File No. S360/S370-30
Order No. GC26-3794-1

OS

IBM Systems Reference Library

# OS Data Management Macro Instructions

## Release 21.7

This publication is intended for application programmers who are writing programs in assembler language; it contains a brief description of each macro instruction and a description of each operand that can be specified in a macro instruction. Descriptions of the macro instructions for the following data management access methods are contained in this publication.

- Basic Direct Access Method (BDAM)
- Basic Indexed Sequential Access Method (BISAM)
- Basic Partitioned Access Method (BPAM)
- Basic Sequential Access Method (BSAM)
- Queued Indexed Sequential Access Method (QISAM)
- Queued Sequential Access Method (QSAM)

This publication does not contain descriptions of macro instructions for specialized application programs such as teleprocessing, graphics, magnetic character readers, optical character reader-sorters, optical character readers, or the time sharing option (TSO).

Before using this publication to code macro instructions, you should be familiar with the information contained in *OS Data Management Services Guide*.

**Second Edition (June 1973)**

This edition replaces the previous edition (numbered GC26-3794-0) and makes that edition obsolete.

This edition applies to OS Release 21.7. It contains the data management macro instructions that were previously described in *IBM System/360 Operating System: Supervisor and Data Management Macro Instructions,* GC28-6647-5, which is now obsolete. Supervisor macro instructions are now described in *OS Supervisor Services and Macro Instructions,* GC28-6646.

Significant system changes are summarized under "Summary of Amendments" following the list of figures. Each technical change is marked by a vertical line to the left of the change.

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using this publication, consult the latest *IBM System/360 and System/370 Bibliography,* GA22-6822, and the technical newsletters that amend that bibliography, to learn which editions and technical newsletters are applicable and current.

Requests for copies of IBM publications should be made to the IBM branch office that serves you.

Forms for readers' comments are provided at the back of the publication. If the forms have been removed, comments may be addressed to IBM Corporation, Programming Center—Publishing, Department D58, Monterey and Cottle Roads, San Jose, California 95193.

# PREFACE

This publication contains descriptions and definitions for the data management macro instructions available in the assembler language. When used with *OS Data Management Services Guide*, this publication provides application programmers who are coding programs in assembler language with the necessary information to code the macro instructions.

This publication is divided into the following parts:

- "Introduction," which contains a general description of macro instructions, the rules to be followed when macro instructions are coded, and a description of the notational conventions used throughout the publication.
- "Macro Instruction Descriptions," which describes the function of each macro instruction and defines how each macro instruction is to be coded. The macro instructions are presented in alphabetic order with each macro instruction beginning on a right-hand page. The standard form of each macro instruction is described first, followed by the description of the list and execute form instructions; the list and execute forms are available only for those macro instructions that pass parameters in a list.
- "Appendix A: Status Information Following an Input/Output Operation," which includes information about error indications available following an input/output operation.
- "Appendix B: Data Management Macro Instructions Available by Access Method," which lists the macro instructions available for each of the data management access methods.
- "Appendix C: Device Capacities," which lists device capacities that can be used as a guide when coding the blocksize and logical record length operands in the DCB macro instruction.
- "Appendix D: DCB Exit List Format and Contents," which describes the format and content of the data control block exit list.
- "Appendix E: Control Characters," which contains information about the control characters used to control spacing and skipping (printers) and stacker selection (card read punch or card punch).
- "Index," which provides topic references to information in this book.

## Prerequisite Publications

Before programs are coded using data management macro instructions, the user should be familiar with the information contained in the following publications:

*OS Assembler Language,* GC28-6514

*OS Data Management Services Guide,* GC26-3746

*OS Introduction,* GC28-6534

*OS Supervisor Services and Macro Instructions,* GC28-6646

## Related Macro Instruction Publications

The following publications contain descriptions of macro instructions for specialized applications such as teleprocessing, graphics, and magnetic/optical character recognition devices:

*OS BTAM,* GC30-2004

*OS Data Management Macros and Services for IBM 1285, 1287, and 1288 Optical Readers,* GC21-5004

*OS Data Management Services and Macro Instructions for IBM 1419/1275,* GC21-5006

*OS GPS for IBM 2250 Display Unit,* GC27-6909

*OS GPS for IBM 2260 Display Station (Local Attachment),* GC27-6912

*OS GPS for IBM 2280 and 2282 Film Units,* GC27-6927

*OS QTAM MCP,* GC30-2005

## Related System Publications

The following publications contain additional information about the operating system. Depending on the requirements of the individual installation, an application programmer may need these publications to code programs for the data management access methods.

*OS Advanced Checkpoint/Restart,* GC28-6708

*OS Data Management for System Programmers,* GC28-6550

*OS Job Control Language Reference,* GC28-6704

*OS Linkage Editor and Loader,* GC28-6538

*OS Programmer's Guide to Debugging,* GC28-6670

*OS System Control Blocks,* GC28-6628

*OS Utilities,* GC28-6586

# CONTENTS

# FIGURES

# SUMMARY OF AMENDMENTS

## Release 21.7

### *New Devices*

Release 21.7 supports the new 3333 Disk Storage. Capacity data for this device has been added to the tables in Appendix C.

### *Miscellaneous Changes*

Descriptions of the CHKPT and WAIT macro instructions have been added to this book. The CHKPT macro instruction was formerly described in *OS Supervisor Services and Macro Instructions,* GC28-6646. The WAIT macro instruction is a duplication of the macro as described in *OS Supervisor Services and Macro Instructions.*

Additional changes were made to:

- Clarify the description of the temporary close (**TYPE=T**) option (CLOSE macro instruction).
- Redefine the contents of register 1 when control is passed to an exit routine (see Appendix D).
- Expand the description of the LEAVE option (FEOV macro instruction for BSAM and QSAM).
- Add a completion code that indicates a backspacing request was ignored on a SYSIN or SYSOUT data set (BSP macro instruction).
- Specify that the return code from SYNADAF is in register 0 rather than register 15.
- Expand the definition of the **Z** byte, which indicates where the system found a directory entry (BLDL macro instruction).
- Delete tape labels as a source of buffer alignment information (BFALN parameter of the DCB macro instruction for all access methods).

Minor technical changes have been made to the BUILDRCD, CHECK, CLOSE, CNTRL, DCB (for all access methods), OPEN, POINT, PRTOV, PUTX, READ, SETL, SETPRT, STOW, SYNADAF, and WRITE (BISAM) macro instructions.

## Release 21

### *Organization of the Publication Changed*

The organization of the *IBM System/360 Operating System: Supervisor and Data Management Macro Instructions* publication has been changed as follows:

- Supervisor macro instructions are now contained in the *OS Supervisor Services and Macro Instructions* publication, GC28-6646. These macro instructions include: ABEND, ATTACH, CALL, CHAP, CHKPT, DELETE, DEQ, DETACH, DOM, DXR, ENQ, EXTRACT, FREEMAIN, GETMAIN, IDENTIFY, LINK, LOAD, POST, RETURN, SAVE, SEGLD, SEGWT, SNAP, SPIE, STAE, STIMER, TIME, TTIMER, WAIT, WAITR, WTL, WTO, WTOR, and XCTL.

- Macro instructions for the time sharing option (TSO) are now contained in the *OS Time Sharing Option Guide to Writing a Terminal Monitor Program or a Command Processor,* GC28-6764. These macro instructions include: GTSIZE, RTAUTOPT, SPAUPOPT, STATUS, STAUTOCP, STAUTOLN, STAX, STBREAK, STCC, STCLEAR, STCOM, STSIZE, STTIMEOU, TCLEARQ, TGET, and TPUT.
- Data management macro instructions are included in this publication. These macro instructions include: BLDL, BSP, BUILD, BUILDRCD, CHECK, CLOSE, CNTRL, DCB, DCBD, ESETL, FEOV, FIND, FREEBUF, FREEDBUF, FREEPOOL, GET, GETBUF, GETPOOL, NOTE, OPEN, POINT, PRTOV, PUT, PUTX, READ, RELEX, RELSE, SETL, SETPRT, STOW, SYNADAF, SYNADRLS, TRUNC, WRITE, and XLATE.

### *DCB Macro Instruction Described by Access Method*

The description of the DCB macro instruction has been rewritten by access method; separate descriptions are included for BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM.

### *Addition of New Card Reader and Card Punch: New Programming Feature*

The programming support for the 3505 card reader and 3525 card punch are included in this edition. The macro instructions changed to support these two devices are CLOSE, CNTRL, DCB (BSAM and QSAM), OPEN and PRTOV.

### *Additional Problem Program Exit Provided: New Programming Feature*

An additional problem program exit is provided to allow the problem program to attempt error recovery or ignore or delay abnormal termination when an ABEND condition occurs during open, close, or end-of-volume processing.

### *DOS/OS Tape Data Set Compatibility: New Programming Feature*

The DOS/OS Interchange feature allows the operating system to recognize and bypass embedded checkpoint records written in DOS tape data sets. The macro instructions changed to support this feature are CNTRL and POINT (also see the OPTCD operand for BSAM and QSAM for the OPTCD that is supplied in the DD statement for the data set).

### *Additional Information Added*

Three appendixes have been added to aid in coding programs:

- Appendix C contains device capacities to aid in determining maximum record length or blocksize for various input/output devices. This information can also be used to determine the optimum blocking factor when blocked records are used.
- Appendix D contains the format of the problem program exit list. This appendix shows the hexadecimal code for each type of exit and a brief description of exit list processing.
- Appendix E contains a description of the control characters that can be used for stacker selection or printer spacing and skipping.

### *Miscellaneous Changes*

- The format of the publication has been changed to aid in coding operands. The description of each operand includes the type of notation that can be used when the macro instruction operand is coded.

- Macro instructions that require a data control block address for an open data set have been clarified to indicate that the data set must be open.
- Clarification for the following macro instructions has been included: BLDL, BUILDRCD, CHECK, CLOSE, ESETL, FREEPOOL, GET, NOTE, OPEN, POINT, PRTOV, PUT, READ, RELEX, STOW, and WRITE.

# INTRODUCTION

## Data Management Macro Instructions

A set of macro instructions is provided by IBM for communicating 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 processing of the macro instruction by the assembler program results in a macro expansion, generally consisting of executable instructions and data in the form of assembler-language statements. The data fields are the parameters to be passed to the access method routine; the executable instructions generally consist of a branch around the data fields, instructions to load registers, and either a branch instruction or supervisor call (SVC) to give control to the proper program. The exact macro expansion appears as a part of the assembler listing.

A listing of a macro definition from SYS1.MACLIB (the library in which macro definitions are stored) can be obtained by using the utility program IEBPTPCH, which is described in the *OS Utilities* publication.

Before macro instructions are coded using this publication, the user should be familiar with the information contained in the *OS Data Management Services Guide.*

When programs that request supervisor services are being coded, the user should be familiar with the information contained in the *OS Supervisor Services and Macro Instructions* publication.

When programs are being coded for more specialized applications such as teleprocessing, graphics, and character recognition, the publication that describes the specific access method and/or device type should be used. Publications containing descriptions of the macro instructions for teleprocessing, graphics, and character recognition devices are listed in the preface of this publication.

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 provides a list of the macro instructions available for each access method.

## Coding Aids

The symbols [ ], { }, and ,... are used in this publication to help define the macro instructions. These symbols are not coded; they are only to indicate how a macro instruction may be written; their general definitions are given below:

If more than one item is enclosed in brackets, one or none of the items may be coded. For example, if this is shown

```
[REREAD]
[,LEAVE]
```

you may code REREAD or LEAVE, or neither REREAD nor LEAVE.

One of the operands from the vertical stack within braces must be coded, depending on which of the associated services is desired. For example, if this is shown

{INPUT}
{OUTPUT}

you must code either INPUT or OUTPUT, but not both INPUT and OUTPUT.

,... indicates that more than one set of operands may be designated in the same macro instruction.

## Coding Macro Instructions

Data management macro instructions are written in the assembler language and, as such, are subject to the rules contained in the *OS Assembler Language* publication. Data management macro instructions, like all assembler language instructions, are written in the following format:

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

The operands are used to specify services and options to be used and are written according to the following general rules:

- If the selected operand is shown in bold capital letters (for example, **MACRF=WL**), code the operand exactly as shown.
- If the selected operand 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 italic lowercase letters (for example, *dcb address*), substitute the indicated address, name, or value.
- If the operand is a combination of bold capital letters and italic 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 italic lowercase letters.
- Commas and parentheses are coded exactly as shown, except that the comma following the last operand coded should be omitted. The use of commas and parentheses is indicated by brackets and braces in the same manner as brackets and braces indicate the use of operands.
- Several macro instructions contain the designation **'S'**. This operand, when used, must have the apostrophe on both sides of the **S**.

When substitution of a name, value, or address is required, the notation used to specify the operand depends on the operand being coded. The following shows two examples of the notations used to indicate how an operand can be coded.

**DDNAME=***symbol*

In the above example, the only type of operand that can be coded is a valid assembler-language symbol.

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

In the above example, the operand that can be substituted can be an RX-type address, any of the general registers 2 through 12, or general register 1.

The following describes the meaning of each notation used to show how an operand can be coded.

symbol

When this notation is shown, the operand can be any valid assembler-language symbol.

decimal digit

When this notation is shown, the operand can be any decimal digit up to the maximum value allowed for the specific operand being described.

(2-12)

When this notation is shown, the operand specified can be any of the general registers 2 through 12. All registers as operands must be coded in parentheses; for example, if register 3 is coded, it is coded as (3). When one of the registers 2 through 12 is used, it can be coded as a decimal digit, symbol (equated to a decimal digit), or an expression that results in a value of 2 through 12.

(1)

When this notation is shown, general register 1 can be used as an operand. When used as an operand in a macro instruction, the register must be specified as the decimal digit 1 enclosed in parentheses as shown above.

(0)

When this notation is shown, general register 0 can be used as an operand. When used as an operand in a macro instruction, the register must be specified as the decimal digit 0 enclosed in parentheses as shown above.

RX-Type Address

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

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

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

A-Type Address

When this notation is shown, the operand can be specified as any address that can be written as a valid assembler-language A-type address constant. An A-type address constant can be written as an absolute value, a relocatable symbol, or 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, refer to the *OS Assembler Language* publication.

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 non-relocatable symbol, a self-defining term, or the length attribute reference. For more details about absolute expressions, refer to the *OS Assembler Language* publication.

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, refer to the *OS Assembler Language* publication.

## *Rules for Register Usage*

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

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

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

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

- If the register is to contain a value, it must be placed in the low-order portion of the register unless the macro instruction description states otherwise. Any unused bits in the register should be set to zero.
- If the register is to contain an 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.

### *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.

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

```
Name     Operation  Operand                                  Comments
NAME1    OP1        OPERAND1,OPERAND2,OPERAND3,OPERAND4,OPERAND5,OPERAND6,OPERAND7,OPEX
                    RAND8                                    THIS IS ONE WAY
NAME2    OP2        OPERAND1,OPERAND2,                       THIS IS ANOTHER WAY          X
                    OPERAND3,                                                             X
                    OPERAND4
```

# MACRO INSTRUCTION DESCRIPTIONS

# BLDL—Build a Directory Entry List (BPAM)

The BLDL macro instruction is used to complete a list of information from the directory of a partitioned data set. The problem program must supply an area in main storage; the area 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 instruction is issued. Data set member names in the list must be in alphameric order. All read and write operations using the same data control block must have been tested for completion before the BLDL macro instruction is issued.

The BLDL macro instruction is written as follows:

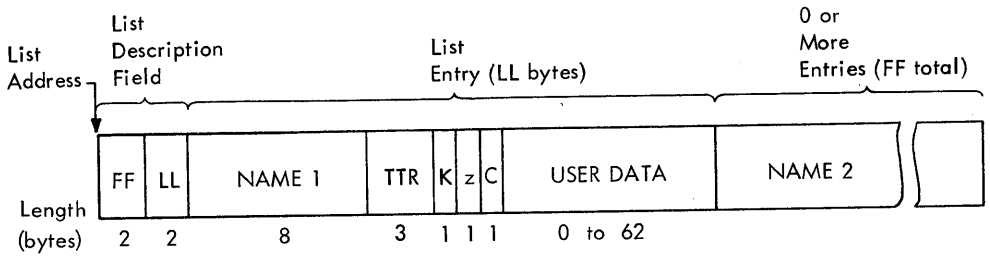
| [symbol] | BLDL | dcb address, list address |
|---|---|---|

*dcb address*—RX-type Address, (2-12), (1), or the Decimal Digit 0

The *dcb address* operand specifies the address of the data control block for an open partitioned data set, or zero can be specified to indicate that the data set is in a job library, step library, or link library.

*list address*—RX-Type Address, (2-12), or (0)

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

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

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

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

**Z** Indicates where the system found the directory entry:

| | |
|---|---|
| 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 eight bits:

| | |
|---|---|
| Bit 0=0 | Indicates a member name. |
| Bit 0=1 | Indicates an alias. |
| Bits 1-2 | Indicate the number of TTRN fields (maximum of three) in the user data field. |
| Bits 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 instruction, 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 one of the following return codes; the three high-order bytes of register 15 are set to zero.

| Hexadecimal Code | Meaning |
|---|---|
| 00 | Successful completion. |
| 04 | 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 | A permanent input/output error was detected when the system attempted to search the directory. |

# BSP—Backspace a Physical Record (BSAM—Magnetic Tape and Direct Access Only)

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

**Magnetic Tape:** A backspace is always made toward the load point.

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

With SYSIN or SYSOUT data sets, a BSP macro instruction is ignored but a completion code is returned.

The BSP macro instruction is written as follows:

| [symbol] | BSP | dcb address |
|---|---|---|

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

The *dcb address* operand specifies the address of the data control block for the volume to be backspaced. The data set on the volume to be backspaced must be opened before issuing the BSP macro 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 zero.

| Hexadecimal Code | Meaning |
|---|---|
| 00 | Successful completion |
| 04 | Unsuccessful completion (includes encountering a tapemark or beginning of an extent) |
| 08 | A backspacing request was ignored on a SYSIN or SYSOUT data set |

# BUILD—Build a Buffer Pool (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM)

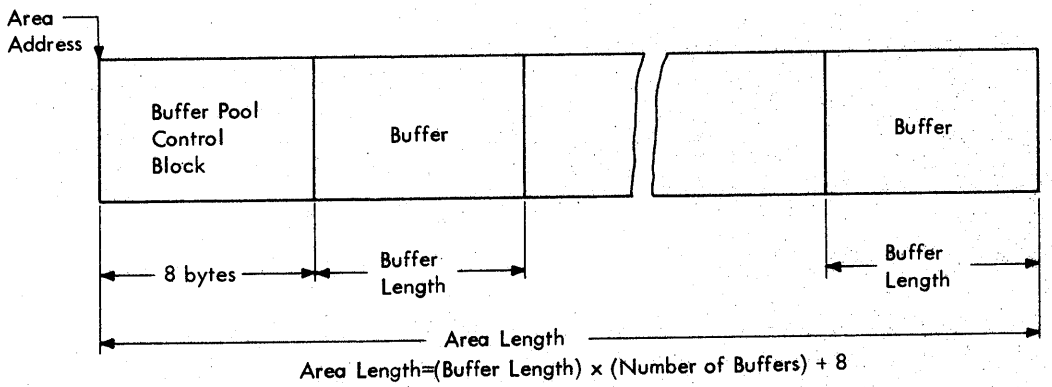
The BUILD macro instruction is used to construct a buffer pool in a main-storage area provided by the problem program. The buffer pool may be used by more than one data set through separate data control blocks. Individual buffers are obtained from the buffer pool using the GETBUF macro instruction, and buffers are returned to the buffer pool using a FREEBUF macro instruction. Refer to the *OS Data Management Services Guide* for an explanation of the interaction of the DCB, BUILD, and GETBUF macro instructions in each access method, as well as the buffer size requirements.

The BUILD macro instruction is written as follows:

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

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

The *area address* operand specifies the address of the main-storage area to be used as a buffer pool. The area must start on a fullword boundary. The following illustration shows the format of the buffer pool:




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

The *number-of-buffers* operand specifies the number of buffers in the buffer pool up to a maximum of 255.

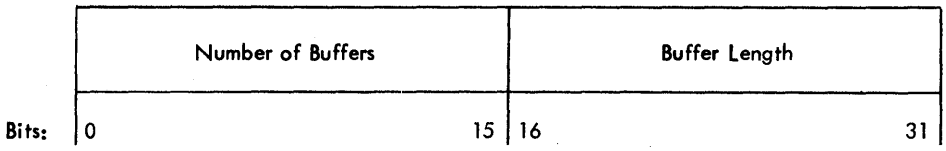
*buffer length*—symbol, decimal digit, absexp, or (2-12)

The *buffer length* operand specifies the length, in bytes, of each buffer in the buffer pool. The value specified for the buffer length must be a fullword multiple; otherwise the system rounds the value specified to the next higher fullword multiple. The maximum length that can be specified is 32,760 bytes. For QSAM, the buffer length must be at least as large as the value specified in the blocksize (DCBBLKSI) field of the data control block.

(0)—Coded as shown

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

Register 0



# BUILDRCD—Build a Buffer Pool and a Record Area (QSAM)

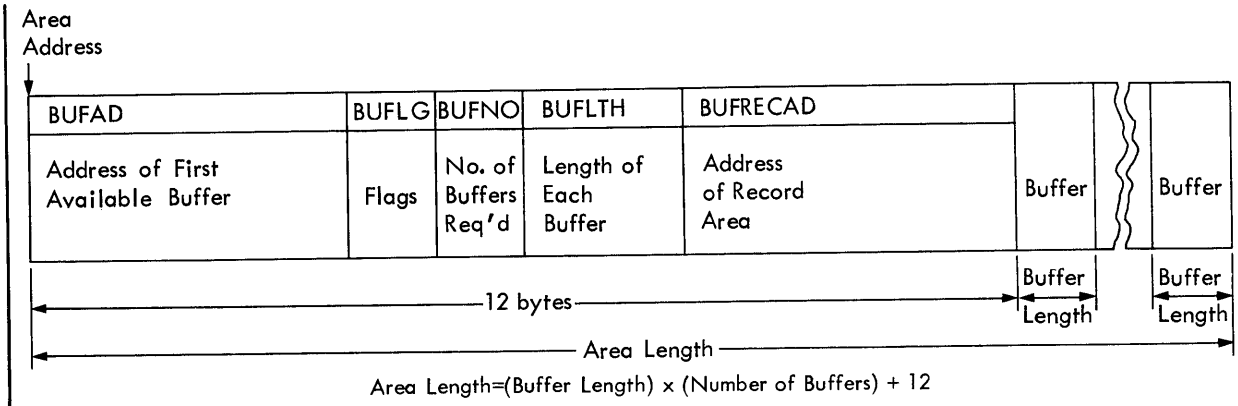
The BUILDRCD macro instruction causes a buffer pool and a record area to be constructed in a user-provided storage area. The macro is used only for variable-length, spanned records processed in QSAM locate mode. Use of this macro before the data set is opened, or before the end of the DCBEXIT routine, will automatically invoke a logical-record interface rather than a segment interface for variable-length, spanned records.

The standard form of the BUILDRCD macro instruction 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,<br>record area address[,record area length] |

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

The *area address* operand specifies the address of the main-storage 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:




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

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

The *number of buffers* operand specifies the number of buffers, up to a maximum of 255, to be in the buffer pool.

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

The *buffer length* operand specifies the length, in bytes, of each buffer in the buffer pool. The value specified for the buffer length must be a fullword multiple; otherwise, the system rounds the value specified to the next higher fullword multiple. The maximum length that can be specified is 32,760.

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

The *record area address* operand specifies the address of the main-storage area to be used as a record area. The area must start on a doubleword boundary and have a length of the maximum logical record (LRECL) plus 32 bytes.

*record area length*—symbol, decimal digit, absexp, or (2-12)

The *record area length* operand specifies the length of the record area to be used. The area must be as long as the maximum length logical record plus 32 bytes for control information. If the record area length operand is omitted, the problem program must store the record area length in the first four bytes of the record area.

**Note:** It is the user's responsibility to release the buffer pool and the record area after a CLOSE macro instruction has been issued for all the data control blocks using the buffer pool and the record area.

# BUILDRCD—List Form

The list form of the BUILDRCD macro instruction 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 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 list form only.

The list form of the BUILDRCD macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | BUILDRCD | area address,number of buffers,buffer length,<br>record area address[,record area length]<br>,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**—Coded as shown

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

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

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

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

# BUILDRCD—Execute form

A remote control program parameter list is referred to, and can be modified by, the execute form of the BUILDRCD macro instruction. 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 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 for the execute form only.

The execute form of the BUILDRCD macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | BUILDRCD | [area address],[number of buffers],<br>[buffer length],[record area address],<br>[record area length],<br>MF=(E,{control program list address<br>(1)}) |

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

*number of buffers*—absexp

*buffer length*—absexp

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

*record area length*—absexp

**MF=(E,** { *control program list address* <br> { **(1)** } **})**

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

**E**—Coded as shown

*control program 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 instruction causes the active task to be placed in the wait condition, if necessary, until the associated input or output operation is completed. The input or output operation is then tested for errors and exceptional conditions. If the operation is completed successfully, control is returned to the instruction following the CHECK macro instruction. 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 instruction.

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

**When Reading:** Volume switching is automatic. The end-of-data-set (EODAD) routine is given control if an input request is made after all the records have been retrieved.

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

For BPAM and BSAM, a CHECK macro instruction must be issued for each input and output operation, and must be issued in the same order as the READ or WRITE macro instructions were issued for the data set. For BDAM or BISAM, either a CHECK or WAIT macro instruction can be used. However, if both a CHECK and WAIT macro instruction are used, the CHECK macro instruction must be issued after the WAIT macro instruction.

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

The CHECK macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | CHECK | decb address [ ,DSORG= { IS / ALL } |

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

The *decb address* operand specifies the address of the data event control block created by the associated READ or WRITE macro instruction or used by the associated input or output operation.

**DSORG=** {**IS**}
{**ALL**}

The **DSORG** operand specifies the type of data set organization.

The following describes the characters that can be coded.

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

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

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

# CHKPT—Take a Checkpoint for Restart Within a Job Step (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM)

The CHKPT macro instruction establishes a checkpoint for the job step. If the step terminates abnormally, it is automatically restarted from the checkpoint. On restart, execution resumes with the instruction that follows the CHKPT instruction. If the step again terminates abnormally (before taking another checkpoint), it is again restarted from the checkpoint. When several checkpoints are taken, the step is automatically restarted from the most recent checkpoint.

