UNISYS

PUBLICATIONS UPDATE

System 80

OS/3
Consolidated Data
Management
Macroinstructions
Programming Guide

UP-9979 Rev. 1-A

This Library Memo announces the release and availability of Update A to the *System 80 OS/3 Consolidated Data Management Macroinstructions Programming Guide*, UP-9979 Rev. 1.

This guide is a standard library item (SLI). It is part of the standard library provided automatically with the purchase of the product.

This update provides an index. The content of the manual is unchanged.

You can order the update only or the complete manual with all updates. To receive only the update, order UP-9979 Rev. 1-A; to receive the complete manual and all updates, order UP-9979 Rev. 1.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists MB00, SAB, and SAE	Mailing Lists MB08, MB18, and MBW (13 pages plus Memo)	Library Memo for UP-9979 Rev. 1-A
		RELEASE DATE: March 1990

UNISYS

PUBLICATIONS REVISION

System 80

OS/3
Consolidated Data
Management
Macroinstructions
Programming
Guide

UP-9979 Rev. 1

This Library Memo announces the release and availability of the *System 80 OS/3 Consolidated Data Management Macroinstructions Programming Guide*, UP-9979 Rev. 1.

This document is a standard library item (SLI). It is part of the standard library provided automatically with the purchase of the product.

This revision incorporates miscellaneous grammatical corrections and minor technical changes.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists MBOO, SAB, and SAE	Mailing Lists MB08, MB18, and MBW (228 pages plus Memo)	Library Memo for UP-9979 Rev. 1
		RELEASE DATE: January 1990

UNISYS

System 80 OS/3 Consolidated Data Management Macroinstructions

Programming Guide

January 1990

Priced Item

Printed in U S America
UP-9979 Rev. 1

UNISYS

System 80
OS/3
Consolidated Data Management Macroinstructions

Programming Guide

OS/3 Release 13.0

March 1990

Printed in U S America
UP-9979 Rev. 1 - Update A

Priced Item

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THIS DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Program Product License or Agreement to purchase or lease equipment. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such License or Agreement. Unisys cannot accept financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special, or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this publication should be forwarded to Unisys Corporation either by using the Business Reply Mail form at the back of this manual or by addressing remarks directly to Unisys Corporation, OS/3 Systems Product Information Development, P.O. Box 500, Mail Station E5-114, Blue Bell, Pennsylvania, 19424, U.S.A.

PAGE STATUS SUMMARY
ISSUE: Update A - UP-9979 Rev. 1

Part/Section	Page Number	Update Level
Cover		Orig.
Title Page/Disclaimer		A
PSS	iii	A
About This Guide	Tab Breaker v thru vii	Orig. Orig.
Contents	ix thru xv	Orig.
1	Tab Breaker 1 thru 5	Orig. Orig.
2	Tab Breaker 1 thru 57	Orig. Orig.
3	Tab Breaker 1 thru 30	Orig. Orig.
4	Tab Breaker 1 thru 16	Orig. Orig.
Appendix A	Tab Breaker 1 thru 13	Orig. Orig.
Appendix B	Tab Breaker 1 thru 3	Orig. Orig.
Appendix C	Tab Breaker 1 thru 40	Orig. Orig.
Appendix D	Tab Breaker 1 thru 28	Orig. Orig.
Index	Tab Breaker 1 thru 8	Orig. A*
User Comments Form		
Back Cover		

Part/Section	Page Number	Update Level

Part/Section	Page Number	Update Level

* New pages

About This Guide

Purpose and Audience

This guide is designed to instruct the programmer in the use of the consolidated data management macroinstructions for the Unisys Operating System/3 (OS/3). Its intended audience is those programmers who will use basic assembly language (BAL) to code their programs.

Organization

The information in this guide is presented as follows:

Section 1. Introduction

Describes how the data management macroinstructions are used in a program, the type of data management macroinstructions, and the conventions that are used to present them.

Section 2. Declarative Macroinstructions

Describes the functions performed by the declarative macroinstructions, shows their formats and provides examples of their use.

Section 3. Imperative Macroinstructions

Describes the functions performed by the imperative macroinstructions, shows their formats and provides examples of their use.

Section 4. Workstation Considerations

Describes those things you must consider when you use a workstation, such as the different modes of operation, the use of function keys, and how to manage the presentation of data on the screen.

Appendix A. Common Data Interface Block (CDIB)

Shows the format of the common data interface block (CDIB), describes the contents of the individual fields, and explains how to use the information in these fields in your program.

Appendix B. Error Flags

Describes the error flags.

Appendix C. Labels for Disk and Diskette Files

Shows the format of the labels for disk and diskette files and describes the contents of the individual fields.

Appendix D. Magnetic Tape Labels

Shows the format of the labels for magnetic tape files and describes the contents of the individual fields.

Related Product Information

The following Unisys documents may be helpful in understanding and implementing the information presented in this guide. Throughout this guide, when we refer you to another document, use the version that applies to the software level in use at your site.

Job Control Programming Guide **(UP-9986)**

Provides information on the format and use of job control statements.

Consolidated Data Management Programming Guide **(UP-9978)**

Describes how the component works and provides information concerning the use of different types of files in a program.

Assembler Programming Guide **(UP-8913)**

Describes the System 80 assembly language.

System Messages Reference Manual **(UP-8076)**

Lists and describes the messages that are displayed on the system console during program execution.

Models 3-6 and 8-20 Installation Guide **(UP-8839)**

This guide provides the system administrator with the information and procedures needed to install, tailor, and maintain OS/3 software in a System 80 Models 3-6 and 8-20 environment.

Model 7E Installation Guide **(7002 3858)**

This guide provides the system administrator with the information and procedures needed to install, tailor, and maintain OS/3 software in a System 80 Model 7E environment.

Model 3-6 and 8-20 Operations Guide **(UP-8859)**

This guide describes the hardware configuration of the System 80 Models 3-6 and 8-20 and presents procedures for initializing the system. It also covers all commands and procedures used in the OS/3 environment.

Model 7E Operations Guide **(7002 3866)**

This guide describes the hardware configuration of the System 80 Model 7E and presents procedures for initializing the system. It also covers all commands and procedures used in the OS/3 environment.

Notation Conventions

See 1.1.4, "Macroinstruction Conventions."

Contents

Appendix B. Error Flags

Appendix C. Labels for Disk and Diskette Files

Appendix D. Magnetic Tape Labels

Index

User Comments Form

Figures

Tables

Section 1
Introduction

1.1. General

When you use consolidated data management to write a program, you must include:

- data management declarative macroinstructions in your program to define your files; and

- imperative macroinstructions to cause data to be brought in from and sent out to these files.

Consolidated data management provides you with a single set of standardized macroinstructions for all types of files rather than specialized macroinstructions for each type of file. As a result, the writing of a program is greatly simplified.

1.1.1. Declarative Macroinstructions

There are two declarative macroinstructions, Common Data Interface Block (CDIB) and Resource Information Block (RIB). They are used to supply information concerning the characteristics of your file. As implied by the term declarative, these macroinstructions generate nonexecutable code such as constants and storage areas for variables. Consequently, these instructions should be kept separate from your main processing.

The format of a declarative macroinstruction is:

LABEL	ΔOPERATIONΔ	OPERAND
[name]	xxxx	keyword-1=x,...,keyword-n=x

A symbolic name can appear in the label field. This name can have a maximum of seven characters for the CDIB macroinstruction and eight characters for the RIB macroinstruction. It must begin with an alphabetic character. The appropriate verb or code must appear in the operation field. One or more keyword parameters (consisting of a word or code immediately followed by an equal sign and one specification) can appear in the operand field. The keyword parameters can be written in any order and must be separated by commas.

1.1.2. Imperative Macroinstructions

Imperative macroinstructions are used to make formal requests to data management to process the files you have defined for your program. These instructions cause data management to perform such actions as opening and closing your files and retrieving data from or transferring data to your files. You include these instructions in the main processing portion of your program at those points where you need these services.

The format of an imperative macroinstruction is:

LABEL	ΔOPERATIONΔ	OPERAND
[name]	xxxx	yyyy,...,zzzz

A symbolic name can appear in the label field. The name can have a maximum of eight characters and must begin with an alphabetic character. The appropriate verb or code must appear in the operation field. The positional parameters must be written in the operand field in the order that is specified and be separated by commas. When a positional parameter is omitted, the comma must be retained to indicate the omission, except in the case where you omit trailing parameters.

1.1.3. Assembler Rules for Operand Fields

As you know, you must include data management macroinstructions in a BAL program in order to process the files that are involved with that program. Because these instructions are part of the program and will be assembled along with the BAL instructions, the same rules that apply to the operand fields of the BAL instructions also apply to the operand fields in the data management macroinstructions. These rules are:

1. The operand field of a macroinstruction begins in column 16 and may not extend beyond column 71. An operand may be continued onto the next line by inserting an arbitrary nonblank character in column 72. Each continuation line starts in column 16.

2. The operand field is terminated by the first blank that is not enclosed within apostrophes. As the operand specification is usually completed before column 40, columns 41 through 71 are available for comments, but at least one blank space must occur between the end of the operand specification and the beginning of the comments.

3. Comments are not continued by the insertion of a nonblank character in column 72. Lengthy comments can be entered by coding an asterisk (*) in column 1. Operands may be continued onto the next line by placing a comma after the last operand on the first line and a nonblank character in column 72. However, if you omit the comma and at least one blank exists between the last operand on the first line and the nonblank character in column 72, the second line of operands is treated as comments. Because the second line is treated as comments and not as part of your operand specification, the assembler does not flag the missing commas as an error. Up to two comment lines are permitted.

1.1.4. Macroinstruction Conventions

The conventions used to present the macroinstructions are as follows:

- Positional parameters must be written in the order specified in the operand field and must be separated by commas. When a positional parameter is omitted, the comma must be retained to indicate the omission, except for omitted trailing parameters.

  Examples:

  Assume that the DMINP macroinstruction has four optional parameters: A, B, C, and D.

  ```
  IN1 DMINP FILE1,A
  IN2 DMINP FILE2,A,B
  IN3 DMINP FILE3,A,B,C
  IN4 DMINP FILE4,A,B,C,D
  IN5 DMINP FILE5,A,B,,D
  ```

- A keyword parameter consists of a word or a code immediately followed by an equal sign, which is, in turn, followed by a specification. Keyword parameters can be written in any order in the operand field. Commas are required only to separate parameters; however, a comma must neither be coded in column 16 of a continuation line nor follow the last keyword of a string.

  Examples:

  Assume that the RIB macroinstruction has three optional keyword parameters: BFSZ, OPTN, and RCFM.

  ```
  IN1 RIB BFSZ=256,RCFM=FIXUNB
  IN2 RIB RCFM=FIXUNB,OPTN=YES
  IN3 RIB OPTN=YES,RCFM=FIXUNB,BFSZ=256
  IN4 RIB OPTN=YES
  ```

- Capital letters, commas, equal signs, and parentheses must be coded exactly as shown. The exceptions are those acronyms that are part of generic terms representing information to be supplied by the user and the commas preceding keyword parameters of declarative macroinstructions. (These commas serve to remind the user that keyword parameters coded in a string must be separated by commas.)

  Examples:

  ```
  ACCESS=EXC
  CKPTREC=NO
  KEY1=(80,10,DUP,CHG)
  ```

- Lowercase letters and words are generic terms representing information that must be supplied by the user.

  Examples:

  ```
  n
  size
  location
  name
  ```

- Information contained within braces represents mandatory entries of which one must be chosen.

  Examples:

  $$\left\{\begin{array}{l}\texttt{cdibname}\\ \texttt{(1)}\\ \texttt{1}\end{array}\right\}$$

- Information contained within brackets represents optional entries that (depending upon program requirements) are included or omitted. Braces within brackets signify that one of the specified entries must be chosen if that parameter is to be included.

  Examples:

  ```
  [ASCII=YES]
  ```

  $$\left[\texttt{,}\left\{\begin{array}{l}\texttt{workarea}\\ \texttt{(0)}\\ \texttt{0}\end{array}\right\}\right]$$

- An optional parameter that has a list of optional entries may have a default specification that is supplied by the operating system when the parameter is not specified by the user. Although the default may be specified by the user with no adverse effect, it is considered unnecessary to do so. For each reference, when a default specification occurs in the format delineation, it is printed on a shaded background.

  If, by parameter omission, the operating system performs some complex processing other than parameter insertion, it is explained under an if omitted heading in the parameter description.

  Examples:

```
[,CKPTREC={NO }]
[         {YES}]

[,MODE={SEQ}]
[      {RAN}]
```

- An ellipsis (series of three periods) indicates the presence of a variable number of entries.

  Example:

```
param-1,...,param-n
```

- Commas are required when positional parameters are omitted, except after the last parameter specified.

  Example:

```
param-1,param-2,,param-4
```

Section 2
Declarative Macroinstructions

2.1 General

The declarative macroinstructions are used to specify file and processing characteristics and as parameter-passing mechanisms. There are two declarative macroinstructions that are used with all types of files. These are the CDIB and RIB macroinstructions.

2.2. Define a Common Data Interface Block (CDIB)

The CDIB macroinstruction identifies a logical file and establishes a common data interface block for the file. The common data interface block is the function-passing mechanism for the file and it is referenced each time you issue an imperative macroinstruction.

Format

LABEL	ΔOPERATIONΔ	OPERAND
name	CDIB	FILENAME=LFD-name

Label

Specifies the name of the CDIB. This name cannot exceed seven characters.

Keyword Parameter FILENAME:

`FILENAME=LFD-name`

Specifies the name assigned to the file by the LFD job control statement in the device assignment set for the file. This name is placed in the file name field of the CDIB.

If omitted, the name specified in the label field is placed in the file name field of the CDIB.

Example

```
INPUT01 CDIB FILENAME=PAYUPDT
```

2.3. Define a Resource Information Block (RIB)

The RIB macroinstruction is used to describe the file characteristics and processing requirements. It is used in combination with the CDIB macroinstruction when you issue an OPEN imperative macroinstruction to open the file. This description consists of specifying the appropriate keyword parameters in the operand field of the RIB macroinstruction. The expansion is an encoded representation of the keywords specified on the call line.

The following describes the set of keyword parameters that pertain to each of the devices that can contain your files. Keyword parameters that pertain to more than one device can be specified in an RIB macroinstruction. When the file is opened, data management determines the device type from the device assignment set you specified in your program execution job control stream. It then scans the RIB and only uses those parameters that pertain to the device. Parameters that pertain to other devices are ignored.

In those cases where a default value is shown for a keyword parameter, you do not have to specify that parameter if the default value meets your requirements. If a parameter with a default value is not specified, data management uses the default value.

2.3.1. RIB Macroinstruction for Disk (MIRAM) Files and Format Label Diskette (MIRAM) Files

Format

```
LABEL      ΔOPERATIONΔ     OPERAND
------------------------------------------------
name        RIB          ┌ACCESS=┌ EXC  ┐┐
                         │       │ SRDO ││
                         │       │ SRDF ││
                         │       { SRD  }│
                         │       │ EXCR ││
                         │       │ SADD ││
                         └       └ UCP  ┘┘
                         [,BFSZ=n]
                         ┌,CACHE=┌SYSGEN┐┐
                         │       {YES   }│
                         └       └NO    ┘┘
```

```
LABEL      ΔOPERATIONΔ     OPERAND
--------------------------------------------------------------------
name        RIB            [,INDA=symbol]
(cont.)
                           ⎡,INDS=⎧0⎫⎤
                           ⎣      ⎩n⎭⎦

                           ,IOA1=symbol
                           [,IOA2=symbol]

                           ⎡,IORG=⎧NO ⎫⎤
                           ⎣      ⎩(r)⎭⎦

                           [,KARG=symbol]

                           ⎡,KEYn= size[,loc] ,⎧NDUP⎫⎤⎡,⎧NCHG⎫⎤⎤
                           ⎣                   ⎩DUP ⎭⎦⎣ ⎩CHG ⎭⎦⎦

                           ⎡,MODE=⎧SEQ⎫⎤
                           ⎣      ⎩RAN⎭⎦

                           ⎡,OPTN=⎧NO ⎫⎤
                           ⎣      ⎩YES⎭⎦

                           ⎡,PROC=⎧UNK⎫⎤
                           ⎣      ⎩KEY⎭⎦

                           ⎡,RCB=⎧YES⎫⎤
                           ⎣     ⎩NO ⎭⎦

                           ⎡,RCFM=⎧FIXUNB⎫⎤
                           ⎣      ⎩VARUNB⎭⎦

                           [,RCSZ=n]

                           ⎡,RETR=⎧INF⎫⎤
                           ⎣      ⎩MOD⎭⎦

                           [,SKAD=symbol]
```

```
LABEL      ΔOPERATIONΔ      OPERAND
-----------------------------------------------
name         RIB         [,TRUNC={NO }]
(cont.)                  [       {YES}]

                         [,VMNT={NO }]
                         [      {ONE}]

                         [,VRFY={NO }]
                         [      {YES}]

                         [,WKFM={NO  }]
                         [      {VAR }]
                         [      {VARI}]

                         [,WORK={NO }]
                         [      {YES}]
```

Keyword Parameter ACCESS:

> This keyword parameter specifies the file share requirements. The requirements indicate whether or not multiple *logical access paths* (LAPs) can access the physical file at the same time and the type of processing (read and/or write use) that each LAP can perform. (See the Consolidated Data Management Programming Guide, UP-9978, for additional information on file sharing.)

Table 2-1 summarizes the ACCESS parameter specifications.

Table 2-1. Summary of ACCESS Parameter Specifications

ACCESS Parameter Specification	Use Permitted	
	Current LAP	Other LAPs (and compatible specifications)
EXC	Read/write	Read (SRDF*)
SRDO	Read	Read (SRDO SRDF* SRD)
SRDF	Read	Read/write (EXC SRDO SRDF* SRD EXCR SADD UCP
SRD	Read	Read/write (SRDO SRDF* SRD EXCR SADD UCP)
EXCR	Read/write	Read (SRDF* SRD)
SADD	Read/write	Read/write (SRDF* SRD SADD)
UCP	Read/write	Read/write (SRDF* SRD UCP)

* See the SDRF description for an explanation of conditions that affect SRDF specifications.

Keyword Parameter BFSZ:

BFSZ=n

Specifies the length of the I/O area (data buffer) in bytes. If the full size of the buffer cannot be used, it will be automatically rounded down to the appropriate size.

There is no actual blocking performed; the records are in spanned format where one record immediately follows the next. Therefore, the file can be accessed using a buffer size that is different from the buffer size used at file creation. When the file is opened, data management verifies that the specified buffer size is at least as large as the *minimum allowable value.* A buffer size that exceeds the minimum allowable value can be specified to enhance performance.

The algorithm for determining the minimum allowable value in logical sectors is as follows:

$$\frac{S}{LSS} = N \text{ and } R$$

where:

S

Is the slot size. The slot size equals record size + 1 for files with RCB and fixed-length records. Otherwise, slot size equals the record size.

LSS

Is the logical sector size. The logical sector size is equal to the physical sector size, which is 256.

N

Is the number of full sectors per slot.

R

Is the remainder.

The minimum allowable value in logical sectors is:

N; if R=0

N+1; if R divides evenly into LSS

N+2; otherwise

After you have determined what the minimum allowable value in logical sectors is, you then multiply this number by the logical sector size to determine the length (n) of your I/O area.

(See "Disk (MIRAM) and Format Label Diskette (MIRAM) File Label Parameters" in this section, for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter CACHE:

This keyword parameter, the CACHE IOGEN parameter, and the REMOVE cache command provide selective caching of MIRAM disk files. The two basic reasons not to cache data from a file are when:

- A file is accessed randomly. Caching may not improve performance, but may impact it.
- Performance is not a priority in file processing and you wish to reserve the cache buffer for other processing.

For additional information on the keyword parameter CACHE, refer to the Consolidated Data Management Progranning Guide, UP-9978.

`CACHE=SYSGEN`
Specifies that caching/no caching is to be based on the CACHE IOGEN parameter, which is the default specification.