Automatic restart from a checkpoint is suppressed if:

1. The job step completion code is not one of a set of codes specified at system generation.
2. The operator does not authorize the restart.
3. The restart definition parameter of the JOB or EXEC statement specifies no restart (RD=NR) or no checkpoint (RD=NC or RD=RNC).
4. The CANCEL operand appears in the last CHKPT macro instruction issued before abnormal termination.

Under any of these conditions, automatic checkpoint restart does not occur. Automatic step restart (restart from the beginning of the job step) can occur, except under condition 1 or 2, or when the job step was restarted from a checkpoint prior to abnormal termination. Automatic step restart is requested through the restart definition parameter of the JOB or EXEC statement (RD=R or RD=RNC).

When automatic restart is suppressed or unsuccessful, a deferred restart can be requested by submitting a new job. The new job can specify restart from the beginning of the job step or from any checkpoint for which there is an entry in the checkpoint data set.

The checkpoint data set contains the information necessary to restart the job step from a checkpoint. The control program records this information when the CHKPT macro instruction is issued. The macro refers to the data control block for the data set, which must be on a magnetic tape or direct-access volume. A tape can have standard labels, nonstandard labels, or no labels.

If the checkpoint data set is not open when CHKPT is issued, the control program opens the data set and then closes it after writing the checkpoint entry. If the data set is physically sequential and is opened by the control program, the checkpoint entry is written over the previous entry in the data set, unless the DD statement specifies DISP=MOD. By writing entries alternately into two checkpoint data sets, it is possible to keep the entries for the two most recent checkpoints while deleting those for earlier checkpoints.

The data control block for the checkpoint data set must specify:
**DSORG=PS** or **PO, RECFM=U** or **UT, MACRF=(W), BLKSIZE=***nnn,* and **DDNAME=***any name*

where *nnn* is at least 600 bytes, but not more than 32,760 bytes for magnetic tape and not more than the track length for direct access. (If the data set is opened by the

control program, blocksize need not be specified; the device-determined maximum blocksize is assumed if no blocksize is specified.) For 7-track tape, the data control block must specify TRTCH=C; for direct access, it must specify or imply KEYLEN=0. To request chained scheduling, OPTCD=C and NCP=2 must be specified. With direct access, OPTCD=W can be specified to request validity checking for write operations, and OPTCD=WC can be specified to combine validity checking and chained scheduling.

The standard form of the CHKPT macro instruction is written as shown below. Information about the list and execute forms follows this description.

| | | |
|---|---|---|
| [symbol] | CHKPT | {dcb address [,checkid address [,checkid length]] / CANCEL [,'S']} |

*dcb address*

The *dcb address* operand specifies the address of the data control block for the checkpoint data set.

*checkid address*

The *checkid address* operand specifies the address of the checkpoint identification field. The contents of the field are used when the job step is to be restarted from the checkpoint. They are used by the control program in requesting operator authorization for automatic restart. You can use it for requesting deferred restart.

If the next operand specifies the length of the field (*checkid length*), or if it is omitted to imply a length of eight bytes, the field must contain the checkpoint identification when the CHKPT macro instruction is issued. If the next operand is written as **'S'**, the identification is generated and placed in the field by the control program. If both operands are omitted, the control program generates the identification, but does not make it available to the problem program. In each case, the identification is written in a message to the operator.

The control program writes the checkpoint identification as part of the entry in the checkpoint data set. For a sequential data set, the identification can be any combination of up to 16 letters, digits, printable special characters, and blanks. For a partitioned data set, it must be a valid member name of up to eight letters and digits, starting with a letter. The identification for each checkpoint should be unique.

If the control program generates the identification, the identification is eight bytes in length. It consists of the letter C followed by a seven-digit decimal number. The number is the total number of checkpoints taken by the job, including the current checkpoint, checkpoints taken earlier in the job step, and checkpoints taken by any previous job steps.

*checkid length*

The *checkid length* operand specifies the length in bytes of the checkpoint identification field. The maximum length is 16 bytes if the checkpoint data set is physically sequential, 8 bytes if it is partitioned. For a partitioned data set, the field can be longer than the actual identification, if the unused portion is blank. If the operand is omitted, the implied length is eight bytes.

If you code **'S'** the control program supplies the checkpoint identification. The implied field length is eight bytes.

**CANCEL**

The **CANCEL** operand cancels the request for automatic restart from the most recent checkpoint. If another checkpoint is taken before abnormal termination, the job step can be restarted at that checkpoint.

When control is returned, register 15 contains one of the following return codes:

| Hexadecimal Code | Meaning |
|---|---|
| 00 | *Successful completion.* Code 00 is also returned if the RD parameter was coded as RD=NC or RD=RNC to totally suppress the function of CHKPT. |
| 04 | *Restart has occurred* at the checkpoint taken by the CHKPT macro instruction during the original execution of the job. A request for another restart of the same checkpoint is normally in effect. If a deferred restart was performed and RD=NC, NR, or RNC was specified in the resubmitted deck, a request for another restart is not in effect. |
| 08 | *Unsuccessful completion.* A checkpoint entry was not written, and a restart from this checkpoint was not requested. A request for an automatic restart from a previous checkpoint remains in effect.<br><br>One of the following conditions exists:<br><br>• The parameters passed by the CHKPT macro instruction are invalid.<br><br>• The CHKPT macro instruction was executed in an exit routine other than the end-of-volume exit routine.<br><br>• A STIMER macro instruction has been issued, and the time interval has not been completed.<br><br>• A WTOR macro instruction has been issued, and the reply has not been received.<br><br>• The checkpoint data set is on a direct-access volume and is full. Secondary space allocation was requested and performed. (Secondary space allocation is performed for a checkpoint data set, but the allocated space is not used. However, had secondary allocation not been requested, the job step would have been abnormally terminated.)<br><br>• In a system with MVT or MFT with subtasking, the job step comprises more than one task.<br><br>• In a system with MVT, the job step has been allocated storage through the rollout/rollin option. |

- A graphics-type DSORG has been found in an open DCB. Graphic devices are not supported in Checkpoint/Restart.

0C — *Unsuccessful completion.* An uncorrectable error occurred in writing the checkpoint entry or in completing queued access method input/output operation that were begun before the CHKPT macro instruction was issued. A partial, invalid checkpoint entry may have been written. If the entry has a programmer-specified *checkid*, and the checkpoint data set is sequential, a different *checkid* should be specified the next time CHKPT is executed. If the data set is partitioned, a different *checkid* need not be specified. This code is also returned if the checkpoint routine tries to open the checkpoint data set and the DD statement for the data set is missing.

10 — *Successful completion with possible error condition.* The task has control, by means of an explicit or implied use of the ENQ macro instruction, of a serially reusable resource; if the task terminates abnormally, it will not have control of the resource when the job step is restarted. The user's program must, therefore, reissue the ENQ macro instruction.

14 — *Unsuccessful completion.* An end-of-volume condition occurred while writing a checkpoint entry on a tape data set. The checkpoint was cancelled, but processing of the user's program continues.

When one of the errors indicated by code 08, 0C, 10, or 14 occurs, the system prints an error message on the operator's console. The message indicating code 08 or 0C contains a code that further identifies the error. The operator should report the message contents to the programmer.

**Note:** Successful use of the CHKPT macro instruction requires some care in the selection of checkpoints. For a detailed discussion of checkpoint requirements, refer to *OS Advanced Checkpoint/Restart.*

# CHKPT—List Form

The list form of the CHKPT instruction is used to construct a control program parameter list.

The description of the standard form of the CHKPT macro provides the explanation of the function of each operand. The description of the standard form also indicates which operands are optional and which are required in at least one of the list and execute forms. The format description below indicates the optional and required operands in the list form only. Note that the CANCEL operand, which can be coded in the standard form, cannot be coded in the list form.

The list form of the CHKPT macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | CHKPT | [dcb address],[checkid address],[checkid length]<br>MF=L ['S'] |

*symbol*

The *symbol* operand specifies any symbol that is valid in the assembler language.

*address*

The *address* operand specifies any address that may be written in an A-type address constant.

*length*

The *length* operand specifies any absolute expression that is valid in the assembler language.

**MF=L**

The **MF=L** operand indicates the list form of the CHKPT macro instruction.

# CHKPT—Execute Form

A control program parameter list is referred to, and can be modified by, the execute form of the CHKPT macro.

The description of the standard form of the CHKPT macro provides the explanation of the function of each operand. The description of the standard form also indicates which operands are optional and which are required in at least one of the list and execute forms. The format description below indicates the optional and required operands for the execute form only. Note that the CANCEL operand, which can be coded in the standard form, cannot be coded in the execute form.

The execute form of the CHKPT macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | CHKPT | [dcb address],[checkid address],[checkid length / 'S']<br>,MF=(E,{control program list address / (1)}) |

*symbol*

The *symbol* operand specifies any symbol that is valid in assembler language.

*address*

The *address* operand specifies any address that is valid in an RX-type instruction, or one of the general registers 2 through 12, previously loaded with the indicated address. You may designate the register symbolically or with an absolute expression; always code it in parentheses.

*length*

The *length* operand specifies any absolute expression that is valid in assembler language, or one of the general registers 2 through 12, previously loaded with the indicated value. You may designate the register symbolically or with an absolute expression; always code it in parentheses.

**MF=(E,** { *control program list address* **(1)** **})**

This operand specifies the execute form of the macro instruction using a control program parameter list. The address of the control program parameter list can be coded as described under address, or can be loaded into register 1, in which case code **MF=(E,(1))**.

# CLOSE—Logically Disconnect a Data Set (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM)

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

Associated data sets for a 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 *OS Data Management Services Guide.*

A FREEPOOL macro instruction should normally follow a CLOSE macro instruction 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 instruction 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] |

*dcb address*—A-Type Address or (2-12)

The *dcb address* operand specifies the address of the data control block for the opened data set that is to be closed.

*option*

These options indicate the volume positioning that is to occur when the data set is closed. The option operand is ignored for ISAM data sets and also for data sets on other than magnetic tape or direct-access devices. The options are:

**REREAD** — Specifies that the current volume is to be positioned to reprocess the data set.

**LEAVE** — Specifies that the current volume is to be positioned to the logical end of the data set. (beginning if RDBACK)

**REWIND** — Specifies that the current volume is to be positioned at the load point, regardless of the direction of processing. **REWIND** cannot be specified when **TYPE=T** is specified.

**DISP** — Specifies that the current volume is to be positioned according to the position implied by the **DISP** parameter of the corresponding DD statement, 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** | Rewind and unload the current volume. |

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

**TYPE=T**—Coded as shown

You can code CLOSE **TYPE=T** and temporarily close 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 don't have to issue another OPEN macro instruction to continue processing the same data set. This option cannot be used in a SYNAD routine.

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

If you code the release (RLSE) operand on the DD statement for the data set, it is ignored by temporary close, but any unused space will be released when you finally issue the normal CLOSE macro instruction.

Refer to *OS Data Management Services Guide* for additional information and coding restrictions.

# CLOSE—List Form

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

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

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

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

A zeroed work area on a fullword 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 one before this list can be used with the execute-form instruction.

A parameter list constructed by a CLOSE macro instruction, 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 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 CLOSE macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | CLOSE | ( [dcb address] , [option], . . . ) [, TYPE=T] , MF=L |

*dcb address*—A-Type Address

*option*—Same as standard form

**TYPE=T**—Coded as shown

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

**MF=L**—Coded as shown

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

# CLOSE—Execute Form

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

The description of the standard form of the CLOSE 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 CLOSE macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | CLOSE | [([dcb address], [option],...)] [,TYPE=T]<br>,MF=(E,{data management list address}) |

*dcb address*—RX-Type Address or (2-12)

*option*—Same as standard form

**TYPE=T**—Same as standard form

**MF=(E,** { *data management list address* **})**
{ **(1)** }

This operand specifies that the execute form of the CLOSE macro instruction is being used, and an existing data management parameter list (created by a list-form instruction) will be used. The **MF=** operand is coded as described in the following:

**E**—Coded as shown

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

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

The CNTRL macro instruction is used to control magnetic tape drives (BSAM only) and to control online card readers, 3525 card punches (read and print features), and printers (BSAM and QSAM). The MACRF operand of the DCB macro instruction must specify a C. The CNTRL macro instruction must not be used for SYSOUT data sets that are temporarily stored on a direct-access device. For BSAM, all input and output operations must be tested for completion before the CNTRL macro instruction is issued. The control facilities available are as follows:

**Card Reader:** Provides stacker selection.

QSAM—The CNTRL macro instruction is issued whenever it is necessary to read a new card. For unblocked records, a CNTRL macro instruction should be issued after every input request except the last. For blocked records, a CNTRL macro instruction is issued after the last logical record on each card is retrieved, except for the last input request. The move mode of the GET macro instruction must be used, and the number of buffers (BUFNO field of the DCB) must be one.

BSAM—The CNTRL macro instruction should be issued after every input request. If, however, the device is allocated to SYSIN, the CNTRL macro instruction does not need to be issued after the request because the CLOSE macro instruction places the last card in the same stacker as the preceding card.

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

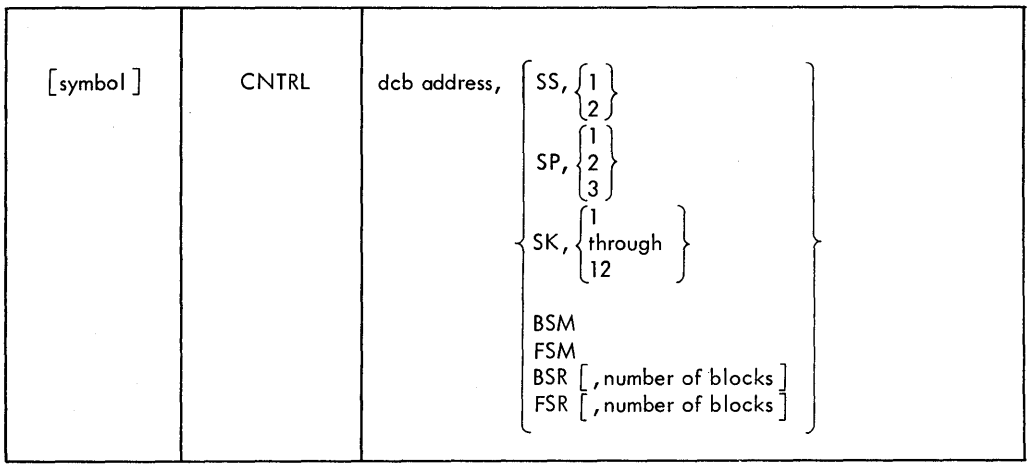
**Magnetic Tape:** Provides method of forward spacing and backspacing (BSAM only).

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

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

**3525 Printing:** Provides line spacing or a skip to a specific printing line on the card. The card contains 25 printing lines; the odd numbered lines 1 through 23 correspond to the printer skip channels 1 through 12 (see the **SK** operand). For additional information about 3525 printing operations, refer to *OS Data Management Services Guide.*

The CNTRL macro instruction is written as follows:




*dcb address*

The *dcb address* operand specifies the address of the data control block for the data set opened for the online device.

**SS, {1}**
**{2}**

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

**{1}**
**SP, {2}**
**{3}**

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

**{1}**
**SK, through**
**{12}**

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

**BSM**—Coded as shown

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

**FSM**—Coded as shown

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

**BSR**—Coded as shown

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

**FSR**—Coded as shown

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

*number of blocks*—symbol, decimal digit, absexp, or (2-12)

The *number-of-blocks* operand specifies the number of blocks to backspace (see **BSR** operand) or forward space (see **FSR** operand) the magnetic tape. The maximum value that can be specified is 32,767. If the *number-of-blocks* operand is omitted, one 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 routine is designated, the task is abnormally terminated. Register contents, when control is passed to the error analysis routine, are shown in Appendix A. If a tapemark is encountered for **BSR** or **FSR,** control is returned to the processing program, and register 15 contains a count of the uncompleted forward spaces or backspaces. If the operation is completed normally, register 15 contains the value zero.

# DCB—Construct a Data Control Block (BDAM)

The data control block for a basic direct access method (BDAM) data set is constructed during assembly of the problem program. The DCB macro instruction must not be coded within the first 16 bytes of a control section (CSECT). The **DSORG** and **MACRF** operands must be coded in the DCB macro instruction, but the other operands can be supplied from other sources. Each of the BDAM DCB operand descriptions contains a heading, "Source." The information under this heading describes the sources from which an operand can be supplied to the data control block.

Before a DCB macro instruction for a BDAM data set is coded, the following characteristics of direct data sets should be considered.

- The problem program must synchronize I/O operations by issuing a CHECK or WAIT macro instruction to test for completion of read and write operations.
- A BDAM data set is created using the basic sequential access method (BSAM). A special operand (**MACRF=WL**) specifies that BSAM is being used to create a BDAM data set. Operand descriptions for the BDAM DCB macro instruction include information about both creating and processing a BDAM data set.
- Although a BDAM data set can contain blocked records, the problem program must perform all blocking and deblocking of records. BDAM provides only the capability to read or write a data block, but the data block can contain multiple logical records assembled by the problem program.
- When a BDAM data set is being created, buffers can be acquired automatically, but buffer control must be provided by the problem program. The problem program must place data in the output buffer before issuing a WRITE macro instruction to write the data block.
- When a BDAM data set is being processed, the problem program can control all buffering, or dynamic buffering can be specified in the DCB macro instruction and subsequently requested in a READ macro instruction.
- The actual organization of a direct data set is determined by the programmer to meet the needs of the application. The data set can be processed by using one of the following addressing methods:

1. Actual device addresses (in the form MBBCCHHR).
2. Relative track addresses (in the form TTR). These addresses specify a track (and a record on the track) of the direct-access device relative to the beginning of the data set.
3. Relative block addresses can be used with fixed-length records. These addresses specify a data block relative to the beginning of the data set.

For additional information about the characteristics of BDAM data sets, refer to *OS Data Management Services Guide.*

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

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

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

**F** — Specifies that each buffer is aligned on a fullword boundary that is not also a doubleword boundary.

**D** — Specifies that each buffer is aligned on a doubleword boundary.

If the BUILD macro instruction is used to construct the buffer pool or if the problem program controls all buffering, the problem program must provide the main-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. If both the **BFALN** and **BFTEK** operands are specified, they must be supplied from the same source.

**BFTEK=R**

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

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

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

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

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

**BLKSIZE**=*absexp* (maximum value is 32,760)

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

The actual value that can be specified in the **BLKSIZE** operand depends on the record format and the type of direct-access device being used. If the track-overflow feature is being used or if variable-length spanned records are being 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 of this publication. For additional information about device capacity and space allocation, refer to *OS Data Management Services 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.

**BUFCB**=*relexp*

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

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

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

**BUFL**=*absexp* (maximum value is 32,760)

The **BUFL** operand specifies the length, in bytes, of each buffer in the buffer pool when the buffers are acquired automatically (create BDAM) or dynamically (existing BDAM).

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

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

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

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

**BUFNO**=*absexp* (maximum value is 255)

The **BUFNO** operand specifies the number of buffers to be constructed by a BUILD macro instruction, or it specifies the number of buffers and/or segment work areas to be acquired by the system.

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

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

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

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

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

**DDNAME**=*symbol*

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

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

**DSORG=** {**DA**}
{**DAU**}

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

**DA** — Specifies a direct organization data set.

**DAU** — Specifies a direct organization data set that contains location-dependent information.

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

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

**EXLST=***relexp*

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

Refer to Appendix D of this publication for the format and requirements of exit list processing. For additional information about exit list processing, refer to *OS Data Management Services Guide.*

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

**HIARCHY=** {**0**} {**1**}

The **HIARCHY** operand specifies the main-storage hierarchy in which the buffer pool is to be constructed. The following describes the characters that can be specified:

**0** — Specifies that the buffer pool is constructed in processor storage.

**1** — Specifies that the buffer pool is constructed in IBM 2361 Core Storage.

The storage hierarchy can also be specified in a GETPOOL macro instruction. If **HIARCHY** is omitted from all sources, the system constructs the buffer pool in processor storage.

The buffer pool is constructed in the user region or partition within the indicated hierarchy; if space is not available within the indicated hierarchy, the task is abnormally terminated. The **HIARCHY** operand is ignored in systems that do not have hierarchy support. The **HIARCHY** operand must not be specified for MVT systems with Model 65 multiprocessing.

**Source:** The **HIARCHY** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or in the **HIARCHY** operand of a GETPOOL macro instruction.

**KEYLEN=***absexp* (maximum value is 255)

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

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

**LIMCT**=*absexp*

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

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

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

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

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  }    [X  ]    [C])                    }
       {(W  {A  }    [C])                             }
            {K  }
            {I  }
            {AK }
            {AI }
            {KI }
            {AKI}
       {(R  {K  }    [X  ]    [C]  ,W  {A  }    [C]) }
            {I  }    [S  ]             {K  }
            {KI }    [XS ]             {I  }
                                       {AK }
                                       {AI }
                                       {KI }
                                       {AKI}
```

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

**A** — Specifies that data blocks are to be added to the data set.

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

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

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

**R** — Specifies that READ macro instructions are used. READ macro instructions can be issued when the data set is opened for INPUT, OUTPUT, or UPDAT.

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

**W** — Specifies that WRITE macro instructions are used. WRITE macro instructions can be issued only when the data set is opened for OUTPUT OR UPDAT.

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

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

**OPTCD= [R][E][F][W]**
**[A]**

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

**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.

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

**W** — Specifies that the system performs a validity check for each record written. If the device is a 2321, the system performs a validity check for each write operation whether it is requested or not.

**Note:** If **OPTCD=A** and **R** are both omitted, the system requires that READ and WRITE macro instructions provide 3-byte relative track addresses (in the form TTR).

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 (other than **A** or **R**) in the DCB macro or DD card when you reopen your data set, or zero out the **OPTCD=A** or **R** bits in the data control block exit routine. Note that the first method will prevent the open routines from merging any of the other **OPTCD** bits from the format-1 DSCB into the DCB. Both methods will update the **OPTCD** in the DSCB if the open is for OUTPUT, OUTIN, or UPDAT.

**Source:** The **OPTCD** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or by the problem program before an OPEN macro instruction is issued to open the data set.

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

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

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

**F** — Specifies that the data set contains fixed-length records.

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

**T** — Specifies that the track-overflow feature is used with the data set. The track-overflow feature allows a record to be partially written on one track and the remainder is written on the following track (if required).

**U** — Specifies that the data set contains undefined-length records.

**V** — Specifies that the data set contains variable-length records.

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

**SYNAD=***relexp*

The **SYNAD** operand specifies the address of the error analysis routine to be given control when an uncorrectable input/output error occurs. The contents of the registers when the error analysis routine is given control are described in Appendix A of this publication.

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 which uses the address in register 14 to return control to the system. When control is returned in this manner, the system returns control to the problem program and proceeds as though no error had been encountered. When a BDAM data set is being created, a return from the error analysis routine to the system causes abnormal termination of the task.

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

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

# DCB—Construct a Data Control Block (BISAM)

The data control block for the basic indexed sequential access method (BISAM) is constructed during assembly of the problem program. The DCB macro instruction must not be coded within the first 16 bytes of a control section (CSECT). The **DSORG** and **MACRF** operands must be coded in the DCB macro instruction, but the other DCB operands can be supplied from other sources. Each BISAM DCB operand description contains a heading, "Source." The information under this heading describes the sources from which the operand can be supplied to the data control block.

Before a DCB macro instruction for a BISAM data set is coded, the following characteristics of BISAM data sets should be considered:

- BISAM cannot be used to create an indexed sequential data set.
- BISAM performs the functions of direct retrieval of a logical record by key, direct update-in-place for a block of records, direct insertion of a new record in its correct key sequence.
- Buffering can be controlled by the problem program, or dynamic buffering can be specified in the DCB macro instruction and subsequently requested in a READ macro instruction.
- The problem program must synchronize I/O operations by issuing a CHECK or WAIT macro instruction to test for completion of Read and Write operations.
- Additional **DCB** operands provide the capability of reducing input/output operations by defining main-storage work areas to contain the highest level master index and the records being processed.

For additional information about the characteristics of BISAM processing, refer to *OS Data Management Services Guide.*

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

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

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

**F** — Specifies that each buffer is on a fullword boundary that is not also a doubleword boundary.

**D** — Specifies that each buffer is on a doubleword boundary.

If the BUILD macro instruction is used to construct the buffer pool or the problem program controls all buffering, the problem program must provide a main-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.

**BUFCB**=*relexp*

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

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

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

**BUFL**=*absexp* (maximum value is 32,760)

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

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

If the problem program controls all buffering, the **BUFL** operand is not required. However, an ISAM data set requires additional buffer space for system use. For a description of the buffer length required for various ISAM operations, refer to *OS Data Management Services Guide.*

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

**BUFNO**=*absexp* (maximum value is 255)

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

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

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

**DDNAME=***symbol*

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

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

**DSORG=IS**

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

**Source:** The **DSORG** operand must be coded in the DCB macro instruction as well as in the DD statement unless it is for a data set passed from a previous job step. In this case, **DSORG** may be omitted from the DD DCB field.

**EXLST=***relexp*

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

Refer to Appendix D of this publication for the format and requirements for exit list processing. For additional information about exit list processing, refer to *OS Data Management Services Guide.*

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

**HIARCHY=** {**0**} / {**1**}

The **HIARCHY** operand specifies the main-storage hierarchy in which the buffer pool is constructed. The following describes the characters that can be specified.

**0** — Specifies that the buffer pool is constructed in processor storage.

**1** — Specifies that the buffer pool is constructed in IBM 2361 Core Storage.

The **HIARCHY** operand can also be specified in a GETPOOL macro instruction. If the **HIARCHY** operand is omitted from all sources, the buffer pool is constructed in processor storage.

The buffer pool is constructed within the user region or partition within the indicated hierarchy; if space is not available within the indicated hierarchy, the task is abnormally terminated. The **HIARCHY** operand is ignored in systems that do not have hierarchy support. The **HIARCHY** operand must not be specified for MVT systems with Model 65 multiprocessing.

**Source:** The **HIARCHY** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or in the **HIARCHY** operand of a GETPOOL macro instruction.