`CACHE=YES`
specifies the file to be processed with caching. Use this specification when you use the IOGEN parameter CACHE=NOMI and you want caching for this file. It is not necessary to use this if you specify IOGEN CACHE=YES. This specification applies only to MIRAM data files.

`CACHE=NO`
Specifies the file to be processed with no caching. Use this specification when you do not want caching for this file.

The CACHE IOGEN parameter is used to remove a device from cache. You can also use the REMOVE cache command for this purpose. Refer to the appropriate operations guide for further information on the disk cache facility, and for descriptions of all cache commands.

For further information on the CACHE IOGEN parameter, refer to the appropriate installation guide.

Keyword Parameter INDA:

`INDA=symbol`
Specifies the symbolic address of the main storage area where index blocks are processed during keyed operations. This area must be half-word aligned and must immediately precede the primary I/O data buffer area IOA1. This parameter is required for all keyed operations and it requires that all related keyword parameters also be specified: INDS, KARG, and KEYn. If any are missing, it is assumed keyed operations were not intended to be used.

Keyword Parameter INDS:

`INDS=0`
Specifies that the length of the index area is 0; consequently, no keyed operations can be performed.

`INDS=n`
Specifies the number of bytes to be used in main storage for the index area named in the INDA. The length must be a multiple of 256, up to a maximum of 32,512 bytes.

(See "Disk (MIRAM) and Format Label Diskette (MIRAM) File Label Parameters" in this section, for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter IOA1:

IOA1=symbol

Specifies the symbolic address of the primary I/O area (data buffer). This area must be half-word aligned.

This area must immediately follow the index buffer (INDA) if specified.

This parameter must be specified.

Keyword Parameter IOA2:

IOA2=symbol

Specifies the symbolic address of the secondary I/O area (data buffer). It must be half-word aligned, the same size as the primary I/O area (IOA1).

If index operations are to be performed, this area must immediately follow IOA1. The use of a secondary buffer is only permitted when performing keyed or unkeyed sequential output or unkeyed sequential input.

Keyword Parameter IORG:

IORG=NO

Indicates that records are to be processed in a work area. (See WORK parameter.)

IORG=(r)

Specifies the number of the general register to be used to point to the current record in the I/O area when you do not process records in a work area.

Registers 2 through 12 may be used. If you specify IORG=(r), this automatically sets the WORK=NO specification. (See WORK parameter.)

Keyword Parameter KARG:

KARG=symbol

Specifies the symbolic address of the field in your program where keys are placed to execute the retrieval of records. The length of this area is equal to the largest key in your program plus 3 bytes. (The minimum length is 6.) This parameter is required for all keyed operations.

Used as a storage area during the processing of certain imperatives and should be considered volatile. This field must be initialized before issuing any imperative that requires a key argument.

Keyword Parameter KEYn:

```
KEYn  (size[,loc] ⎡,⎧NDUP⎫⎤ ⎡ , ⎧NCHG⎫⎤
                  ⎣ ⎩DUP ⎭⎦ ⎣   ⎩CHG ⎭⎦
```

Specifies one of up to five keys for an indexed file, that is, $1 \leq n \leq 5$. There must be a KEYn parameter for each key in the file. The size of the key may range from 1 to 80 bytes. The loc parameter specifies the number of bytes preceding the key. If loc is omitted, 0 is assumed for fixed records. For variable records, however, loc is required. DUP specifies that duplicate keys are permitted. NDUP specifies that they are not allowed and is the default. CHG specifies that the key can change during update. NCHG specifies that it cannot change and is the default case.

(See "Disk (MIRAM) and Format Label Diskette (MIRAM) File Label Parameters" in this section, for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter MODE:

`MODE=SEQ`
Specifies sequential processing.

`MODE=RAN`
Specifies random processing.

For further information, see the DMINP and DMOUT macroinstruction descriptions in 3.4 and 3.5.

Keyword Parameter OPTN:

This keyword parameter determines the action that data management takes when a file resource is unavailable. A file resource is unavailable if:

- no device has been assigned to the file in the control stream (no device assignment set, that is, DVC - LFD statements); or

- the device is not available at execution time and the OPT positional parameter is included in the DVC job control statement.

`OPTN=NO`
If the file resource is unavailable, the file will not be treated as an optional file; that is, error status will be reported when an OPEN macroinstruction is issued for the file.

`OPTN=YES`
If the file resource is unavailable, the file will be treated as an optional file; that is, successful status will be reported when an OPEN imperative macroinstruction is issued for the file.

If a DMINP imperative macroinstruction is issued for the file, exception status will be reported. The error subcode will be set to X'00' to indicate exception status due to an end-of-file condition.

If any other imperative macroinstruction is issued for the file, successful status will be reported, but the function will be ignored.

For further information on status reporting, see Appendix A.

Keyword Parameter PROC:

PROC=UNK
Specifies unkeyed operations.

PROC=KEY
Specifies keyed operations.

Keyword Parameter RCB:

RCB=YES
Each record will contain a record control byte (RCB). If the records are fixed-length, the RCB will be appended to the front of each record. If the records are variable-length, the third byte in the 4-byte overhead is used as the RCB.

The presence of RCBs allows you to logically delete records from the file by marking them as void records. The records are not physically removed from the file. The marking process consists of setting the high order bit in the RCB. This is accomplished by using the DMDEL imperative macroinstruction (see 3.7).

In addition, the presence of RCBs also provides the following:

- If you are retrieving records sequentially, the deleted records will be skipped over.
- If you are randomly retrieving records and your search argument specifies the key or relative record number of a deleted record, it will result in a no-find-error condition.
- If you are outputting records randomly and you direct a record to a point beyond the last record in the file, any gap will be filled with void records.
- If you are randomly outputting records and you direct a record to a point within the file limits, an error condition will result if a valid (not deleted) record is at that point.

RCB=NO

Indicates that each record is not to contain an RCB.

(See "Disk (MIRAM) and Format Label Diskette (MIRAM) File Label Parameters" in this section, for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter RCFM:

RCFM=FIXUNB

Specifies fixed-length unblocked records.

Because there is no concept of blocked or undefined records for a disk (MIRAM) file or a format label diskette (MIRAM) file, the specification of RCFM=FIXBLK and RCFM=UNDEF in this case will be treated exactly as if RCFM=FIXUNB was specified.

RCFM=VARUNB

Specifies variable-length records. The first four bytes of a variable-length record are the record descriptor word (RDW). The first two bytes of the RDW contain the physical record size that you supply on output and data management supplies on input. The physical record size is the sum of RDW and data (logical record).

Variable-length records are contained within a fixed-size slot, where slot size equals the RCSZ specification. Because there is no concept of blocked records for a disk (MIRAM) file or a format label diskette (MIRAM) file, the specification of RCFM=VARBLK in this case will be treated exactly as if RCFM=VARUNB was specified.

(See "Disk (MIRAM) and Format Label Diskette (MIRAM) File Label Parameters" in this section, for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter RCSZ:

RCSZ=n

Specifies the length of each record in bytes.

For variable-length records, this reflects the slot size and should include the 4-byte RDW. The presence or absence of RCBs does not affect this parameter.

(See "Disk (MIRAM) and Format Label Diskette (MIRAM) File Label Parameters" in this section, for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter RETR:

RETR=INF

Specifies that records are to be retrieved for information purposes only. If this is specified, update or delete operations are not permitted.

RETR=MOD

Specifies that records are to be retrieved for modification. If this is specified, update or delete operations are permitted.

Keyword Parameter SKAD:

SKAD=symbol

Specifies the symbolic address of the field in your program where the relative disk address is placed for use in processing files by relative record number. This field is a 4-byte full-word aligned field. The first record is relative record 1.

This parameter is required if nonkeyed random input operations, nonkeyed random output operations, or nonkeyed select record operations are to be performed.

If this parameter is specified and the function request was successful, the following information is presented back to you in the SKAD field:

- the number of records in the file + 1 for an OPEN imperative macroinstruction only, or;
- the relative record number of the record in question for DMINP, DMOUT, and DMSEL (RECORD) imperative macroinstructions only.

Keyword Parameter TRUNC:

This keyword parameter specifies the manner in which input truncation will be reported. Input truncation occurs when the actual size of the record in question exceeds that which the program requested. It can only occur if WKFM=VARI is specified.

TRUNC=NO

Truncation is ignored. Successful status is reported and the program is not aware that truncation occurred.

TRUNC=YES

Truncation is reported as exception status due to an input truncation condition.

For further information on status reporting, see Appendix A.

Keyword Parameter VMNT:

VMNT=NO

Specifies that the file is to be processed with all volumes online. Random operations are permitted.

VMNT=ONE

Specifies that the file is to be processed with only one volume online at any time. A file that is created in this manner must be processed in this manner. Nonkeyed random operations or keyed random output operations are not permitted.

(See "Disk (MIRAM) and Format Label Diskette (MIRAM) File Label Parameters" in this section, for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter VRFY:

VRFY=NO

Specifies that data management is not to check the parity of output records.

VRFY=YES

Specifies that data management is to check the parity of output records after they have been written to disk. If it detects bad parity, data management will indicate output parity check status and return control to your program inline. If you specify this parameter, it will result in an increase in execution time for output operations.

Keyword Parameter WKFM:

When work-area processing is used (WORK=YES), this parameter specifies whether or not the work area is in variable format; that is, a 4-byte record descriptor word (RDW) followed by data (logical record). If the format is variable, the first 2 bytes of the RDW contain the physical record size that is the sum of RDW and data. The work-area format may be different from the record format as specified by the RCFM parameter.

WKFM=NO

Specifies that the work-area format is identical to the record format.

WKFM=VAR

Specifies that the work-area format is variable. For output operations, you must specify the physical record size in the RDW.

For input operations, data management will specify the physical size of the record in the RDW. Because the entire record is always moved into the work area on an input operation, the work area must be large enough to hold the largest record that could be expected.

WKFM=VARI

Specifies that the work-area format is variable. For output operations, you must specify the physical record size in the RDW.

For input operations, you must specify (in the RDW) the maximum number of bytes that are to be transferred to the work area. If the specified size is less than the actual record size, the record will be truncated. (See the TRUNC parameter for additional information on the status that is reported.) If the specified size is greater than the actual record size, the value in the RDW is changed to reflect the actual size of the record moved to the work area and successful status is reported.

Keyword Parameter WORK:

WORK=NO

Specifies that records are not to be processed in a work area. (See IORG parameter.)

WORK=YES

Specifies that input and output records will be processed in a work area rather than in the I/O area. When you issue input, update, or output operations, you specify the address of the work area with the appropriate imperative macroinstruction. If both IORG=(r) and WORK=YES are specified, the IORG parameter is ignored. When work area processing is being used, the work area and the I/O area should not be the same buffer.

Work-area processing must be used for all output, keyed update, and keyed delete operations.

Example:

```
RIB1 RIB BFSZ=512,RCSZ=80,IOA1=IN1,RCB=NO,RCFM=FIXUNB
```

This example shows the RIB specification for a disk file that is to be processed sequentially. The buffer size is 512 bytes and the record size is 80 bytes. All of the default values for the other keyword parameters are to be applied to the file.

Disk (MIRAM) and Format Label Diskette (MIRAM) File Label Parameters

When a file is created, the values specified for the BFSZ, INDS, KEYn, RCB, RCFM, and VMNT keyword parameters are saved in the format label for that file. The format label for a disk/diskette (MIRAM) file is described in Appendix C.

Each time you open a file after it has been created, data management will compare the values in the file format label with those you have specified. If they do not match, the file will not be opened and an error status is reported. The exception to this rule is that the setting in the format label for the RCB parameter is always used regardless of what you specify.

If the BFSZ, INDS, KEYn, RCB, RCFM, RCSZ, and VMNT parameters are defaulted, the values are determined as follows:

- For a new file, that is, one that is being created (newly allocated) or one for which the INIT option is specified in the LFD job control statement, the following values are used:

  BFSZ=minimum allowable value. (See the BFSZ parameter for the algorithm used to determine the minimum allowable value.)

  INDS=0

  KEYn=(0,0)

  RCB=NO

  RCFM=FIXUNB

  RCSZ=255 if RCFM=FIXUNB and RCB=YES; otherwise, 256. (The purpose of having 255 or 256 as the default is to ensure that the slot size is 256.)

  VMNT=NO

- For an existing file (one that is to be read, extended, or updated), the values that were saved in the file format label will be used. The exceptions to this are the INDS and VMNT parameters, which must be explicitly specified for existing files if the defaults were not used at file creation.

  Note that the buffer size that is used when the file is created is not saved in the file format label. It is not saved because the buffer can vary as long as it is at least as large as the minimum allowable value. If the BFSZ parameter is defaulted in this case, the minimum allowable value will be used. (See the BFSZ parameter for the algorithm used to determine the minimum allowable value.)

2.3.2. RIB Macroinstruction for Tape Files

Format

```
LABEL    ΔOPERATIONΔ    OPERAND
-----------------------------------------------
name        RIB         [,ASCII=YES]
                        [,BFSZ=n]

                        ⎡          ⎧NO ⎫ ⎤
                        ⎢ ,BKNO=   ⎨   ⎬ ⎥
                        ⎣          ⎩YES⎭ ⎦

                        ⎡          ⎧0⎫ ⎤
                        ⎢ ,BUFOFF= ⎨ ⎬ ⎥
                        ⎣          ⎩n⎭ ⎦

                        ⎡            ⎧NO ⎫ ⎤
                        ⎢ ,CKPTREC=  ⎨   ⎬ ⎥
                        ⎣            ⎩YES⎭ ⎦

                        ⎡        ⎧UNLOAD⎫ ⎤
                        ⎢,CLRW=  ⎨YES   ⎬ ⎥
                        ⎣        ⎩NO    ⎭ ⎦

                        ⎡          ⎧NO    ⎫ ⎤
                        ⎢ERROPT=   ⎨SKIP  ⎬ ⎥
                        ⎣          ⎩IGNORE⎭ ⎦

                        ⎡          ⎧NO  ⎫ ⎤
                        ⎢,FILABL=  ⎨NSTD⎬ ⎥
                        ⎣          ⎩STD ⎭ ⎦

                        ,IOA1=symbol
                        [,IOA2=symbol]

                        ⎡        ⎧NO ⎫ ⎤
                        ⎢,IORG=  ⎨   ⎬ ⎥
                        ⎣        ⎩(r)⎭ ⎦

                        ⎡          ⎧NO ⎫ ⎤
                        ⎢,LENCHK=  ⎨   ⎬ ⎥
                        ⎣          ⎩YES⎭ ⎦
```

```
LABEL      ΔOPERATIONΔ     OPERAND
---------------------------------------------------
           RIB             [,OPRW=  {NO }]
                                    {YES}

                           [,OPTN=  {NO }]
                                    {YES}

                           [,RCFM=  {FIXUNB}]
                                    {FIXBLK}
                                    {UNDEF }
                                    {VARUNB}
                                    {VARBLK}

                           [,RCSZ=  {n  }]
                                    {(r)}

                           [,READ=  {FORWARD}]
                                    {BACK   }

                           [,TPMARK=  {NO }]
                                      {YES}

                           [,TRANS=  {NO   }]
                                     {ASCII}

                           [,TRUNC=  {NO }]
                                     {YES}

                           [,TYPEFLE=  {INPUT }]
                                       {OUTPUT}

                           [,ULABEL=  {NO }]
                                      {YES}
```

```
LABEL    ΔOPERATIONΔ    OPERAND
                        [,VARBLD=(r)]

                        ⎡        ⎧NO  ⎫⎤
                        ⎢,WKFM=  ⎨VAR ⎬⎥
                        ⎣        ⎩VARI⎭⎦

                        ⎡,WORK=  ⎧NO ⎫⎤
                        ⎣        ⎩YES⎭⎦
```

Keyword Parameter ASCII:

ASCII=YES
: Specifies that the file is to be processed as an ASCII tape. If this parameter is specified, TRANS=ASCII and FILABL=STD must also be specified. If omitted, data management assumes that the file is to be processed as an EBCDIC tape.

Keyword Parameter BFSZ:

BFSZ=n
: Specifies the length of the I/O area (data buffer) in bytes. If the full size of the buffer cannot be used, it will be automatically rounded down to the appropriate size.

(See "Tape File Label Parameters" in this section, for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter BKNO:

BKNO=NO
: Specifies that block numbers are not to be created during output operations and that the sequence of the block numbers are not to be checked during input operations.

BKNO=YES
: Specifies that block numbers are to be created during output operations and that the sequence of the block numbers be checked during input operations. If this parameter is specified, a 4-byte storage area, full-word aligned, must precede each I/O area that is used.

Keyword Parameter BUFOFF:

BUFOFF=0
Specifies that there are no block prefixes. Used only with ASCII files.

BUFOFF=n
Specifies the length of a block prefix in bytes where $1 \leq n \leq 99$. Used only with ASCII files.

Keyword Parameter CKPTREC:

CKPTREC=NO
Specifies that checkpoint records are to be processed as data.

CKPTREC=YES
Specifies that checkpoint records are to be bypassed on input files. This parameter is ignored for output files.

Keyword Parameter CLRW:

CLRW=UNLOAD
Specifies that the tape is to be rewound and unloaded after the file is closed.

CLRW=YES
Specifies that the tape is to be rewound but not unloaded after the file is closed.

CLRW=NO
Specifies that the tape is not to be rewound after the file is closed.

Keyword Parameter ERROPT:

ERROPT=NO
Specifies that data management is to return an unsuccessful indication when a parity error is detected.

ERROPT=SKIP
Specifies that if a parity error is detected in an input record, the record is to be bypassed; that is, the record is not made available for processing.

ERROPT=IGNORE
Specifies that if a parity error is detected, it is ignored. The record can be processed as though no error occurred.

Keyword Parameter FILABL:

FILABL=NO
Specifies that all volumes of the file are unlabeled. (Not valid for ASCII files.)

FILABL=NSTD
Specifies that all volumes of the file contain nonstandard labels. (Not valid for ASCII files.)

FILABL=STD
Specifies that all volumes of the file contain standard labels that conform to system conventions.

Keyword Parameter IOA1:

IOA1=symbol
Specifies the symbol address of the primary I/O area (data buffer). If BKNO=YES, a 4-byte storage area must be reserved immediately preceding the I/O area. The 4-byte storage area and the I/O area must be full-word aligned. If the BKNO parameter is defaulted or BKNO=NO is specified, the I/O area must be half-word aligned.

This parameter must be specified.

Keyword Parameter IOA2:

IOA2=symbol
Specifies the symbolic address of a secondary I/O area. This area must be the same size as IOA1. If BKNO=YES, a 4-byte storage area must be received immediately preceding the secondary I/O area. The 4-byte storage area and the secondary I/O area must be full-word aligned.

If the BKNO parameter is defaulted or BKNO=NO, the secondary I/O area must be half-word aligned.

Keyword Parameter IORG:

(See 2.3.1 for a detailed explanation of the IORG parameter.)

Keyword Parameter LENCHK:

LENCHK=NO
Specifies that data management is not to check the block length specified in the block length prefix of variable length records in ASCII files against the physical block length.

LENCHK=YES
Specifies that data management is to check the block length specified in the block length prefix of variable length records in ASCII files against the physical block length.

Keyword Parameter OPRW:

OPRW=NO
Specifies that the tape is not to be rewound before labels are checked when a file is opened.

OPRW=YES
Specifies that the tape is to be rewound before labels are checked when a file is opened.

Keyword Parameter OPTN:

(See 2.3.1 for a detailed explanation of the OPTN parameter.)

Keyword Parameter RCFM:

`RCFM=FIXUNB`
Specifies fixed-length unblocked records.

`RCFM=FIXBLK`
Specifies fixed-length blocked records.

`RCFM=UNDEF`
Specifies undefined records.

`RCFM=VARUNB`
Specifies variable unblocked records.

`RCFM=VARBLK`
Specifies variable blocked records.

(See "Tape File Label Parameters" in this section for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter RCSZ:

`RCSZ=n`
Specifies the length of each record in bytes.

This parameter is only used for fixed-length records. This parameter is not used for variable-length records because you must place the record length in the first two bytes of each variable-length record and data management expects to find it there.

`RCSZ=(r)`
Specifies the number of the general register that is to contain the block length when you are reading or writing undefined records. Registers 2 through 12 may be used.

If this parameter is specified, RCFM=UNDEF must also be specified.

(See "Tape File Label Parameters" in this section for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter READ:

READ=FORWARD
Specifies that the tape is to be read in a forward direction.

READ=BACK
Specifies that the tape is to be read backward. Multivolume files cannot be read backward because the volume serial number is not present in the trailer labels.

Keyword Parameter TPMARK:

TPMARK=NO
Specifies that data management is not to write a tape mark. In this case, it is your responsibility to distinguish between labels and data.

TPMARK=YES
Specifies that data management is to write a tape mark to separate labels from data for output files with nonstandard labels or no labels.

Keyword Parameter TRANS:

TRANS=NO
Specifies that no translation is to be performed; that is, data management assumes that the tape is an EBCDIC file.

TRANS=ASCII
Specifies that the tape is to be processed as an ASCII file. If this parameter is specified, ASCII=YES and FILABL=STD must also be specified.

Keyword Parameter TRUNC:

(See 2.3.1 for a detailed explanation of the TRUNC parameter.)

Keyword Parameter TYPEFLE:

TYPEFLE=INPUT
Specifies that the file is an input file to be read. You cannot issue an output function to this file.

TYPEFLE=OUTPUT
Specifies that the file is an output file to be written. You cannot issue an input function to this file.

If omitted, data management assumes that the file is an input file unless you have specified that the file is to be extended. In the latter case, data management treats the file as an output file.

Keyword Parameter ULABEL:

This parameter specifies whether or not optional standard or nonstandard user header labels (UHLs) and user trailer labels (UTLs) are to be processed. A standard labeled file may have a maximum of 8 UHLs and 8 UTLs and these must follow the standard 80-byte format as described in Appendix D. For a nonstandard labeled file, there are no limitations as to the number of user labels or their contents.

ULABEL=NO

Specifies that no user label processing is to be performed. If the file is an output file, no user labels can be written on it. If it is an input file, any user labels will be bypassed.

ULABEL=YES

Specifies that user label processing is to be performed.

When user label processing is necessary during file or volume open and close processing, the appropriate function request (OPEN, CLOSE, DMFEV, DMINP, or DMOUT macroinstruction) is interrupted and exception status due to a *user label processing required condition* is reported. Also, one of the following alphabetic characters is placed in the device dependent field, CT$LCODE, to indicate when user label processing is being performed:

- O indicates at file or volume open;
- V indicates at volume close; and
- F indicates at file close.

If you intend to process user labels, you must include a test in your program after the appropriate macroinstruction to determine if it has been interrupted because user label processing required exception status has been reported. If so, you must then transfer control to a user label processing routine where you issue a DMLAB macroinstruction (see 3.20) to process these labels. No other macroinstruction can be issued until user label processing is completed. The DMLAB (IO) macroinstruction must be issued to read or write successive user labels. When there are no more user labels to process, the DMLAB (END) macroinstruction must be issued to terminate user label processing and to complete the original function request that was interrupted.

If the original function request was an OPEN or CLOSE macroinstruction, the open or close process will be completed and the appropriate status will be reported.

User label processing is necessary during the close of the current volume and the open of the next volume if the original function request was a DMFEV, DMINP, or DMOUT macroinstruction. When the DMLAB (END) macroinstruction is issued, the close process is completed, the volumes are swapped, and then the open process for the new volume is started. If label processing is required during the open of the new volume, user label processing requested exception status is reported for the DMLAB (END) macroinstruction. (This is exactly the same as was reported for the original function request.) At this point, you should transfer control to the beginning of your user label processing routine, and the DMLAB (IO) macroinstruction should be used to process the user labels during this open procedure. When there are no more user labels to process, the DMLAB (END) macroinstruction is again issued. The open process and the original function request are completed and the status that is reported is that of the DMFEV, DMINP, or DMOUT macroinstruction. Note that user label processing is bypassed at volume close when a DMFEV macroinstruction is issued to an input file. (For additional information on status reporting, see Appendix A.)

Keyword Parameter VARBLD:

`VARBLD=(r)`

Required for output files with variable-length blocked records when you are processing these in an I/O area rather than a work area. r is the number of the general register that data management loads with the number of bytes of residual space left in the current I/O area.

Keyword Parameter WKFM:

(See 2.3.1 for a detailed explanation of the WKFM parameter.)

Keyword Parameter WORK:

`WORK=NO`

Specifies that records are not to be processed in a work area. (See the IORG parameter.)

`WORK=YES`

Specifies that records will be processed in a work area rather than in an I/O area. The IORG parameter cannot be specified when WORK=YES is used. The address of the work area must be specified each time you issue an input or output imperative macroinstruction.

Example

```
RIB2 RIB BFSZ=512,FILABL=STD,IOA1=OUT,TYPEFLE=OUTPUT,RCSZ=512
```

This shows the RIB specification for the output tape file. The buffer and record size is 512 bytes, and the file is an output file. All the default values for the other keyword parameters are to be applied to the file.

Tape File Label Parameters

When a tape file is created with standard labels, the values specified for the BFSZ, RCFM, and RCSZ keyword parameters are saved in the tape labels. The tape labels are described in Appendix D.

Each time you open a file after it has been created, data management will compare the values in the tape labels with those you have specified. If they do not match, the file will not be opened and an error status is reported.

If the BFSZ, RCFM, and RCSZ parameters are defaulted (not specified), the values are determined as follows:

- For a new file, that is, TYPEFLE=OUTPUT and the EXTEND option is not specified in the LFD job control statement, the following values are used:

  BFSZ=256
  RCFM=FIXUNB
  RCSZ=256

- For an existing file, that is, TYPEFLE=INPUT or TYPEFLE=OUTPUT and the EXTEND option is specified in the LFD job control statement, the values that were saved in the tape labels will be used.

Note that if the file does not have standard labels, the new file default value will always be used if the parameter is defaulted regardless of whether it is a new or existing file.

2.3.3. RIB Macroinstruction for Printer Files

Format

```
LABEL      ΔOPERATIONΔ      OPERAND
---------------------------------------------------
name         RIB          [BFSZ=n]

                          ⎡            ⎧YES⎫ ⎤
                          ⎢,CONTROL=   ⎨   ⎬ ⎥
                          ⎣            ⎩CTL⎭ ⎦

                          ,IOA1=symbol
                          [,IOA2=symbol]

                          ⎡        ⎧NO ⎫ ⎤
                          ⎢,IORG=  ⎨   ⎬ ⎥
                          ⎣        ⎩(r)⎭ ⎦

                          ⎡        ⎧NO ⎫ ⎤
                          ⎢,OPTN=  ⎨   ⎬ ⎥
                          ⎣        ⎩YES⎭ ⎦

                          [,PRAD=n]

                          ⎡            ⎧SKIP  ⎫ ⎤
                          ⎢,PRINTOV=   ⎨REPORT⎬ ⎥
                          ⎣            ⎩YES   ⎭ ⎦

                          ⎡        ⎧FIXUNB⎫ ⎤
                          ⎢,RCFM=  ⎨VARUNB⎬ ⎥
                          ⎣        ⎩UNDEF ⎭ ⎦

                          [,RCSZ=(r)]

                          ⎡         ⎧NO ⎫ ⎤
                          ⎢,TRUNC=  ⎨   ⎬ ⎥
                          ⎣         ⎩YES⎭ ⎦

                          ⎡       ⎧OFF⎫ ⎤
                          ⎢,UCS=  ⎨   ⎬ ⎥
                          ⎣       ⎩ON ⎭ ⎦
```

LABEL	ΔOPERATIONΔ	OPERAND
name	RIB	$\left[\text{,WKFM=}\begin{Bmatrix}\text{NO}\\\text{VAR}\\\text{VARI}\end{Bmatrix}\right]$ $\left[\text{,WORK=}\begin{Bmatrix}\text{NO}\\\text{YES}\end{Bmatrix}\right]$

Keyword Parameter BFSZ:

BFSZ=n

Specifies the length of the I/O area (data buffer) in bytes. If the full size of the buffer cannot be used, then it will be rounded down to the appropriate size.

If omitted, the buffer size is automatically set as follows:

- Buffer size is 120 if CONTROL=YES and RCFM=FIXUNB.
- Buffer size is 121 if CONTROL=CTL and RCFM=FIXUNB (allows for a 1-byte device independent control character that is used to control spacing and skipping).
- Buffer size is 128 if CONTROL=YES and RCFM=VARUNB (allows for the 8-byte overhead required for variable-length records).
- Buffer size is 129 if CONTROL=CTL and RCFM=VARUNB (allows for a 1-byte control character and the 8-byte overhead required for variable-length records).

Note that the control character and the overheads are not printed; however, the buffer must be large enough to hold them.

Note also that if you want to print an odd number of characters, the buffer size must be one byte larger than the number of characters to be printed.

Keyword Parameter CONTROL:

CONTROL=YES

Indicates that spacing or skipping of lines is to be controlled by using the DMCTL, SK/SP imperative macroinstruction (see 3.13).

CONTROL=CTL

Indicates that a device independent, 2-digit, hexadecimal control character will be used with each data record to control skipping or spacing.

The control characters are shown in Table 2-2. The use of device independent characters allows a single character to be used for a particular function with any printer. With this set of characters, however, some substitutions have to be made to compensate for the characteristics of the various printers. These substitutions are shown in Table 2-2.

Table 2-2. Device Independent Control Character Codes for Printers

Function	DI Code (Hex.)	Printers			
		0789	0770	0776	9246
No-op	00				
Print and space n lines (Note 5)					
n= 0	10				
1	01				
2	02				
3	03				
4	04				
5	05				
6	06				
7	07				
8	08				
9	09				
10	0A				
11	0B				
12	0C				
13	0D				
14	0E				
15	0F				

continued

Table 2-2. Device Independent Control Character Codes for Printers (cont.)

Function	DI Code (Hex.)	Printers			
		0789	0770	0776	9246
Print and skip to code n (Note 4)					
n=1 (OV)	11		Code 12 (OV)	Code 12 (OV)	Code 12 (OV)
2	12				
3	13				
4	14				
5	15				
6	16				
7 (HP)	17		Note 2 (OV)	Code 7 (HP) Note 2	Note 2 (OV)
8	18	Code 2			
9	19	Code 1 (OV)	Note 1 (OV)	Code 12 (OV)	Note 1 (OV)
10	1A	Code 3			
11	1B	Code 4			
12	1C	Code 1 (OV)	Note 1 (OV)	Code 12 (OV)	Note 1 (OV)
13	1D	Code 5			
14	1E	Code 7 (HP)	Code 7 (HP)	Code 7 (HP)	Code 7 (HP)
15	1F	Code 7 (HP)	Code 7 (HP)	Code 7 (HP)	Code 7 (HP)
Space n lines (Note 5)			Note 3	Note 3	Note 3
n = 1	51				
2	52				
3	53				
4	54				
5	55				
6	56				
7	57				
8	58				
9	59				
10	5A				
11	5B				
12	5C				
13	5D				
14	5E				
15	5F				
Skip to code n (Note 4)					
n= 1 (OV)	21		Code 12 (OV)	Code 12 (OV)	Code 12 (OV)
2	22				
3	23				
4	24				
5	25				
6	26				
7 (HP)	27		Note 2	Code 7 (HP) Note 2	Note 2
8	28	Code 2			
9	29	Code 1 (OV)	Note 1	Code 12 (OV)	Note 1
10	2A	Code 3			
11	2B	Code 4			
12	2C	Code 1 (OV)	Note 1	Code 12 (OV)	Note 1
13	2D	Code 5			
14	2E	Code 7 (HP)	Code 7 (HP)	Code 7 (HP)	Code 7 (HP)
15	2F	Code 7 (HP)	Code 7 (HP)	Code 7 (HP)	Code 7 (HP)

continued

Table 2-2. Device Independent Control Character Codes for Printers (cont.)

LEGEND:

OV	Overflow code
HP	Home paper code

NOTES:

1. Code 12 is the primary forms overflow code on the 0770 and 9246 printers. Code 9 can also be detected as forms overflow code, using the DMCTL,PRTOV macro. If the DMCTL,PRTOV macro is not used, however, code 12 should be used as overflow code, and code 9 should not be placed in the VFB.

2. Code 7 must be used as the home paper code on the 0770, 9246 and 0776 printers.

3. For the 0770 and 0776 printers, line spacing is software-controlled via the DENSITY parameter of the VFB statement.

4. Code n specifies channel code CD1 through CD15. (See 3.13.)

5. Code n specifies number of lines to be spaced. (See 3.13.)

Table 2-3. Overflow and Home Paper Control Character Codes for Printers

Code	Printer			
	0789	0770	0776	9246
Overflow	Code 1	Code 9 Code 12*	Code 12	Code 9 Code 12*
Home paper	Code 7	Code 7	Code 7	Code 7

* Code 12 is the primary code; code 9 should not be used with the 0770 or 9246 printer unless the DMCTL,PRTOV macro is used.

Keyword Parameter IOA1:

IOA1=symbol

Specifies the symbolic address of the primary I/O area (data buffer). This area must be half-word aligned so that the character to be printed in print position 1 (excluding the control character) is on the half-word boundary. If you are going to print an odd number of characters, this area should be one byte larger than the number of characters to be printed; that is, when you define this area you should allocate an even number of bytes for it.

This parameter must be specified.

Keyword Parameter IOA2:

IOA2=symbol

Specifies the symbolic address of the secondary I/O area (data buffer). The conditions that apply to IOA1 also apply to this area.

Keyword Parameter IORG:

(See 2.3.1 for a detailed description of the IORG parameter.)

Keyword Parameter OPTN:

(See 2.3.1 for a detailed description of the OPTN parameter.)

Keyword Parameter PRAD:

PRAD=n

Specifies the number of lines the form is to be advanced after printing. n is the number of lines and ranges from 1 through 15.

If omitted, 1 is assumed unless CONTROL=CTL is specified, in which case the control character determines line advancement.

Keyword Parameter PRINTOV:

PRINTOV=SKIP

Specifies that, upon detecting a forms overflow condition, data management will cause an automatic skip to the home paper position.

PRINTOV=REPORT

Specifies that, upon detecting a forms overflow condition, data management will inform you by reporting exception status due to a forms overflow condition. See Appendix A for additional information on exception status.

PRINTOV=YES

Specifies that the DMCTL, PRTOV imperative macroinstruction (see 3.14) will be used to control the detection of forms overflow and subsequent actions.

Keyword Parameter RCFM:

RCFM=FIXUNB

Specifies fixed-length unblocked records. Because there is no concept of blocked records for a printer file, RCFM=FIXBLK will be treated exactly as RCFM=FIXUNB.

RCFM=VARUNB

Specifies variable-length unblocked records. Because there is no concept of blocked records for a printer file, RCFM=VARBLK will be treated exactly as RCFM=VARUNB.

RCFM=UNDEF
Specifies undefined records.

Keyword Parameter RCSZ:

RCSZ=(r)
Specifies the number of the general register that holds the size of the output record. Registers 2 through 12 may be used. This parameter is required for output files with undefined record format.

Keyword Parameter TRUNC:

(See 2.3.1 for a detailed explanation of the TRUNC parameter.)

Keyword Parameter UCS:

UCS=OFF
Specifies that printer mismatches are to be ignored. Mismatches occur whenever the printer attempts to print a bit configuration that is not present in its load code buffer. The nonprintable character is replaced by a blank.

UCS=ON
Specifies that the operator is to be notified when printer mismatches occur.

Keyword Parameter WKFM:

(See 2.3.1 for a detailed explanation of the WKFM parameter.)

Keyword Parameter WORK:

WORK=NO
Specifies that records are not to be processed in a work area. (See IORG parameter).

WORK=YES
Specifies that records will be processed in a work area rather than in the I/O area. When you issue an output operation, you must specify the address of the work area with the appropriate macroinstruction. If both IORG=(r) and WORK=YES are specified, the IORG parameter is ignored. When work area processing is being used, the work area and the I/O area should not be the same buffer.

Example

```
RIB3 RIB IOA1=OUT
```

This shows the RIB specification for a printer file. The symbolic address of IOA1 is OUT. All of the default values for the other keyword parameters are to be used.

2.3.4. RIB Macroinstruction for Card (Reader and Punch) Files

Format

```
LABEL       ΔOPERATIONΔ    OPERAND
-----------------------------------------------------
name          RIB          ⎡,AUE=⎧NO ⎫⎤
                           ⎣     ⎩YES⎭⎦
                           [,BFSZ=n]
                           ⎡,CDMODE= ⎧STD   ⎫⎤
                           ⎢         ⎨CC    ⎬⎥
                           ⎣         ⎩BINARY⎭⎦
                           ,IOA1=symbol
                           [,IOA2=symbol]
                           ⎡,IORG= ⎧NO ⎫⎤
                           ⎣       ⎩(r)⎭⎦
                           [,ITBL=symbol]
                           ⎡,OPTN= ⎧NO ⎫⎤
                           ⎣       ⎩YES⎭⎦
                           ⎡,ORLP= ⎧NO ⎫⎤
                           ⎣       ⎩YES⎭⎦
                           [,OTBL=symbol]
                           [,OUBLKSZ=n]
                           ⎡,RCFM= ⎧FIXUNB⎫⎤
                           ⎢       ⎨VARUNB⎬⎥
                           ⎣       ⎩UNDEF ⎭⎦
                           [,RCSZ=(r)]
                           ⎡,STUB= ⎧NO⎫⎤
                           ⎢       ⎨51⎬⎥
                           ⎣       ⎩66⎭⎦
```

```
LABEL       ΔOPERATIONΔ        OPERAND
---------------------------------------------------
                         [,TRANS= ⎧NO   ⎫ ]
                         [        ⎨YES  ⎬ ]
                         [        ⎩ASCII⎭ ]

                         [,TRUNC= ⎧NO ⎫ ]
                         [        ⎩YES⎭ ]

                         [,TYPEFLE= ⎧INPUT ⎫ ]
                         [          ⎨OUTPUT⎬ ]
                         [          ⎩INOUT ⎭ ]

                         [,WKFM= ⎧NO  ⎫ ]
                         [       ⎨VAR ⎬ ]
                         [       ⎩VARI⎭ ]

                         [,WORK= ⎧YES⎫ ]
                         [       ⎩NO ⎭ ]
```

Keyword Parameter AUE:

AUE=NO
Specifies that error processing is not to be inhibited when a validity check error is detected on a nonbinary input file. A validity check error will cause the unique unit error indicator to be set.

AUE=YES
Specifies that error processing is to be inhibited when a validity check error is detected on a nonbinary input file. A validity check error will cause a PIOCS message, which indicates the problem, to be displayed on the system console. The card containing the error will be the last card in the stacker. The operator has the option of having the card reread.

Keyword Parameter BFSZ:

BFSZ=n

Specifies the length of the I/O area (data buffer) in bytes. For variable-length records, n specifies the maximum size for records (this includes the block and record headers but only the record header is punched). If a value is specified that is greater than what is required, it will be rounded down to the appropriate value depending upon the particular device and the CDMODE, RCFM, and STUB keyword specifications.

If omitted, the buffer size is automatically set as follows:

- For fixed-length records, the buffer size is 80.
- For variable-length records, the buffer size is 84.

Keyword Parameter CDMODE:

CDMODE=STD

Specifies that the cards are to be read or punched in EBCDIC if TRANS=NO is specified. If TRANS=ASCII, the cards are to be read or punched in ASCII.

CDMODE=CC

Specifies that the cards are to be read or punched in compressed card code.

CDMODE=BINARY

Specifies that the cards are to be read or punched in column binary (image) mode; a buffer size and I/O area of 160 bytes is required for one 80-column card.

Keyword Parameter IOA1:

IOA1=symbol

Specifies the symbolic address of the primary I/O area (data buffer). This area must be half-word aligned so that the first character to be read or punched is on the half-word boundary. If you are going to read or punch an odd number of characters, this area should be one byte larger than the number of characters; that is, when you define this area you should allocate an even number of bytes.