```
MACRF= {(R  [S]  [C])                    }
       {(W  {U}  [C])                    }
            {A}
            {UA}
       {(R  [S]  [C],W    {U}  [C]) }
            [U]           {A}
            [US]          {UA}
```

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

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

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

**R** — Specifies the READ macro instructions are used. When **R** is coded, the routines that allow the FREEDBUF macro instruction to be used are also included.

**S** — Specifies that dynamic buffering is requested in READ macro instructions.

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

**W** — Specifies that WRITE macro instructions are used.

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

**MSHI**=*relexp*

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

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

**MSWA**=*relexp*

The **MSWA** operand specifies the address of the main-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, refer to *OS Data Management Services Guide.*

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

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

**NCP**=*absexp* (maximum value is 99)

The **NCP** operand specifies the maximum number of READ/WRITE macro instructions that are issued before the first CHECK (or WAIT) macro instruction is issued to test for completion of the I/O operation. The maximum number that can be specified may be less than 99 depending on the limit established when the operating system is generated. If the **NCP** operand is omitted, one 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 an OPEN macro instruction is issued to open the data set.

**SMSI**=*absexp* (maximum value is 65,535)

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

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

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

**SMSW**=*absexp* (maximum value is 65,535)

The **SMSW** operand specifies the length, in bytes, of a work area in main storage that is used by BISAM. This operand is optional, but the system acquires a minimum-size work area if the operand is omitted. The **SMSW** operand is coded only when the **MSWA** operand is also coded. If the **SMSW** operand is coded but the size specified is less than the minimum required, the task is abnormally terminated. *OS Data Management Services 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 (eight 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 (eight bytes each) and data fields contained on one direct-access device track plus additional space for one logical record (**LRECL** value).

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

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

**SYNAD**=*relexp*

The **SYNAD** operand specifies the address of the error anlaysis routine given control when an uncorrectable input/output error occurs. The contents of the registers when the error analysis routine is given control are described in Appendix A of this publication.

The error anlaysis 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 which 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.

For ISAM data sets, if the error analysis routine receives control from the Close routine, bit 3 of the IOBFLAG1 field in the input/output block is set to one. In this case, the error analysis routine must not issue a CLOSE macro instruction. To complete Close processing, the error analysis routine must return control to the Close routine with a branch to the address in register 14.

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

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

# DCB—Construct a Data Control Block (BPAM)

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

Before a DCB macro instruction for a BPAM data set is coded, the following characteristics of partitioned data sets should be considered:

- The entire partitioned data set must reside on one direct-access volume, but several such data sets, on the same or different volumes, can be concatenated for input.
- When a partitioned data set is being created, the first (or only) DD statement for the data set must contain a **SPACE** parameter defining the size of the entire data set and its directory. From this information, the system allocates space for the data set and pre-formats the data set directory. As subsequent data set members are added, they are added in the space originally allocated.
- A single member of a partitioned data set can be added or retrieved using BSAM or QSAM without using the BLDL, FIND, or STOW macro instructions. In this case, the data set member is being processed as a sequential data set (**DSORG=PS**). Processing a member in this manner does not provide the full capability of the basic partitioned access method. For more information about processing a member using BSAM or QSAM, refer to *OS Data Management Services Guide.*
- A single member or multiple members can be added, retrieved, or updated using BPAM (many of the routines used by BPAM are actually BSAM routines).
- Buffers for a BPAM data set can be acquired automatically, but buffer control must be provided by the problem program. The problem program must issue a READ macro instruction that provides a buffer address to fill an input buffer, and it must place the data in an output buffer before issuing a WRITE macro instruction to write a data block.
- Although a BPAM data set can contain blocked records, the problem program must perform all blocking and deblocking of records. BPAM provides only the capability to read or write a data block, but the data block can contain multiple logical records assembled by the problem program.
- The STOW macro instruction can be used to add, delete, change, or replace a member name or alias in the directory.
- Multiple members of the data set can be processed by building a list of member locations (with a BLDL macro instruction) and using the FIND macro instruction (in conjunction with the list) to locate the beginning of each member.
- The problem program must synchronize I/O operations by issuing a CHECK macro instruction for each READ or WRITE macro instruction issued.

These characteristics of partitioned data sets and the basic partitioned access method are described in more detail in *OS Data Management Services Guide.*

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

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

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

**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 a main-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 is 32,760)

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

The actual blocksize that can be specified depends on the record format and the type of direct-access device being used. If the track-overflow feature is used, the blocksize can be up to the maximum. If the track-overflow feature is not used, the maximum blocksize 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 of this publication. For additional information about device capacity and space allocation, refer to the *OS Data Management Services Guide.*

For variable-length records, the value specified in the **BLKSIZE** operand must include the maximum logical record length (up to 32,756 bytes) plus four bytes for the block descriptor word (BDW).

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

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

**BUFCB**=*relexp*

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

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

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

**BUFL**=*absexp* (maximum value is 32,760)

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

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

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

**BUFNO**=*absexp* (maximum value is 255)

The **BUFNO** operand specifies the number of buffers to be constructed by a BUILD macro instruction, or it specifies the number of buffers to be acquired automatically by the system.

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

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

**DDNAME**=*symbol*

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

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

**DSORG=** {**PO**}
{**POU**}

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

**PO** — Specifies a partitioned data set organization.

**POU** — Specifies a partitioned data set organization and that the data set contains location-dependent information.

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

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

**EODAD**=*relexp*

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

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

**EXLST**=*relexp*

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

Refer to Appendix D of this publication for the format and requirements of the exit list processing. For additional information about exit list processing, refer to *OS Data Management Services Guide.*

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

**HIARCHY=** {**0**}
{**1**}

The **HIARCHY** operand specifies the main-storage hierarchy in which the buffer pool is constructed. The following describes the characters that can be specified.

**0** — Specifies that the buffer pool is constructed in processor storage.

**1** — Specifies that the buffer pool is constructed in IBM 2361 Core Storage.

The **HIARCHY** operand can also be specified in a GETPOOL macro instruction. If the **HIARCHY** operand is omitted from all sources, the system constructs the buffer pool in processor storage.

The buffer pool is constructed in the user region or partition within the indicated hierarchy; if space is not available within the indicated hierarchy, the task is abnormally terminated. The **HIARCHY** operand is ignored in systems that do not have hierarchy support. The **HIARCHY** operand must not be specified for MVT systems with Model 65 multiprocessing.

**Source:** The **HIARCHY** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or in the **HIARCHY** operand of a GETPOOL macro instruction.

**KEYLEN**=*absexp* (maximum value is 255)

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

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

**LRECL**=*absexp* (maximum value is 32,760)

The **LRECL** operand specifies the length, in bytes, of each fixed-length logical record in the data set; It is required only for fixed-length records. The value specified in the **LRECL** operand cannot exceed the value specified in the **BLKSIZE** operand.

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

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

**MACRF=** **{(R)}**
**{(W)}**
**{(R,W)}**

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

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

**W** — Specifies that WRITE macro instructions are used. This operand automatically provides the capability to use the NOTE macro instruction with the data set.

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

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

**NCP=***absexp* (maximum value is 99)

The **NCP** operand specifies the maximum number of READ and WRITE macro instructions that will be issued before the first CHECK macro instruction is issued. The maximum number may be less than 99 depending on the limit established when the operating system is generated. If chained scheduling is specified, **NCP** must be specified as more than one. If the **NCP** operand is omitted, one 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 an OPEN macro instruction is issued to open the data set.

**OPTCD=** {**C**}
{**W**}
{**WC**}

The **OPTCD** operand specifies the optional services performed by the system. The following describes the characters that can be specified; they can be specified in any order and no commas are required between characters.

**C** — Specifies that chained scheduling is used.

**W** — Specifies that the system performs a validity check for each record written. If the device is a 2321, the system performs a validity check for each write operation whether it is requested or not.

**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**] [**A**] }
[**T**] [**M**]
[**BT**]

{**F** [**B**] [**A**] }
[**T**] [**M**]
[**BT**]

The **RECFM** operand specifies the record format and characteristics of the data set being created or processed. All the record formats shown above can be specified, but in those formats that show blocked records, the problem program must perform the blocking and deblocking of logical records; BPAM recognizes only data blocks. When **RECFM** is not specified, undefined format is assumed. The following describes the characters that can be specified.

**A** — Specifies that the records in the data set contain American National Standards Institute (ANSI) control characters. Refer to Appendix E for a description of control characters.

**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. Refer to Appendix E for a description of control characters.

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

**U** — Specifies that the data set contains undefined-length records.

**V** — Specifies that the data set contains variable-length records.

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

**SYNAD**=*relexp*

The **SYNAD** operand specifies the address of the error analysis (SYNAD) routine to be given control when an uncorrectable input/output error occurs. The contents of the registers when the error analysis routine is given control are described in Appendix A.

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

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

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

# DCB—Construct a Data Control Block (BSAM)

The data control block for the basic sequential access method (BSAM) is constructed during assembly of the problem program. The DCB macro instruction must not be coded within the first 16 bytes of a control section (CSECT). The **DSORG** and **MACRF** operands must be coded in the DCB macro instruction, but the other DCB operands can be supplied, to the data control block, from other sources. Each DCB operand description contains a heading, "Source." The information under this heading describes the sources from which an operand can be supplied.

Before a DCB macro instruction for creating or processing a BSAM data set is coded, the following characteristics of BSAM data sets should be considered:

- Although several record formats with blocked records can be specified for BSAM, the problem program must perform all blocking and deblocking of records. BSAM provides only the capability to read or write a data block, but the block can contain one or more logical records assembled by the problem program.
- Buffers for a BSAM data set can be acquired automatically, but buffer control must be provided by the problem program. The problem program must issue a READ macro instruction that provides a buffer address to fill an input buffer, and it must place the data in an output buffer before issuing the WRITE macro instruction to write a data block.
- The problem program must synchronize I/O operations by issuing a CHECK macro instruction for each READ and WRITE macro instruction issued.
- BSAM provides capability for nonsequential processing by using the NOTE and POINT macro instructions.
- Keys for direct-access device records can be read or written using BSAM.
- Specifying the **DEVD** operand in the DCB macro instruction can make the program device dependent.

These characteristics of basic sequential access method data sets are described in more detail in *OS Data Management Services Guide.*

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

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

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

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

The following describes the characters that can be specified.

**F** — Specifies that each buffer is on a fullword boundary that is not also a doubleword boundary.

**D** — Specifies that each buffer is on a doubleword boundary.

If the BUILD macro instruction is used to construct the buffer pool or if the problem program controls all buffering, the problem program must provide a main-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 block exit routine. If both the **BFALN** and **BFTEK** operands are specified, they must be supplied by the same source.

**BFTEK=R**

The **BFTEK=R** operand specifies that BSAM is used to read unblocked variable-length spanned records with keys from a BDAM data set. Each read operation reads one segment of the record and places it in the area designated in the READ macro instruction. The first segment enters at the beginning of the area, but all subsequent segments are offset by the length of the key (only the first segment has a key). The problem program must provide an area in which to 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 is 32,760)

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

The actual value that can be specified in the **BLKSIZE** operand depends on the device type and the record format being used. Device capacity is shown in Appendix C of this publication. For additional information about device capacity, refer to *OS Data Management Services Guide.* For direct-access devices when the track-overflow feature 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 for a SYSOUT data set, the value specified in the **BLKSIZE** operand must be an integral multiple of the value specified for the logical record length (**LRECL**); otherwise the system will adjust the blocksize downward to the nearest multiple.

If variable-length records are used, the value specified in the **BLKSIZE** operand must include the maximum logical record length (up to 32,756 bytes) plus the

four bytes required for the block descriptor word (BDW). For format-D variable-length records (ASCII data sets), the minimum value for **BLKSIZE** is 18 and the maximum value is 2,048.

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

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

**BUFCB**=*relexp*

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

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

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

**BUFL**=*absexp* (maximum value is 32,760)

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

If the data set contains 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 controls all buffering or if the buffer pool is 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** subparameter of a DD statement, or by the problem program before completion of the data control block exit routine.

**BUFNO=***absexp* (maximum value is 255)

The **BUFNO** operand specifies the number of buffers constructed by a BUILD macro instruction or the number of buffers to be acquired automatically by the system.

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

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

**BUFOFF=** { *absexp* }
{**L**}

The **BUFOFF** operand specifies the length, in bytes, of the block prefix used with an ASCII tape data set. When BSAM is used to read an 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 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 ASCII code; any character that cannot be translated is replaced with a substitute character. 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 (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.

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.

**DDNAME=***symbol*

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

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

```
DEVD=  {DA  [,KEYLEN=absexp]
       {TA  [,DEN=   {0}][,TRTCH=  {C}]
                     {1}          {E}
                     {2}          {T}
                     {3}          {ET}

       {PT  [,CODE=  {A}]
                     {B}
                     {C}
                     {F}
                     {I}
                     {N}
                     {T}

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

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

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

The **DEVD** operand specifies the device type on which the data set can or does reside. The device types above are shown with the optional operand(s) that can be coded when a particular device is used. The devices are listed in order of device-independence. For example, if **DEVD=DA** is coded in a DCB macro instruction (or the **DEVD** operand is omitted, which causes a default to **DA**), the data control block constructed during assembly could later be used for any of the other devices, but if **DEVD=RD** is coded, the data control block can be used only

with a card reader or card reader punch. Unless you are certain that device interchangeability is not required, you should either code **DEVD=DA** or omit the operand and allow it to default to **DA**.

If system input is directed to an intermediate storage device, the **DEVD** operand is omitted, and the job control language for the problem program designates the system input device to be used. Likewise, if system output is directed to an intermediate storage device, the **DEVD** operand is omitted, and the job control language for the problem program designates the system output device to be used.

The following describes the device type and the optional operands that can be specified for each device type:

**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**).

**KEYLEN=***absexp*

The **KEYLEN** operand can be specified only for data sets that reside on direct-access devices. Since 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.

**TA** — 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=** {**0**}
{**1**}
{**2**}
{**3**}

The **DEN** operand specifies the recording density in the number of bits-per-inch per track as shown in the following chart.

| DEN | 7-Track Tape | 9-Track Tape | 9-Track Tape (Phase Encoded) | 9-Track Tape (Dual Density) |
|---|---|---|---|---|
| 0 | 200 | — | — | — |
| 1 | 556 | — | — | — |
| 2 | 800 | 800 | — | 800 (NRZI) |
| 3 | — | — | 1600 | 1600 (PE) |

NRZI is for non-return-to-zero-inverse mode
PE is for phase encoded mode

Specifying DEN=0 for a 7-track, 3420 tape will result in a 556 bits-per-inch recording density, but corresponding messages and tape labels will indicate a 200 bits-per-inch recording density.

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

**TRTCH=** {**C**}
{**E**}
{**ET**}
{**T**}

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

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

**E** — Specifies even parity with no translation or conversion.

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

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

**PT** — Specifies that the data control block is used for a paper tape device (or any of the other devices following **PT**). If **PT** is coded, the following optional operand can be coded.

**CODE=** {A}
{B}
{C}
{F}
{I}
{N}
{T}

The **CODE** operand specifies the code in which the data was punched. The system converts these codes to EBCDIC code. If the **CODE** operand is not supplied by any source, **CODE=I** is assumed. The following describes the characters that can be specified.

**A** — Specifies 8-track tape in ASCII code.

**B** — Specifies Burroughs 7-track tape.

**C** — Specifies National Cash Register 8-track tape.

**F** — Specifies Friden 8-track tape.

**I** — Specifies IBM BCD perforated tape and transmission code with 8 tracks.

**N** — Specifies that no conversion is required.

**T** — Specifies Teletype[1] 5-track tape.

**PR** — 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}

[1] Trademark of Teletype Corporation

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

**0** — Specifies that spacing is suppressed (no space).

**1** — Specifies single-spacing.

**2** — Specifies double-spacing (one blank line between printed lines).

**3** — Specifies triple-spacing (two blank lines between printed lines).

**PC** — 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]** **[R]**
**[E]**

The **MODE** operand specifies the mode of operation for the card punch. The following describes the characters that can be specified. If the **MODE** operand is omitted, **E** is assumed.

**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 main 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 six 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).

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

**STACK=** **{1}**
**{2}**

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

**1** — Specifies stacker number 1.

**2** — Specifies stacker number 2.

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

The **FUNC** operand defines the type of 3525 card punch data sets that are used. **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.

**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] [O]**
**[E] [R]**

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

**C** — Specifies that the cards to be read are in card image mode. In card image mode, the 12 rows in each card column are read into two consecutive bytes of main 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 only).

**R** — Specifies that the program runs in read-column-eliminate mode (3505 card reader and 3525 card reader only).

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

**STACK= {1}**
**{2}**

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

**1** — Specifies stacker number 1.

**2** — Specifies stacker number 2.

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

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

**D** — Specifies that the data protection option is to be used. The data protection option prevents punching information into

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

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

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

**R** — Specifies that the data set is for reading cards.

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

**W** — Specifies that the data set is for printing. See the description of the character **X** for associated punch and print data sets.

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

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

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

**DSORG= {PS}**
**{PSU}**

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

**PS** — Specifies a physical sequential data set.

**PSU** — Specifies a physical sequential data set that contains location-dependent information that would make it unmovable.

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

**EODAD**=*relexp*

The **EODAD** operand specifies the address of the routine given control when the end of an input data set is reached. Control is given to this routine when a READ macro instruction is issued and there are no additional input 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. The end of data routine is entered when the CHECK macro instruction determines that the READ macro instruction reached the end of the data. If the end of the data set is reached but no EODAD address has been supplied, the task is abnormally terminated. See *OS Data Management Services Guide* for additional information on the EODAD routine.

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

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

**EXLST**=*relexp*

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

Refer to Appendix D of this publication for the format and requirements of exit list processing. For additional information about exit list processing, refer to *OS Data Management Services Guide.*

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

**HIARCHY=** {**0**}
{**1**}

The **HIARCHY** operand specifies the main-storage hierarchy in which the buffer pool is constructed. The following describes the characters that can be specified.

**0** — Specifies that the buffer pool is constructed in processor storage.

**1** — Specifies that the buffer pool is constructed in IBM 2361 Core Storage.

The **HIARCHY** operand can also be specified in the **HIARCHY** operand of a GETPOOL macro instruction. If the **HIARCHY** operand is omitted from all sources, the buffer pool is constructed in processor storage.

The buffer pool is constructed in the user region or partition within the indicated hierarchy; if space is not available in the indicated hierarchy, the task is abnormally terminated. The **HIARCHY** operand is ignored in systems that do not have hierarchy support. The **HIARCHY** operand must not be specified for MVT systems with Model 65 multiprocessing.

**Source:** The **HIARCHY** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or in the **HIARCHY** operand of a GETPOOL macro instruction.

**KEYLEN**=*absexp* (maximum value is 255)

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

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

**LRECL=** { *absexp* }
{**X**}

The **LRECL** operand specifies the length, in bytes, for fixed-length records, or it specifies the maximum length, in bytes, for variable-length records. **LRECL=X** is used for variable-length spanned records that exceed 32,756 bytes. Except when variable-length spanned records are used, the value specified for the **LRECL** operand cannot exceed the value specified for the **BLKSIZE** operand.

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

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

For variable-length records, the value specified in **LRECL** must include the maximum data length (up to 32,752 bytes) plus 4 bytes for the RDW.

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

When BSAM is used to create a BDAM data set with variable-length spanned records, the **LRECL** value should be the maximum data length (up to 32,752) plus four bytes for the record descriptor word (RDW), or if the logical record length is greater than 32,756 bytes, **LRECL=X** is specified.

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

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

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

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

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

**P** — Specifies that POINT macro instructions are used with the data set being created or processed. Specifying **P** in the **MACRF** operand also automatically provides the capability of using NOTE macro instructions with the data set.

**R** — Specifies that READ macro instructions are used.

**W** — Specifies that WRITE macro instructions are used.

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

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

**NCP=** *absexp* (maximum value is 99)

The **NCP** operand specifies the maximum number of READ/WRITE macro instructions that will be issued before the first CHECK macro instruction is issued to test for completion of the I/O operation. The maximum number may be less than 99 depending on the limit established when the operating system is generated. If chained scheduling is specified (**OPTCD=C**), **NCP** must be specified as more than one. If the **NCP** operand is omitted, one 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 an OPEN macro instruction is issued to open the data set.

**OPTCD=** {**B** } [**T**]
{**C** }
{**H** }
{**Q** }
{**U** }
{**UC** }
{**W** }
{**WC** }
{**Z** }
{**ZC** }

The **OPTCD** operand specifies the optional services that are used with the BSAM data set. Two of the optional services (**OPTCD=B** and **OPTCD=H**) cannot be specified in the DCB macro instruction. They are requested in the DCB subparameter of a DD statement. Since all optional services requests must be supplied by the same source, the **OPTCD** operand must be omitted from the DCB macro instruction if either of these options is requested in a DD statement. The following describes the characters that can be specified; these characters can be specified in any order (in one of the combinations shown above), and no commas are required between characters.

**C** — Requests that chained scheduling be used. **OPTCD=C** cannot be specified if **BFTEK=R** is specified for the same data control block. Also, chained scheduling cannot be specified for associated data sets or printing on a 3525.

**Q** — Requests that ASCII tape records in an input data set be converted to EBCDIC code after the input record has been read, or it requests that an output record in EBCDIC code be converted to ASCII code before the record is written.

**T** — Requests the user totaling facility. If this facility is requested, the **EXLST** operand should specify the address of an exit list to be used.

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

**W** — Specifies that the system performs a validity check on each record written on a direct-access device. If the device is a 2321 data cell, the system performs a validity check for each write operation whether it is requested or not.

**Z** — For magnetic tape, input only, the **Z** option requests 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 selected when the operating system is generated. **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 requests the system to use the search direct option to accelerate the input operations for a data set. **OPTCD=Z** cannot be specified when **RECFM=UT, FS, FBT, VS,** or **VBS.**

**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 (EOF) 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 SL or AL tape, the EOV routine will treat EOF labels as EOV labels until the volume serial list is exhausted. When no more volumes are available, control is passed to the user's end-of-data routine. This option allows SL or AL tapes to be read out of order 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 DSCB for direct-access devices, 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**] [**A**]}
[**S**] [**M**]
[**T**]
[**BS**]
[**BT**]

{**D** [**B**] [**A**]}

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

The **RECFM** operand specifies the record format and characteristics of the data set being created or processed. All the record formats shown above can be specified, but in those record formats that specify blocked records, the problem program must perform the blocking and deblocking of logical records; BSAM recognizes only data blocks. When RECFM is not specified, undefined format is assumed. The following describes the characters that can be specified:

**A** — Specifies that the records in the data set contain American National Standards Institute (ANSI) control characters. Refer to Appendix E for a description of control characters.

**B** — Specifies that the data set contains blocked records.

**D** — Specifies that the data set contains variable-length ASCII tape records. See **OPTCD=Q** and the **BUFOFF** operand for a description of how to specify 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. Refer to Appendix E for a description of control characters. **RECFM=M** cannot be used with ASCII data sets.

**S** — For fixed-length records, **S** specifies that the records are written as standard blocks; the data set does not contain any truncated blocks or unfilled tracks, with the exception of the last block or track in the data set.

For variable-length records, **S** specifies that a record can span more than one block. Spanned records can be read (reading a BDAM data set) or written (creating a BDAM data set) using BSAM.

**T** — Specifies that the track-overflow feature is used with the data set. The track-overflow feature allows a record to be written partially on one track of a direct-access device and the remainder of the record written on the following track (if required). Chained scheduling cannot be used if the track-overflow feature is used.

**U** — Specifies that the data set contains undefined-length records.

**V** — Specifies that the data set contains variable-length records.

**Note: RECFM=V** cannot be specified for a card reader data set or an ASCII tape data set.

**Note:** The record format **RECFM=VBS** does not provide the spanned record function; if this format is used, the problem program must block and segment the records.

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

**SYNAD=***relexp*

The **SYNAD** operand specifies the address of the error analysis (SYNAD) routine to be given control when an uncorrectable input/output error occurs. The contents of the registers when the error analysis routine is given control are described in Appendix A of this publication.

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 which uses the address in register 14 to return control to the system. If control is returned to the system, the system returns control to the problem program and proceeds as though no error had been encountered.

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

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

# DCB—Construct a Data Control Block (QISAM)

The data control block for a queued indexed sequential access method (QISAM) data set is constructed during assembly of the problem program. The DCB macro instruction must not be coded within the first 16 bytes of a control section (CSECT). The **DSORG** and **MACRF** operands must be coded in the DCB macro instruction, but the other DCB operands can be supplied from other sources. Each QISAM DCB operand description contains a heading, "Source." The information under this heading describes the sources which can supply the operand to the data control block.

Before a DCB macro instruction for a QISAM data set is coded, the following characteristics of QISAM should be considered:

- The characteristics of a QISAM data set are established when the data set is created; these characteristics cannot be changed without reorganizing the data set. The following DCB operands establish the characteristics of the data set and can be coded only when creating the data set: **BLKSIZE, CYLOFL, KEYLEN, LRECL, NTM, OPTCD, RECFM,** and **RKP.**
- The data set can contain the following record formats: Unblocked fixed-length records (**F**), blocked fixed-length records (**FB**), unblocked variable-length records (**V**), or blocked variable-length records (**VB**).
- QISAM can create an indexed sequential data set (QISAM, load mode), add additional data records at the end of the existing data set (QISAM, resume load mode), update a record in place, or retrieve records sequentially (QISAM, scan mode).
- The track-overflow feature cannot be used to create an ISAM data set.
- When an indexed sequential data set is being created, space for the prime area of the data set, the overflow area of the data set, and the cylinder/master index(es) for the data set can be allocated on the same or separate volumes. For information about space allocation, refer to *OS Job Control Language Reference* manual.
- The system automatically creates one track index for each cylinder in the data set and one cylinder index for the entire data set. The DCB **NTM** and **OPTCD** operands can be specified to indicate that the data set requires a master index(es); the system creates and maintains up to three levels of master indexes. *OS Data Management Services Guide* contains additional information about indexes for indexed sequential data sets.
- A record deletion option can be specified (**OPTCD=L**) when the ISAM data set is created. This option allows a record to be flagged for deletion by placing a hexadecimal value of 'FF' in the first data byte of the record (first byte of a fixed-length record or fifth byte of a variable-length record). Records marked for deletion are ignored during sequential retrieval by QISAM.
- Reorganization statistics can be obtained by specifying **OPTCD=R** when the ISAM data set is created. These statistics can be used by the problem program to determine the status of the overflow areas allocated to the data set. Reorganization of ISAM data sets is described in *OS Data Management Services Guide.*

- When an ISAM data set is created, the records must be written with the keys in ascending order.