This parameter must be specified.

Keyword Parameter IOA2:

IOA2=symbol

Specifies the symbolic address of the secondary I/O area (data buffer). The conditions that apply to IOA1 also apply to this area.

Keyword Parameter IORG:

(See 2.3.1 for a detailed explanation of the IORG parameter.)

Keyword Parameter ITBL:

ITBL=symbol

Specifies the symbolic address of your 256-byte translation table that is to be used for input translation. If this parameter is specified, TRANS=YES must also be specified.

Keyword Parameter OPTN:

(See 2.3.1 for a detailed explanation of the OPTN parameter.)

Keyword Parameter ORLP:

ORLP=NO

Specifies that a combined file is to be processed in the nonoverlap mode when using a punch unit with a prepunch read station.

In this mode you can read a card and punch data into that card. You do this in your program by issuing a DMINP imperative macroinstruction and then a DMOUT macroinstruction. This will cause a card to be read and then cause data to be punched on that card.

ORLP=YES

Specifies that a combined file is to be processed in the overlap mode when using a card/punch unit with a prepunch read station.

In this mode you overlap input and output processing by alternating prepunched data cards with blank cards in your input deck. Then, in your program, you use the DMINP imperative macroinstruction to read a prepunched card and the DMOUT macroinstruction to punch data onto a successive blank card.

The possible combinations are:

1. Alternating DMINP and DMOUT macroinstructions, when used with alternating prepunched and blank cards, produce valid results if overlap is specified. Each DMINP macroinstruction applies to prepunched input data cards, and each DMOUT macroinstruction applies to punching data into a blank card.

2. Multiple DMINP macroinstructions between single DMOUT macroinstructions (when used with multiple prepunched cards between single blank cards) produce valid results if, in every case, the number of DMINP macroinstructions corresponds to the number of prepunched cards between each of the blanks that the DMOUT macroinstructions reference.

3. Multiple DMINP and multiple DMOUT macroinstructions (when used with multiple prepunched cards between multiple blanks) produce valid results if the number of DMINP macroinstructions and DMOUT macroinstructions and the number of prepunched and blank cards are consistent through the program.

Keyword Parameter OTBL:

OTBL=symbol
: Specifies the symbolic address of your 256-byte translation table that is to be used for output translation. If this parameter is specified, TRANS=YES must also be specified.

Keyword Parameter OUBLKSZ:

OUBLKSZ=n
: Specifies the length of the secondary I/O area (data buffer) in bytes.

If omitted, the value specified by the BFSZ parameter is used.

Keyword Parameter RCFM:

RCFM=FIXUNB
: Specifies fixed-length unblocked records.

Because there is no concept of blocked records for a card file, RCFM=FIXBLK will be treated exactly as RCFM=FIXUNB.

This must be specified for input or combined files.

RCFM=VARUNB
: Specifies variable-length unblocked records. The first four bytes of a variable-length record are the record descriptor word (RDW). The first two bytes of the RDW contain the physical record size that you supply on output and data management supplies on input. The physical record size specified includes the four bytes for RDW.

Because there is no concept of blocked records for a card file, RCFM=VARBLK will be treated exactly as RCFM=VARUNB.

This specification can only be used for output files.

RCFM=UNDEF
: Specifies undefined records.

This specification can only be used for output files.

Keyword Parameter RCSZ:

RCSZ=(r)
Specifies the number of the general register that holds the size of the output record when undefined records are used. Registers 2 through 12 may be used.

If this parameter is specified, RCFM=UNDEF must also be specified.

Keyword Parameter STUB:

STUB=NO
Specifies that 80-column cards are being processed.

STUB=51
Specifies that 51-column cards are being processed. This specification can be used only when the card unit has the 51-column stub card read feature.

STUB=66
Specifies that 66-column cards are being processed. This specification can be used only when the card unit has the 66-column stub card read feature.

Keyword Parameter TRANS:

TRANS=NO
Specifies that no translation is to be performed.

TRANS=YES
Specifies that input or output translation is to be performed using the translation table you supplied. If this parameter is specified, the ITBL or OTBL parameter must be specified. In addition, when TRANS=YES is specified, the CDMODE parameter is ignored.

TRANS=ASCII
Specifies that the system-supplied ASCII translation is to be used.

Keyword Parameter TRUNC:

(See 2.3.1 for a detailed explanation of the TRUNC parameter.)

Keyword Parameter TYPEFLE:

This parameter applies only to a punch device that has the read feature. If you are using a device without the read feature, this parameter is ignored and the appropriate specification is automatically determined (INPUT for reader file; OUTPUT for punch file).

TYPEFLE=INPUT
Specifies that the file is an input file. This parameter can only be specified if you are using a punch device with the read feature strictly as a card reader.

TYPEFLE=OUTPUT

Specifies that the file is an output file. This parameter can only be specified if you are using a punch device with the read feature strictly as a card punch.

TYPEFLE=INOUT

Specifies that the file is a combined file. This parameter can only be specified if you are using a punch device with the read feature to both read and punch cards. (See the ORLP parameter for more information).

Keyword Parameter WKFM:

(See 2.3.1 for a detailed explanation of the WKFM parameter.)

Keyword Parameter WORK:

WORK=YES

Specifies that records will be processed in a work area rather than in the I/O area. When you issue an input or output operation, you must specify the address of the work area with the appropriate macroinstruction. If both IORG=(r) and WORK=YES are specified, the IORG parameter is ignored. When work-area processing is being used, the work area and the I/O area should not be the same buffer.

WORK=NO

Specifies that records are not to be processed in a work area. (See the IORG parameter.)

Example

```
RIB4 RIB IOA1=INP,TYPEFLE=INOUT,ORLP=YES
```

This shows the RIB specification for a combined file that is to be processed in the overlap mode. The symbolic address of IOA1 is INP, and all of the default values for the other keyword parameters are to be used.

2.3.5. RIB Macroinstruction for Data Set Label Diskette Files

Format

LABEL	ΔOPERATIONΔ	OPERAND
name	RIB	[BFSZ=n] ,IOA1=symbol [,IOA2=symbol] $\left[\text{,IORG=}\begin{Bmatrix}\text{NO}\\\text{(r)}\end{Bmatrix}\right]$ $\left[\text{,MODE=}\begin{Bmatrix}\text{SEQ}\\\text{RAN}\end{Bmatrix}\right]$ $\left[\text{,OPTN=}\begin{Bmatrix}\text{NO}\\\text{YES}\end{Bmatrix}\right]$ $\left[\text{,RCFM=}\begin{Bmatrix}\text{FIXUNB}\\\text{VARUNB}\end{Bmatrix}\right]$ [,RCSZ=n] $\left[\text{,RECATR=}\begin{Bmatrix}\text{UU}\\\text{BS}\\\text{BU}\end{Bmatrix}\right]$ $\left[\text{,RETR=}\begin{Bmatrix}\text{INF}\\\text{MOD}\end{Bmatrix}\right]$ [,SKAD=symbol] $\left[\text{,TRUNC=}\begin{Bmatrix}\text{NO}\\\text{YES}\end{Bmatrix}\right]$ $\left[\text{,VMNT=}\begin{Bmatrix}\text{NO}\\\text{ONE}\end{Bmatrix}\right]$

LABEL	ΔOPERATIONΔ	OPERAND
		$\left[,\text{VKFM=}\begin{Bmatrix}\text{NO}\\ \text{VAR}\\ \text{VARI}\end{Bmatrix}\right]$
		$\left[,\text{WORK=}\begin{Bmatrix}\text{YES}\\ \text{NO}\end{Bmatrix}\right]$
		$\left[,\text{WPROTC=}\begin{Bmatrix}\text{YES}\\ \text{ERASE}\\ \text{NO}\end{Bmatrix}\right]$
		$\left[,\text{WPROTO=}\begin{Bmatrix}\text{YES}\\ \text{ERASE}\\ \text{NO}\end{Bmatrix}\right]$

Keyword Parameter BFSZ:

BFSZ=n

Specifies the length of the I/O area (data buffer) in bytes. If the full size of the buffer cannot be used, it will be automatically rounded down to the appropriate size.

The file can be accessed using a buffer size that is different from the buffer size used at file creation. When the file is opened, data management verifies that the specified buffer size is at least as large as the minimum allowable value. A buffer size that exceeds the minimum allowable value may be specified to enhance performance.

The algorithm for determining the minimum allowable value in logical sectors is as follows:

$$\frac{S}{LSS} = N \text{ and } R$$

where:

S

Is the slot size. The slot size is equal to the record size.

LSS

Is the logical sector size. The logical sector size is determined by the RECATR parameter.

N

Is the number of full sectors per slot.

R

Is the remainder.

The minimum allowable value in logical sectors is:

N;	if R=0
N+1;	if R divides evenly into LSS
N+2;	otherwise

After you have determined what the minimum allowable value in logical sectors is, you then multiply this number by the logical sector size to determine the length of your I/O area.

(See "Label Parameters for a Data Set Label Diskette File" in this section, for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter IOA1:

(See 2.3.1 for a detailed explanation of the IOA1 parameter.)

Keyword Parameter IOA2:

(See 2.3.1 for a detailed explanation of the IOA2 parameter.)

Keyword Parameter IORG:

(See 2.3.1 for a detailed explanation of the IORG parameter.)

Keyword Parameter MODE:

(See 2.3.1 for a detailed explanation of the MODE parameter.)

Keyword Parameter OPTN:

(See 2.3.1 for a detailed explanation of the OPTN parameter.)

RCFM=FIXUNB

Specifies fixed-length unblocked records.

Because there is no concept of blocked records for a data set label diskette file, RCFM=FIXBLK and RCFM=UNDEF will be treated exactly as RCFM=FIXUNB.

RCFM=VARUNB

Specifies variable-length unblocked records. The first four bytes of a variable-length record are the record descriptor word (RDW). The first two bytes of the RDW contain the physical record size that you supply on output and data management supplies on input. The physical record size specified includes the four bytes for RDW.

Because there is no concept of blocked records for a data set label diskette file, RCFM=VARBLK will be treated exactly as RCFM=VARUNB.

(See "Label Parameters for a Data Set Label Diskette File" in this section, for additional information on label checking and default values when a file is opened.)

Keyword Parameter RCSZ:

RCSZ=n

Specifies the length of the record in bytes. For variable-length records, this reflects the slot size and should include the 4-byte RDW.

(See "Label Parameters for a Data Set Label Diskette File" in this section, for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter RECATR:

RECATR=UU

Specifies that the records are unblocked and unspanned. The record size must be less than or equal to the physical sector size (PSS) of the diskette. The logical sector size (LSS) is equal to the record size.

RECATR=BS

Specifies that the records are blocked and spanned. This must be specified if the record size is greater than the physical sector size (PSS) of the diskette. The logical sector size (LSS) is equal to the PSS.

RECATR=BU

Specifies that the records are blocked and unspanned. If this is specified, the record size must be less than or equal to the physical sector size (PSS) of the diskette. The logical sector size (LSS) is equal to n times the record size, where n is the number of whole records that fit in a physical sector.

Note that the physical sector size (PSS) will be 128, 256, or 512 bytes depending on the recording mode that was established when the diskette was prepped.

(See "Label Parameters for a Data Set Label Diskette File" in this section, for additional information on label checking and default values for this parameter when a file is opened.)

Keyword Parameter RETR:

RETR=INF

Specifies that records are to be retrieved for information purposes only. If this is specified, update operations are not permitted.

RETR=MOD

Specifies that records are to be retrieved for modification. If this is specified, update operations are permitted.

Keyword Parameter SKAD:

(See 2.3.1 for a detailed explanation of the SKAD parameter.)

Keyword Parameter TRUNC:

(See 2.3.1 for a detailed explanation of the TRUNC parameter.)

Keyword Parameter VMNT:

VMNT=NO

Specifies that the file is to be processed with all volumes online. Random operations are permitted.

VMNT=ONE

Specifies that the file is to be processed with only one volume online at a time. Random operations are not permitted. This must be specified for multivolume files.

Keyword Parameter WKFM:

(See 2.3.1 for a detailed explanation of the WKFM parameter.)

Keyword Parameter WORK:

(See 2.3.1 for a detailed explanation of the WORK parameter.)

Keyword Parameter WPROTC:

`WPROTC=YES`
Specifies that the file is to be marked in the file label as write protected when it is closed.

`WPROTC=ERASE`
Specifies that the file is to be marked in the file label as *not* write protected when it is closed.

`WPROTC=NO`
Specifies that the existing write protection indicator in the file label is not to be changed when the file is closed.

Keyword Parameter WPROTO:

`WPROTO=YES`
Specifies that the file is to be marked in the file label as write protected when the file is opened.

`WPROTO=ERASE`
Specifies that the file is to be marked in the file label as *not write protected when it is opened.*

`WPROTO=NO`
Specifies that the existing write protection indicator in the file label is not to be changed when the file is opened.

Example

```
RIB5 RIB BKSZ=512,IOA1=IN,WPROTO=YES,VMNT=ONE
```

This shows the RIB specification for a data set label diskette file. The file is to be write protected when it is opened, the buffer size is 512 bytes, the symbolic address of IOA1 is IN, and the file is to be processed with only one volume online at a time.

Label Parameters for a Data Set Label Diskette File

When a data set label diskette file is created, the values specified for the BFSZ, RCFM, RCSZ, and RECATR keyword parameters are saved in the diskette data set label. The diskette data set label is described in Appendix C.

Each time you open a file after it has been created, data management will compare the values in the diskette data set label with those you have specified. If they do not match, the file will not be opened and an error status is reported.

If the BFSZ, RCFM, RCSZ, and RECATR parameters are defaulted (not specified), the values are determined as follows:

- For a new file (one that is being created (newly allocated) or one for which the INIT option is specified in the LFD job control statement), the following values are used:

  BFSZ=minimum allowable value (see the BFSZ parameter for the algorithm used to determine the minimum allowable value).

  RCFM=FIXUNB

  RCSZ=128

  RECATR=UU

- For an existing file (one that is to be read, extended or updated), the values that were saved in the diskette data set label will be used.

  Note that the buffer size that is used when the file is created is not saved in the diskette data set label. It is not saved because the buffer can vary as long as it is at least as large as the minimum allowable value. If the buffer size is defaulted in this case, the minimum allowable value will be used. (See the BFSZ parameter for the algorithm used to determine the minimum allowable value.)

2.3.6. RIB Macroinstruction for Workstation Files

Format

```
LABEL      ΔOPERATIONΔ     OPERAND
--------------------------------------------------------
name        RIB            [BFSZ=n]

                           ⎡ ,CONFRE= ⎧NO ⎫ ⎤
                           ⎣          ⎩YES⎭ ⎦
                           [,FNKEYS=symbol]

                           ⎡,INCSR= ⎧YES⎫ ⎤
                           ⎣        ⎩NO ⎭ ⎦

                           ⎡,INPTRAN= ⎧YES⎫ ⎤
                           ⎣          ⎩NO ⎭ ⎦
                           [,IOA1=symbol]
                           [,IOA2=symbol]

                           ⎡,IOOPT= ⎧YES⎫ ⎤
                           ⎣        ⎩NO ⎭ ⎦
                           [,IORG=(r)]

                           ⎡,IREC= ⎧APPEND⎫ ⎤
                           ⎢       ⎨NL    ⎬ ⎥
                           ⎣       ⎩NP    ⎭ ⎦

                           ⎡,ISOE= ⎧NO ⎫ ⎤
                           ⎣       ⎩REC⎭ ⎦
                           [,ITBL=symbol]

                           ⎡,KBRDIN= ⎧LOCK  ⎫ ⎤
                           ⎣         ⎩UNLOCK⎭ ⎦

                           ⎡,KBRDOUT= ⎧LOCK  ⎫ ⎤
                           ⎣          ⎩UNLOCK⎭ ⎦
```

```
LABEL      ΔOPERATIONΔ      OPERAND

name        RIB            ⎡,LOG=  ⎧NO    ⎫⎤
                           ⎢       ⎪BOTH  ⎪⎥
                           ⎢       ⎨INPUT ⎬⎥
                           ⎣       ⎩OUTPUT⎭⎦

                           ⎡,OPTN=  ⎧NO ⎫⎤
                           ⎣        ⎩YES⎭⎦

                           ⎡,OREC=  ⎧APPEND⎫⎤
                           ⎢        ⎨NL    ⎬⎥
                           ⎢        ⎪NP    ⎪⎥
                           ⎣        ⎩NLB   ⎭⎦

                           ⎡,OSOE=  ⎧BLK⎫⎤
                           ⎢        ⎨NO ⎬⎥
                           ⎣        ⎩REC⎭⎦

                           [,OTBL=symbol]

                           ⎡,PMODE=  ⎧CNTRLDD⎫⎤
                           ⎣         ⎩WSAM   ⎭⎦

                           ⎡,RCFM=  ⎧FIXBLK⎫⎤
                           ⎢        ⎪FIXUNB⎪⎥
                           ⎢        ⎨UNDEF ⎬⎥
                           ⎢        ⎪VARBLK⎪⎥
                           ⎣        ⎩VARUNB⎭⎦

                           ⎡,RCSZ=  ⎧n  ⎫⎤
                           ⎢        ⎨(r)⎬⎥
                           ⎣        ⎩80 ⎭⎦

                           ⎡,SCREND=  ⎧NP    ⎫⎤
                           ⎢          ⎨SCROLL⎬⎥
                           ⎣          ⎩WRAP  ⎭⎦

                           [,SPACELN=n]
```

LABEL	ΔOPERATIONΔ	OPERAND
name	RIB	$\left[\text{,STRTLN=}\begin{Bmatrix}\text{FIRST}\\ \text{nn}\end{Bmatrix}\right]$ $\left[\text{,TRUNC=}\begin{Bmatrix}\text{NO}\\ \text{YES}\end{Bmatrix}\right]$ $\left[\text{,WAIT=}\begin{Bmatrix}\text{NO}\\ \text{YES}\end{Bmatrix}\right]$ $\left[\text{,WKFM=}\begin{Bmatrix}\text{NO}\\ \text{VAR}\\ \text{VARI}\end{Bmatrix}\right]$ $\left[\text{,WORK=}\begin{Bmatrix}\text{NO}\\ \text{YES}\end{Bmatrix}\right]$

Keyword Parameter BFSZ:

BFSZ=n
: Specifies the length of the I/O area (data buffer) in bytes.

If omitted, a buffer size will be established that is equal to the physical screen size plus the number of control bytes necessary to provide screen management.

Keyword Parameter CONFRE:

CONFRE=NO
: Specifies that the connecting and freeing of a workstation is not to be reported.

CONFRE=YES
: Specifies that the connecting and freeing of a workstation is to be reported. If this is specified, control is returned to the caller and exception status due to a connecting or freeing of a workstation is reported.

For further information on status reporting, see Appendix A.

Keyword Parameter FNKEYS:

FNKEYS=symbol

Specifies that the program processes function keys and specifies the symbolic address of a 4-byte area where the function keys will be returned to the program. The 4-byte area has the following form:

Bit Byte	0	1	2	3	4	5	6	7
0	F1	F2	F3	F4	F5	F6	F7	F8
1	F9	F10	F11	F12	F13	F14	F15	F16
2	F17	F18	F19	F20	F21	F22	Reserved	
3	Reserved						Message Waiting	

When a particular function key is pressed, the bit position in the table that corresponds to that key is set. For further information, see 4.2.3.

Keyword Parameter INCSR:

INCSR=YES

Specifies that the character under the cursor is to be returned to the user buffer.

INCSR=NO

Specifies that the character under the cursor is *not* to be returned to the user buffer.

Keyword Parameter INPTRAN:

This parameter can be used only when PMODE=WSAM is specified.

INPTRAN=YES

Specifies that lowercase to uppercase translation is to be performed on input.

INPTRAN=NO

Specifies that lowercase to uppercase translation is *not* to be performed on input.

Note: *INPTRAN=YES overrides the SCREEN LOWER interactive services command. INPTRAN=NO overrides the SCREEN UPPER interactive services command.*

Keyword Parameter IOA1:

IOA1=symbol
: Specifies the symbolic address of an I/O area (data buffer). This area must be half-word boundary aligned.

If omitted, data management will get a buffer from the system pool.

Keyword Parameter IOA2:

IOA2=symbol
: Specifies the symbolic address of a secondary I/O area (data buffer). This area must be half-word boundary aligned.

Keyword Parameter IOOPT:

IOOPT=YES
: Specifies that IOA1 will be the input buffer and IOA2 will be the output buffer.

IOOPT=NO
: Specifies that IOA1 and IOA2 will be used interchangeably for I/O operations. If this is specified, IORG=(r) must also be specified.

Keyword Parameter IORG:

IORG=(r)
: Specifies the number of the general register to be used to point to the current record in the I/O area when you do not process records in a work area.

Registers 2 through 12 may be used. If this parameter is specified, this automatically sets the WORK=NO specification. (See the WORK parameter.)

Keyword Parameter IREC:

This parameter can be used only when PMODE=WSAM is specified.

IREC=APPEND
: Specifies that after data is read from the screen, the cursor is not moved. The next data record will immediately follow the previous record.

IREC=NL
: Specifies that after data is read from the screen, the remainder of the line is blanked and the cursor is advanced to the next line.

IREC=NP
: Specifies that after data is read from the screen, the screen is cleared and the cursor is returned to its home position on the screen (column 1 on the line specified by the STRTLN parameter).

Keyword Parameter ISOE:

This parameter can be used only when PMODE=WSAM is specified.

ISOE=NO
Specifies that a start-of-entry character (SOE) will not be displayed after an input record is read from the screen.

ISOE=REC
Specifies that a start-of-entry character (SOE) will be displayed after an input record is read from the screen.

Keyword Parameter ITBL:

This parameter can be used only when PMODE=WSAM is specified.

ITBL=symbol
Specifies the symbolic address of a 256-byte input translation table that you supply.

Keyword Parameter KBRDIN:

The parameter can be used only when PMODE=WSAM is specified.

KBRDIN=LOCK
Specifies that the workstation keyboard will be locked after an input operation is performed.

KBRDIN=UNLOCK
Specifies that the workstation keyboard will be unlocked after an input operation is performed.

Keyword Parameter KBRDOUT:

This parameter can be used only when PMODE=WSAM is specified.

KBRDOUT=LOCK
Specifies that the workstation keyboard is to be locked after an output operation is performed.

KBRDOUT=UNLOCK
Specifies that the workstation keyboard is to be unlocked after an output operation is performed.

Keyword Parameter LOG:

This parameter can be used only when PMODE=WSAM is specified.

`LOG=NO`
Specifies that data sent to or read from the screen is *not to be logged.*

`LOG=BOTH`
Specifies that data sent to or read from the screen is to be logged.

`LOG=INPUT`
Specifies that data read from the screen is to be logged.

`LOG=OUTPUT`
Specifies that data sent to the screen is to be logged.

Keyword Parameter OPTN:

(See 2.3.1 for a detailed explanation of the OPTN parameter.)

Keyword Parameter OREC:

This parameter can be used only when PMODE=WSAM is specified.

`OREC=APPEND`
Specifies that after data is displayed on the screen, the cursor is not moved. The next data record will be displayed immediately following the previous record.

`OREC=NL`
Specifies that after data is displayed on the screen, the remainder of the line is blanked and the cursor is advanced to the next line.

`OREC=NP`
Specifies that before each data record is displayed, the screen is cleared. After the data is displayed, the cursor is returned to its home position (column 1 on the line specified by the STRTLN parameter).

`OREC=NLB`
Specifies that before each data record is displayed, the remainder of the current line is blanked and the cursor is advanced to the next line. The data is displayed on that line and the remainder of the line is blanked.

Keyword Parameter OSOE:

This parameter can be used only when PMODE=WSAM is specified.

`OSOE=BLK`
Specifies that a start-of-entry character (SOE) is to be inserted at the end of a logical block of records.

`OSOE=NO`
Specifies that a start-of-entry character (SOE) will not be displayed after each output record is displayed on the screen.

`OSOE=REC`
Specifies that a start-of-entry character (SOE) will be displayed at the character position specified by the OREC parameter after a data record is displayed on the screen.

Keyword Parameter OTBL:

This parameter can be used only when PMODE=WSAM is specified.

`OTBL=symbol`
Specifies the symbolic address of a 256-byte output translation table that you supply.

Keyword Parameter PMODE:

`PMODE=CNTRLDD`
Specifies that the device dependent mode will be used to display and retrieve data from the workstation. In this mode, you control screen management by including device dependent control character sequences within your data records.

`PMODE=WSAM`
Specifies that screen management will be controlled in accordance with the specifications you provide in the IREC, ISOE, KBRDIN, KBRDOUT, OREC, OSOE, SCREND, SPACELN, and STRTLN parameters.

Keyword Parameter RCFM:

`RCFM=FIXBLK`
Specifies that the records are fixed-length and blocked. Record length is defined by the RCSZ=n parameter. The records are blocked based on the physical screen size for output operations if PMODE=WSAM; otherwise, they are processed as fixed-unblocked records.

RCFM=FIXUNB
Specifies that the records are fixed-length and unblocked. Record length is defined by the RCSZ parameter.

RCFM=UNDEF
Specifies that the records are of undefined length and unblocked. The record length is contained in the register specified in the RCSZ=(r) parameter. A maximum record size of screen size is assumed for PMODE=WSAM.

RCFM=VARBLK
Specifies that the records are variable-length and blocked. The physical record length is specified in the first two bytes of the record descriptor word (RDW) that appears on the front of the record. If this parameter is specified, the RCSZ parameter is the maximum record size for PMODE=WSAM. The records are blocked based on the physical screen size for output operations if PMODE=WSAM and SCREND=NP; otherwise, the records are processed as variable-unblocked records.

RCFM=VARUNB
Specifies that the records are variable-length and unblocked. The physical record length is specified in the first two bytes of the record descriptor word (RDW) that appears on the front of the record. If this parameter is specified, the RCSZ parameter is the maximum record size for PMODE=WSAM.

Keyword Parameter RCSZ:

RCSZ=n
Specifies the length of the record, that is, the number of characters. This should be less than or equal to the physical screen size.

RCSZ=(r)
Specifies the number of the general register that is to contain the record length. Registers 2 through 12 may be used.

RCSZ=80
Specifies that the record length is 80 characters.

Keyword Parameter SCREND:

This parameter can be used only when PMODE=WSAM is specified.

SCREND=NP
Specifies that when the display reaches the end of the screen, the screen is cleared, and the cursor is returned to its home position, that is, column 1 on the line specified by the STRTLN parameter.

SCREND=SCROLL
Specifies that when the display reaches the end of the screen, the screen scrolls up.

SCREND=WRAP
Specifies that when the display reaches the end of the screen, the remainder of the display continues at the cursor home position without clearing the screen.

Note: *SCREND=SCROLL or NP overrides the SCREEN WRAP interactive services command. SCREND=WRAP or NP overrides the SCREEN SCROLL or ROLL interactive services command.*

Keyword Parameter SPACELN:

This parameter can be used only when PMODE=WSAM is specified.

SPACELN=n
Specifies the action to be taken when scrolling is in effect. It allows you to specify the number of lines on the screen to be available for input. The number of lines specified can range from 1 to 9.

Keyword Parameter STRTLN:

This parameter can be used only when PMODE=WSAM is specified.

STRTLN=FIRST
Specifies that the first line on the screen is the first line, that is, line 1.

STRTLN=nn
Specifies the number of the first line on the screen.

For further information, see the SCREND parameter.

Keyword Parameter TRUNC:

(See 2.3.1 for a detailed explanation of the TRUNC parameter.)

Keyword Parameter WAIT:

WAIT=NO
Specifies that when an I/O operation is performed, control will be immediately returned to the program. If this is specified, a DMWTF imperative macroinstruction must follow the I/O operation in your program to ensure that the transfer of a record between the program and the workstation has been completed.

WAIT=YES

Specifies that when an I/O operation is performed, control will not be returned to the program until the transfer of a record between the program and the workstation is completed.

Keyword Parameter WKFM:

(See 2.3.1 for a detailed description of the WKFM parameter.)

Keyword Parameter WORK:

WORK=NO

Specifies that records are not to be processed in a work area (see IORG parameter).

WORK=YES

Specifies that records will be processed in a work area rather than an I/O area. When you issue an input or output operation, you specify the address of the work area with the appropriate imperative macroinstruction. If both IORG=(r) and WORK=YES are specified, the IORG parameter is ignored.

Example

```
RIB7 RIB BFSZ=200, IOA1=IN1, IOA2=OUT1
```

This shows the RIB specification for a workstation file in which the input and output buffers are identified, and the buffer size is 200 characters rather than the default value.

Section 3
Imperative Macroinstructions

3.1. General

You use the imperative macroinstructions in your program to open and close files, bring data in from and send it out to the various peripheral devices in your system, and to control such things as print form movement and workstation screen formatting. The instructions you use to cause these actions are: OPEN, CLOSE, DMINP, DMOUT, DMDEL, DMUPD, DMFEV, DMERS, DMCTL, DMSEL, DMDSP, DMLAB, and DMBRK.

For the most part, these instructions can be used with all types of files. There are exceptions, however, and these are covered in the descriptions in the following subsections.

3.1.1. Register Conventions

When you use the imperative macroinstructions, data management expects the following registers to be used as indicated:

- Register 0

  - The RIB address must be loaded in this register for the OPEN macroinstruction when a RIB is used.

  - The work-area address must be loaded in this register for those macroinstructions that use work-area processing.

- Register 1

  The CDIB address must always be loaded in this register.

If symbolic notation is used, the expansion of the imperative macroinstruction will load register 0 and/or register 1 with the appropriate address.

All registers are returned unchanged when control is received back from an imperative macroinstruction. The only exception to this is that register 14 is destroyed when control is received back from an OPEN or CLOSE macroinstruction.

3.1.2. Status Checking

Status checking is required after each imperative macroinstruction. This is necessary because control is always returned inline. For additional information on status checking, see Appendix A.

3.2. Open a File (OPEN)

You use this macroinstruction to initialize a file. When this instruction is issued, a logical access path (LAP) is established to the physical file. Only one file can be opened when you issue this instruction. If you want to access more than one file, you must issue an OPEN instruction for each of the files. This instruction must be issued for a file before you attempt any processing.

Format

```
LABEL     ΔOPERATIONΔ   OPERAND
------------------------------------------------------
[name]    OPEN          ⎧cdibname⎫ ⎡ ,⎧(ribname)⎫ ⎤
                        ⎨(1)     ⎬ ⎢  ⎨(0)      ⎬ ⎥
                        ⎩1       ⎭ ⎣  ⎩0        ⎭ ⎦
```

Positional Parameter 1:

`cdibname`
Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
Indicates that you have preloaded register 1 with the address of the CDIB for the file to be opened.

Positional Parameter 2:

`ribname`
Is the symbolic address of the RIB associated with the file to be opened.

`(0) or 0`
Indicates that you have preloaded register 0 with the address of the RIB associated with the file. If you load register 0 with a value of zero, this will indicate that there is no RIB associated with the file. (This also means that a RIB cannot reside at relative address X'000000'.)

If a symbolic address is specified for both positional parameter 1 and positional parameter 2, the comma shown in the instruction format is optional.

Example

```
OPEN FILE1,(RIB1)
```

3.3. Close a File (CLOSE)

This macroinstruction is used to terminate file processing. Once a file is closed, it cannot be accessed until it is reopened by issuing an OPEN macroinstruction. Only one file can be closed when you issue this instruction. If you have more than one file open in your program, you must issue a CLOSE instruction for each of the files.

Format

LABEL	ΔOPERATIONΔ	OPERAND
[name]	CLOSE	cdibname (1) 1

Positional Parameter 1:

cdibname
: Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

(1) or 1
: Indicates that you have preloaded register 1 with the address of the CDIB associated with the file to be closed.

Example

```
1.  CLOSE FILE1
2.  CLOSE 1
```

1. The file, whose CDIB has the symbolic address FILE1, is to be closed.

2. The file, the address of whose associated CDIB has been preloaded into register 1, is to be closed.

3.4. Retrieve a Record (DMINP)

The DMINP macroinstruction retrieves a single record from a file. This macroinstruction applies to all files except printer files.

Format:

```
LABEL     ΔOPERATIONΔ    OPERAND
--------------------------------------------------------------------
[name]     DMINP         ⎡cdibname⎤ ⎡  ⎡workarea⎤ ⎤⎡ ⎧LOCK  ⎫⎤
                         ⎨(1)     ⎬ ⎢, ⎨(0)     ⎬ ⎥⎢,⎩UNLOCK⎭⎥
                         ⎣1       ⎦ ⎣  ⎣0       ⎦ ⎦⎣         ⎦

                         ⎡   ⎧SEQ⎫⎤
                         ⎢,, ⎩RAN⎭⎥
                         ⎣        ⎦
```

Positional Parameter 1:

`cdibname`
: Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
: Indicates that you have preloaded register 1 with the address of the CDIB for the file.

Positional Parameter 2:

`workarea`
: Is the symbolic address of a work area that is to receive the record.

`(0) or 0`
: Indicates that you have preloaded the address of the work area that is to receive the record in register O.

Positional Parameter 3:

This parameter applies only to workstation files and only when PMODE=WSAM is specified when the file is opened. It is ignored for all other types of files. It temporarily overrides the KBRDIN RIB parameter that was established when the file is opened. If this parameter is specified, positional parameters 4 and 5 cannot be specified.

`LOCK`
: After the input operation is performed, the keyboard is to be locked.

`UNLOCK`
: After the input operation is performed, the keyboard is to be unlocked.

Positional Parameter 4:

This parameter is always a comma. It is used only when positional parameter 5 is specified.

Positional Parameter 5:

This parameter applies only to disk (MIRAM) files, format label diskette (MIRAM) files, and data set label diskette files. It is ignored for all other types of files. It temporarily overrides the MODE RIB specification that was established when the file is opened. If this parameter is specified, positional parameters 3 and 4 must be commas.

SEQ

Sequential retrieval based on current sequential position. It is:

1. the next highest record number for unkeyed retrieval (PROC=UNK specified when the file is opened); or

2. the next highest key for keyed retrieval (PROC=KEY specified when the file is opened).

The current sequential position will be modified.

RAN

Random retrieval based on the argument you supply. It is:

1. the key that is placed in the KARG field for keyed retrieval (PROC=KEY specified when the file is opened); or

2. the relative record number in the SKAD field for unkeyed random retrieval (PROC=UNK specified when the file is opened).

The current sequential position will be modified.

If a record is not found (no-find condition), this is reported via an error status and the error code X'31'. For additional information, see Appendix A.

Examples

```
1.  DMINP WFILE,WSIN,UNLOCK

2.  DMINP 1,0,,,SEQ
```

1. A record is to be read from the workstation screen and placed in the work area named WSIN. After the input operation is performed, the keyboard is to be unlocked.

2. A record is to be retrieved based on the current sequential position from the file whose address was preloaded into register 1, and the record is to be placed in the work area whose address has been preloaded in register 0.

3.5. Output a Record (DMOUT)

This macroinstruction places a record in a file. This form of the macroinstruction applies to all files.

Format

```
LABEL    ΔOPERATIONΔ   OPERAND
--------------------------------------------------------------
[name]     DMOUT       ⎧cdibname⎫ ⎡ , ⎧workarea⎫ ⎤
                       ⎨(1)     ⎬ ⎢   ⎨(0)     ⎬ ⎥
                       ⎩1       ⎭ ⎣   ⎩0       ⎭ ⎦

                       ⎡ , ⎧LOCK  ⎫ ⎤⎡ , ⎧SEQ⎫ ⎤
                       ⎣   ⎩UNLOCK⎭ ⎦⎣   ⎩RAN⎭ ⎦
```

Positional Parameter 1:

`cdibname`
Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
Indicates that you have prelocked register 1 with the address of the CDIB for the file.

Positional Parameter 2:

`workarea`
Is the symbolic address of the work area that contains the record that is to be sent to the file.

`(0) or 0`
Indicates that you have preloaded the address of the work area that contains the record to be sent to the file in register 0.

Positional Parameter 3:

This parameter applies only to workstation files and only when PMODE=WSAM is specified when the file is opened. It is ignored for all other types of files. It temporarily overrides the KBRDOUT RIB parameter that was established when the file is opened. If this parameter is specified, positional parameter 4 cannot be specified.

LOCK
After the record is displayed on the workstation screen, the keyboard is to be locked.

UNLOCK
After the record is displayed on the workstation screen, the keyboard is to be unlocked.

Positional Parameter 4:

This parameter applies only to disk (MIRAM) files, format label diskette (MIRAM) files, and data set label diskette files. It is ignored for all other types of files. It temporarily overrides the MODE RIB specification that was established when the file is opened. If this parameter is specified, positional parameter 3 must be a comma.

SEQ
The record is to be placed at the end of the file at the next available record position.

RAN
The record is to be placed in the relative record position according to the record number that is in the SKAD field.

Examples

```
1.  DMOUT WFILE,WSOUT,LOCK
2.  DMOUT 1,0,,SEQ
```

1. The record contained in the work area named WSOUT is to be displayed on the workstation screen. After the record is displayed, the keyboard is to be locked.

2. The record contained in the work area whose address was preloaded in register 0 is to be placed in the file whose address was preloaded in register 1. The record is to be placed at the end of the file at the next available record position.

3.6. Update a Record (DMUPD)

The DMUPD macroinstruction is used to update the most recently retrieved record from a disk (MIRAM) file, format label diskette (MIRAM) file, or a data set label diskette file. When you use this instruction, the updated record is placed back in the file at the point where it was originally retrieved.

Format

LABEL	ΔOPERATIONΔ	OPERAND
[name]	DMUPD	$\begin{Bmatrix}\text{cdibname}\\(1)\\1\end{Bmatrix},\left[\begin{Bmatrix}\text{workarea}\\(0)\\0\end{Bmatrix}\right]$

Positional Parameter 1:

`cdibname`
: Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
: Indicates that you have preloaded register 1 with the address of the CDIB for the file.

Positional Parameter 2:

`workarea`
: Is the symbolic address of a work area that contains the updated record that is to be placed back in the file.

`(0) or 0`
: Indicates that you have preloaded the address of the work area that contains the updated record in register 0.

Examples

```
1.|  DMUPD UPFILE,CHREC
2.|  DMUPD 1,0
```

1. The updated record contained in the work area named CHREC is to be placed back in the file.

2. The updated record contained in the work area whose address was preloaded in register 0 is to be placed back in the file whose address was preloaded in register 1.

3.7. Delete a Record (DMDEL)

You use this macroinstruction to logically delete the most recently retrieved record from a disk (MIRAM) or format label diskette (MIRAM) file. This can be used only if the records in the file contain a record control byte (RCB). The record and any index entries pointing to the record are marked as void. When a deleted record is encountered in subsequent sequential processing, it is bypassed. If you are retrieving records randomly and you specify the relative record number or the key of a deleted record, it is treated as a no-find situation.

Format

LABEL	ΔOPERATIONΔ	OPERAND
[name]	DMDEL	cdibname (1) 1

Positional Parameter 1

`cdibname`
Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
Indicates that you have preloaded register 1 with the address of the CDIB.

Example

```
DMDEL DFILE
```

The most recently retrieved record from the file named DFILE is to be deleted.

3.8. Terminate a Volume (DMFEV)

This macroinstruction allows you to terminate processing on the current volume of a tape, disk (MIRAM) file, format label diskette (MIRAM) file, or data set label diskette file that is being processed with only one volume online at a time. This instruction is ignored for files with all volumes mounted.

When this instruction is issued, the current volume is closed and a mount message is issued requesting that the next volume of the file be mounted. After the next volume of the file is mounted, it is opened and processing continues.

Format

LABEL	ΔOPERATIONΔ	OPERAND
[name]	DMFEV	cdibname (1) 1

Positional Parameter 1:

`cdibname`
Is the symbolic address of the CDIB declarative macroinstruction that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
Indicates that you have preloaded register 1 with the address of the CDIB macroinstruction.