These characteristics of queued indexed sequential access method data sets are described in more detail in *OS Data Management Services Guide.*

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

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

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

**F** — Specifies that each buffer is on a fullword boundary that is not also a doubleword boundary.

**D** — Specifies that each buffer is on a doubleword boundary.

If the **BUILD** macro instruction is used to construct the buffer pool, the problem program must provide a main-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 is device-dependent)

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

Track capacity of the direct-access device being used must be considered when the **BLKSIZE** for an ISAM data set is specified. For fixed-length records, the sum of the key length, data length, and device overhead plus 10 bytes (for ISAM use) must not exceed the capacity of a single track on the direct-access device being used. For variable-length records the sum of the key length, block-descriptor word length, record-descriptor word length, data length, and device overhead plus 10 bytes (for ISAM use) must not exceed the capacity of a single track on the direct-access device being used. Device capacity and device overhead are described in Appendix C of this publication. For additional information about device capacity and space allocation, refer to *OS Data Management Services Guide.*

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

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

**BUFCB=** *relexp*

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

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

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

**BUFL=** *absexp* (maximum value is 32,760)

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

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

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

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

**BUFNO=** *absexp* (maximum value is 255)

The **BUFNO** operand specifies the number of buffers to be constructed by a BUILD macro instruction, or it specifies the number of buffers to be acquired automatically by the system. If the **BUFNO** operand is omitted, the system automatically acquires two buffers.

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

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

**CYLOFL=** *absexp* (maximum value is 99)

The **CYLOFL** operand specifies the number of tracks on each cylinder that is reserved as an overflow area. The overflow area is used to contain records that are forced off prime area tracks when additional records are added to the prime area track in ascending key sequence. ISAM maintains pointers to records in the overflow area so that the entire data set is logically in ascending key sequence. Tracks in the cylinder overflow area are used by the system only if **OPTCD=Y** is specified. For a more complete description of the cylinder overflow area, refer to the space allocation section of *OS Data Management Services 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 must be omitted, allowing the data set label to supply the operand.

**DDNAME=***symbol*

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

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

**DSORG=** {**IS**}
{**ISU**}

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

**IS** — Specifies an indexed sequential data set organization.

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

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

**EODAD=***relexp*

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

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

**EXLST=***relexp*

The **EXLST** operand specifies the address of the problem program exit list. The **EXLST** operand is required only 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.

Refer to Appendix D of this publication for the format and requirements for exit list processing. For additional information about exit list processing, refer to *OS Data Management Services Guide.*

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

**HIARCHY=** {**0**} / {**1**}

The **HIARCHY** operand specifies the main-storage hierarchy in which the buffer pool is constructed. The following describes the characters that can be specified.

**0** — Specifies that the buffer pool is constructed in processor storage.

**1** — Specifies that the buffer pool is constructed in IBM 2361 Core Storage.

The **HIARCHY** operand can also be specified in a GETPOOL macro instruction. If the **HIARCHY** operand is omitted from all sources, the buffer pool is constructed in processor storage.

The buffer pool is constructed within the user region or partition within the indicated hierarchy; if space is not available within the indicated hierarchy, the task is abnormally terminated. The **HIARCHY** operand is ignored in systems that do not have hierarchy support. The **HIARCHY** operand must not be specified for MVT systems with Model 65 multiprocessing.

**Source:** The **HIARCHY** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or in the **HIARCHY** operand of a GETPOOL macro instruction.

**KEYLEN=***absexp* (maximum value is 255)

The **KEYLEN** operand specifies the length, in bytes, of the key associated with each record in an indexed sequential data set. When block 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 which 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 label to supply the key length value.

**LRECL=***absexp* (maximum value is device-dependent)

The **LRECL** operand specifies the length, in bytes, for fixed-length records, or it specifies the maximum length, in bytes, for variable-length records. The value specified in the **LRECL** operand cannot exceed the value specified in the **BLKSIZE** operand. When fixed unblocked records are used and the relative key position (as specified in the **RKP** operand) is zero, the value specified in the

**LRECL** operand should include only the data length (the key is not written as part of the fixed, unblocked record when **RKP=0**).

The track capacity of the direct-access device being used must be considered if maximum length logical records are being used. For fixed-length records, the sum of the key length, data length, and device overhead plus 10 bytes (for ISAM use) must not exceed the capacity of a single track on the direct-access device being used. For variable-length records, the sum of the key length, data length, device overhead, block-descriptor-word length, and record-descriptor-word length plus 10 bytes (for ISAM use) must not exceed the capacity of a single track on the direct-access device being used. Device capacity and device overhead are described in Appendix C of this publication. For additional information about device capacity and space allocation, refer to *OS Data Management Services 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]][,PU]) }
                 [I]        }
```

The **MACRF** operand specifies the type of macro instructions, the transmittal mode, and type of search to be used with the data set being processed. The operand can be coded in any of the combinations shown above; the following describes the characters that can be coded.

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

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

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

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

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

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

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

**K** — Specifies that a key or key class is used to search for a record (or the first record) to be read.

**PU** — Specifies that PUTX macro instructions are 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)

The **NTM** operand specifies the number of tracks to be contained 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 up to three levels of master indexes. The **NTM** operand is ignored unless the master index option (**OPTCD=M**) is selected.

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

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

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

**I** — Specifies that the system uses the independent overflow areas to contain overflow records. Allocated independent overflow areas are not used unless **OPTCD=I** is specified.

**L** — Specifies that the data set will contain records flagged for deletion. A record is flagged for deletion by placing a hexadecimal value of 'FF' in the first data byte. Records flagged for deletion remain in the data set until the space is required for another record to be added to the track. Records flagged for deletion 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 zero (**RKP=0**), or it cannot be specified for variable-length records if the relative key position is four (**RKP=4**).

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

**M** — Specifies that the system creates and maintains a master index(es) according to the number of tracks specified in the **NTM** operand.

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

**U** — Specifies that the system accumulates track index entries in main storage and writes them as a group for each track of the track index. **OPTCD=U** can be specified only for fixed-length records. The entries are written in fixed-length unblocked format.

**W** — Specifies that the system performs a validity check on each record written. If the device is a 2321 data cell, the system performs a validity check for each write operation whether it is requested or not.

**Y** — Specifies that the system uses 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 must be omitted.

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

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

**B** — Specifies that the data set contains blocked records.

**F** — Specifies that the data set contains fixed-length records.

**V** — Specifies that the data set contains variable-length records.

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

**RKP=** *absexp*

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

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

If variable-length records (blocked or unblocked) are used, **RKP** must be four or greater if the delete option is not specified; if the delete option is specified, **RKP** must be specified as five or greater. The four 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 must be omitted.

**SYNAD=** *relexp*

The **SYNAD** operand specifies the address of the error analysis routine given control when an uncorrectable input/output error occurs. The contents of the registers when the error analysis routine is given control are described in Appendix A of this publication.

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 which 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 ISAM data sets, if the error analysis routine receives control from the Close routine, bit 3 of the IOBFLAG1 field in the input/output block is set to one. In this case, the error analysis routine must not issue a CLOSE macro instruction. To complete close processing, the error analysis routine must return control to the Close routine with a branch to the address in register 14.

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

# DCB—Construct a Data Control Block (QSAM)

The data control block for the queued sequential access method (QSAM) is constructed during assembly of the problem program. The DCB macro instruction must not be coded within the first 16 bytes of a control section (CSECT). The **DSORG** and **MACRF** operands must be coded in the DCB macro instruction, but the other DCB operands can be supplied, to the data control block, from other sources. Each DCB operand description contains a heading, "Source." The information under this heading describes the sources from which the operand can be supplied.

Before a DCB macro instruction for creating or processing a QSAM data set is coded, the following characteristics of QSAM data sets should be considered.

- All record formats can be processed.
- Automatic blocking and deblocking of records is provided.
- Automatic buffer control is provided; this function fills input buffers when they are empty and writes output buffers when they are full.
- A logical record interface is provided; a GET macro instruction retrieves the next sequential logical record from the input buffer, and a PUT macro instruction places the next sequential logical record in the output buffer.
- I/O operations are synchronized automatically.
- Four transmittal modes (move, locate, data, and substitute) are provided. These transmittal modes provide flexibility in buffer management and data movement between buffers.
- Keys for direct-access device records cannot be read or written using QSAM.
- Specifying the **DEVD** operand in the DCB macro instruction can cause the program to be device-dependent.

These characteristics of queued sequential access method data sets are described in more detail in *OS Data Management Services Guide.*

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

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

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

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

The following describes the characters that can be specified.

**F** — Specifies that each buffer is on a fullword boundary that is not also a doubleword boundary.

**D** — Specifies that each buffer is on a doubleword boundary.

When exchange buffering (**BFTEK=E**) is specified and the records are in blocked fixed-length format, each buffer segment is aligned as specified in the **BFALN** operand.

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**}
{**E**}
{**A**}

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

**S** — Specifies that simple buffering is used.

**E** — Specifies that exchange buffering is used. Exchange buffering can be used only with record formats (**RECFM** operand) **F, FB, FBS,** or **FS**; the track-overflow feature cannot be used with exchange buffering. If exchange buffering is used with ASCII tape records, the **BUFOFF** operand must be zero (no block prefix).

**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. When a logical record interface is requested, the system uses the simple buffering technique.

To use the simple or exchange buffering technique efficiently, the user should be familiar with the four transmittal modes for QSAM and the buffering techniques as described in *OS Data Management Services 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 32,760)

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

The actual value that can be specified in the **BLKSIZE** operand depends on the device type and record format being used. Device capacity is shown in Appendix C of this publication. For additional information about device capacity, refer to *OS Data Management Services Guide*. For direct-access devices when the track-overflow feature is used or variable-length spanned records are being processed, the **BLKSIZE** operand can be up to the maximum value. For other record formats used with direct-access devices, the value specified in the **BLKSIZE** operand cannot exceed the capacity of a single track.

Since QSAM provides a logical record interface, the device capacities shown in Appendix C 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 for a SYSOUT data set, the value specified in the **BLKSIZE** operand must be an integral multiple of the value specified in the **LRECL** operand; otherwise, the system will adjust the blocksize downward to the nearest multiple. 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 32,756 bytes) plus four bytes required for the block descriptor word (BDW). For format-D variable-length records, the minimum blocksize is 18 bytes and the maximum is 2048 bytes.

If 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 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.

**BUFCB**=*relexp*

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

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

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

**BUFL=** *absexp* (maximum value is 32,760)

The **BUFL** operand specifies the length, in bytes, of each buffer in the buffer pool when the buffer pool is acquired automatically. The system acquires buffers with a length equal to the value specified in the **BLKSIZE** operand if the **BUFL** operand is omitted; 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 ASCII tape records with a block prefix, the value specified in the **BUFL** operand must also include the length of the block prefix.

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

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

**BUFNO=** *absexp* (maximum value is 255)

The **BUFNO** operand specifies the number of buffers in the buffer pool constructed by a BUILD or BUILDRCD macro instruction, or it specifies the number of buffers to be acquired automatically. If the **BUFNO** operand is omitted and the buffers are acquired automatically, the system acquires three buffers if the device is a 2540 card read punch or two buffers for any other device type.

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

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

**BUFOFF=** { *absexp* }
{**L**}

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

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

**L** — Specifies that the block prefix is 4 bytes long and contains the block length. **BUFOFF=L** is used when format-D records (ASCII) are processed. QSAM uses the four bytes as a block-descriptor word (BDW).

**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.

**DDNAME**=*symbol*

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

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

```
                                                                    }
DEVD=  {DA                                                          }
       {TA  [,DEN=   {0}]  [,TRTCH={C}]
                     {1}           {E}
                     {2}           {T}
                     {3}           {ET}

       {PT  [,CODE= {A}]                                            }
                    {B}
                    {C}
                    {F}
                    {I}
                    {N}
                    {T}

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

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

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

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

If system input is directed to an intermediate storage device, the **DEVD** operand is omitted, and the job control language for the problem program must designate the system input to be used. Similarly, if system output is directed to an intermediate storage device, the **DEVD** operand is omitted, and the job control language for the problem program must designate the system output to be used.

The following describes the device type and the optional operands that can be specified for each device type.

**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**).

**TA** — 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=** **{0}**
**{1}**
**{2}**
**{3}**

The **DEN** operand specifies the recording density in the number of bits-per-inch per track as shown in the following chart.

| DEN | 7-Track Tape | 9-Track Tape | 9-Track Tape (Phase Encoded) | 9-Track Tape Dual Density |
|---|---|---|---|---|
| 0 | 200 | — | — | — |
| 1 | 556 | — | — | — |
| 2 | 800 | 800 | — | 800 (NRZI) |
| 3 | — | — | 1600 | 1600 (PE) |

NRZI is for non-return-to-zero-inverse mode
PE is for phase encoded mode

Specifying DEN=0 for a 7-track, 3420 tape will result in a 556 bits-per-inch recording density, but corresponding messages and tape labels will indicate a 200 bits-per-inch recording density.

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

**TRTCH=** {C}
{E}
{ET}
{T}

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

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

**E** — Specifies even parity with no translation or conversion.

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

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

**PT** — Specifies that the data control block is used for a paper tape device (or any of the other devices following **PT**). If **PT** is coded, the following optional operand can be coded.

**CODE=** {A}
{B}
{C}
{F}
{I}
{N}
{T}

The **CODE** operand specifies the code in which the data was punched. The system converts these codes to EBCDIC code. If the **CODE** operand is not supplied by any source, **CODE=I** is assumed. The following describes the characters that can be specified.

**A** — Specifies 8-track tape in ASCII code.

**B** — Specifies Burroughs 7-track tape.

**C** — Specifies National Cash Register 8-track tape.

**F** — Specifies Friden 8-track tape.

**I** — Specifies IBM BCD perforated tape and transmission code with 8-tracks.

**N** — Specifies that no conversion required.

**T** — Specifies Teletype[1] 5-track tape.

[1] Trademark of Teletype Corporation

**PR** — Specifies that the data control block is used for an on-line printer (or any of the other device types following **PR**). If **PR** is coded, the following optional operand can be coded.

**PRTSP=** {**0**}
{**1**}
{**2**}
{**3**}

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

**0** — Specifies that spacing is suppressed (no space).

**1** — Specifies single-spacing.

**2** — Specifies double-spacing (one blank line between printed lines).

**3** — Specifies triple-spacing (two blank lines between printed lines).

**PC** — 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**] [**R**]
[**E**]

The **MODE** operand specifies the mode of operation for the card punch. The following describes the characters that can be specified. If the **MODE** operand is omitted, **E** is assumed.

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

**E** — Specifies that cards are punched in EBCDIC code.

**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 is specified in the DCB subparameter of a DD statement, either **C** or **E** must be specified if **R** is specified.

**STACK=** {**1**}
{**2**}

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

**1** — Specifies stacker number 1.

**2** — Specifies stacker number 2.

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

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

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

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

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

**R** — Specifies that the data set is for reading cards.

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

**W** — Specifies that the data set is for printing. See the description of the character **X** for associated punch and print data sets.

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

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

**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] [O]**
**[E] [R]**

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

**C** — Specifies that the cards to be read are in card image mode. In card image mode, the 12 rows of each card column are read into two consecutive bytes of main storage. Rows 12 through 3 are read into the low-order 6 bits of one byte, and rows 4 through 9 are read into the low-order 6 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 only).

**R** — Specifies that the program runs in read column eliminate mode (3505 card reader and 3525 card punch, read feature).

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

**STACK= {1}**
**{2}**

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

**1** — Specifies stacker number 1.

**2** — Specifies stacker number 2.

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

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

**D** — Specifies that the data protection option is to be used. The data protection option prevents punching information into card

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

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

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

**R** — Specifies that the data set is for reading cards.

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

**W** — Specifies that the data set is for printing. See the description of the character **X** for associated punch and print data sets.

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

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

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

**DSORG=** {**PS**}
{**PSU**}

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

**PS** — Specifies a physical sequential data set.

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

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

**EODAD**=*relexp*

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

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

**EROPT=** {**ACC**}
{**SKP**}
{**ABE**}

The **EROPT** operand specifies the action taken by the system when 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 or for output operations to a printer.

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

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

**SKP** — Specifies that the block that caused the error is skipped. Specifying **SKP** also causes the buffer associated with the data block to be released. This action can be specified when a data set is opened for INPUT, RDBACK, or UPDAT.

**ABE** — Specifies that the error results in the abnormal termination of the task. This action can be specified when the data set is opened for INPUT, OUTPUT, RDBACK, or UPDAT.

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

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

**EXLST**=*relexp*

The **EXLST** operand specifies the address of the problem program exit list. The **EXLST** operand is required if the problem program requires additional processing

for user labels, user totaling, data control block exit routine, end-of-volume, block count exits, to define a forms control buffer (FCB) image, or to use the DCB ABEND exit for ABEND condition analysis.

Refer to Appendix D of this publication for the format and requirements of exit list processing. For additional information about exit routine processing, refer to *OS Data Management Services Guide.*

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

**HIARCHY=** **{0}**
**{1}**

The **HIARCHY** operand specifies the main-storage hierarchy in which the buffer pool is constructed. The following describes the characters that can be specified:

**0** — Specifies that the buffer pool is constructed in processor storage.

**1** — Specifies that the buffer pool is constructed in IBM 2361 Core Storage.

The storage hierarchy can also be specified in a GETPOOL macro instruction. If the **HIARCHY** operand is omitted from all sources, the system constructs the buffer pool in processor storage.

The buffer pool is constructed in the user region or partition within the indicated hierarchy; if space is not available indicated hierarchy, the task is abnormally terminated. The **HIARCHY** operand is ignored in systems that do not have hierarchy support. The **HIARCHY** operand must not be specified for MVT systems with Model 65 multiprocessing.

**Source:** The **HIARCHY** operand can be supplied in the DCB macro instruction, in the DCB subparameter of a DD statement, or in the **HIARCHY** operand of a GETPOOL macro instruction.

**LRECL=** { *absexp* } **{X}**

The **LRECL** operand specifies the length, in bytes, for fixed-length logical records, or it specifies the maximum length, in bytes for variable-length or undefined-length (output only) logical records. The value specified in the **LRECL** operand cannot exceed the value specified in the BLKSIZE operand except when variable-length spanned records are used.

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

For variable-length logical records, the value specified in the **LRECL** operand must include the maximum data length (up to 32,752) plus four bytes for the record-descriptor word (RDW).

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

For variable-length spanned records, the logical record length (**LRECL**) can exceed the value specified in the BLKSIZE operand, and a variable-length spanned record can exceed the maximum blocksize. When the logical record length exceeds the maximum blocksize, the logical record length is specified as **LRECL=X**.

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

```
MACRF= {(G  {M[C]}    )                  }
       {L   }
       {T   }
       {D   }

     {(P  {M   }    [C]  )              }
       {L   }
       {T   }
       {D   }

     {(G  {M[C]}    ,P   {M}[C] )       }
       {L   }          {L}
       {T   }          {T}
       {D   }          {D}
```

The **MACRF** operand specifies the type of macro instructions (GET, PUT or PUTX, CNTRL, RELSE, and TRUNC) and the transmittal modes (move, locate, data, and substitute) that are used with the data set being created or processed. The operand can be coded in any of the combinations shown above; the following describes the characters that can be coded:

**C** — Specifies that the CNTRL macro instruction is used with the data set. If the CNTRL macro instruction is specified, the data set should be for a card reader (stacker selection) or printer (carriage and spacing control). The CNTRL option can be specified with GET in the move mode only.

**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.

T — Specifies that the substitute transmittal mode is used; the system substitutes a buffer for a work area contained in the problem program.

**Note:** For data sets on paper tape that are processed by QSAM, only **MACRF=(GM)** can be specified.

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

**OPTCD=** {**B** } [**T**]
{**C** }
{**H** }
{**Q** }
{**U** }
{**UC** }
{**W** }
{**WC** }
{**Z** }
{**ZC** }

The **OPTCD** operand specifies the optional services used with the QSAM data set. Two of the optional services (**OPTCD=B** and **OPTCD=H**) cannot be specified in the DCB macro instruction. They are requested in the DCB subparameter of a DD statement. Since all optional services codes must be supplied by the same source, the **OPTCD** operand must be omitted from the DCB macro instruction if either of these options is requested in a DD statement. The following describes the characters that can be specified.

**C** — Requests that chained scheduling be used. **OPTCD=C** cannot be specified when either **BFTEK=A** or **BFTEK=R** is specified for the same data control block. Also, chained scheduling cannot be specified for associated data sets or printing on a 3525.

**Q** — Request that ASCII tape records in an input data set be converted to EBCDIC code when the input record has been read, or an output record in EBCDIC code be converted to ASCII code before the record is written.

**T** — Requests the user totaling facility. If this facility is requested, the EXLST operand should specify the address of an exit list to be used.

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

**W** — Specifies that the system performs a validity check for each record written on the direct-access device being used. If the device is a 2321 data cell, the system performs a validity check whether it is requested or not.

**Z** — For magnetic tape, input only, the **Z** option requests 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 selected when the operating system is generated. **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 requests the system to use the search direct option to accelerate the input operations for a data set. **OPTCD=Z** cannot be specified when **RECFM=UT, FS, FBT, VS,** or **VBS**.

**Source:** 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 SL or AL tape, the EOV routine will treat EOF labels as EOV labels until the volume serial list is exhausted. When no more volumes are available, control is passed to the user's end-of-data routine. This option allows SL or AL tapes to be read out of order or to be concatenated to another tape using one DD statement.

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

```
RECFM= {U  [T  ]    [A]}
                    [M]

       {V  [B  ]    [A]}
           [S  ]    [M]
           [T  ]
           [BS ]
           [BT ]

       {D  [B  ]    [A]}

       {F  [B  ]    [A]}
           [S  ]    [M]
           [T  ]
           [BS ]
           [BT ]
```

The **RECFM** operand specifies the record format and characteristics of the data set being created or processed. All record formats can be used in QSAM. When **RECFM** is not specified, undefined format is assumed. The following describes the characters that can be specified.

**A** — Specifies that the records in the data set contain American National Standards Institute (ANSI) control characters. Refer to Appendix E for a description of control characters.

**B** — Specifies that the data set contains blocked records.

**D** — Specifies that the data set contains variable-length ASCII tape records. See **OPTCD=Q** and the **BUFOFF** operand for a description of how to specify 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. Refer to Appendix E for a description of control characters. **RECFM=M** cannot be used with ASCII data sets.

**S** — For fixed-length records, **S** specifies that the records are written as standard blocks; the data set does not contain any truncated blocks or unfilled tracks, with the exception of the last block or track in the data set.

For variable-length records, **S** specifies that a record can span more than one block. If spanned records are used, exchange buffering (**BFTEK=E**) cannot be specified.

**T** — Specifies that the track-overflow feature is used with the data set. The track-overflow feature allows a record to be written partially on one track and the remainder of the record on the following track (if required). Chained scheduling (**OPTCD=C**) and exchange buffering (**BFTEK=E**) cannot be used if the track-overflow feature is used.

**U** — Specifies that the data set contains undefined-length records.

**V** — Specifies that the data set contains variable-length records.

**Note:** **RECFM=V** cannot be specified for a card reader data set or an ASCII tape data set.

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

**SYNAD=***relexp*

The **SYNAD** operand specifies the address of the error analysis routine given control when an uncorrectable input/output error occurs. The contents of the registers when the error analysis routine is given control are described in Appendix A of this publication.

The error analysis routine must not use the save area pointed to by register 13, because this area is used by the system. The system does not restore registers when it regains control from the error analysis routine. The error analysis routine can issue a RETURN macro instruction that uses the address in register 14 to return control to the system.

If the error condition was the result of a data-validity error, the control program takes the action specified in the **EROPT** operand; otherwise, the task is abnormally terminated. The control program takes these actions when the **SYNAD** operand is omitted or when the error analysis routine returns control.

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

# DCBD—Provide Symbolic Reference to Data Control Blocks (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM)

The DCBD macro instruction is used to generate a dummy control section that provides symbolic names for the fields in one or more data control blocks. The names and attributes of the fields appear as part of the description of each data control block in *OS System Control Blocks* publication. Attributes of the symbolically named fields in the dummy section are the same as the fields in the data control blocks, with the exception of fields containing 3-byte addresses. The symbolically named fields containing 3-byte addresses have length attributes of four and are aligned on fullword boundaries.

The name of the dummy control section generated by a DCBD macro instruction is IHADCB. The use of any of the symbolic names provided by the dummy section must be preceded by a USING instruction specifying IHADCB and a dummy section base register (which contains the address of the actual data control block). The DCBD macro instruction can only be issued once within any assembled module; however, the resulting symbolic names can be used for any number of data control blocks by changing the address in the dummy section base register. The DCBD macro instruction can be coded at any point in a control section; if coded at any point other than at the end of a control section; however, the control section must be resumed by coding a CSECT instruction.

The DCBD macro instruction is written as follows:

| | | |
|---|---|---|
| blank | DCBD | [DSORG=([GS] [[BS] [,DA] [,IS] [,LR] [,PO] [,PS] [QS]])] [,DEVD=([DA] [,PC] [,PR] [,PT] [,RD] [,TA] [,MR])] |

**DSORG= ([GS] )**
**([BS][,DA][,IS][,LR][,PO][,PS][,QS] )**

The **DSORG** operand specifies the types of data control blocks for which symbolic names are provided. If the **DSORG** operand is omitted, the **DEVD** operand is ignored, and symbolic names are provided only for the "foundation block" portion that is common to all data control blocks. One or more of the following pairs of characters can be specified; each pair of characters must be separated by a comma:

**BS** — Specifies a data control block for a sequential data set and basic access method.

**DA** — Specifies a data control block for a direct data set.

**IS** — Specifies a data control block for an indexed sequential data set.

**LR** — Specifies a dummy section for the logical record length field (DCBLRECL) only.

**PO** — Specifies a data control block for a partitioned data set.

**PS** — Specifies a data control block for a sequential data set. **PS** includes both **BS** and **QS**.

**QS** — Specifies a data control block for a sequential data set and queued access method.

**GS** — Specifies a data control block for graphics; this operand cannot be used in combination with any of the above.

**DEVD=[DA][,PC][,PR][,PT][,RD][,TA][,MR]**

The **DEVD** operand specifies the types of devices on which the data set can reside. If the **DEVD** operand is omitted and a sequential data set is specified in the **DSORG** operand, symbolic names are provided for all of the device types listed below. One or more of the following pairs of characters can be specified; each pair of characters must be separated by a comma:

**DA** — Direct-access device

**PC** — Online punch

**PR** — Online printer

**PT** — Paper tape

**RD** — Online card reader or read punch feed

**TA** — Magnetic tape

**MR** — Magnetic character reader

# ESETL—End Sequential Retrieval (QISAM)

The ESETL macro instruction ends the sequential retrieval of data from an indexed sequential data set and causes the buffers associated with the specified data control block to be released. An ESETL macro instruction must separate SETL macro instructions issued for the same data control block.