Example

```
DMFEV AFILE
```

Processing is to be terminated on the current volume of the file named AFILE.

3.9. Select a Record Search (DMSEL, RECORD)

This macroinstruction is used to establish the point where the sequential retrieval of records for disk (MIRAM) files, format label diskette (MIRAM) files, and data set label diskette files is to begin. For data set label diskette files, it establishes the relative record number where retrieval is to begin. For disk (MIRAM) files and format label diskette (MIRAM) files, it establishes the relative record number or record key depending on whether unkeyed or keyed processing is being used.

It can also be used to change the key of reference for an indexed disk (MIRAM) file or a format label diskette (MIRAM) file.

Format

```
LABEL    | ΔOPERATIONΔ |   OPERAND
---------+-------------+---------------------------------------------
[name]   |   DMSEL     | ⎧cdibname⎫ ,RECORD ⎡, ⎧EQ ⎫ ⎤ [,KREF]
         |             | ⎨(1)     ⎬         ⎢  ⎪GT ⎪ ⎥
         |             | ⎩1       ⎭         ⎢  ⎨GE ⎬ ⎥
         |             |                    ⎢  ⎪BOF⎪ ⎥
         |             |                    ⎣  ⎩EOF⎭ ⎦
```

Positional Parameter 1

`cdibname`
Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2:

RECORD

Indicates that you are establishing the point where retrieval is to begin.

Positional Parameter 3:

This parameter is required for data set label diskette files and is optional when positional parameter 4 is used to change the key of reference for an indexed disk (MIRAM) file or an indexed format label diskette (MIRAM) file.

EQ

Establish the starting point at the record whose key is equal to the key specified in the KARG field (indexed disk or diskette file), or whose relative record number is equal to the relative record number specified in the SKAD field.

GT

Establish the starting point at the record whose key is greater than the key specified in the KARG field (indexed disk or diskette file) or whose relative record number is greater than the relative record number specified in the SKAD field.

GE

Establish the starting point at the record whose key is greater than or equal to the key specified in the KARG RIB parameter (indexed disk or diskette file) or whose relative record number is greater than or equal to the relative record number specified in the SKAD field.

BOF

Establish the starting point at the record with the lowest key value (indexed disk or diskette file) or with the lowest relative record number.

EOF

Establish the starting point at the end of the file. If you follow this function with a sequential input function, an end-of-file condition will result.

Note: *If the EQ, GT, or GE option is specified for an indexed file (keyed processing) and the operation is successful, the key of the record pointed to is returned in the KARG field.*

If the EQ, GT, or GE option is specified for a nonindexed file (nonkeyed processing) and the operation is successful, the relative record number of the record pointed to is returned in the SKAD field.

If the BOF option is specified for an indexed file (keyed processing) and the operation is successful, the key of the record pointed to is not returned in the KARG field. If you want to establish the starting point at the beginning of the file and have the key of the record pointed to returned in the KARG field, you must specify the GE option and set the KARG field to zeros.

If the BOF option is specified for a nonindexed file (nonkeyed processing) and the operation is successful, the relative record number of the record pointed to is not returned in the SKAD field. If you want to establish the starting point at the beginning of the file and have the relative record number of the record pointed to returned in the SKAD field, you must specify the GE option and set the SKAD field to ones.

Positional Parameter 4:

This parameter is used only for indexed disk or diskette files.

KREF

Specifies that the key of reference is to be changed. Register 0 must be preloaded with the value that corresponds to n in the KEYn RIB parameter ($1 \leq n \leq 5$). The KREF parameter may be used with positional parameter 3 if you want to first change the key of reference, and then establish a starting point.

Example

```
DMSEL XFILE,RECORD,EQ,KREF
```

The key of reference for the file named XFILE is to be changed and the starting point for sequential retrieval is to be at the record whose key is equal to the key in the KARG field.

3.10. Controlling Tape Functions (DMCTL, TAPE CONTROL)

You use this form of the DMCTL macroinstruction to control such functions as positioning the tape reel, erasing the tape, and writing tape marks.

Format

LABEL	ΔOPERATIONΔ	OPERAND	
[name]	DMCTL	cdibname	BSF
		(1)	BSR
		1	ERG
			FSF
			FSR
			REW
			RUN
			WTM

Positional Parameter 1:

`cdibname`
Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2:

`BSF`
Backspace to tape mark.

`BSR`
Backspace to interrecord gap.

`ERG`
Erase gap (write blank tape).

`FSF`
Forward space to tape mark.

`FSR`
Forward space to interrecord gap.

`REW`
Rewind tape.

`RUN`
Rewind and unload tape.

`WTM`
Write tape mark.

Example

```
DMCTL TFILE,WTM
```

Write a tape mark on the tape file named TFILE.

3.11. Controlling Short Blocks on Tape (DMCTL, TRUNC)

This form of the DMCTL macroinstruction is used with blocked output records to write short blocks on tape. Its use is optional with fixed-length blocked records, but required when you are building variable-length blocked records in the I/O buffer. When you issue this macroinstruction, you notify data management that the current block terminates at this point and is to be written to tape.

When you issue a DMOUT macroinstruction after building a variable-length record in the I/O area, data management supplies you with the number of bytes of residual space remaining in the buffer by placing this number in the general register you specified in the VARBLD RIB parameter. If you determine that the current record will not fit, you issue this form of the DMCTL macro to write out the current block without it. The entire output area is then available to you for building the next block beginning with the current record.

Format

LABEL	ΔOPERATIONΔ	OPERAND
[name]	DMCTL	{cdibname ,TRUNC (1) 1}

Positional Parameter 1

`cdibname`
Is the symbolic address of the CDIB declarative macroinstruction that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2:

`TRUNC`
Indicates that you are writing a short block on tape.

Example

```
DMCTL YFILE,TRUNC
```

Write a short block on the tape file named YFILE.

3.12. Controlling Skipping to the Next Block on Tape (DMCTL, RELSE)

When you are processing an input tape file with blocked records, you may reach a point in a block where you want to skip over the remaining records and begin processing the first record of the next block. You use this form of the DMCTL macroinstruction to cause this skip. When you do this, your next DMINP macroinstruction will retrieve the first record of the next block.

Format

LABEL	ΔOPERATIONΔ	OPERAND
[name]	DMCTL	{cdibname (1) 1} ,RELSE

Positional Parameter 1:

`cdibname`
Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2:

`RELSE`
Indicates that you are skipping to the next block on tape.

Example

```
DMCTL ZFILE,RELSE
```

This causes the remaining records in the current block in the file named ZFILE to be skipped over.

3.13. Controlling Printer Forms (DMCTL, SK/SP)

This form of the DMCTL macroinstruction is used to control spacing or skipping for printer forms. Form motion can occur before, after, or both before and after a line is printed. If you use this instruction, CONTROL=YES must be specified in the RIB associated with the file.

Format

```
LABEL      ΔOPERATIONΔ     OPERAND
[name]       DMCTL         ⎧cdibname⎫ , ⎧SK⎫ [,m][,n]
                           ⎨(1)     ⎬   ⎩SP⎭
                           ⎩1       ⎭
```

Positional Parameter 1:

`cdibname`
Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2:

`SK`
Indicates form skipping.

`SP`
Indicates form spacing.

Positional Parameter 3:

`m`
Specifies either the number of lines (0 through 15) to be spaced before printing or the channel code (1 through 15) for skipping before printing. If this parameter is not specified, the number of lines spaced will be determined by the PRAD keyword parameter in the RIB.

Positional Parameter 4:

`n`
Specifies either the number of lines (0 through 15) to be spaced after printing or the channel code (1 through 15) for skipping after printing.

Example

```
DMCTL PFILE,SP,5,3
```

Space five lines before printing the next line and three after printing it.

Programming Considerations:

Because of differences between the various printers, substitutions have to be made for some skip codes on each type of printer. Table 3-1 lists these substitutions.

Table 3-1. Device Skip Code Table for Printers

Function	Printer Code Substitution			
	0789	0770	0776	9246
Skip to code n, m				
m,n=1 (OV)		Code 12 (OV)	Code 12 (OV)	Code 12 (OV)
2				
3				
4				
5				
6				
7 (HP)		Note 2	Code 7 (HP) Note 2	Note 2
8	Code 2			
9	Code 1 (OV)	Note 4 (OV)	Code 12 (OV)	Note 4 (OV)
10	Code 3			
11	Code 4			
12	Code 1 (OV)	Note 4 (OV)	Code 12 (OV)	Note 4 (OV)
13				
14	Code 7 (HP)	Code 7 (HP)	Code 7 (HP)	Code 7 (HP)
15	Code 7 (HP)	Code 7 (HP)	Code 7 (HP)	Code 7 (HP)

LEGEND:

OV	Overflow code or channel
HP	Home paper code

NOTES:

1. A blank in the code substitution column indicates that no substitution is made; data management skips to the code you have specified.
2. Code 7 must be used as the home paper code on the 0770, 9246 and 0776 printers.
3. A skip to code 7 control causes a skip to the home paper code on all printers.
4. Code 12 is the primary forms overflow control code for the 0770 and 9246 printers. Code 9 can also be detected as forms overflow code, using the DMCTL,PRTOV macro. If the DMCTL,PRTOV macro is not used, however, code 12 should be used as overflow code, and code 9 should not be placed in the VFB.

3.14. Controlling Print Overflow Action (DMCTL, PRTOV)

You use this form of the DMCTL macroinstruction to control printer action when forms overflow is detected. This instruction can be used only if PRINTOV=YES is specified in the RIB associated with the file.

Format

```
LABEL        ΔOPERATIONΔ   OPERAND
--------------------------------------------------------------
[name]       DMCTL         ⎡cdibname⎤ ,PRTOV, ⎡⎡ 9⎤⎤
                           ⎨(1)     ⎬         ⎢⎨12⎬⎥ .
                           ⎣1       ⎦         ⎣⎣  ⎦⎦

                           ⎡ , ⎧SKIP  ⎫ ⎤
                           ⎣   ⎩REPORT⎭ ⎦
```

Positional Parameter 1

cdibname
: Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

(1) or 1
: Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2

PRTOV
: Indicates that you are controlling printer action when forms overflow is detected.

Positional Parameter 3:

9
: Specifies that channel 9 is to set the overflow condition.

12
: Specifies that channel 12 is to set the overflow condition.

Positional Parameter 4:

SKIP
: When forms overflow is detected, an automatic skip to the home paper position is performed.

REPORT
: When a forms overflow is detected, data management will inform you by reporting exception status due to a forms overflow condition. See Appendix A for additional information on exception status.

Example:

```
DMCTL OFILE,PRTOV,REPORT
```

When a forms overflow is detected, data management will inform you by reporting exception status due to a forms overflow condition.

3.15. Controlling Stacker Selection (DMCTL, SS)

This form of the DMCTL macroinstruction is used to select which one of the output stackers a punched card is to be placed.

If this instruction is issued for a file that does not support stacker selection, it is ignored.

Format

LABEL	ΔOPERATIONΔ	OPERAND
[name]	DMCTL	$\left\{\begin{matrix}\text{cdibname}\\ \text{(1)}\\ \text{1}\end{matrix}\right\}\text{,SS}\left[\text{,}\left\{\begin{matrix}\text{1}\\ \text{2}\end{matrix}\right\}\right]$

Positional Parameter 1:

cdibname
Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

(1) or 1
Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2:

SS
Indicates that you are using stacker selection.

Positional Parameter 3:

1
Indicates that the punched card is to be placed in output stacker 1.

2
Indicates that the punched card is to be placed in output stacker 2.

Example

```
DMCTL CFILE,SS,1
```

Place the punched card in output stacker 1.

3.16. Controlling Workstation Screen Management (DMCTL, Workstation Control)

This form of the DMCTL macroinstruction allows you to perform dynamic screen management. This includes line spacing, cursor positioning, full or partial clearing of the screen, locking or unlocking the keyboard, and inserting start of entries (SOEs).

Format

```
LABEL     ΔOPERATIONΔ    OPERAND
--------------------------------------------------------------
[name]      DMCTL        ⎡ ⎡cdibname⎤ ⎡ , ⎡CSAL⎤ , ⎡symbol⎤ ⎤ ⎤
                         ⎢ ⎨(1)     ⎬ ⎢   ⎢CSUP⎥   ⎨(0)   ⎬ ⎥ ⎥
                         ⎢ ⎣1       ⎦ ⎢   ⎢LNSP⎥   ⎣0     ⎦ ⎥ ⎥
                         ⎢            ⎣   ⎣PC  ⎦            ⎦ ⎥
                         ⎨                                    ⎬
                         ⎢ ⎡cdibname⎤ ⎡ , ⎡LOCK  ⎤ ⎤          ⎥
                         ⎢ ⎨(1)     ⎬ ⎢   ⎢NL    ⎥ ⎥          ⎥
                         ⎢ ⎣1       ⎦ ⎢   ⎢UNLOCK⎥ ⎥          ⎥
                         ⎣            ⎣   ⎣SOE   ⎦ ⎦          ⎦
```

Positional Parameter 1

cdibname
: Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

(1) or 1
: Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2:

CSAL
: Specifies that the cursor is positioned to line y and character position x, and the remaining portion of the screen is cleared of both protected and unprotected data. y and x are two consecutive half-word fields. The symbolic address of these fields or the register the address has been preloaded in must be specified in positional parameter 3. If the values of both fields are specified as zero, the cursor is positioned to the home position and the screen is cleared from home position to the end of the screen.

CSUP

Specifies that the cursor is positioned to line y and character position x, and the remaining portion of the screen is cleared of unprotected data. y and x are two consecutive half-word fields. The symbolic address of these fields or the register that the address has been preloaded in must be specified in positional parameter 3. If the values of both fields are specified as zero, the cursor is positioned to the home position and the screen is cleared from the home position to the end of the screen.

LNSP

Specifies that y lines are to be spaced immediately, and the cursor is positioned at character position 1 of the next available line. y is a half-word field. The symbolic address of this field or the register the address has been preloaded in must be specified in positional parameter 3. If the value of y exceeds the screen size, spacing will stop at the end of the screen.

PC

Specifies that the cursor is to be positioned to line y and character position x. y and x are two consecutive half-word fields. The symbolic address of these fields or the register the address has been preloaded in must be specified in positional parameter 3. If only y is specified, the cursor is positioned to that line; that is, no character positioning occurs. If only x is specified, the cursor is positioned to that character position; that is, no line positioning occurs. If the values of both fields are specified as zero, the cursor is positioned to the home position. The home position is specified by the STRTLN keyword parameter if PMODE=WSAM in the RIB associated with the file, or the home position is line 1 and character position 1 if PMODE=CNTRLDD.

LOCK

Indicates that the workstation keyboard is to be locked.

NL

Space fill the remainder of the current line and position the cursor to the first character position on the next line.

UNLOCK

Indicates that the workstation keyboard is to be unlocked.

SOE

Display on SOE at the present cursor position.

Positional Parameter 3:

symbcl

Is the symbolic address of the line number and character position entry for the CSAL, CSUP, LNSP, or PC operand.

(0) or 0

Specifies that you have preloaded register 0 with the address of the line number and character position entry.

Example

```
      DMCTL FILE1,PC,POSIT
FILE1 CDIB  FILENAME=WFILE
POSIT DC    X'00060004'
```

Position the cursor to line 6, character position 4.

3.17. Ensuring Completion of a Record Transfer for a Workstation (DMWTF)

If you immediately continue processing after you issue a DMINP, DMOUT, or DMCTL macroinstruction, unpredictable results may occur. You can avoid this problem by specifying WAIT=YES in the RIB associated with the file. If this was not specified, you must issue a DMWTF macro after each DMINP, DMOUT, or DMCTL macroinstruction to ensure that the transfer is completed.

Format

LABEL	ΔOPERATIONΔ	OPERAND
[name]	DMWTF	cdibname (1) 1

Positional Parameter 1:

cdibname
: Is the symbolic address of the CDIB declarative macroinstruction that contains the name assigned to the file by the LFD job control statement.

(1) or 1
: Indicates that you have preloaded register 1 with the address of the CDIB macroinstruction.

Example

```
DMINP AFILE, WORK
DMWTF AFILE
    .
    .
    .
AFILE CDIB FILENAME=WFILE
WORK DC    size of record
```

Control is to be returned to the program when an I/O operation for any workstation associated with the file has been completed.

3.18. Displaying Messages in the Workstation System/Command Zone (DMDSP)

The DMDSP macroinstruction is used to display messages in the system/command zone (first 2 lines) of the system console, the master workstation, an individual workstation in a file, or all workstations associated with a file. It also allows you to select the log that the message is to be placed on and to request a reply to the message. The message is displayed on line 2 and the reply is entered on line 1.

Format

```
LABEL       ΔOPERATIONΔ    OPERAND
-------------------------------------------------------------------
[name]       DMDSP         ⎡cdibname⎤ , ⎧output work area⎫
                           ⎨(1)     ⎬   ⎩(r)             ⎭
                           ⎣1       ⎦

                           ⎡ , ⎡60                    ⎤⎤⎡ , ⎡SJL ⎤⎤
                           ⎢   ⎨output message length ⎬⎥⎢   ⎨SOJL⎬⎥
                           ⎣   ⎣(r)                   ⎦⎦⎣   ⎣SO  ⎦⎦

                           [,REP]⎡ , ⎧input work area⎫⎤
                                 ⎣   ⎩(r)            ⎭⎦

                           ⎡ , ⎡60          ⎤⎤[,OWNMSG]
                           ⎢   ⎨reply length⎬⎥
                           ⎣   ⎣(r)         ⎦⎦
```

Positional Parameter 1

`cdibname`
Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

If a message is to be sent to the system console, the file name in the CDIB must be $Y$CON.

If a message is to be sent to the master workstation, the file name in the CDIB must be $Y$MAS.

Note that you do not have to issue an OPEN or CLOSE macroinstruction in your program for either $Y$CON or $Y$MAS.

`(1) or 1`
Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2:

`output work area`
Is the symbolic address of an output work area that contains the message to be displayed.

`(r)`
Indicates that you have preloaded the address of the output work area in a register. General register 0 or registers 2 through 12 may be used.

Positional Parameter 3:

`60`
Specifies that the length of the output message is 60 bytes.

`output message length`
Specifies the length of the output message.

`(r)`
Indicates that you have preloaded a register with the address of the output message length entry. Register 0 or registers 2 through 12 may be used.

Positional Parameter 4:

`SJL`
Send the message to the job log without displaying it.

`SOJL`
Display the message and send it to the associated log (if configured) and the job log.

`SO`
Display the message and send it to the associated log (if configured).

Positional Parameter 5:

This parameter can be used only when positional parameter 4 is specified as SO or SJL.

`REP`
Indicates that a reply is requested.

If omitted, positional parameters 6 and 7 are ignored.

This parameter must not be used if you are sending a message to the system console or to all workstations associated with a file.

Positional Parameter 6:

`input work area`
Is the symbolic address of an input work area that is to receive the reply.

`(r)`
Indicates that you have preloaded a register with the address of the input work area. Register 0 or registers 2 through 12 may be used.

If omitted, positional parameter 7 is ignored and the output work area and output message length are used.

Positional Parameter 7:

`60`
Specifies that the length of the reply is 60 bytes.

`reply length`
Specifies the length of the reply.

`(r)`
Indicates that you have preloaded a register with the address of the reply length entry. General register 0 or registers 2 through 12 may be used.

Positional Parameter 8:

`OWNMSG`
Indicates that the message is not to be retrieved from the system message file regardless of the message text contents.

Example

```
         DMDSP $Y$MAS,WORK2,30,,,,,OWNMSG
           .
           .
           .
$Y$MAS CDIB

WORK2  DC  message of 30 characters
```

Display a 30-character message on the master workstation.

3.19. Create a Breakpoint in a Printer or Punch Spool File (DMBRK)

This macroinstruction is used to create a breakpoint in a printer or punch spool file. It should only be issued for a file that has been opened. It will be ignored if your system does not have spooling capability.