The ESETL macro instruction is written as follows:

| [symbol] | ESETL | dcb address |
|---|---|---|

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

The *dcb address* operand specifies the address of the data control block opened for the indexed sequential data set being processed.

# FEOV—Force End of Volume (BSAM and QSAM)

The FEOV macro instruction causes the system to assume an end-of-volume condition, and causes automatic volume switching. Volume positioning for magnetic tape can be specified by the option operand. If no option is coded, the positioning specified in the OPEN macro instruction is used. Output labels are created as required and new input labels are verified. The standard exit routines are given control as specified in the data control block exit list. For BSAM, all input and output operations must be tested for completion before the FEOV macro instruction is issued. The end-of-data-set (EODAD) routine is given control if an input FEOV macro instruction is issued for the last volume of an input data set.

The FEOV macro instruction is written as follows:

| [symbol] | FEOV | dcb address [,REWIND ,LEAVE] |
|---|---|---|

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

The *dcb address* operand specifies the address of the data control block for an opened sequential data set.

The following operands request optional services.

**REWIND** — Requests that the system position the tape at the load point regardless of the direction of processing.

**LEAVE** — Requests that the system position the tape at the logical end of the data set on that volume; this option causes the tape to be positioned at a point after the tapemark that follows the trailer labels. Note that multiple tape units must be available to achieve this positioning. If only one tape unit is available, its volume is rewound and unloaded.

**Note:** If an FEOV macro is issued for a spanned multivolume data set that is being read using QSAM, errors may occur when the next GET macro is issued following an FEOV macro if the first segment on the new volume is not the first segment of a record. The errors include duplicate records, program checks in the user program, and invalid input from the variable spanned data set.

# 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 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 instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | FIND | dcb address, {name address, D / relative address list, C} |

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

The *dcb address* operand specifies the address of the data control block for the opened partitioned data set being processed.

*name address*—RX-Type Address, (2-12), or (0)

The *name address* operand specifies the main-storage address of a doubleword that contains the data set member name. The name must start in the first byte of the doubleword and be padded on the right (if necessary) to complete the eight-byte doubleword.

**D** — Specifies that only a member name has been supplied, and the access method must search the directory of the data set indicated in the data control block to find the location of the member.

*relative address list*—RX-Type Address, (2-12), or (0)

The *relative address list* operand specifies the main-storage address that contains the relative address (TTRK) for the beginning of a data set member. The relative address can be a list entry completed by using a BLDL macro instruction for the data set being processed, or the relative address can be supplied by the problem program.

**C** — Specifies that a relative address has been supplied, and no directory search is required. The relative address supplied is used directly by the access method for the next input operation.

### *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 zero.

*name address,* **D**

00 - Successful execution

04 - Name not found

08 - Permanent input/output error found during directory search

*relative address list,* **C**

00 - At all times. If the relative address is in error, execution of the next READ macro instruction causes control to be passed to the error analysis (SYNAD) routine.

# 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 instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | FREEBUF | dcb address, register |

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

The *dcb address* operand specifies the address of the data control block for an opened data set to which the buffer pool has been assigned.

*register*—(2-12)

The *register* operand specifies one of registers 2 through 12 that contains the address of the buffer being returned to the buffer pool.

# FREEDBUF—Return a Dynamically Obtained Buffer (BDAM and BISAM)

The FREEDBUF macro instruction causes the system to return a buffer to the buffer pool assigned to the specified data control block. The buffer must have been acquired through dynamic buffering; that is, by coding 'S' for the *area address* operand in the associated READ macro instruction.

**Note:** A buffer acquired dynamically can also be released by a WRITE macro instruction; refer to the description of the WRITE macro instruction for BDAM or BISAM.

The FREEDBUF macro instruction is written as follows:

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

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

The *decb address* operand specifies the address of the data event control block (DECB) used or created by the READ macro instruction that acquired the buffer dynamically.

**K** — Specifies that BISAM is being used.

**D** — Specifies that BDAM is being used.

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

The *dcb address* operand specifies the address of the data control block for the opened data set being processed.

# FREEPOOL—Release a Buffer Pool (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM)

The FREEPOOL macro instruction causes an area of main storage, previously assigned as a buffer pool for a specified data control block, to be released. The main-storage area must have been acquired either automatically or by the execution of a GETPOOL macro instruction. For queued access methods, the FREEPOOL macro instruction must not be issued until after a CLOSE macro instruction has been issued for all the data control blocks using the buffer pool. For basic access methods, the FREEPOOL macro instruction can be issued as soon as the buffers are no longer required. A buffer pool should be released only once, regardless of the number of data control blocks sharing the buffer pool.

If **BFALN=F** is supplied from a source other than the DCB macro instruction, the CLOSE macro instructions removes the bit that designates fullword alignment from the data control block. In this case, if a FREEPOOL macro instruction is issued after the CLOSE macro instruction, the system does not release the complete buffer area (eight bytes are not released).

The FREEPOOL macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | FREEPOOL | dcb address |

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

The *dcb address* operand specifies the address of a data control block to which the buffer pool has been assigned.

# GET—Obtain Next Logical Record (QISAM)

The GET macro instruction causes the system to retrieve the next record. Control is not returned to the problem program until the record is available.

The GET macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | GET | dcb address [,area address] |

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

The *dcb address* operand specifies the address of the data control block for the opened input data set being retrieved.

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

The *area address* operand specifies the main-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 the 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 the move mode has been specified in the data control block, the *area address* operand must specify the main-storage 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 must be closed or an ESETL macro instruction must be issued. An attempt to continue to use the data set will have unpredictable results.
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 routine are described in Appendix A.
3. When the key of an unblocked fixed-length record is retrieved with the data, the address of the key is returned as follows (see the SETL macro instruction):

   Locate mode — The address of the key is returned in register 0.

   Move mode — The key appears in front of the record in the main-storage area.
4. If a GET macro instruction is issued for a data set and the previous request issued for the same data set was an OPEN, ESETL, or unsuccessful SETL (no record found), a SETL B (key and data) is invoked automatically, and the first record in the data set is returned.

# GET—Obtain Next Logical Record (QSAM)

The GET macro instruction causes the system to retrieve the next record. Various modes are available and are specified in the DCB macro instruction. In the locate mode, the GET macro instruction locates the next sequential record or record segment to be processed. The system returns the address of the record in register 1 and places the length of the record or segment in the logical-record-length (DCBLRECL) field of the data control block. The user can process the record within the input buffer or move the record to a work area.

In the move mode, the GET macro instruction moves the next sequential record to the user's work area. This work area must be big 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. (This feature provides compatibility with the substitute mode GET.) The record length is placed in the DCBLRECL field. The move mode can be used only with simple buffering.

In the data mode, which is available only for variable-length spanned records, the GET macro instruction moves only the data portion of the next sequential record to the user's work area.

In the substitute mode, the GET macro instruction transfers ownership of the next sequential record in a data set from the system to the user. In return, the ownership of a work area is transferred from the user to the system for future use as an input buffer. There is no movement of data. The address of an input buffer containing the record is returned to the user in register 1 after the instruction is executed. The system returns the record length in the DCBLRECL field. For undefined-length records, the DCBLRECL field is equal to the BLKSIZE field for chained scheduling. The substitute mode can be used only with exchange buffering and cannot be used with variable-length records.

If the ASCII translation routines are included when the operating system is generated, translation can be requested by coding LABEL=(,AL) or (,AUL) in the DD statement, or it can be requested by coding OPTCD=Q in the DCB macro instruction or DCB subparameter of the DD statement. When translation is requested, all QSAM records whose record format (RECFM operand) is F, FB, D, DB, or U are automatically translated from ASCII code to EBCDIC code as soon as the input buffer is full. For translation to occur correctly, all input data must be in ASCII code.

The GET macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | GET | dcb address [,area address] |

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

The *dcb address* operand specifies the address of the data control block for the opened input data set being retrieved.

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

The *area address* operand specifies the main-storage address of an area into which the system is to move the record (move or data mode), or it specifies the main-storage address of an area to be exchanged for the buffer containing the record (substitute mode). The move, locate, data, or substitute 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, data, or substitute mode, the system assumes that register 0 contains the area address. The following describes the operation of the four modes:

*Locate mode:* If the 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.

When retrieving variable-length spanned records, 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.) 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. To process a record when the logical record length is greater than 32,756 bytes, LRECL=X must be specified in the data control block, and the problem program must assemble the segments into a complete logical record.

*Move mode:* If the move mode has been specified in the data control block, the *area address* operand specifies the main-storage address of an area in the problem program into which the system will move the record.

For variable-length spanned records, the system constructs the record-descriptor word in the first four bytes of the main-storage area and assembles one or more segments into the data portion of the logical record; the segment descriptor words are removed.

*Data mode:* If the 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 main storage area in the problem program into which the system will move the data portion of the logical record; a record-descriptor word is not constructed when data mode is used.

*Substitute mode:* If the substitute mode is specified in the data control block, the *area address* operand specifies the main-storage address of an area in the problem program that will be exchanged for the buffer containing the record. The system returns the address of the buffer containing the record in register 1.

## *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 job step.

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 routine are described in Appendix A.

# GETBUF—Obtain a Buffer (BDAM, BISAM, BPAM, and BSAM)

The GETBUF macro instruction causes the control program to obtain a buffer from the buffer pool assigned to the specified data control block and to return the address of the buffer in a designated register. The BUFCB field of the data control block must contain the address of the buffer pool control block when the GETBUF macro instruction is issued. The system returns control to the instruction following the GETBUF macro instruction. The buffer obtained must be returned to the buffer pool using a FREEBUF macro instruction.

The GETBUF macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | GETBUF | dcb address, register |

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

The *dcb address* operand specifies the address of the data control block that contains the buffer pool control block address.

*register*—(2-12)

The *register* operand specifies one of the registers 2 through 12 in which the system is to place the address of the buffer obtained from the buffer pool. If no buffer is available, the contents of the designated register are set to zero.

# GETPOOL—Build a Buffer Pool (BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM)

The GETPOOL macro instruction causes a buffer pool to be constructed in a main-storage area provided 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 instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | GETPOOL | dcb address, {number of buffers, buffer length / (0)} [,HIARCHY={0/1}] |

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

The *dcb address* operand specifies the address of the data control block to which the buffer pool is assigned. Only one buffer pool can be assigned to a data control block.

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

The *number-of-buffers* operand specifies the number of buffers in the buffer pool up to a maximum of 255.

*buffer length* — symbol, decimal digit, absexp, or (2–12)

The *buffer length* operand specifies the length, in bytes, or each buffer in the buffer pool. The value specified for the buffer length must be a doubleword multiple; otherwise the system rounds the value specified to the next higher doubleword multiple. The maximum length that can be specified is 32,760 bytes. For QSAM, the *buffer length* must be at least as large as the value specified in the blocksize (DCBBLKSI) field in the data control block.

(0)—Coded as shown

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

Register 0

| Bits: | Number of Buffers | | Buffer Length | |
|---|---|---|---|---|
| | 0 | 15 | 16 | 31 |

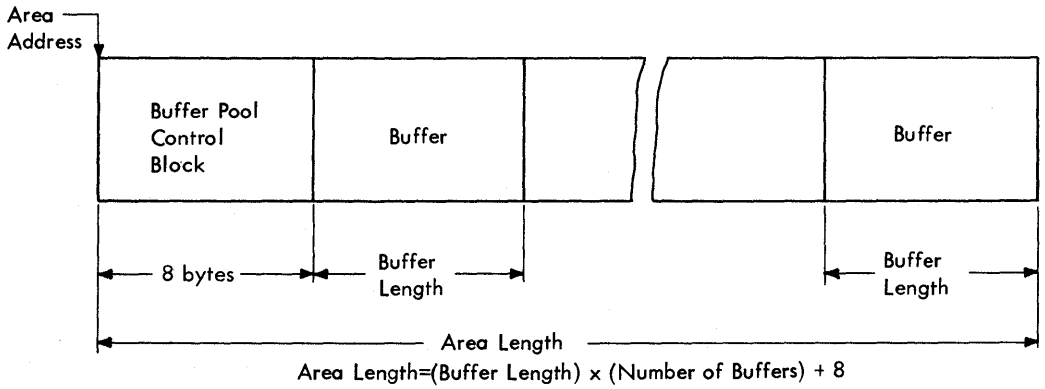
**HIARCHY=** {**0**}
{**1**}

The **HIARCHY** operand specifies the main-storage hierarchy in which the buffer pool is constructed. If the **HIARCHY** parameter is omitted, the buffer pool is formed in the main-storage hierarchy indicated in the data control block. If no **HIARCHY** parameter is specified in the data control block, hierarchy 0 is assigned. The **HIARCHY** operand is ignored in an operation system that does not have main-storage hierarchy support. The following characters can be specified in the **HIARCHY** operand.

**0** — Specifies that the buffer pool is constructed in processor storage.

**1** — Specifies that the buffer pool is constructed in IBM 2361 Core Storage.

The following illustration shows the format of the buffer pool. The buffer pool and the associated main-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.



# NOTE—Provide Relative Position (BPAM and BSAM—Tape and Direct Access Only)

The NOTE macro instruction causes the system to return the relative 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 relative position, in terms of the current volume, is returned in register 1 as follows:

**Magnetic Tape:** The block number is in binary, right-adjusted in register 1 with high-order bits set to zero. Do not use a NOTE macro instruction for tapes without standard labels when:

- The data set is opened for RDBACK (specified in the OPEN macro instruction).
- The DISP parameter of the DD statement for the data set specifies DISP=MOD.

**Direct-Access Device:** TTRz format, where:

TT is a 2-byte relative track number.

R is a 1-byte block (record) number on the track indicated by TT.

z is a byte set to zero.

**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 macro instruction, the track capacity of the direct-access device is returned in register 0.

The NOTE macro instruction is written as follows:

| [symbol] | NOTE | dcb address |
|---|---|---|

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

The *dcb address* operand specifies the address of the data control block opened for the partitioned or sequential data set being processed.

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

The OPEN macro instruction causes the specified data control block(s) to be completed and the data set(s) identified in the data control block(s) to be prepared for processing. Input labels are analyzed and output labels are created. Control is given to exit routines as specified in the data control block exit list. The processing method (option 1) is designated to provide correct volume positioning for the data set and define the processing mode (INPUT, OUTPUT, etc.) 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.

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, refer to *OS Data Management Services Guide.*

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)],...) |

*dcb address*—A-Type Address or (2-12)

The *dcb address* operand(s) specifies the address of the data control block(s) for the data set(s) to be prepared for processing.

*options*

The *options* operands shown in the following illustration indicate the volume positioning available based on the device type and access method being used. If option 1 is omitted, INPUT is assumed. If option 2 is omitted, DISP is assumed. Option 1 must be coded if option 2 is coded. Options 1 and 2 are ignored for BISAM and QISAM (in the scan mode), and the data control block indicates the operation. OUTPUT or OUTIN must be specified when creating a data set.

| 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 ]<br>[OUTPUT ]<br>[RDBACK] | [,REREAD ]<br>[,LEAVE ]<br>[,DISP ] | [INPUT ]<br>[OUTPUT ]<br>[UPDAT ] | [,REREAD ]<br>[,LEAVE ]<br>[,DISP ] | [INPUT ]<br>[OUTPUT ] | — |
| BSAM | [INPUT ]<br>[OUTPUT ]<br>[INOUT ]<br>[OUTIN ]<br>[RDBACK] | [,REREAD ]<br>[,LEAVE ]<br>[,DISP ] | [INPUT ]<br>[OUTPUT ]<br>[INOUT ]<br>[OUTIN ]<br>[UPDAT ] | [,REREAD ]<br>[,LEAVE ]<br>[,DISP ] | [INPUT ]<br>[OUTPUT ] | — |
| QISAM (Load Mode) | — | — | [OUTPUT ] | — | — | — |
| BPAM, BDAM | — | — | [INPUT ]<br>[OUTPUT ]<br>[UPDAT ] | — | — | — |

The following describes the options shown in the preceding illustration. All option operands are coded as shown.

| **Option 1** | **Meaning** |
|---|---|
| INPUT | Input data set. |
| INOUT | The data set is first used for input and, without reopening, it is used as an output data set. The data set is processed as INPUT 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, it is used as an input data set. The data set is processed as OUTPUT if LABEL=(,,,OUT) is specified in the DD statement. |
| RDBACK | Input data set, positioned to read backward. |
| UPDAT | Data set to be updated in place. |
| **Option 2** | **Meaning** |
| LEAVE | Positions the current volume to the logical end of the data set. |
| REREAD | Positions the current volume to reprocess the data set when volume switching occurs. |
| DISP | Performs volume positioning implied by the DISP parameter of the DD control statement, as follows: |

| *DISP Parameter* | *Action* |
|---|---|
| PASS | Forward space to the end of the data set on the current volume |
| DELETE | Rewind the current volume |
| KEEP, CATLG, or UNCATLG | Rewind and unload the current volume |

**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.

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 cause the results indicated:

| *Error* | *Result* |
|---|---|
| Opening 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. |
| Opening a data control block when a corresponding DD statement has not been provided. | A "DD STATEMENT MISSING" message is issued. An attempt to use the data set causes unpredictable results. |

The last of these errors can be detected by testing bit 3 of the DCBOFLGS field in the data control block. Bit 3 is set to 0 in the case of an error and can be tested by the sequence:

```
TM DCBOFLGS,X'10'
BZ ERRORRTN          (Branch to user's error routine)
```

Executing the two instructions shown above requires writing a DCBD macro instruction in the program, and a base register must be defined with a USING statement before the instructions are executed.

# OPEN—List Form

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

The list consists of a one-word entry for each DCB in the parameter list; the high-order byte is used for the options and the three low-order bytes are used for the DCB address. The end of the list is indicated by a one 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 one 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 one 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 instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | OPEN | ([dcb address], [(options)],...),MF=L |

*dcb address*—A-Type Address

**MF=L**—Coded as shown

The **MF=L** operand specifies that the OPEN macro instruction is used to create a data management parameter list that is referenced by an execute form instruction.

# 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 instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | OPEN | [([dcb address], [(options)],...)]<br>,MF=(E, { data management list address / (1) }) |

*dcb address*—RX-Type Address or (2-12)

**MF=(E,** { *data management list address* / **(1)** } **)**

This operand specifies that the execute form of the OPEN macro instruction is used, and an existing data management parameter list (created by a list-form instruction) is used. The **MF=** operand is coded as follows:

**E**—Coded as shown

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

# POINT—Position to a Relative Block (BPAM and BSAM—Tape and Direct Access Only)

The POINT macro instruction causes the system to start processing the next READ or WRITE operation at the specified block in the data set on the current volume. All input and output operations using the same data control block must have been tested for completion before the POINT macro instruction is issued. When processing a data set that has been opened for UPDAT, the POINT macro instruction must be followed by a READ macro instruction. When processing an output data set, the POINT macro instruction must be followed by a WRITE macro instruction prior to closing the data set, unless a CLOSE macro instruction (with TYPE=T specified) was issued prior to the POINT macro instruction. Issuing a POINT macro instruction for the system input data set or a system output data set on magnetic tape results in an effective NOP instruction.

The POINT macro instruction is written as follows:

| [symbol] | POINT | dcb address,block address |
|---|---|---|

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

The *dcb address* operand specifies the address of the data control block for the opened data set that is to be positioned.

*block address*—RX-Type Address, (2-12), or (0)

The *block address* operand specifies the 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. 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 zero; add one if reading tape backward. Do not use the POINT macro instruction for tapes without standard labels when:

- The data set is opened for RDBACK.
- The DD statement for the data set specifies DISP=MOD.

If OPTCD=H is indicated in the data control block, the POINT macro instruction can be used to perform record positioning on DOS tapes that contain embedded checkpoint records. Any embedded checkpoint records that are encountered during the record positioning are bypassed and are not counted as blocks spaced over. OPTCD=H must be specified in a job control language DD statement. Do not use the POINT macro instruction to backspace DOS 7-track tapes that are written in data convert mode and that contain embedded checkpoint records.

*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 zero; 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 previous description of TTRz).

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 READ or WRITE macro instruction is executed.

# PRTOV—Test for Printer Carriage Overflow (BSAM and QSAM—Online Printer and 3525 Card Punch, Print Feature)

The PRTOV macro instruction is used to control the page format for an online printer when carriage control characters are not being used or to supplement the carriage control characters that are being used.

The PRTOV macro instruction causes the system to test for an overflow condition on the specified channel (either channel 9 or channel 12) of the printer carriage control, and either skip the printer carriage to the line corresponding to channel 1, or transfer control to the exit address, if one is specified. Overflow is detected after printing the line that follows the line corresponding to channel 9 or channel 12.

When the PRTOV macro instruction is used with a 3525 card punch, print feature, channel 9 or 12 can be tested. If an overflow condition occurs, control is passed to the overflow exit routine if the overflow exit address is coded, or a skip to channel 1 (first print-line of the next card) occurs.

When requesting overprinting (for example, to underscore a line), the PRTOV macro instruction is issued before the first PUT or WRITE macro instruction only. The PRTOV macro instruction should be issued only when the device type is an online printer. PRTOV cannot be used to request overprinting on the 3525.

The PRTOV macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | PRTOV | dcb address,{9 / 12}[,overflow exit address] |

*dcb address*—RX-Type Address or (2-12)

The *dcb address* operand specifies the address of the data control block opened for output to an online printer or 3525 card punch with a print feature.

**9** — Coded as shown

**12** — Coded as shown

These operands specify which channel 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 the *OS Data Management Services Guide.*

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

The *overflow exit address* operand specifies the address of the user-supplied routine to be given control when an overflow condition is detected on the specified channel. If this operand is omitted, the printer carriage skips to the first line of the next page or the 3525 skips to the first line of the next card before executing the next PUT or WRITE macro instruction.

When the overflow exit routine is given control, the contents of the registers are as follows:

| *Register* | *Contents* |
|---|---|
| 0 and 1 | The contents are destroyed. |
| 2 - 13 | The same contents as before the macro instruction was executed. |
| 14 | Return address. |
| 15 | Overflow exit routine address. |

# PUT—Write Next Logical Record (QISAM)

The PUT macro instruction causes the system to write a record into an indexed sequential data set. If the move mode is used, the PUT macro instruction moves a logical record into an output buffer from which it is written. If the locate mode is specified, the address of the next available output buffer segment is available in register 1 after the PUT macro instruction is executed. The logical record can then be constructed in the buffer for output as the next record. The records are blocked by the system (if specified in the data control block) before being placed in the data set. The system uses the length specified in the record length (DCBLRECL) field of the data control block as the length of the record currently being written. When constructing blocked variable-length records in the locate mode, the problem program may either specify the maximum record length once in the DCBLRECL field of the data control block or provide the actual record length in the DCBLRECL field before issuing each PUT macro instruction. Use of the maximum record length may result in more but shorter blocks, since 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 instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | PUT | dcb address [,area address] |

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

The *dcb address* operand specifies the address of the data control block for the opened ISAM data set.

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

The *area address* operand specifies the main-storage 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 the 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 the move mode has been specified in the data control block, the *area address* operand must specify the main-storage 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 zero contains the area address.

### *Put Routine Exit*

The error analysis (SYNAD) routine is given control if the output operation could not be completed satisfactorily. The contents of the registers when the error analysis routine is given control are described in Appendix A.

# PUT—Write Next Logical Record (QSAM)

The PUT macro instruction causes the system to write a record in a sequential data set. Various modes are available and are specified in the DCB macro instruction. In the locate mode, the address of an area within an output buffer is returned in register 1 after the macro instruction is executed. The user should subsequently construct, at this address, the next sequential record or record segment. The move mode of the PUT macro instruction causes a logical record to be moved into an output buffer. In the data mode, which is available only for variable-length spanned records, the PUT macro instruction moves only the data portion of the record into one or more output buffers. When the substitute mode is specified, the macro transfers ownership of a work area containing a record to the control program. In return, the ownership of a buffer segment is transferred to the user, for use as a work area. There is no movement of data in main storage.

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. For variable-length spanned records, the system segments the record according to the record length, buffer length, and amount of unused space remaining in the output buffer. The smallest segment created will be 5 bytes, 4 for the segment descriptor word plus one byte of data.

If the ASCII translation routines are included when the operating system is generated, translation can be requested by coding LABEL=(,AL) or (,AUL) in the DD statement, or it can be requested by coding OPTCD=Q in the DCB macro instruction or DCB subparameter of the DD statement. When translation is requested, all QSAM records whose record format (RECFM operand) is F, FB, D, DB, or U are automatically translated from EBCDIC code to ASCII code. For translation to occur correctly, all output data must be in EBCDIC code; any EBCDIC character that cannot be translated into an ASCII character is replaced by a substitute character.

The PUT macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | PUT | dcb address [,area address] |

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

The *dcb address* operand specifies the address of the data control block for the data set opened for output.

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

The *area address* operand specifies the address of a main-storage area that contains the record to be written (move or data mode), or it specifies the address of a main-storage area to be exchanged for a buffer (substitute mode). The move, locate, data, or substitute 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, data, or substitute mode, the system assumes that register zero contains the area address. The following describes the operation of the four modes:

*Locate Mode:* If the locate mode is specified, the *area address* operand must be omitted. The system returns the address of the next available buffer in register 1; this is the buffer into which the next record is placed.

When variable-length spanned records are used and a record area has been provided for a logical record interface (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 32,756 bytes). The system segments the record and writes all segments, providing proper control codes for each segment. If, for variable-length spanned records, an area has not been provided, the actual length remaining in the buffer will be returned in register 0. In this case, it is the user's responsibility to segment the records and process them in terms of record segments. 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 instruction; QSAM returns the address of the buffer, but the record is not written until the next PUT macro instruction is issued.