Format

LABEL	ΔOPERATIONΔ	OPERAND
[name]	DMBRK	cdibname (1) 1

Positional Parameter 1:

cdibname
Is the symbolic address of the CDIB that contains the name that was assigned to the file of the LFD job control statement.

(1) or 1
Indicates that you have preloaded register 1 with the address of the CDIB.

Example

```
DMBRK PFILE
```

Create a breakpoint on the spool file whose CDIB is named PFILE.

3.20. Processing Optional User Tape Labels (DMLAB)

This macroinstruction is used to read or write optional standard or nonstandard user header and trailer labels (UHLs and UTLs) and to terminate the processing of these labels.

If you intend to process optional user labels, ULABEL=YES must be specified in the RIB for the file, and you must include the DMLAB instruction in the label processing routine in your program.

Format

LABEL	ΔOPERATIONΔ	OPERAND
[name]	DMLAB	cdibname (1) 1 , workarea (0) 0 , IO END

Positional Parameter 1:

cdibname

Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

(1) or 1

Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2:

This parameter specifies the work area that is to receive the optional label for an input file or contains the label for an output file. It must be full-word aligned. It cannot be the same area as the work area used to process records nor can it be the same area as any of the file I/O (data buffer) areas.

- Standard Labeled File Considerations

  The work area will contain only the 80-byte label. If block numbering is specified (BKNO=YES in the RIB), the work area must be immediately preceded by a 4-byte full-word aligned field.

- Nonstandard Labeled File Considerations

  The work area is a variable format; that is, it consists of a 4-byte record descriptor word (RDW) followed by the label. The first 2 bytes of the RDW contain the label size and this value includes the 4-byte RDW. Data management supplies the label size on input and you must supply it for output. For output, the label size cannot exceed the file buffer size.

If block numbering is specified (BKNO=YES), the 4-byte RDW will serve as the block number field. On output, the label size in the RDW in the work area will be overlayed with the block number when control is returned from the DMLAB macroinstruction. Consequently, the label size must be placed in the RDW in the work area before you write each label.

If backward reading is specified (READ=BACK in the RIB), the size of the work area must be at least the file buffer size + 4.

Positional Parameter 3:

IO

Specifies that the next label is to be read from an input file or written to an output file.

When control is returned, one of the following status conditions will be reported.

- Successful Status

  This is reported if the label was successfully read or written. In addition, the CT$LCODE field in the CDIB will contain the appropriate alphabetic character to indicate when the label processing is being performed: 0 indicates at file or volume open, V indicates at volume close, and F indicates at file close.

- Error Status

  This is reported if an error occurred.

- Exception Status

  If no more labels can be read or written, exception status due to a *user label termination processing required condition* is reported. (This is the only exception status that is reported for the DMLAB (IO) macroinstruction.) When this condition is reported, you must then issue the DMLAB (END) macroinstruction to terminate label processing.

END

Specifies that user label processing is to be terminated (no more labels can be read or written) and that the last nonlabel related function request (OPEN, CLOSE, DMFEV, DMINP, or DMOUT macroinstruction that was interrupted) is to be completed.

When control is returned, the status that is reported is actually the status of the last function request that was interrupted. For example, if a DMINP instruction was interrupted, the status that is reported for the DMLAB (END) instruction is that of the DMINP instruction. The status reported could be any of the exception conditions such as end-of-file, user label processing required, and so on.

Note also that when control is returned for the DMLAB (END) instruction, the contents of register 0 will be changed to reflect the value that was in it when the last function request was interrupted. For example, if successful status is reported when control is returned for the DMLAB (END) instruction and the last function request was a DMINP instruction with work area processing (WORK=YES specified in the RIB), the register will be pointing to the record in the original work area that was used by the DMINP instruction.

See Appendix A for an example of how to use the DMLAB macroinstruction.

3.21. Select a Screen Format (DMSEL, SCREEN)

This macroinstruction, in conjunction with screen format services, lets you select and display a previously created screen stored in the format file $Y$FMT or a MIRAM user library. For information concerning screen format services, see the Screen Format Services Technical Overview, UP-9977.

Format

LABEL	ΔOPERATIONΔ	OPERAND
[name]	DMSEL	{cdibname (1) 1} ,SCREEN, {screen format name (0) 0}

Positional Parameter 1:

`cdibname`
Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2:

`SCREEN`
Indicates that you are selecting a screen format from a screen format file.

Positional Parameter 3:

`screen format name`
Is the symbolic address of an 8-byte area that contains the name of the screen format to be selected.

`(0) or 0`
Indicates that you have preloaded the address of the screen format name in register 0.

Example

```
DMSEL WORKSTN,SCREEN,S1
```

Selects the screen format named in the 8-byte area S1.

3.22. Select a Menu (DMSEL, MENU)

This macroinstruction, in conjunction with menu services, lets you select and display a previously created menu stored in the format file, $Y$FMT, or a MIRAM user library. For information concerning menu services, see the Menu Services Technical Overview, UP-9317.

Format

LABEL	ΔOPERATIONΔ	OPERAND
[name]	DMSEL	$\left\{\begin{array}{l}\texttt{cdibname}\\ \texttt{(1)}\\ \texttt{1}\end{array}\right\}$,MENU, $\left\{\begin{array}{l}\texttt{tag}\\ \texttt{(0)}\\ \texttt{0}\end{array}\right\}$

Positional Parameter 1:

`cdibname`
Is the symbolic address of the CDIB that contains the name that was assigned to the file by the LFD job control statement.

`(1) or 1`
Indicates that you have preloaded register 1 with the address of the CDIB.

Positional Parameter 2:

`MENU`
Indicates that you are selecting a menu from a menu format file.

Positional Parameter 3:

`tag`
Is the symbolic address of an 8-byte area that contains the name of the menu to be selected.

`(0) or 0`
Indicates that you have preloaded the address of the menu name in register 0.

Note: *Specifying a null menu name (i.e., eight bytes of hexadecimal zeros), turns off menu processing.*

Section 4
Workstation Considerations

4.1. General

When you are dealing with workstation files, you have a choice of two areas on a workstation where you can enter or display data. These areas are called the data zone and the system/command zone. The data zone consists of the entire screen, and the command zone consists of the first two lines of the screen. The data zone is used for processing application programs; the system/command zone is used for sending messages to an individual workstation, all workstations, the system console, or the master workstation.

4.2. Data Zone Operations

Data zone operations consist of transferring data to and from the workstation and performing screen management. There are two modes that you can use to transfer data and perform screen management. These are the WSAM mode (PMODE=WSAM) and the device dependent mode (PMODE=CNTRLDD).

The imperative macroinstructions that you use to process data in the data zone are: OPEN, CLOSE, DMOUT, DMINP, DMCTL, and DMWTF.

4.2.1. WSAM Mode

Using the WSAM mode requires that you specify PMODE=WSAM in the RIB associated with the workstation file or that you accept WSAM as the default specification.

This mode provides you with the means to have the workstation screen dynamically managed for both input and output operations. As a result, you do not have to be concerned with controlling the screen each time you perform an input or output operation.

When you use this mode, your data records must contain only *pure data.* Control characters and protected characters must not be included in your data records. If they are included, the dynamic screen management will be disrupted and the screen will not appear as you expected.

Controlling Screen Management in the WSAM Mode

Screen management is controlled in the WSAM mode by the IREC, OREC, ISOE, OSOE, KBRDIN, KBRDOUT, SCREND, STRTLN, and SPACELN keyword parameters in the RIB for the file. If any of these parameters are not specified, the default option is assumed when the file is opened. If a RIB is not specified for the file, the default options are assumed for all of these parameters when the file is opened.

The primary screen control that is performed in the WSAM mode is line end recovery and screen end recovery. Line end recovery occurs at the end of each input/output record. Screen end recovery occurs when the logical screen end is detected. Operator intervention is required when logical screen end is detected during the execution of a DMOUT or DMCTL macroinstruction. Screen end recovery is automatic if the logical screen end is detected during the execution of a DMINP macroinstruction.

- Line End Recovery Operations

  The line end recovery operations you want performed are specified by the IREC, OREC, ISOE, KBRDIN, and KBRDOUT keyword parameters in the RIB for the file. The options you choose for these parameters will determine how the screen will appear, that is, the position of the cursor, the position of the SOE character, whether the next record that is displayed or entered will follow the previous record, and whether the keyboard is to be locked or unlocked after an input or output operation is performed. For detailed information on these parameters, see 2.3.6.

- Screen End Recovery Operations

  The screen end recovery operations you want performed are specified by the SCREND keyword parameter in the RIB for the file. There are three options: SCREEN=NP, WRAP, or SCROLL. Before we consider each of these options, a general discussion of what happens when logical screen end is detected is necessary.

  When logical screen end is detected, all further output (DMOUT and DMCTL instructions) will cease until screen end recovery operations are performed. Once this is done, the remaining output will continue. To prevent current output data from being erased by the screen end recovery operations, the operator is informed when logical screen end is detected that he must press the F19 function key to continue. This is done by displaying:

  ```
  ▪MSGWAIT▪
  ```

  on the workstation's twenty-fifth line. If data is being entered (DMINP instruction) and the end of the screen is reached, the screen end recovery operations are automatically performed because no data should be current; however, the operator should not press the XMIT key until he has read the current screen.

You must also consider that logical screen end can be caused by not only insufficient space for data before current data is erased, but also by a lack of space to perform proper line end recovery. Consequently, while the data may appear to fit, the line end recovery operations may not be able to be performed unless the F19 function key is pressed. As you can see, if logical screen end is caused by insufficient space for line recovery operations, screen end recovery takes the place of line end recovery.

- SCREND=NP

  When logical screen end is detected (the physical bottom of the screen in this case) during the execution of a DMOUT instruction, all further output will cease until the operator presses the F19 function key. When the F19 function key is pressed, the screen will be cleared from position 1 of the line specified by the STRTLN parameter in the RIB to the screen end. The remaining output will continue starting at the top of the screen at position 1 of the line specified by the STRTLN RIB parameter.

  If logical screen end is detected during the execution of a DMINP instruction, the screen end recovery actions will be performed automatically; that is, the F19 function key does *not* have to be pressed.

- SCREND=WRAP

  If you specify this, data is always maintained on the screen; that is, data will be overlayed, not erased. Logical screen end, in this case, is based upon the most recent input line on the screen. If the line containing the most recent input is to be overlayed by the execution of a DMOUT or DMCTL instruction, all output will cease until the F19 function key is pressed because the most recent input line is considered to be the logical screen end. After the F19 function key is pressed, the most recent input line is overlayed and the succeeding output will wrap around the screen; that is, the output will continue, starting at the top of the screen, and overlay the existing lines on the screen.

  If the current instruction is a DMINP instruction, this input line becomes the most recent input line and, therefore, is the new logical screen end. The succeeding output will wrap around the screen until the most recent input line is to be overlayed.

  If there are no input lines on the screen, the logical screen end is the physical bottom of the screen. When logical screen end is detected in this case, all output will cease until the F19 function key is pressed. When the F19 function key is pressed, the succeeding output will wrap around the screen and overlay the existing lines on the screen starting with the top line.

- SCREND=SCROLL

  The screen action in this mode differs from the other modes we have discussed in that the screen action is a continuous process rather than one that occurs only at logical screen end. This action consists of the screen scrolling continually, that is, moving up. When a DMOUT or DMCTL instruction is executed, the screen is scrolled, starting at the top (as specified by the STRTLN RIB parameter), the necessary number of lines to accommodate the display. The display starts from one line above the area defined by the SPACELN RIB parameter; consequently, the available output space is the area from the line specified by the STRTLN RIB parameter to one line above the area defined by the SPACELN RIB parameter. Logical screen end is detected when current data is about to be scrolled off the screen. At this point, all output ceases until the F19 function key is pressed.

  Inputs (DMINP instructions) cause the current data to be considered as old data; that is, any scrolling of current data off the screen that results from the input will not cause logical screen end. This will occur when the following outputs fill the available output space and current data is about to be scrolled off the screen. If input is transmitted from within the area defined by the SPACELN RIB parameter, it will be scrolled to one line above this area. If input is transmitted from within the available output space, the cursor will be placed at position 1 of the area defined by the SPACELN RIB parameter.

Using WSAM Mode Screen Management

The coding that follows provides an example of a typical use for a workstation file, that is, a question and answer dialog.

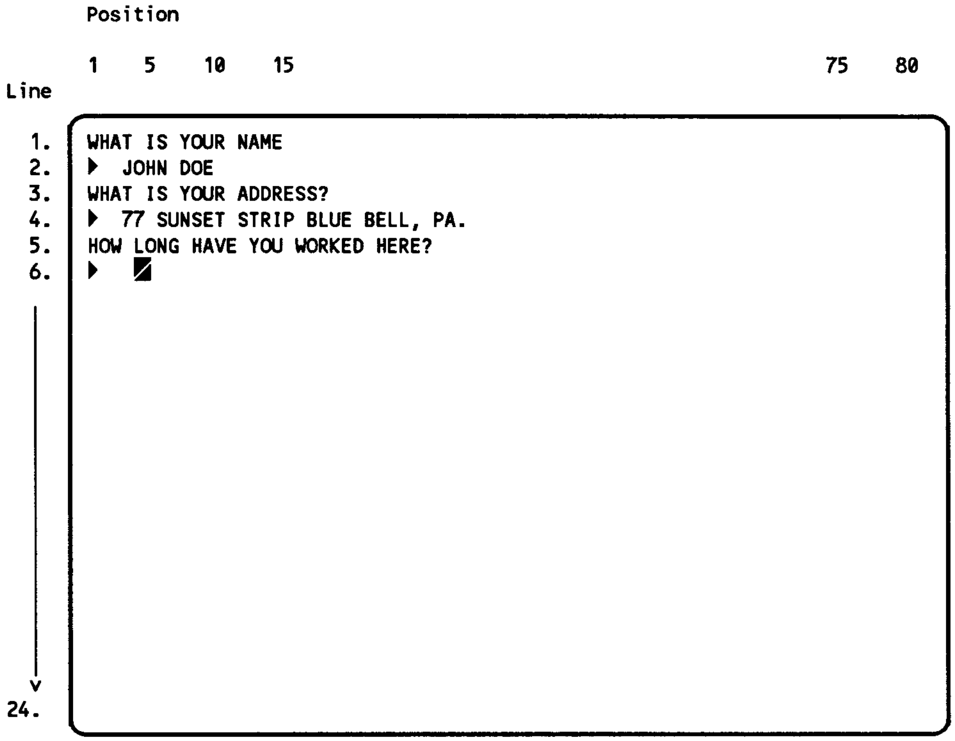
```
  .
  .
  .
OPEN WKSTN
DMOUT WKSTN,Q1
DMINP WKSTN,NAME
  .
  .
  .
DMOUT WKSTN,Q2
DMINP WKSTN,ADDRS
  .
  .
  .
DMOUT WKSTN,Q3
DMINP WKSTN,YRSWRK
  .
  .
  .
```

```
WKSTN  CDIB FILENAME=WFILE
Q1     DC   CL80'WHAT IS YOUR NAME?'
Q2     DC   CL80'WHAT IS YOUR ADDRESS?'
Q3     DC   CL80'HOW LONG HAVE YOU WORKED HERE?'
NAME   DC   CL80' '
ADDRS  DC   CL80' '
YRSWRK DC   CL80' '
```

As you can see, the RIB is defaulted; that is, PMODE=WSAM, SCREND=NP, IREC=NL, OREC=NL, OSOE=REC, ISOE=NO, STRTLN=FIRST, RCFN=FIXBLK, RCSZ=80, WORK=YES, INPTRAN=NO, KBRDIN=LOCK, and KBRDOUT=UNLOCK. This means that:

- the dialog begins at the top of the screen;
- the questions (output) and answers (input) appear on separate successive lines;
- a start-of-entry (SOE) character precedes each answer;
- the keyboard will be unlocked after output and locked after input; and
- the logical screen end is the physical bottom of the screen. Also, when logical screen end is detected, the dialog will cease until the F19 function key is pressed. At that point, the screen will be cleared and the dialog will continue starting at the top of the screen.

When this coding is executed, the dialog will proceed on the screen as shown:



```
WHAT IS YOUR NAME
▶ JOHN DOE
WHAT IS YOUR ADDRESS?
▶ 77 SUNSET STRIP BLUE BELL, PA.
HOW LONG HAVE YOU WORKED HERE?
▶  █
```

I/O Area (Data Buffer) Allocation for WSAM Mode

If you have specified PMODE=WSAM and WORK=YES in the RIB for the file, it is recommended that you do not specify IOA1=symbol; the WSAM mode almost always will allocate its own buffer because of its need for a varying number of control characters to perform screen management. If you do specify IOA1=symbol in this case, this area will be used only if it is large enough to handle the varying number of control characters.

If you specify PMODE=WSAM and WORK=NO in the RIB for the file, you must specify IOA1=symbol. You can also specify IOA2=symbol in this case; however, this secondary buffer is usually wasted.

4.2.2. Device Dependent Mode

Using the device dependent mode requires that you specify PMODE=CNTRLDD in the RIB associated with the workstation file.

When you use this mode, you have absolute control of the screen; you have the capability of displaying protected and unprotected data, repositioning the cursor, deleting lines, and so on. This is accomplished by including control character sequences in the data records. In order to do this, it is necessary to know the device control character sequences so that you will be aware of what will be placed in your input buffer and work area (if specified) when you make an input request and what you must place in these areas to perform screen management. Table 4-1 shows the control character sequences and their hexadecimal equivalents that are received on input or that you must use to perform screen management on output.

In addition, when you position the cursor or the SOE character, you must indicate the specific location on the screen where you want to place it. This is necessary because each location on the screen is determined by a pair of coordinates, Y and X. The Y coordinate represents the line number, and the X coordinate represents the character position on each line. Each coordinate has a specific hexadecimal value that must be used. Figure 4-1 shows the hexadecimal values for each of these coordinates.

Table 4-1. Control Character Sequences

Function	Output		Input	
	Control Character Sequence	Code Sent to Workstation	Control Character Sequence	Code from Workstation
Cursor positioning	ESC VT Y X SI	270BYYXX0F	-	
SOE positioning	-	-	ESC VT Y X NUL SI	270BYX00F
Cursor return (new line)	CR	0D	CR	OD
Erase to end of display	ESC a	2781	-	-
Erase to end of line	ESC b	2782	-	-
Delete in line	ESC c	2783	-	-
Delete in display	ESC C	27C3	-	-
Insert in line	ESC d	2784	-	-
Insert in display	ESC D	27C4	-	-
Scan left	ESC g	2787	-	-
Scan right	ESC h	2788	-	-
Scan down	ESC i	2789	-	-
Scan up	ESC f	2786	-	-
Character erase (space)	SP	40	SP	40
Tab	HT	05	-	-
Tab stop set	ESC HT	2705	-	-
Tab stop	-	-	HT	-
Message waiting	BEL	2F	-	2F
Cursor to home	ESC e	2785	-	-
Insert line	ESC j	2791	-	-
Delete line	ESC k	2792	-	-
Erase field	ESC K	27D2	-	-
Erase display	ESC M	27D4	-	-

continued

Table 4-1. Control Character Sequences (cont.)