*Move Mode:* If the move mode has been specified in the data control block, the *area address* operand specifies the address of the main-storage area that contains the record to be written. The system moves the record to an output buffer before control is returned. The address of the main-storage area is returned in register 1 (this action provides compatibility with substitute mode operations, and makes it possible for the problem program to be used in instances where substitute mode is requested but cannot be supported by the system).

*Data Mode:* If the 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 a main-storage area in the problem program that contains the data portion of the record to be written. The system moves the data portion of the record to an output buffer before control is returned. The user must place the total data length in the DCBPRECL (not DCBLRECL) field of the data control block before the PUT macro instruction is issued.

*Substitute Mode:* If the substitute mode is specified in the data control block, the *area address* operand specifies the address of a main-storage area in the problem program that contains the next record to be written. The area is exchanged for an empty buffer. The address of the empty buffer is returned in register 1.

### *Put Routine Exit*

The error analysis (SYNAD) routine is given control if the output operation could not be completed satisfactorily. The contents of the registers when the error analysis routine is given control are described in Appendix A.

# 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 record must always have been brought into main storage by a locate mode GET macro instruction. The logical records are blocked by the control program, as specified in the data control block, before they are placed in the output data set. The control program uses the length specified in the DCBLRECL field as the length of the record currently being stored. Control is not returned to the user's program until the control program has processed the record.

The PUTX macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | PUTX | dcb address [,input dcb address] |

*dcb address* RX-Type Address, (2-12), or (1)

The *dcb address* operand specifies the address of the data control block for a data set opened for output.

*input dcb address*—RX-Type Address, (2-12), or (0)

The *input dcb address* operand specifies the address of a data control block opened for input. The PUTX macro instruction can be used for the following modes:

*Output Mode:* This mode is used with QSAM only. The *input dcb address* operand specifies the address of the data control block opened for input. If this operand is omitted, the system assumes that register 0 contains the input dcb address.

*Update Mode:* The *input dcb address* operand is omitted for update mode.

## *PUTX Routine Exit*

The error analysis (SYNAD) routine is given control if the operation is not completed satisfactorily. The contents of the registers when the error analysis routine is given control are described in Appendix A.

# READ—Read a Block (BDAM)

The READ macro instruction causes a block to be retrieved from a data set and placed in a designated area of main 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, is constructed as part of the macro expansion.

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

| | | |
|---|---|---|
| [symbol] | READ | decb name,type,dcb address,{area address / 'S'},{length / 'S'},<br>{key address / 'S'},block address[,next address] |

*decb name*—symbol

The *decb name* operand specifies the name assigned to the data event control block created as part of the macro expansion.

*type* — {**DI** **[F]** **[R]** **[X]** **[RU]** }

{**DK** **[F]** **[R]** **[X]** **[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 it requests that the system provide block position feedback into the area specified by the *block address* operand. The descriptions of the WRITE and RELEX macro instructions contain a description of releasing a data block that is under exclusive control. This character can be coded as a suffix to **DI** or **DK** as shown above.

**R** — Requests that the system provide next address feedback into the area specified by the *next address* operand. When R is coded, the feedback is the relative track address of the next data record. This character can be coded as a suffix to **DI** or **DK, DIF, DIX, DKF,** or **DKX** as shown above, but it can be coded only for use with variable-length spanned records.

**RU** — Requests that the system provide next address feedback into the area specified by the *next address* operand. When **RU** is coded, the feedback is the relative track address of the next capacity record (R0) or data record whichever occurs first. These characters can be coded as a suffix to **DI, DK, DIF, DIX, DKF,** or **DKX,** but it can be coded only for use with variable-length spanned records.

*dcb address*—A-Type Address or (2-12)

The *dcb address* operand specifies the address of the data control block opened for the data set to be read.

*area address*—A-Type Address, (2-12), or **'S'**

The *area address* operand specifies the address of the main-storage area into which the data block is to be placed. If **'S'** is coded instead of an address, dynamic buffering is requested (dynamic buffering must also be specified in the MACRF operand of the DCB macro instruction). When dynamic buffering is used, the system acquires a buffer and places its address in the data event control block.

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

The *length* operand specifies the number of data bytes to be read up to a maximum of 32,760. If **'S'** is coded instead of a length, the number of bytes to be read is taken from the data control block.

*key address*—A-Type Address, (2-12), **'S'**, or 0

The *key address* operand specifies the address of the main-storage area for the key of the desired data block. If the search operation is made using a key, the area must contain the key. Otherwise, the key is read into the designated area. If the key is read and **'S'** was coded for the area address, **'S'** can also be coded for the key address; the key and data are read sequentially into the buffer acquired by the system. If the key is not to be read, specify 0 instead of an address or **'S'**.

*block address*—A-Type Address or (2-12)

The *block address* operand specifies the address of the main-storage 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 main-storage 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 was originally presented by the READ macro instruction, and the field can be either three or eight bytes long depending on the type of addressing.

If OPTCD=F has not been specified, feedback (if requested) is in the form of an actual device address, and the field must be eight bytes long.

*next address*—A-Type Address or (2-12)

The *next address* operand specifies the address of the main-storage area where the system places the relative address of the next record. The length operand must be specified as **'S'**. When the *next address* operand is specified, an **R** or **RU** must be added to the *type* operand (for example, **DIR** or **DIRU**). The **R** indicates that the next address returned is the next data record. **RU** indicates that the next address returned is for the next data or capacity record, whichever occurs first. The *next address* operand can be coded only for use with variable-length spanned records.

# READ—Read a Block (BPAM and BSAM)

The READ macro instruction causes a block to be retrieved from a data set and placed in a designated area of main storage. Control may be returned to the problem program before the block is retrieved. The input operation must be tested for completion using a CHECK macro instruction. A data event control block, shown in Appendix A, is constructed as part of the macro expansion.

If the OPEN macro instruction specifies UPDAT, both the READ and WRITE macro instruction must refer to the same data event control block. Refer to the list form of the READ or WRITE macro instruction for a description of how to construct a data event control block; refer to the execute form of the READ or WRITE macro instruction for a description of how to modify an existing data event control block.

The standard form of the READ macro instruction is written as follows (the list and execute forms are shown following the descriptions of the standard form instructions):

| | | |
|---|---|---|
| [symbol] | READ | decb name,type,dcb address,area address, { length / 'S' } |

*decb name*—symbol

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

*type* — **{SF} {SB}**

The *type* operand is coded as shown to specify the type of read operation.

**SF** — Specifies normal, sequential forward, retrieval.

**SB** — Specifies a read backward operation; this operand can be specified only for magnetic tape with format-F or format-U records.

*dcb address*—A-Type Address or (2-12)

The *dcb address* operand specifies the address of the data control block for the opened data set to be read.

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

The *area address* operand specifies the address of the main-storage area into which the record is placed. When a READ SB macro instruction is issued, the area address must be the address of the last byte of the area into which the record is read. If the data set contains keys, the key is read into the buffer followed by the data.

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

The *length* operand specifies the number of data bytes to be read, to a maximum of 32,760. (If the data is translated from ASCII code to EBCDIC code, the maximum number of bytes that can be read is 2048.) A number can be coded only for format-U records. The number of bytes to be read is taken from the data control block if **'S'** is coded instead of a number. (This operand is ignored for format-F or format-V records.) For format-D records only, the length of the record just read is automatically inserted into the DCBLRECL field if BUFOFF=(L) is not specified in the data control block.

# READ—Read a Block (Offset Read of Keyed BDAM Data Set Using BSAM)

The READ macro instruction causes a block to be retrieved from a data set and placed in a designated area of main storage. The data set is a BDAM data set and its record format is unblocked variable-length spanned records. BFTEK=R must be specified in the data control block. Control may be returned to the problem program before the block is retrieved. The input operation must be tested for completion using a CHECK macro instruction. A data event control block, shown in Appendix A, is constructed as part of the macro expansion.

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

| | | |
|---|---|---|
| [symbol] | READ | decb name,type,dcb address,area address |

*decb name*—symbol

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

*type*—**SF**

The *type* operand is coded as shown to specify the type of read operation.

**SF** Specifies normal, sequential, forward operation.

*dcb address*—A-Type Address or (2-12)

The *dcb address* operand specifies the address of the data control block for the opened BDAM data set to be read.

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

The *area address* operand specifies the address of the main-storage area into which the record is placed.

When a spanned BDAM data set is created with keys, only the first segment of a record has a key; successive segments do not. When a spanned record is retrieved by the READ macro instruction, the system places a segment in a designated area addressed by the *area address* operand. The problem program must assemble all the segments into a logical record. Since 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 the same relative position.

# READ—Read a Record (BISAM)

The READ macro instruction causes a block containing a specified logical record to be retrieved from a data set. The block is placed in a designated area of main 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.

Control may be returned to the problem program before the block is retrieved. The input operation must be tested for completion using a WAIT or CHECK macro instruction.

The standard form of the READ macro instruction is written as follows for BISAM (the list and execute forms are shown following the descriptions of the standard form):

| | | |
|---|---|---|
| [symbol] | READ | decb name,type,dcb address,{area address / 'S'},<br>{length / 'S'},key address |

*decb name*—symbol

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

*type* — **{K} {KU}**

The *type* operand is coded as shown to specify the type of read operation.

**K** Specifies normal retrieval.

**KU** Specifies that the record retrieved is to be updated and returned to the data set; the system saves the device address to be returned.

When an ISAM data set is being updated with a READ KU macro instruction and a WRITE K macro instruction, both the READ and WRITE macro instructions must refer to 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 refer to the same list.

*dcb address*—A-Type Address or (2-12)

The *dcb address* operand specifies the address of the data control block for the opened data set to be read.

*area address*—A-Type Address, (2-12), or **'S'**

The *area address* operand specifies the address of the main-storage area into which the data block is placed. The first sixteen bytes of this area are used by the system and do not contain information from the data block. Dynamic buffering is specified by coding **'S'** instead of an address; the address of the acquired main-storage area is returned in the data event control block. Indexed sequential buffer and work area requirements are described in *OS Data Management Services Guide.*

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

The *length* operand specifies the number of bytes to be read up to a maximum of 32,760. 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)

The *key address* operand specifies the address of a main-storage area containing the key of a logical record in the block that is to be retrieved. When the input operation is completed, the main-storage address of the logical record is placed in the data event control block.

# READ—List Form

The list form of the READ macro instruction is used to construct a data management parameter list in the form of a data event control block (DECB). Refer to Appendix A for a description of the various fields of the DECB for each access method.

The description of the standard form of the READ macro instruction provides the explanation of the function of each operand. The description of the standard form also indicates the operands used for each access method, as well as 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 instruction is written as follows:

| | | |
|---|---|---|
| [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 or **'S'**

*area address* — A-Type Address or **'S'**

*length* — symbol, decimal digit, absexp, or **'S'**

*key address* — A-Type Address or **'S'**

*block address* — A-Type Address

*next address* — A-Type Address

**MF=L** — Coded as shown

The **MF=L** operand specifies that the READ macro instruction is used to create a data event control block that can be referenced by an execute-form instruction.

# READ—Execute Form

A remote data management parameter list (data event control block) is used in, and can be modified by, the execute form of the READ macro instruction. The data event control block can be generated by the list form of either a READ or WRITE macro instruction.

The description of the standard form of the READ macro instruction provides the explanation of the function of each operand. The description of the standard form also indicates the operands used for each access method, as well as 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 instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | READ | decb address,type,[dcb address],[area address / 'S'], [length / 'S'],[key address / 'S'],[block address],[next address] ,MF=E |

*decb address* — RX-Type Address or (2-12)

*type* — Code one of the types shown in the standard form

*dcb address* — RX-Type Address or (2-12)

*area address* — RX-Type Address, (2-12), or **'S'**

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

*key address* — RX-Type Address, (2-12), or **'S'**

*block address* — RX-Type Address, or (2-12)

*next address* — RX-Type Address or (2-12)

**MF=E** — Coded as shown

The **MF=E** operand specifies that the execute form of the READ macro instruction is used, and that an existing data event control block (specified in the *decb address* operand) is used by the access method.

# RELEX—Release Exclusive Control (BDAM)

The RELEX macro instruction causes release of a data block from exclusive control. The block must have been requested in an earlier READ macro instruction which specified either **DIX** and **DKX.** The RELEX macro instruction must be issued by the same task that opened the data set.

**Note:** A WRITE macro instruction which specifies either **DIX** or **DKX** can also be used to release exclusive control.

The RELEX macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | RELEX | D,dcb address,block address |

**D**—Specifies direct access.

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

The *dcb address* operand specifies the address of the data control block for a BDAM data set opened for processing. The *dcb address* operand must specify the same data control block as designated in the associated READ macro instruction.

*block address*—RX-Type Address, (2-12), or (0)

The *block address* operand specifies the address of the main-storage area containing the relative block address, relative track address, or actual device address of the data block being released. The *block address* operand must specify the same main-storage area as 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 zero.

| Hexadecimal Code | Meaning |
|---|---|
| 00 | Operation completed successfully. |
| 04 | The specified data block list. |
| 08 | The relative track address, relative block address, or actual device address was not within the data set. |

# RELSE—Release an Input Buffer (QISAM and QSAM Input)

The RELSE macro instruction causes immediate release of the current input buffer. The next GET macro instruction retrieves the first record from the next input buffer. For variable-length spanned records (QSAM), the input data set is spaced to the next segment which starts a logical record in a subsequent block. Thus, one or more blocks of data or records may be skipped. The RELSE macro instruction is ignored if a buffer has just been completed or released, or if the records are unblocked.

The RELSE macro instruction is written as follows:

| [symbol] | RELSE | dcb address |
|---|---|---|

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

The *dcb address* operand specifies the address of the data control block for the opened input data set.

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

The SETL macro instruction causes the control program to start processing the next input request at the specified record or device address. Sequential retrieval of records using the GET macro instruction continues from that point until the end of the data set is encountered or a CLOSE or ESETL macro instruction is issued. An ESETL macro instruction must be issued between SETL macro instructions that specify the same data set.

The SETL macro instruction can specify that retrieval is to start at the beginning of the data set, at a specific address on the device, at a specific record, or at the first record of a specific class of records. For additional information on SETL functions, see *OS Data Management Services Guide.*

The SETL macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | SETL | dcb address, { K[H], lower limit address<br>KC, lower limit address<br>KD[H], lower limit address<br>KCD, lower limit address<br>I, lower limit address<br>ID, lower limit address<br>B<br>BD } |

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

The *dcb address* operand specifies the address of the data control block opened for the indexed sequential data set being processed.

The following operands are coded as shown, and they specify the starting point and type of retrieval.

**K** — Specifies that the next input operation begins at the record containing the key specified in the *lower-limit address* operand.

**KC** — Specifies that the next input operation begins at the first record of the key class specified in the *lower-limit address* operand. If the first record of the specified key class has been deleted, retrieval begins at the next nondeleted record regardless of key class.

**H** — This option, 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 begins 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 begins 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 begins with the record at the actual device address specified in the *lower-limit address* operand.

**ID** — Specifies that the next input operation begins 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 begins with the first record in the data set.

**BD** — Specifies that the next input operation begins 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)

The *lower-limit address* operand specifies the address of the main-storage area containing the key, key class, or actual device address that designates the starting point for the next input operation. If I or ID has been specified, this area must contain the actual device address (in the form MBBCCHHR) of a prime data record; the other types require that the key or key class be contained in this area.

## *SETL Exit*

The error analysis (SYNAD) routine is given control if the operation could not be completed successfully. The exceptional condition code and general registers are set as shown in Appendix A. If the SETL macro instruction is not reissued, retrieval starts at the beginning of the data set.

# SETPRT—Load UCS and FCB Images (BSAM and QSAM)

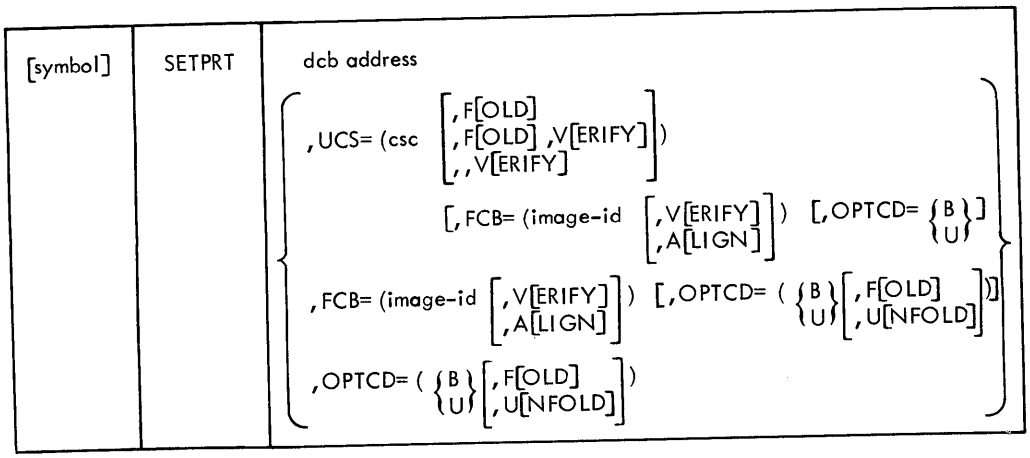
The SETPRT macro instruction is used with printers that have a universal character set (UCS) buffer or a forms control buffer (FCB). When a SETPRT macro instruction is issued, UCS and FCB images are fetched from the image library and loaded from main storage into their respective buffers. The SETPRT macro instruction is also used to block or unblock printer data checks.

IBM character sets and forms control images are included in the image library at system generation; user-defined character sets and forms control images can be added to the image library as described in *OS Data Management for System Programmers* publication. The FCB image can also be defined in the problem program using the exit list (EXLST) parameter of the DCB macro instruction.

When BSAM is being used, all write operations must be checked for completion before the SETPRT macro instruction is issued; any incomplete write operations are purged. Issuing the SETPRT macro instruction for a device other than an online UCS printer results in a NOP instruction.

**Note:** Do not use SETPRT with EXCP programming.

The standard form of the SETPRT macro instruction is written as follows (the list and execute form are shown following the standard form):




*dcb address*—A-Type Address or (2-12)

The *dcb address* operand specifies the address of the data control block for the data set to be printed; the data set must be opened for output before the SETPRT macro instruction is issued.

**UCS=**—A character code with options

The **UCS** operand specifies that the UCS buffer is to be loaded from the image library. When the **UCS** operand is specified, the **FCB** and **OPTCD** operands can also be specified.

*csc* (character set code)

The *csc* operand specifies the character set to be loaded. A character set is identified by a 1-4 character code. Codes for standard IBM character sets are as follows:

*1403 Printer:* **AN, HN, PCAN, PCHN, PN, QN, QNC, RN, SN, TN, XN,** and **YN**

*3211 Printer* **A11, H11, G11, P11,** and **T11**

For descriptions of the standard IBM character sets, refer to *OS Operator's Procedures;* codes for user-designed character sets are defined by the installation.

**F** or **FOLD**
Specifies that the character set image is to be loaded in the fold mode. The fold mode is most often used when the EBCDIC code for lowercase alphabetic characters is printed as uppercase characters by a print train with lowercase type.

**V** or **VERIFY**
Requests that the character set image be displayed on the printer for visual verification.

**FCB=**—A character code with options

The **FCB** operand specifies that the forms control buffer (FCB) is to be loaded from the image library. When the **FCB** operand is specified, the **OPTCD** operand can also be specified.

*image id*

The *image id* operand specifies the forms control image to be loaded. A forms control image is identified by a 1-4 character code. IBM-supplied images are identified by *image id* value of STD1 and STD2; user-designed forms control images are defined by the installation. For descriptions of the standard forms control images, refer to the *OS System Generation* publication.

**V** or **VERIFY**
Requests that the forms control image be displayed on the printer for visual verification. This operand allows forms alignment using the WTOR macro instruction.

**A** or **ALIGN**
Allows forms alignment using the WTOR macro instruction.

**OPTCD=**—A printer option code

The **OPTCD** operand specifies whether UCS printer data checks are blocked or unblocked and if the printer is to operate in fold or normal mode.

**B**
Specifies that UCS printer data checks are blocked; this option updates the DCBOPTCD field of the data control block.

**U**
Specifies that UCS printer data checks are unblocked; this option updates the DCBOPTCD field of the data control block.

**F** or **FOLD**
Specifies that printing is in fold mode.

U or **UNFOLD**
Specifies that printing is in normal mode; this operand causes fold mode to revert to normal mode.

## *Completion 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. Bits 16-23 indicate the result of the attempt to load the forms control buffer (FCB). Bits 24-31 indicate the result of the attempt to load a universal-character-set (UCS) buffer. For completion codes 18, 1C, 20, and 24, bits 24-31 apply to both FCB and UCS loading. The codes in the following table are in hexadecimal.

| Bits 16-23 FCB Code | Bits 24-31 UCS Code | Meaning |
|---|---|---|
| 00 | 00 | Successful completion. |
| 04 | 04 | The operator canceled the load because either the image could not be found in the image library or, in the case of the UCS image, the requested chain or train was not available. |
| 08 | 08 | A permanent I/O error was detected when the BLDL macro instruction was issued to locate the image in the image library. |
| 0C | 0C | A permanent I/O error persisted after two attempts were made to load the FCB/UCS buffer. |
| 10 | 10 | A permanent I/O error was detected when an attempt was made to display the character set image or forms control image on the printer for visual verification. |
| 14 | 14 | The operator canceled the load because the wrong image was displayed for visual verification. |

| Bits 24-31 FCB and UCS Code | Meaning |
|---|---|
| 18 | No operation was performed for one of the following reasons:<br>• The data control block was not open.<br>• The data control block was not valid for a sequential data set.<br>• The SETPRT parameter list was not valid.<br>• The output device was not a UCS printer.<br>• SETPRT was used with EXCP programming. |
| 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. |

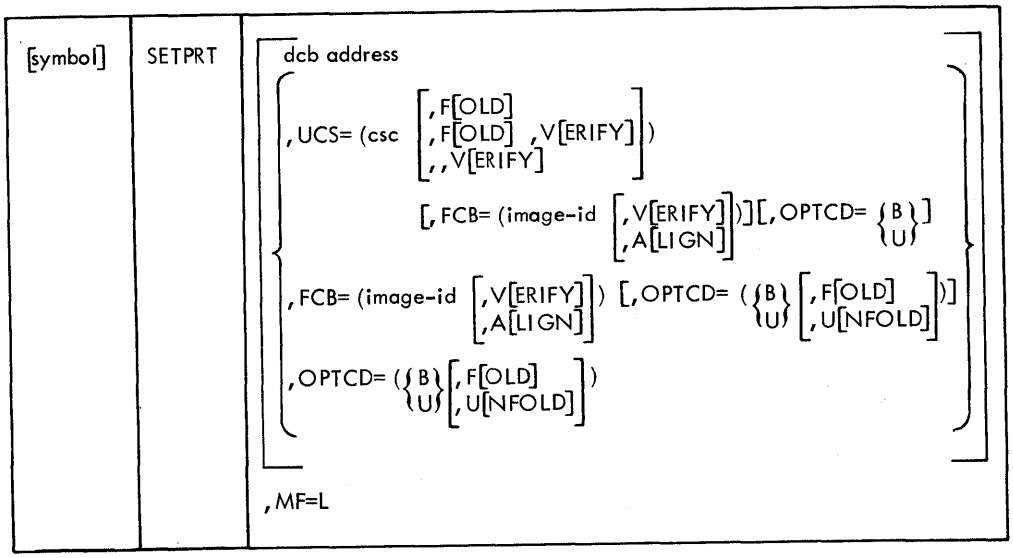
| | |
|---|---|
| | No operation was performed because an uncorrectable error occurred when the Block Data Check or Reset Block Data Check command was issued by SETPRT. |
| 20 | Not enough space has been provided for the IMAGELIB control blocks. Increase the amount of space allocated for the job step. |
| 24 | SYS1.IMAGELIB cannot be opened to load the specified UCS/FCB image. |

# 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 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 for the list form only.

The list form of the SETPRT macro instruction is written as follows:




*dcb address*—A-Type Address

**UCS=**—A character code with options

It is coded as described in the standard form of the macro instruction.

**FCB=**—A character code with options

It is coded as described in the standard form of the macro instruction.

**OPTCD=**—A printer option code

It is coded as described in the standard form of the macro instruction.

**MF=L**

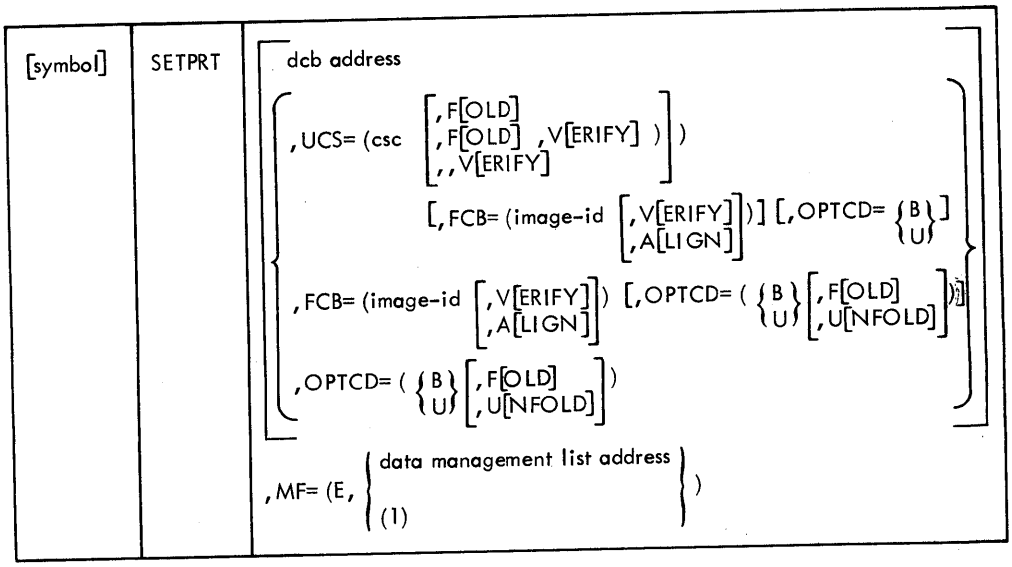
The **MF=L** operand specifies that the list form of the macro instruction is used to create a parameter list that can be referenced by an execute form of the SETPRT macro instruction.

# 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 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 for the execute form only.

The execute form of the SETPRT macro instruction is written as follows:




*dcb address*—RX-Type Address or (2-12)

**UCS=**—A character code with options

It is coded as shown in the standard form of the macro instruction.

**FCB=**—A character code with options

It is coded as shown in the standard form of the macro instruction.

**OPTCD=**—A printer option code

It is coded as shown in the standard form of the macro instruction.

**MF=(E,** { *data management list address* **}**
{**(1)** }

This operand specifies that the execute form of the SETPRT macro instruction is used, and an existing data management parameter list is used.

**E**—Coded as shown

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

# STOW—Update Partitioned Data Set Directory (BPAM)

The STOW macro instruction causes the system to update a partitioned data set directory by adding, changing, replacing, or deleting an entry in the directory. Only one entry can be updated at a time using the STOW macro instruction. If the entry to be added or replaced is a member name, the system writes an end-of-data indication following the member. All input and output operations using the same data control block must have previously been tested for completion.

The STOW macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | STOW | dcb address, list address [,directory action] |

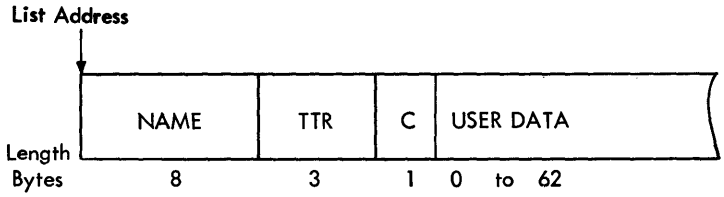
*dcb address*—RX-Type Address, (2-12), or (1)

The *dcb address* operand specifies the address of the data control block for the opened partitioned data set. The STOW macro instruction can be used only when the data set is opened for OUTPUT, UPDAT or OUTIN (BSAM).

*list address*—RX-Type Address, (2-12), or (0)

The *list address* operand specifies the address of the main-storage 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 on which the beginning of the data set is located.

**R**

Specifies the relative block (record) number on the track identified by **TT**.

**Note:** The TTR field shown above must be supplied by the problem program if an alias (alias bit is 1) is being added or replaced. The system supplies the TTR field when a member name is being added or replaced.

**C**

Specifies the type of entry (member or alias) for the name, the number of note list fields (TTRNs), and the length in halfwords, of the user data field. The following describes the meaning of the eight bits.

| | | |
|---|---|---|
| Bit 0=0 | — | Indicates a member name. |
| Bit 0=1 | — | Indicates an alias. |
| Bits 1 and 2 | — | Indicate the number of TTRN fields (maximum of three) in the user data field. |
| Bits 3-7 | — | Indicate the total number of halfwords in the user data field. |

**Deleting a Directory Entry:** The *list address* operand must specify an 8-byte area that contains the member name or alias to be deleted. The name must begin in the first byte of the area and be padded on the right with blanks, if necessary, to complete the eight 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]**

The *directory action* operand is coded as shown to specify the type of directory action. If the operand is not coded, **A** (add an entry) is assumed.

**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 one of the following return codes in the low-order byte; the three high-order bytes of register 15 are set to zero.

| Code (Hexa-decimal) | Directory Action A | R | D | C |
|---|---|---|---|---|
| 00 | The update of the directory was completed successfully | | | |
| 04 | The directory already contains the specified name. | — | — | The directory already contains the specified new name. |
| 08 | — | The specified name could not be found. | | The specified old name could not be found. |
| 0C | No space left in the directory. The entry could not be added or replaced. | | — | — |
| 10 | A permanent input or output error was detected when attempting to update the directory. Control is not given to the error analysis (SYNAD) routine. | | | |
| 14 | The specified data control block is not open or is opened incorrectly. | | | |
| 18 | Conditional GETMAIN with STOW macro instruction was unsuccessful. | | | |

# 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 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 *OS Data Management for System Programmers.)*

The SYNADAF macro instruction uses register 1 to return the address of a buffer containing a message. The message describes the error, and can be printed by a subsequent PUT or WRITE macro instruction. The message consists of EBCDIC information and is in the form of a variable-length record. The format of the message is shown following the descriptions of the SYNADAF operands.

The system does not use the save area whose address is in register 13. Instead, it provides a save area for its own use, and then makes this area available to the error analysis routine. The system returns the address of the new save area in register 13 and in the appropriate location (word 3) of the previous save area; it also stores the address of the previous save area in the appropriate location (word 2) of the new save area.

The SYNADAF macro instruction passes parameters to the system in registers 0 and 1. When used in a SYNAD routine, it passes the parameters that are in these registers when the routine is entered, and it should therefore be coded at the entry point of the routine. (Refer to Appendix A, Figures 2 and 3.) To save these parameters for use by the SYNAD routine, the system stores them in a parameter save area that follows the message buffer as shown in the message buffer format. The system does not alter the return address in register 14 or the entry point address in register 15.

When a SYNADAF macro instruction is used, a SYNADRLS macro instruction must be used to release the message buffer and save areas, and to restore the original contents of register 13.

The SYNADAF macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | SYNADAF | { ACSMETH=BDAM<br>ACSMETH=BPAM<br>ACSMETH=BSAM<br>ACSMETH=QSAM<br>ACSMETH=BISAM } [,PARM1=parm register] [,PARM2=parm register]<br>{ ACSMETH=EXCP[,PARM1=iob address]<br>ACSMETH=QISAM[,PARM1=dcb address] [,PARM2=parm register] } |

**ACSMETH=BDAM, BPAM, BSAM, QSAM, BISAM, EXCP, or QISAM**

The **ACSMETH** operand specifies the access method used to perform the input/output operation for which error analysis is performed.

**PARM1**=*parm register, iob address,* or *dcb address*—(2-12) or (1)

The **PARM1** operand specifies the address of information that is dependent on the access method being used. For BDAM, BISAM, 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 QISAM, the operand specifies the address of the data control block; for EXCP, it specifies the address of the input/output block. If the operand is omitted, **PARM1=(1)** is assumed.

**PARM2**=*parm register*—(2-12), (0), or RX-Type Address (only if **ACSMETH=QISAM)**

The **PARM2** operand specifies the address of additional information that is dependent on the access method being used. For BDAM, BISAM, BPAM, BSAM, QISAM, and QSAM, the operand specifies a register that contains the information that was in register 0 on entry to the SYNAD routine. For EXCP, the operand is meaningless and should be omitted. If the operand is omitted, except in the case of EXCP, **PARM2=(0)** is assumed.

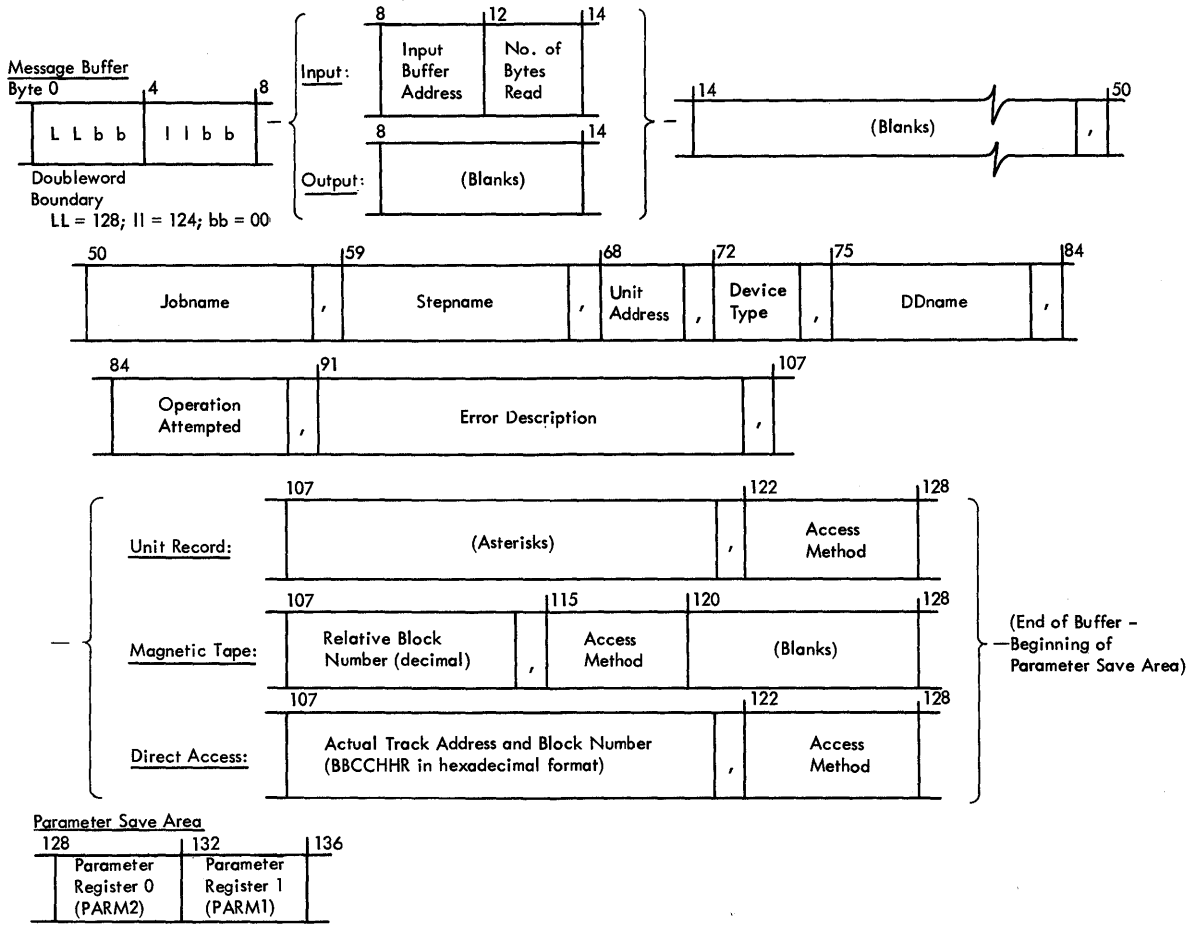
## *Completion Codes*

When the system returns control to the problem program, the low-order byte of register 0 contains one of the following return codes; the three high-order bytes of register 0 are set to zero.

| Hexadecimal Code | Meaning |
|---|---|
| 00 | Successful completion. Bytes 8-13 of the message buffer contain blanks. |
| 04 | Successful completion. Bytes 8-13 of the message buffer contain binary data. |
| 08 | Unsuccessful completion. The message can be printed, but some information is missing in bytes 50-127 and is represented by asterisks. Bytes 8-13 contain either blanks or 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

- The device type field (bytes 72-73) contains UR for a unit record device, TA for a magnetic tape device, or DA for a direct-access device.
- If a message field (bytes 91-105) is not applicable to the type of error that occurred, it contains N/A or NOT APPLICABLE.
- If no data was transmitted, or if the access method is QISAM, bytes 8-13 contain blanks.
- If the access method is BISAM, bytes 68-70, 84-89, and 107-120 contain asterisks.
- If the access method is BDAM, and if the error was an invalid request, bytes 107-120 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 zero. Thus, when control is returned, the save area pointers are the same as before the SYNADAF macro instruction was issued.

The SYNADRLS macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | SYNADRLS | |

When the system returns control to the problem program, the low-order byte of register 0 contains one of the following return codes; the three high-order bytes of register 0 are set to zero.

| Hexadecimal Code | Meaning |
|---|---|
| 00 | Successful completion. |
| 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. |

# 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 being truncated and 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), 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 instruction is written as follows:

| [symbol] | TRUNC | dcb address |
|---|---|---|

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

The *dcb address* operand specifies the address of the data control block for the sequential data set opened for output. The record format in the data control block must not indicate standard blocked records (RECFM=FBS).

# WAIT—Wait for One or More Events (BDAM, BISAM, BPAM, and BSAM)

The WAIT macro instruction is used to inform the control program that performance of the active task cannot continue until one or more specific events, each represented by a different ECB (event control block), have occurred. Bit 0 of each ECB must be set to 0 before it is used. The control program takes the following action:

- For each event that has already occurred (each ECB is already posted), the count of the number of events is decreased by 1.
- If the number of events is 0 by the time the last event control block is checked, control is returned to the instruction following the WAIT macro instruction.
- If the number of events is not 0 by the time the last ECB is checked, control is not returned to the issuing program until sufficient ECBs are posted to bring the number to 0. Control is then returned to the instruction following the WAIT macro instruction.

A full description of the event control block is presented in the publication *IBM System/360 Operating System: System Control Blocks.*

The WAIT macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | WAIT | [number of events,] {ECB=address / ECBLIST=address} |

*number of events*

The *number of events* operand is a decimal integer from 0 to 255. Zero is an effective NOP instruction; one is assumed if the operand is omitted. The number of events must not exceed the number of event control blocks.

**ECB=**

The **ECB** operand is the address of the event control block representing the single event that must occur before processing can continue. The operand is valid only if the number of events is specified as one or is omitted.

**ECBLIST=**

The **ECBLIST** operand is the address of a main-storage area containing one or more consecutive fullwords on a fullword boundary. Each fullword contains the *address* of an event control block; 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.

**Caution:** A job step with all of its tasks in a WAIT condition is terminated upon expiration of the time limits that apply to it.

**Example:** You have previously initiated one or more activities to be completed asynchronously to your processing. As each activity was initiated, you set up an ECB in which bits 0 and 1 were set to 0. You now wish to suspend your task via the WAIT macro instruction until a specified number of these activities has been completed.

Completion of each activity must be made known to the system via the POST macro instruction. POST causes an addressed ECB to be marked complete. If completion of the event satisfies the requirements of an outstanding WAIT, the waiting task is marked ready and will be executed when its priority allows.

# WRITE—Write a Block (BDAM)

The WRITE macro instruction causes the system to add or replace a block in an existing direct data set. (This version of the WRITE macro instruction cannot be used to create a direct data set because no capacity record facilities are provided.) Control may be returned before the block is written. The output operation must be tested for completion using a CHECK or WAIT macro instruction. A data event control block, shown in Appendix A is constructed as part of the macro expansion.

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

| | | |
|---|---|---|
| [symbol] | WRITE | decb name, type, dcb address, {area address / 'S'}, {length / 'S'}, {key address / 'S' / 0}, block address |

*decb name*—symbol

The *decb name* operand specifies the name assigned to the data event control block created as part of the macro expansion.

*type* — **{DA [F]}**
**{DI [F]}**
**[X]**
**{DK [F]}**
**[X]**

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 data block is to be added to the data set in the first available space; the search for available space starts at the device address indicated in the area specified in the *block address* operand. The description of the DCB macro instruction, LIMCT operand, contains a description of the search.

**DI**

Specifies that a data block and key, if any, are to be written at the device address indicated in the area specified in the *block address* operand. Any attempt to write a capacity record (R0) is an invalid request when relative track addressing or actual device addressing are used, but when relative block addressing is used, relative block 0 is the first data block in the data set.

**DK**

Specifies that a data block (only) is to be written using the key in the area specified by the *key address* operand as a search argument; the search for the block starts at the device address indicated in the area specified in the *block address* operand. The description of the DCB macro instruction, LIMCT operand, contains a description of the search.

**F**

Requests that the system provide block position feedback into the area specified in the *block address* operand. This character can be coded as a suffix to **DA, DI,** or **DK** as shown above.

**X**

Requests that the system release the exclusive control requested by a previous READ macro instruction and provide block position feedback into the area specified in the *block address* operand. This character can be coded as a suffix to **DI** or **DK** as shown above.

*dcb address*—A-Type Address or (2-12)

The *dcb address* operand specifies the address of the data control block for the opened BDAM data set.

*area address*—A-Type Address, (2-12), or **'S'**

The *area address* operand specifies the address of a main-storage 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. See Appendix A for a description of the data event control block.

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

The *length* operand specifies the number of data bytes to be written up to a maximum of 32,760. If **'S'** is coded, it specifies that the system uses the value in the blocksize (DCBBLKSI) field as the length. When undefined-length records are used, if the WRITE macro instruction is for update and the length specified differs from the original block, the new block will be truncated or padded with binary zeros accordingly. The problem program can check for this situation in the SYNAD routine.

If the *length* operand is omitted for format-U records, no error indication is given when the program is assembled, but the problem program must insert a length into the data event control block before the WRITE macro instruction is executed.

*key address*—A-Type Address, (2-12), **'S'**, or 0

The *key address* operand specifies the address of the main-storage area that contains the key to be used. **'S'** is specified instead of an address only if the key is contained in an area acquired by dynamic buffering. If the key is not written or used as a search argument, zero is specified instead of a key address.

*block address*—A-Type Address or (2-12)

The *block address* operand specifies the address of a main-storage area that contains the relative block address, relative track address, or actual device address used in the output operation. The length of the area depends on the type of addressing used and if the feedback option (OPTCD=F) is specified in the data control block.

If OPTCD=F has been specified, feedback, when requested, is in the same form as was originally presented by the WRITE macro instruction; the area is either three or eight bytes long depending on the type of addressing.

If OPTCD=F has not been specified, feedback, when requested, is in the form of an 8-byte actual device address (MBBCCHHR); the area must be eight bytes.

# WRITE—Write a Block (BPAM and BSAM)

The WRITE macro instruction causes the system to add or replace a block in a sequential or partitioned data set being created or updated. Control may be returned to the problem program before the block is written. The output operation must be tested for completion using the CHECK macro instruction. A data event control block, shown in Appendix A, is constructed as part of the macro expansion.

If translation from EBCDIC code to ASCII code is requested, issuing multiple WRITE macro instructions for the same record causes an error because the first WRITE macro instruction issued causes the output data in the output buffer to be translated into ASCII code.

If the OPEN macro instruction specifies UPDAT, both the READ and WRITE macro instructions must refer to the same data event control block. Refer to the list form of the READ or WRITE macro instruction for a description of how to construct a data event control block; refer to 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,type,dcb address,area address [,length] [,'S'] |

*decb name*—symbol

The *decb name* operand specifies the name assigned to the data event control block created as part of the macro expansion.

*type*—**SF**

This operand is coded as shown to specify the type of Write operation.

**SF** Specifies normal, sequential forward operation.

*dcb address*—A-Type Address, or (2-12)

The *dcb address* operand specifies the address of the data control block for the opened data set 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)

The *area address* operand specifies the address of a main-storage 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'**

The *length* operand specifies the number of bytes to be written; this operand is specified for only undefined-length records (RECFM=U) or ASCII records (RECFM=D) when the DCB BUFOFF operand is zero. If the data is to be translated from EBCDIC code to ASCII code, the maximum length is 2048; otherwise, the maximum length is 32,760 bytes. **'S'** can be coded to indicate that the value specified in the blocksize (DCBBLKSI) field of the data control block is used as the length to be written. The *length* operand should be omitted for all record formats except format-U and format-D (when BUFOFF=0).

If the *length* operand is omitted for format-U or format-D (with BUFOFF=0) records, no error indication is given when the program is assembled, but the problem program must insert a length into the data event control block before the data set is opened.

# WRITE—Write a Block (Create a BDAM Data Set with BSAM)

The WRITE macro instruction causes the system to add a block to the direct data set being created. For fixed-length blocks, the system writes the capacity record automatically when the current track is filled; for variable- and unspecified-length blocks, a WRITE macro instruction must be issued for the capacity record. Control may be returned before the block is written. The output operation must be tested for completion using a CHECK macro instruction. A data event control block, shown in Appendix A, is constructed as part of the macro expansion.

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

| | | |
|---|---|---|
| [symbol] | WRITE | decb name,type,dcb address,area address[,length[,'S']] [,next address] |

*decb name*—symbol

The *decb name* operand specifies the name assigned to the data event control block created as part of the macro expansion.

*type* — {**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).

**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)

The *dcb address* operand specifies the address of the data control block opened for the data set being created. DSORG=PS (or PSU) and MACRF=WL must be specified in the DCB macro instruction to create a BDAM data set.

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

The *area address* operand specifies the address of the main-storage 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 one 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 operand must be within the main-storage area allocated to the program writing the dummy blocks.

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

The *length* operand 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 32,760. If 'S' is coded, it specifies that the system is to use the length in the blocksize (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 data set is opened.

*next address*—A-Type Address or (2-12)

The *next address* operand specifies the address of a main-storage 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*

After the write has been scheduled and control has been returned to the user's program, the three high-order bytes of register 15 are set to zero; the low-order byte contains one of the following return codes:

| Code | Meaning | | |
|---|---|---|---|
| | Fixed-Length (**SF** or **SD**) | Variable- or Unspecified-Length (**SF** or **SFR**) | (**SZ**) |
| 00 | Block will be written. (If the previous return code was 08, a block is written only if the DD statement specifies secondary space allocation and sufficient space is available. | | Capacity record was written; another track is available. |
| 04 | Block will be written, followed by a capacity record. | Block was not written; write a capacity record record (**SZ**) to complete the current track, then reissue. | |
| 08 | Block will be written, followed by capacity record. The next block requires secondary space allocation. | | Capacity record was written. The next block requires secondary space allocation. This code is not issued if the WRITE SZ is the only WRITE macro instruction issued on a one-track secondary extent. |
| 0C | Block will not be written; issue a CHECK macro instruction for the previous WRITE macro instruction, then reissue the WRITE macro instruction. | | |

# WRITE—Write a Logical Record or Block of Records (BISAM)

The WRITE macro instruction causes the system to add or replace a record or replace an updated block in an existing indexed sequential data set. Control may be returned to the problem program before the block or record is written. The output operation must be tested for completion using a WAIT or CHECK macro instruction. A data event control block, shown in Appendix A, is constructed as part of the macro expansion.

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

| | | |
|---|---|---|
| [symbol] | WRITE | decb name, type, dcb address, {area address / 'S'}, {length / 'S'}, key address |

*decb name*—symbol

The *decb name* operand specifies the name assigned to the data event control block created as part of the macro expansion.

*type* — **{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 read using a READ KU macro instruction 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 refer to the same DECB.

*dcb address*—A-Type Address or (2-12)

The *dcb address* operand 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'**

The *area address* operand specifies the address of the main-storage area containing the record to be written. The first sixteen bytes of this area are used by the system and should not contain your data. When new records are written, the area address of the new record must always be supplied by the problem program. This area may be altered by the system. **'S'** may be coded instead of an address only if the record 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.

Indexed sequential buffer and work area requirements are discussed in *OS Data Management Services Guide.*

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

The *length* operand specifies the number of data bytes to be written, to a maximum of 32,760. If the length is already known to the system (if new records are being added or a block containing an updated record is written), specify **'S'** instead of a length.

*key address*—A-Type Address or (2-12)

The *key address* operand specifies the address of a main-storage area containing the key of the new or updated record. 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, this area may be altered by the system.

# WRITE—List Form

The list form of the WRITE macro instruction is used to construct a data management parameter list in the form of a data event control block (DECB). Refer to Appendix A for a description of the various fields in the DECB for each access method.

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 as well as 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 WRITE macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | WRITE | decb name,type,[dcb address],[area address\|'S'],<br>[length\|'S'],[key address\|'S'],[block address],[next address]<br>,MF=L |

*decb name* — symbol

*type* — Code one of the types shown in the standard form

*dcb address* — A-Type Address

*area address* — A-Type Address or **'S'**

*length* — symbol, decimal digit, absexp, or **'S'**

*key address* — A-Type Address or **'S'**

*block address* — A-Type Address

*next address* — A-Type Address

**MF=L** — Coded as shown

The **MF=L** operand specifies that the WRITE macro instruction is used to create a data event control block that will be referenced by an execute-form instruction.

# WRITE—Execute Form

A remote data management parameter list (data event control block) is used in, and can be modified by, the execute form of the WRITE macro instruction. The data event control block can be generated by the list form of either a READ or WRITE macro instruction.

The description of the standard form of the WRITE macro instruction provides the explanation of the function of each operand. The description of the standard form also indicates the operands used for each access method, as well as 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 WRITE macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | WRITE | decb address,type,[dcb address],[area address\|'S'], [length\|'S'],[key address\|'S'],[block address],[next address] ,MF=E |

*decb address* — RX-Type Address or (2-12)

*type* — Code one of the types shown in the standard form

*dcb address* — RX-Type Address or (2-12)

*area address* — RX-Type Address, (2-12), or **'S'**

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

*key address* — RX-Type Address, (2-12), or **'S'**

*block address* —RX-Type Address or (2-12)

*next address* — RX-Type Address or (2-12)

**MF=E** — Coded as shown

The **MF=E** operand 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 ASCII (BSAM and QSAM)

The XLATE macro instruction is used to translate the data in an area in main storage from ASCII code to EBCDIC code or from EBCDIC code to ASCII code.

The XLATE macro instruction is written as follows:

| | | |
|---|---|---|
| [symbol] | XLATE | area address, length[,TO={A \| E}] |

*area address*—RX-Type Address, symbol, decimal digit, *absexp,* or (2-12)

The *area address* operand specifies the address of the main-storage area that is to be translated.

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

The *length* operand specifies the number of bytes to be translated.

**TO = {A} {E}**

The **TO** operand specifies the type of translation that is requested. The following describes the characters that can be specified. If this operand is omitted, **E** is assumed.

**A** Specifies that translation from EBCDIC code to ASCII code is requested.

**E** Specifies that translation from ASCII code to EBCDIC code is requested.

# APPENDIX A: STATUS INFORMATION FOLLOWING AN INPUT/OUTPUT OPERATION

Following an input/output operation, the control program makes certain status information available to the problem program. This information is a 2-byte exception code, or a 16-byte field of standard status indicators, or both.

Exception codes are provided in the data control block (QISAM), or in the data event control block (BISAM and BDAM). The data event control block is described below, and the exception code lies within the block as shown in the illustration for the data event control block. If a DCBD macro instruction is coded, the exception code in a data control block can be addressed as two 1-byte fields, DCBEXCD1 and DCBEXCD2. The exception codes can be interpreted by referring to Figures 1-3.

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). The contents of registers on entry to the SYNAD routine are shown in Figures 4-6; the status indicators are shown in Figure 7.

## The 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 | Status indicator address | Logical record address | Status indicator address |
| +20 | | Key address | Key address |
| +24 | | Exception code (2 bytes) | Block address |
| +28 | | | Next address |

[1] Exception codes are returned in the second and third bytes of the ECB by the control program.

## The Event Control Block

The event control block (ECB) is used by the control program to test for completion of the read or write operation. The type, length, data control block address, area address, key address, block address, and next address information is taken from the operands of the macro instruction for use by the control program. Exception codes are returned by the control program after the corresponding WAIT or CHECK macro instruction is issued, as indicated in Figure 1; for BDAM, BSAM, and BPAM the control program provides a pointer to status indicators shown in Figure 7.

| Exception Code Bit | READ | WRITE | Condition if On |
|---|---|---|---|
| 0 | X | Type K | Record not found |
| 1 | X | X | Record length check |
| 2 | | Type KN | Space not found |
| 3 | | Type K | Invalid request |
| 4 | X | X | Uncorrectable I/O error |
| 5 | X | X | Unreachable block |
| 6 | X | | Overflow record |
| 7 | | Type KN | Duplicate record |
| 8-15 | | | 8-15 reserved for control program use |

Figure 1. Exception Code Bits—BISAM

*Notes for Figure 1:*

**Record Not Found:** This condition is reported if the logical record with the specified key is not found in the data set, if the specified key is higher than the highest key in the highest level index, or if the record is not in either the prime area or the overflow area of the data set.

**Record Length Check:** This condition is reported, for READ and update WRITE macro instructions, if an overriding length is specified and (1) the record format is blocked, (2) the record format is unblocked but the overriding length is greater than the length known to the control program, or (3) the record is fixed length and the overriding length does not agree with the length known to the control program. This condition is reported for the add WRITE macro instruction if an overriding length is specified.

When blocked records are being updated, the control program must find the high key in the block in order to write the block. (The high key is not necessarily the same as the key supplied by the problem program.) The high key is needed for writing because the control unit for direct-access devices permits writing only if a search on equal is satisfied; this search can be satisfied only with the high key in the block. If the user were permitted to specify an overriding length shorter than the block length, the high key might not be read; then, a subsequent write request could not be satisfied. In addition, failure to write a high key during update would make a subsequent update impossible.

**Space Not Found in Which to Add a Record:** This condition is reported if no room exists in either the appropriate cylinder overflow area or the independent overflow area when a new record is to be added to the data set. The data set is not changed in any way in this situation.

**Invalid Request:** This condition is reported for either of two reasons. First, if byte 25 of the data event control block indicates that this request is an update WRITE macro instruction corresponding to a READ (for update) macro instruction, but the input/output block (IOB) for the READ is not found in the update queue. This condition could be caused by the problem program altering the contents of byte 25 of the data event control block. Second, if a READ or WRITE macro instruction specifies dynamic buffering (that is, **'S'** in the *area address* operand) but the DCBMACRF field of the data control block does not specify dynamic buffering.

**Uncorrectable Input/Output Error:** This condition is reported if the control program's error recovery procedures encounter an uncorrectable error in transferring data between main storage and secondary storage.

**Unreachable Block:** This condition is reported if an uncorrectable input/output error occurs while searching the indexes or following an overflow chain. It is also posted if the data field of an index record contains an improper address (that is, points to the wrong cylinder or track or is an invalid address).

**Overflow Record:** This condition is reported if the record just read is an overflow record. (See the section on direct retrieval and update of an indexed sequential data set in *OS Data Management Services Guide* for consideration during BISAM updating.)

**Duplicate Record Presented for Inclusion in the Data Set:** This condition is reported if the new record to be added has the same key as a record in the data set. However, if the delete option was specified and the record in the data set is marked for deletion, this condition is not reported. Instead the new record replaces the existing record.

If the record format is blocked and the relative key position is zero, the new record cannot replace an existing record that is of equal key and is marked for deletion.

| Exception Code | | Code Set by | | | | | |
|---|---|---|---|---|---|---|---|
| **Field** | **Bit** | **CLOSE** | **GET** | **PUT** | **PUTX** | **SETL** | **Condition if On** |
| DCBEXCD1 | 0 | | | | | Type K | Record Not Found |
| | 1 | | | | | Type I | Invalid actual address for lower limit |
| | 2 | | | X | | | Space not found in which to add a record |
| | 3 | | | | | X | Invalid request |
| | 4 | | X | | | | Uncorrectable input error |
| | 5 | X | | X | X | | Uncorrectable output error |
| | 6 | | X | | | X | Block could not be reached (input) |
| | 7 | X | X | | | | Block could not be reached (update) |
| DCBEXCD2 | 0 | | | X | | | Sequence check |
| | 1 | | | X | | | Duplicate record |
| | 2 | X | | | | | Data control block closed when error routine entered |
| | 3 | | X | | | | Overflow record[1] |
| | 4 | | | X | | | Incorrect record length |
| | 5-7 | | | | | | Reserved for future use |

[1]The SYNAD routine is entered only if bit 4, 5, 6, or 7 of DCBEXCD1 is also on.

Figure 2. Exception Code Bits—QISAM

*Notes for Figure 2:*

**Record Not Found:** This condition is reported if the logical record with the specified key is not found in the data set, if the specified key is higher than the highest key in the highest level index, or if the record is not in either the prime area or the overflow area of the data set.

**Invalid Actual Address for Lower Limit:** This condition is reported if the specified lower limit address is outside the space allocated to the data set.

**Space Not Found in Which to Add a Record:** This condition is reported if the space allocated to the data set is already filled. In the locate mode, a buffer segment address is not provided. In the move mode, data is not moved.

**Invalid Request:** This condition is reported if (1) the data set is already being referred to sequentially by the problem program, (2) the buffer cannot contain the key and the data, or (3) the specified type is not also specified in the DCBMACRF field of the data control block.

**Uncorrectable Input Error:** This condition is reported if the control program's error recovery procedures encounter an uncorrectable error when transferring a block from secondary storage to an input buffer. The buffer address is placed in register 1, and the SYNAD routine is given control when a GET macro instruction is issued for the first logical record.

**Uncorrectable Output Error:** This condition is reported if the control program's error recovery procedures encounter an uncorrectable error when transferring a block from an output buffer to secondary storage. If the error is encountered during closing of the

data control block, bit 2 of DCBEXCD2 is set to 1 and the SYNAD routine is given control immediately. Otherwise, control program action depends on whether load mode or scan mode is being used.

If a data set is being created (load mode), the SYNAD routine is given control when the next PUT or CLOSE macro instruction is issued. In the case of a failure to write a data block, register 1 contains the address of the output buffer, and register 0 contains the address of a work area containing the first 16 bytes of the IOB; for other errors, the contents of register 1 are meaningless. After appropriate analysis, the SYNAD routine should close the data set or end the job step. If records are to be subsequently added to the data set using the queued indexed sequential access method (QISAM), the job step should be terminated by issuing an ABEND macro instruction. (ABEND closes all open data sets. However, an ISAM data set is only partially closed, and it can be reopened in a later job to add additional records by using QISAM). Subsequent execution of a PUT macro instruction would cause reentry to the SYNAD routine, since an attempt to continue loading the data set would produce unpredictable results.

If a data set is being processed (scan mode), the address of the output buffer in error is placed in register 1, the address of a work area containing the first 16 bytes of the IOB is placed in register 0, and the SYNAD routine is given control when the next GET macro instruction is issued. Buffer scheduling is suspended until the next GET macro instruction is reissued.

**Block Could Not be Reached (Input):** This condition is reported if the control program's error recovery procedures encounter an uncorrectable error in searching an index or overflow chain. The SYNAD routine is given control when a GET macro instruction is issued for the first logical record of the unreachable block.

**Block Could Not be Reached (Output):** This condition is reported if the control program's error recovery procedures encounter an uncorrectable error in searching an index or overflow chain.

If the error is encountered during closing of the data control block, bit 2 of DCBEXCD2 is set to 1 and the SYNAD routine is given control immediately. Otherwise, the SYNAD routine is given control when the next GET macro instruction is issued.

**Sequence Check:** This condition is reported if a PUT macro instruction refers to a record whose key has a smaller numerical value than the key of the record previously referred to by a PUT macro instruction. The SYNAD routine is given control immediately; the record is not transferred to secondary storage.

**Duplicate Record:** This condition is reported if a PUT macro instruction refers to a record whose key duplicates that of the record previously referred to by a PUT macro instruction. The SYNAD routine is given control immediately; the record is not transferred to secondary storage.

**Data Control Block Closed When Error Routine Entered:** This condition is reported if the control program's error recovery procedures encounter an uncorrectable output error during closing of the data control block. Bit 5 or 7 of DCBEXCD1 is set to 1, and the SYNAD routine is immediately given control. After appropriate analysis, the SYNAD routine must branch to the address in return register 14 so that the control program can finish closing the data control block.

**Overflow Record:** This condition is reported if the input record is an overflow record.

**Incorrect Record Length:** This condition is reported if the length of the record as specified in the record-descriptor word (RDW) is larger than the value in the DCBLRECL field of the data control block.

| Exception Code Bit | READ | WRITE | Condition if On |
|---|---|---|---|
| 0 | X | X | Record not found |
| 1 | X | X | Record length check |
| 2 | | X | Space not found |
| 3 | X | X | Invalid request—see bits 9-15 |
| 4 | X | X | Uncorrectable I/O error |
| 5 | X | X | End of data |
| 6 | X | X | Uncorrectable error |
| 7 | | Type X | Not read with exclusive control |
| 8 | | | Not used |
| 9 | | X | WRITE to input data set |
| 10 | X | X | Extended search with DCBLIMCT=0 |
| 11 | X | X | Block or track requested was outside data set |
| 12 | | X | Tried to write capacity record |
| 13 | X | X | Specified key as search argument when KEYLEN=0 or no key address supplied |
| 14 | X | X | Request for options not in data control block |
| 15 | | X | Attempt to add fixed-length record with key beginning with hexadecimal FF |

Figure 3. Exception Code Bits—BDAM

*Notes for Figure 3:*

**Record Not Found;** This condition is reported if the search argument is not found in the data set.

**Record Length Check:** This condition occurs for READ and WRITE (update) and WRITE (add). For WRITE (update) variable-length records only, the length in the BDW does not match the length of the record to be updated. For all remaining READ and WRITE (update) conditions the BLKSIZE, when **'S'** is specified in the READ or WRITE macro, or the length given with these macros does not agree with the actual length of the record. For WRITE (add), fixed-length records, the BLKSIZE, when **'S'** is specified in the WRITE macro, or the length given with this macro does not agree with the actual length of the record. For WRITE (add), all other conditions, no error can occur.

**Space Not Found in Which to Add a Record:** This condition occurs if either there is no dummy record when adding an F-format record, or there is no space available when adding a V or U-format record.

**Invalid Request:** Occurs whenever one of the following bits are set to one:

| | | |
|---|---|---|
| Bit 9 | — | A WRITE was attempted for an input data set. |
| Bit 10 | — | An extended search was requested, but LIMCT was zero. |
| Bit 11 | — | A WRITE (add) with fixed-length was attempted with the key beginning with X'FF'. |
| Bit 12 | — | Writing a capacity record(R0) was attempted. |
| Bit 13 | — | A READ or WRITE with key was attempted, but either KEYLEN equaled zero or the key address was not supplied. |
| Bit 14 | — | The READ or WRITE macro options conflict with the OPTCD or MACRF parameters. |
| Bit 15 | — | The relative block or relative track requested was not in the data set. |

**Uncorrectable Input/Output Error:** This condition is reported if the control program's error recovery procedures encounter an uncorrectable error in transferring data between real and secondary storage.

**End of Data:** This only occurs as a result of a READ (type DI, DIF, or DIX) when the record requested is an end-of-data record.

**Uncorrectable error:** Same conditions as for bit 4.

**Not Read With Exclusive Control:** A WRITE, type DIX or DKX, has occurred for which there is no previous corresponding READ with exclusive control.

| Register | Bits | Meaning |
|---|---|---|
| 0 | 0-7 | Not used. |
| | 8-31 | Address of a work area containing the first 16 bytes of the IOB (after an uncorrectable input/output error caused by a GET, PUT, or PUTX macro instruction; original contents destroyed in other cases). If the error condition was detected before I/O was started, register 0 contains all zeros. |
| 1 | 0-7 | Not used. |
| | 8-31 | Address of the buffer containing the error record (after an uncorrectable input/output error caused by a GET, PUT, or PUTX macro instruction while attempting to read or write a data record; in other cases this register contains 0). |
| 2-13 | 0-31 | Contents that existed before the macro instruction was issued. |
| 14 | 0-7 | Not used. |
| | 8-31 | Return address. This address is either an address in the control program's Close routine (bit 2 of DCBEXCD2 is on), or the address of the instruction following the expansion of the macro instruction that caused the SYNAD routine to be given control (bit 2 of DCBEXCD2 is off). |
| 15 | 0-7 | Not used. |
| | 8-31 | Address of the SYNAD routine. |

Figure 4. Register Contents on Entry to SYNAD Routine—QISAM

| Register | Bits | Meaning |
|---|---|---|
| 0 | 0-7 | Not used. |
| | 8-31 | Address of the first IOB sense byte. (Sense information is valid only when associated with a unit check condition.) |
| 1 | 0-7 | Not used. |
| | 8-31 | Address of the DECB. |
| 2-13 | 0-31 | Contents that existed before the macro instruction was issued. |
| 14 | 0-7 | Not used. |
| | 8-31 | Return address. |
| 15 | 0-7 | Not used. |
| | 8-31 | Address of the SYNAD routine. |

Figure 5. Register Contents on Entry to SYNAD Routine—BISAM

| Register | Bits | Meaning |
|---|---|---|
| 0 | 0-7 | Value to be added to the status indicators address to provide the address of the first CCW (QSAM only). |
| | 8-31 | Address of the associated data event control block for BDAM, BPAM, and BSAM; address of the status indicators shown in Figure 7 for QSAM. |
| 1 | 0 | Bit is on for error caused by input operation. |
| | 1 | Bit is on for error caused by output operation. |
| | 2 | Bit is on for error caused by BSP, CNTRL, or POINT macro instruction (BPAM AND BSAM only). |
| | 3 | Bit is on if error occurred during update of existing record or if error did not prevent reading of the record. Bit is off if error occurred during creation of a new record or if error prevented reading of the record. |
| | 4 | Bit is on if the request was invalid. The status indicators pointed to in the data event control block are not present (BDAM, BPAM, and BSAM only). |
| | 5 | Bit is on if an invalid character was found in paper tape conversion (BSAM and QSAM only). |
| | 6 | Bit is on for a hardware error (BDAM only). |
| | 7 | Bit is on if no space was found for the record (BDAM only). |
| | 8-31 | Address of the associated data control block. |
| 2-13 | 0-31 | Contents that existed before the macro instruction was issued. |
| 14 | 0-7 | Not used. |
| | 8-31 | Return address. |
| 15 | 0-7 | Not used. |
| | 8-31 | Address of the error analysis routine. |

Figure 6. Register Contents on Entry to SYNAD Routine—BDAM, BPAM, BSAM, and QSAM

| Offset From Status Indicator Address | | Meaning | Name | |
|---|---|---|---|---|
| Byte | Bit | | | |
| +2 | 0 | Command reject | Sense byte 1 | |
| | 1 | Intervention required | | |
| | 2 | Bus-out check | | |
| | 3 | Equipment check | | |
| | 4 | Data check | | |
| | 5 | Overrun | | |
| | 6,7 | Device-dependent | | |
| +3 | 0-7 | Refer to the appropriate device manual | Sense byte 2 | |
| +8 | 0-7 | Beginning of a channel status word | | |
| +9 | — | Command address | | |
| +12 | 0 | Attention | Status byte 1 (Unit) | |
| | 1 | Status modifier | | |
| | 2 | Control unit end | | |
| | 3 | Busy | | |
| | 4 | Channel end | | |
| | 5 | Device end | | channel |
| | 6 | Unit check—must be on for sense bytes to be meaningful | | status |
| | 7 | Unit exception | | word |
| +13 | 0 | Program-controlled interrupt | Status byte 2 (Channel) | |
| | 1 | Incorrect length | | |
| | 2 | Program check | | |
| | 3 | Protection check | | |
| | 4 | Channel data check | | |
| | 5 | Channel control check | | |
| | 6 | Interface control check | | |
| | 7 | Chaining check | | |
| +14 | — | Count field | | |

Figure 7. Status Indicators for the SYNAD Routine—BDAM, BPAM, BSAM, and QSAM

# 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 |
| NOTE | | | X | X | | |
| OPEN | X | X | X | X | X | 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 |
| 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 blocksize (BLKSIZE) and logical record length (LRECL) operands in the DCB macro instruction. These values can be used to determine the maximum blocksize and logical record length for a given device, and they can be used to determine the optimum blocking factor when records are to be blocked.

## Card Readers and Card Punches

The logical-record length for a card reader or card punch is fixed at 80 bytes; variable-length records are not supported for these devices. 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 shows the record length that can be specified for the various printers. In some cases, two values are shown; 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).

| | |
|---|---|
| 1403 printer | — 120 or 132 bytes |
| 1404 printer | — 120 or 132 bytes |
| 1443 printer | — 120 or 144 bytes |
| 3211 printer | — 132 or 150 bytes |
| 1052 printer keyboard | — 130 bytes |
| 3210 printer keyboard | — 130 bytes |
| 3215 printer keyboard | — 130 bytes |
| 3525 card punch, print feature | — 64 bytes |

## Paper-Tape Reader

2671 paper tape—32,760 bytes

## Magnetic-Tape Units

2400/3400 magnetic-tape units—32,760

(7 tracks and 9 tracks)

## Direct-Access Devices

The following chart shows the capacity of direct-access devices by track, cylinder, and total capacity in bytes.

| Device Type | Volume Type | Maximum Block-size/Track[1] | Tracks/ Cylinder | Number of Cylinders[2] | Total Capacity[1,2] |
|---|---|---|---|---|---|
| 2301 | Drum | 20483 | 8 | 25 | 4,096,600 |
| 2302 | Disk | 4984 | 46 | 246 | 56,398,944 |
| 2303 | Drum | 4892 | 10 | 80 | 3,913,600 |
| 2305-1 | Drum | 14136 | 8 | 48 | 5,428,224 |
| 2305-2 | Drum | 14660 | 8 | 96 | 11,258,880 |
| 2311 | Disk | 3625 | 10 | 200 | 7,250,000 |
| 2314/2319 | Disk | 7294 | 20 | 200 | 29,176,000 |
| 2321 | Data Cell | 2000 | 20 | 980[3] | 39,200,000 |
| 3330/3333 | Disk | 13030 | 19 | 404 | 100,018,280 |

[1] Capacity indicated in bytes (when R0 is used by the IBM programming system).
[2] Excluding alternate cylinders.
[3] A volume is equal to one bin in a 2321 Data Cell.

Each record written on a direct-access device requires some "device overhead." The term device overhead means the space required by the device for address markers, count areas, gaps between the count, key, and data areas, and gaps between blocks. The following formulas can be used to compute the number of bytes required for each data block including the space required for device overhead. Note that any fraction of a byte must be ignored. For example, if the formula computation results in 15.644 bytes, 15 bytes must be used to determine track capacity.

| | | Bytes Required by Each Data Block | | | |
|---|---|---|---|---|---|
| Device | Track | Blocks With Keys | | Blocks Without Keys | |
| Type | Capacity | Bi | Bn | Bi | Bn |
| 2301 | 20483 | 186+KL+DL | 53+KL+DL | 133+DL | DL |
| 2302 | 4984 | 81+(KL+DL)537/512 | 20+KL+DL | 61+(DL)537/512 | DL |
| 2303 | 4892 | 146+KL+DL | 38+KL+DL | 108+DL | DL |
| 2305-1 | 14568[1] | 634+KL+DL | 634+KL+DL | 432+DL | 432+DL |
| 2305-2 | 14858[1] | 289+KL+DL | 289+KL+DL | 198+DL | 198+DL |
| 2311 | 3625 | 81+(KL+DL)537/512 | 20+KL+DL | 61+(DL)537/512 | DL |
| 2314/2319 | 7294 | 146+(KL+DL)534/512 | 45+KL+DL | 101+(DL)534/512 | DL |
| 2321 | 2000 | 100+(KL+DL)537/512 | 16+KL+DL | 84+(DL)537/512 | DL |
| 3330/3333 | 13165[1] | 191+KL+DL | 191+KL+DL | 135+DL | 135+DL |

Bi is any block but the last on the track.
Bn is the last block on the track.
DL is data length.
KL is key length.
[1] This value is different from the maximum blocksize per track because the formula for the last block on the track includes an overhead for this device.

When the track overflow feature is being used or variable-length spanned records are written, the size of a data block or logical record can exceed the capacity of a single track on the direct-access device used.

# APPENDIX D: DCB EXIT LIST FORMAT AND CONTENTS

The following shows the format and contents that must be supplied by the problem program when the EXLST operand is specified in a DCB macro instruction. The exit list must begin on a fullword boundary and each entry in the list requires one fullword.

| Routine Type | Hexadecimal Code | 3-Byte Routine Address—Purpose |
|---|---|---|
| Inactive entry | 00 | Ignored; the entry is not active. |
| Input header label | 01 | Process a user input header label. |
| Output header label | 02 | Create a user output header label. |
| Input trailer label | 03 | Process a user input trailer label. |
| Output trailer label | 04 | Create a user output trailer label. |
| Data control block exit | 05 | Data control block exit routine. |
| End-of-volume | 06 | End-of-volume exit routine. |
| User totaling | 0A | Pointer to user's totaling area. |
| Block count exit | 0B | Block count unequal exit routine. |
| Defer input trailer label | 0C | Defer processing of a user input trailer label from the end-of-data until the CLOSE macro instruction is issued. |
| Defer nonstandard input trailer label | 0D | Defer processing a nonstandard input trailer label on magnetic tape unit from the end-of-data until the CLOSE macro instruction is issued (no exit routine address). |
| FCB Image | 10 | Define an FCB image. |
| DCB ABEND exit | 11 | Allow analysis of ABEND condition and select one of several options. |
| Last entry | 80 | Last entry in list. A high-order bit can be specified with any of the above codes but must always be specified with the last entry. |
| | 67 | ADDRESS OF SFCB BUFFER |

The list can be dynamically shortened during execution by setting the high-order bit of a word to a value of 1. An entry in the list can be made inactive dynamically by setting the high-order byte of the word to a value of hexadecimal 00.

When control is passed to an exit routine, the general registers contain the following information:

| Register | Contents |
|---|---|
| 0 | Variable; the contents depend on the exit routine used. |
| 1 | The three low-order bytes contain either the address of the DCB currently being processed or, when certain exits are taken, the address of the exit parameter list. These exits are: user-label exits (X'01'-'04'), deferred nonstandard input trailer exit (X'0D'), and DCB ABEND exit (X'11'). |
| 2-13 | Contents prior to execution of the macro instruction. |
| 14 | Return address (must not be altered by the exit routine). |
| 15 | Address of the exit routine entry point. |

The conventions for saving and restoring registers are as follows:

- The exit routine must preserve the contents of register 14. It need not preserve the contents of other registers. The control program restores registers 2-13 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, refer to *OS Data Management Services Guide.*

# APPENDIX E: CONTROL CHARACTERS

Each logical record, in all record formats, can contain an optional control character. This control character is used to control stacker selection on a card punch or card read punch, or it is used to control printer spacing and skipping. If a record containing an optional control character is directed to any other device, it is considered to be the first data byte, and it does not cause a control function to occur.

In format-F and format-U records, the optional control character must be in the first byte of the logical record.

In format-V records, the optional control character must be in the fifth byte of the logical record, immediately following the block descriptor word of the record.

Two control character options are available. A control character option is selected by coding the appropriate character in the RECFM operand of the DCB macro instruction. If either option is specified in the data control block, a control character must be included in each record, and other spacing or stacker selection options also specified in the data control block are ignored.

## Machine Code

The record format field in the data control block indicates that the machine code control character has been placed in each logical record. If the record is written, the appropriate byte must contain the command code bit configuration specifying both the write and the desired carriage or stacker select operation. If the record is not written, the byte can specify any command other than write.

Command codes for specific devices are contained in IBM System Reference Library publications describing the control units or devices.

## American National Standards Institute Control Characters

In place of machine code, control characters defined by the American National Standards Institute (ANSI) can be specified. These characters must be represented in EBCDIC code.

American National Standards Institute control characters (ANSI) 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 |
| 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.

# INDEX

Indexes to Systems Reference Library publications are consolidated in *OS Master Index to Reference Manuals*, GC28-6644. For additional information about any subject listed below, refer to other publications listed for the same subject in the master index.

## A



## B

# C

# D

## E

## F



## G



## H



## I

## J



## K



## L



## M

# N



# O



# P

## Q



## R



## S

# T



# U



# V



# W



# X

GC26-3794-1

OS Data Management Macro Instructions (File No. S360/S370-30) Printed in U.S.A. GC26-3794-1

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

OS Data Management Macro Instructions
GC26-3794-1

**Reader's Comment Form**

Your comments about this publication will help us to improve it for you. Comment in the space below, giving specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

If you would like a reply, please provide your name, job title, and business address (including ZIP code).

**Fold on two lines, staple, and mail.** No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

GC26-3794-1

Fold and Staple

First Class Permit
Number 2078
San Jose, California

**Business Reply Mail**

No postage necessary if mailed in the U.S.A.

Postage will be paid by:

**IBM Corporation**
**Programming Center – Publishing**
**Department D58**
**Monterey and Cottle Roads**
**San Jose, California 95193**

Fold and Staple

IBM®

**International Business Machines Corporation**
**Data Processing Division**
**1133 Westchester Avenue, White Plains, New York 10604**
**(U.S.A. only)**

**IBM World Trade Corporation**
**821 United Nations Plaza, New York, New York 10017**
**(International)**

OS Data Management Macro Instructions (File No. S360/S370-30) Printed in U.S.A. GC26-3794-1

OS Data Management Macro Instructions
GC26-3794-1

**Reader's Comment Form**

Your comments about this publication will help us to improve it for you. Comment in the space below, giving specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

If you would like a reply, please provide your name, job title, and business address (including ZIP code).

**Fold on two lines, staple, and mail.** No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

Fold and Staple

[redacted]

First Class Permit
Number 2078
San Jose, California

**Business Reply Mail**

No postage necessary if mailed in the U.S.A.

[redacted]

Postage will be paid by:

**IBM Corporation**
**Programming Center – Publishing**
**Department D58**
**Monterey and Cottle Roads**
**San Jose, California 95193**

Fold and Staple

IBM®

**International Business Machines Corporation**
**Data Processing Division**
**1133 Westchester Avenue, White Plains, New York 10604**
**(U.S.A. only)**

**IBM World Trade Corporation**
**821 United Nations Plaza, New York, New York 10017**
**(International)**

OS Data Management Macro Instructions (File No. S360/S370-30) Printed in U.S.A. GC26-3794-1