Function	Output		Input	
	Control Character Sequence	Code Sent to Workstation	Control Character Sequence	Code from Workstation
Request processor message	-	2F	BEL	2F
Start blink marker	FS	1C	FS	1C
End blink marker	GS	1D	GS	1D
Lock keyboard	DC4 or ESC DC4	3C	-	-
Print	DC2	12	-	-
Print transparent	ESC DC2	2712	-	-
Start of entry (SOE)	RS	1E	RS	IE
Shift in	SI	0F	-	-
Shift out	SO	OE	-	-
Line feed	LF	25	LF	25
Form feed	FF	OC	FF	0C
Transmit (unprotected)	DC1	11	-	
Transmit display	ESC DC1	2711	-	

Y COORDINATE	
LINE NO.	HEX VALUE
1	40
2	5A
3	7F
4	7B
5	5B
6	6C
7	50
8	7D
9	4D
10	5D
11	5C
12	4E
13	6B
14	60
15	4B
16	61
17	F0
18	F1
19	F2
20	F3
21	F4
22	F5
23	F6
24	F7

Y COORDINATE		Y COORDINATE	
COL. NO.	HEX VALUE	COL. NO.	HEX VALUE
1	40	41	C8
2	5A	42	C9
3	7F	43	D1
4	7B	44	D2
5	5B	45	D3
6	6C	46	D4
7	50	47	D5
8	7D	48	D6
9	4D	49	D7
10	5D	50	D8
11	5C	51	D9
12	4E	52	E2
13	6B	53	E3
14	60	54	E4
15	4B	55	E5
16	61	56	E6
17	F0	57	E7
18	F1	58	E8
19	F2	59	E9
20	F3	60	4A
21	F4	61	E0
22	F5	62	4F
23	F7	63	5F
24	F7	64	6D
25	F8	65	79
26	F9	66	81
27	7A	67	82
28	5E	68	83
29	4C	69	84
30	7E	70	85
31	6E	71	86
32	6F	72	87
33	7C	73	88
34	C1	74	89
35	C2	75	91
36	C3	76	92
37	C4	77	93
38	C5	78	94
39	C6	79	95
40	C7	80	96

NOTE:

The addressing sequence will always be:

Y position

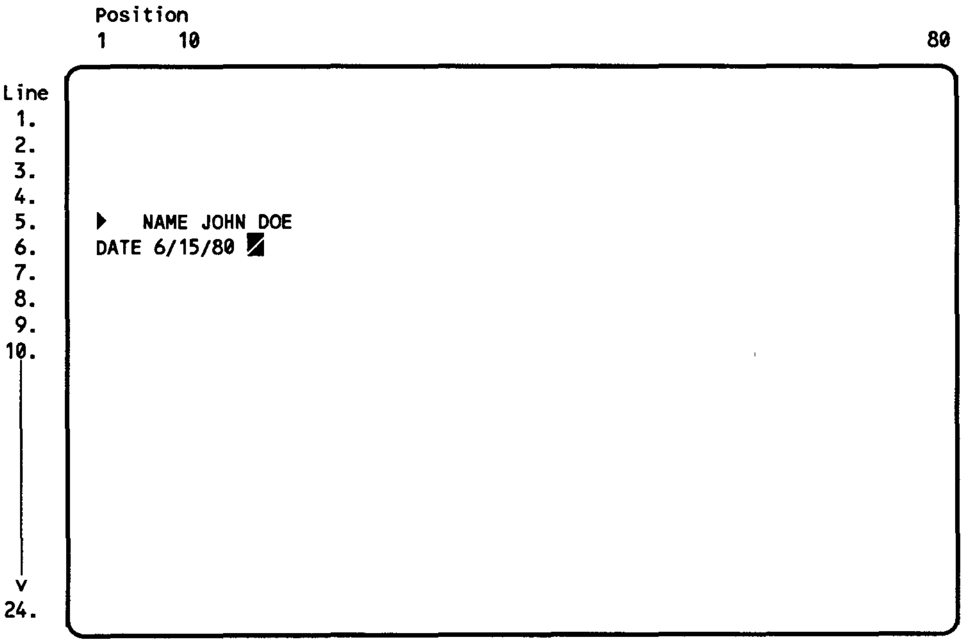
then X position

Figure 4-1. Hexadecimal Values for Workstation Screen Location Coordinates

Device Dependent Mode Input Processing

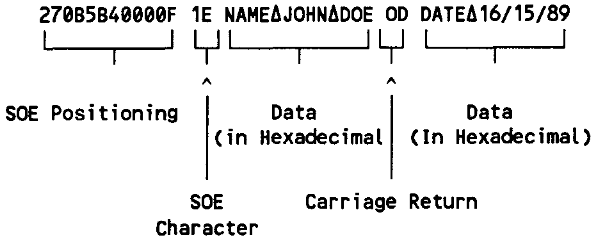
When you issue an input request, the data record is retrieved from the screen and placed in your input buffer and work area (if specified). The record will consist of the hexadecimal equivalents of the control character sequences and the actual data. The data record is transparent to data management. Therefore, if input editing such as the deletion of control character sequences (SOE positioning, carriage returns, and so on) is required, you must perform this in your program.

To show what a data record consists of when it is retrieved from the workstation screen, assume that the following was keyed in to a workstation:



```
▶  NAME JOHN DOE
DATE 6/15/80 ▨
```

When this screen is transmitted, the following data record will be moved to the specified area after issuing an input request.



```
270B5B40000F 1E NAMEΔJOHNΔDOE OD DATEΔ16/15/89
```

Device Dependent Mode Output Processing

As mentioned before, you use control character sequences to perform screen management. This requires that you place the hexadecimal equivalents of the control character sequences (see Table 4-1) in your output buffer or work area and then issue an output request.

The following coding provides an example of how to use the device control character sequences for scrolling a 24-line workstation screen when you use the device dependent mode.

```
     WRKC CDIB  FILENAME=WFILE
     WRKR RIB   PMODE=CNTRLDD
              .
              .
              .
            OPEN   WRKC,(WRKR)
              .
              .
              .
            DMOUT  WRKC,SCRL
              .
              .
              .                                          Control Character
     *    POSITION CURSOR TO HOME POSITION               Sequences
  SCRL  DC    X'270B40400F'                            ESC VT Y X SI
*    DELETE LINE WHERE CURSOR IS POSITIONED
        DC    X'2792'                                  ESC k
*    POSITION CURSOR TO BEGINNING OF LINE 24
        DC    X'270BF7400F'                            ESC VT Y X SI
*    DISPLAY ON LINE 24
        DC    C'NEW LINE 24'
```

The first line in the work area named SCRL positions the cursor to the home position. The second line in the work area causes the line where the cursor is positioned to be erased. All lines following this line (line1) move up one line to take up the space of the deleted line and this leaves line 24 blank. The third line in the work area positions the cursor at the beginning of line 24. The fourth line in the work area causes the text NEW LINE 24 to be displayed on line 24.

4.2.3. Function Keys

There are 22 function code keys on the workstation keyboard: F1 through F22. F1 through F12 can be processed by your program; that is, you can have your program perform a specified action when a particular key is pressed.

F15, F17, and F19 are used for screen management. The remaining keys (F13, F14, F16, F18 and F20 through F22) are not available for your use.

Processing Function Keys in Your Program

To process function keys in your program requires that FNKEYS=symbol be specified in the RIB for the workstation file. Then, in your main program, you provide the action that is to take place when a function key is pressed.

When your program is subsequently executed and in response to an I/O request:

- a function code key is pressed in combination with the FUNCTION key;
- the bit position corresponding to the function code key in the 4-byte area defined by FNKEYS (see 2.3.6) will be set to 1; and
- the action you provided based on this setting will be performed.

Using Function Keys for Screen Management

As mentioned before, the F15, F17, and F19 function code keys can be used for screen management. To use them requires that you press the function code key in combination with the FUNCTION key. The resulting actions are as follows:

F15

Pressing this key indicates that there is no more input (end-of-file).

F17

Pressing this key will temporarily stop the display.

F19

Pressing this key releases the current screen and causes the next portion of the screen to be displayed. This is used when the screen contains less than a complete display or you had previously stopped the display by using the F17 key.

4.3. System/Command Zone Operations

System/command zone operations consist of routing messages to the various logs and displaying the messages on the system console, master workstation, an individual workstation, or all workstations associated with a file.

You use the DMDSP imperative macroinstruction (see 3.18) to perform command zone operations. All messages are displayed on the second line of the screen, and all replies are entered on the first line of the screen.

4.3.1. Sending Messages to the System Console

The file name for the system console is $Y$CON. When you send a message to the system console, you must use this name with the DMDSP macroinstruction and your program must contain a CDIB macroinstruction that has this name as its label. Note that, because $Y$CON is a system file, it does not have to be defined in your program execution job control stream nor does this file have to be opened in your program.

The SO (default) and SOJL log options are the only ones you can specify with the DMDSP macroinstruction (see 3.18) when you send a message to the system console.

The following coding provides an example of sending a 50-character message to the system console:

```
        DMDSP $Y$CON,WORK1,50
          .
          .
          .
$Y$CON  CDIB
WORK1   DC    message of 50 characters
```

The message is displayed on the system console, and it is routed to the console log (if configured).

4.3.2. Sending Messages to the Master Workstation

The file name for the master workstation (the workstation that initiated the job) is $Y$MAS. When you send a message to the master workstation, you must use this with the DMDSP macroinstruction and your program must contain a CDIB macroinstruction that has this name as its label. Note that, because $Y$MAS is a system file, it does not have to be defined in your program execution job control stream nor does this file have to be opened in your program.

Any of the log options (SJL, SOJL, or SO) can be specified with the DMDSP macroinstruction (see 3.18) when you send a message to the master workstation. If the SO (default) is specified and there is no current master workstation, the message is displayed on the system console and routed to the console log (if configured). If SOJL is specified and there is no current master workstation, the message is displayed on the system console and routed to the console log (if configured) and the job log.

The following coding provides an example of sending a 35-character message to the master workstation:

```
          DMDSP $Y$MAS,WORK1,35,SOJL
            .
            .
            .
$Y$MAS CDIB
WORK1  DC    message of 35 characters
```

The message is displayed on the master workstation and routed to the associated log (if configured) and to the job log.

4.3.3. Sending Messages to a Single Workstation

When you send a message to a single workstation, the file name that was assigned to the workstation file in the program execution job control stream must be used with the DMDSP macroinstruction, your program must contain a CDIB macroinstruction that defines this file, and this file must be opened before you issue the DMDSP macroinstruction. Any of the log options (SJL, SOJL, and SO) can be specified with the DMDSP macroinstruction (see 3.18) when you send a message to a single workstation.

The following coding provides an example of sending a 50-character message to a single workstation with a reply requested.

```
           .
           .
           .
          OPEN  FILE1
           .
           .
           .
          DMDSP FILE1,WORK1,50,SO,REP,WORK2,20
           .
           .
           .
FILE1 CDIB    FILENAME=WFILE
WORK1 DC    message of 50 characters
WORK2 DC    XL20'00':  message reply area
```

The 50-character message in WORK1 is displayed on the workstation and the 20-character reply typed in by the operator is placed in WORK2.

4.4. Workstation Multivolume File Processing

The CDIB (Appendix A) that is created for a workstation file contains a subelement ID field that consists of two 1-byte subfields, CW$SUB1 and CW$SUB2.

The first subfield, CW$SUB1, is set to X'00' when the file is opened.

The second subfield, CW$SUB2, identifies the specific workstation that completed its function; that is, the number of the workstation (X'01' - X'FF') is placed in this field. This information, along with the completion status, will be reported for all imperative instructions. After a multivolume file is opened (OPEN instruction), the CW$SUB2 field will contain the number of the first workstation that is reported as being available. If no workstations are presently connected to the file when you attempt to open it, CW$SUB2 will contain X'00'. If you issue an input request (DMINP instruction) to a multivolume workstation file, CW$SUB2 will contain the number of the workstation from which the record is received.

CW$SUB2 is also used to specify the particular workstation in the multivolume file for which you want to have a DMOUT, DMDSP, or DMCTL instruction performed. This requires that you place the number of the workstation into CW$SUB2 before you execute the instruction. If you place X'00' in CW$SUB2 and you issue a DMDSP instruction without a reply, the message will be directed to all actively assigned workstations in the file. If you specify a workstation number when you issue an OPEN, CLOSE, or DMINP instruction, it will be ignored.

The following examples show how the CW$SUB2 field is used to specify a specific workstation and how it is used to identify the particular workstation that completed its function.

Example 1

```
       .
       .
       .
       OPEN  FILE1
       .
       .
       .
       MVI   CW$SUB2,X'02'
       DMOUT FILE1,WORK1
       .
       .
       .
FILE1  CDIB  FILENAME=WFILE
WORK1  DC    size of record
       .
       .
```

Example 2

```
         .
        OPEN  FILE1
         .
         .
         .
NEXT    DMINP FILE1,WORK1
         .
         .
         .
        DMOUT FILE1,WORK1
         .
         .
         .
        B     NEXT
         .
         .
         .
FILE1 CDIB    FILENAME=WFILE
WORK1 DC      size of record
```

1. The record in WORK1 is displayed on workstation 2 of the multivolume file.

2. An input record is read from an active workstation in a multivolume file, and the number of that workstation is placed in CW$SUB2. Then, a record is displayed on the same workstation.

Appendix A
Common Data Interface Block (CDIB)

A.1. General

A common data interface block is created for each file that you define (via a CDIB macroinstruction) in your program. This block is a 44-byte area that resides in the user region. It acts as a generalized function-passing mechanism. It provides a standard method for identifying a resource, indicating a desired function, and receiving status. The format of the CDIB is shown in Figure A-1.

BYTE	0	1	2	3
0	CDIBID* (CD$ID)	CDIB length* (CD$LGTH)	function code (CD$FCOD)	function control 1* (CD$FCTL)
4 ≈ 8	filename (CD$FNME)			
12	interface flags* (CD$FLG)		interface status (CD$ISTAT)	
16	function link* (CD$LINK)		function control 2* (CD$FCTL2)	function control 3* (CD$FCTL3)
20 ≈ 28	device independent completion status (CD$CS)			
32 ≈ 40	device dependent completion status (CD$DDINF)			

* System use only

Figure A-1. Common Data Interface Block (CDIB)

A.2. CDIB Field Definition and Usage

Each of the CDIB fields is described in the following subsections with its corresponding DSECT name shown in parentheses. Whenever you reference a field in the CDIB, you must do it symbolically by using the DSECT name for the field.

The CDIB DSECT can be generated and used as follows:

```
VTOC CDIB=YES
USING CD$CDIB,1
```

When this is done, the position of each field or bit indicator (relative to the beginning of the CDIB) is established via a DSECT equate (EQU); that is, the name of each field or bit indicator is equated to a value that represents its displacement within the CDIB. As a result, the testing of both a field or bit indicator is simplified because you do not worry about their position within the CDIB. The testing of a bit is further simplified because you don't have to know its position within a byte nor do you have to specify a mask to test it. Instead, all that is required is that you specify the length attribute of the bit indicator.

The following coding shows how to test a field and a bit indicator:

- Testing a field

```
CLI CD$ECOD,X'30'
BE processing routine for equal condition
```

- Testing a bit indicator

```
TM CD$ISUCC,L'CD$ISUCC
BZ processing routine for off condition
```

A.2.1. CDIB ID (CD$ID)

This field is for system use only.

A.2.2. CDIB Length (CD$LGTH)

This field is for system use only.

A.2.3. Function Code (CD$FCOD)

This field is intended for system use; however, you can use it for debugging purposes. This field is conditioned by the imperative macroinstructions you issue in your program. The CDIB DSECT expansion shows the values that are associated with the imperative macroinstructions.

A.2.4. Function Control 1 (CD$FCTL)

This field is for system use only.

A.2.5. Filename (CD$FNME)

The 1- to 8-character file name, which you specified in the LFD job control statement in your program execution job control stream, is placed in this field.

A.2.6. Interface Flags (CD$IFLG)

This field is for system use only.

A.2.7. Interface Status (CD$ISTAT)

This field contains indicators that data management uses to indicate the status of a requested function. These indicators, and their relation to other status information, are discussed in "Interface Status Considerations".

Interface Status Considerations

After performing all imperative functions, data management always returns control inline immediately after the imperative macroinstruction. Your program must then check the status of the function to verify whether or not it was performed successfully.

There are three status conditions that can be reported:

- Successful
- Error
- Exception

Data management uses two different methods to report the status: the indicators in the CD$ISTAT field and the program status word (PSW) condition code. It is suggested that, when you write your program, you use the PSW condition code to check for status. If you do, make sure that you do not execute any instructions that would change the condition code before you check it.

The indicators in the CD$ISTAT field are provided for debugging purposes so that you can see the status of the last function that was performed. They can be used if your program is written so that the PSW condition code is changed before it is checked. These indicators are as follows:

- CD$ISUCC

  If this indicator is on, successful status is indicated. If it is off, unsuccessful status is indicated and the CD$IEXC indicator indicates error or exception status.

- CD$IEXC

  If this indicator is on, exception status is indicated. The error code in the CD$ECOD field (see Table A-1 and "Unsuccessful Completion for All Imperative Macroinstructions" in this section) is always X'34' for exception status. If this indicator is off, error status is indicated. The error code in the CD$ECOD field is =/=X'00' and =/=X'34'.

As you can see, a great deal of status information is reported when a requested function is performed. This information has been summarized in Table A-1 so that you can see exactly what is provided when each of the status conditions is reported.

Table A-1. Summary of Status Information

Status Condition	PSW Condition Code	Binary Mask for Conditional Branch	CD$ISUCC Indicator Setting Field	CD$IEXC Indicator Setting	Error Code in CD$ECOD
Successful	0	8	On	N/A	N/A
Error	1	4	Off	Off	=X'00' and =X'34'
Exception	2	2	Off	On	X'34'

Normally when special functions are not requested in the RIB for a file, exception status is only reported when an end-of-file (EOF) condition is detected during the reading of a file. The error subcode in the CD$SCOD field (see Table A-2 and "Unsuccessful Completion for All Imperative Macroinstructions" in this section) will be X'00' to indicate exception status due to an *EOF condition*. Consequently, exception status under normal conditions always indicates EOF; therefore, you do not have to interrogate the CD$SCOD field to determine what caused the exception status.

There are special parameter specifications that you can specify in the RIB for a file that requests that conditions other than EOF be reported as exception status. (These special parameters must be explicitly specified; that is, they are not the defaults for these parameters.) Each of the exception conditions that these special parameters will cause to be reported has a specific error subcode assigned to it, and that code will be placed in the CD$SCOD field when the particular exception condition occurs. The exception conditions, the RIB parameters that cause them to be reported, the error codes that are placed in the CD$SCOD field, and the applicable devices are summarized in Table A-2.

Table A-2. Summary of Exception Condition Information

Exception Condition	RIB Parameter Specification	Error Sudcode Placed in CD$SCOD Field	Applicable Device
Input end-of-file	N/A	X'00'	All input devices
Input truncation	TRUNC=YES	X'01'	All input devices
Function key activated by operator	FNKEYS=symbol	X'02'	Workstation
Connect	CONFRE=YES	X'03'	Workstation
Free	CONFRE=YES	X'04'	Workstation
User tape label processing required	ULABEL=YES	X"05"	Tape
User tape label processing termination required	ULABEL=YES	X'06'	Tape
Forms overflow	PRINTOV=REPORT or YES	X'08'	Printer

If one or more of the special parameters shown in Table A-2 is specified in a RIB for a file, exception status will be reported for conditions other than EOF. Consequently, the CD$SCOD field should be interrogated to determine what condition caused the exception status to be reported.

Status Checking

As we mentioned before, the recommended status checking method is to use the PSW condition code. The following examples show how to use the PSW condition code to perform status checking under normal conditions (only possible exception status is EOF) and under special conditions (exception status in addition to EOF is reported).

- Status Checking Routine for Normal Conditions

```
**** IMPERATIVE MACROINSTRUCTION ****
         BAL   Rx,STATUS   CALL STATUS CHECKING ROUTINE
**** SUCCESSFUL STATUS PROCESSING ****
           .
           .
           .
STATUS   BCR   8,Rx        RETURN INLINE IF SUCESSFUL
         BC    4,ERROR     GO TO ERROR PROCESSING ROUTINE
         B     EOF         GO TO EOF PROCESSING ROUTINE
           .
           .
           .
ERROR    EQU   *           ERROR PROCESSING ROUTINE
           .
           .
           .
EOF      EQU   *           EOF PROCESSING ROUTINE
           .
           .
           .
```

- Status Checking Routine for Special Conditions

```
**** IMPERATIVE MACROINSTRUCTION ****
         BAL   Rx,STATUS   CALL STATUS CHECKING ROUTINE
**** SUCCESSFUL STATUS PROCESSING ****
           .
           .
           .
STATUS   BCR   8,Rx        RETURN INLINE IF SUCCESSFUL
         BC    4,ERROR     GO TO ERROR PROCESSING ROUTINE
**** EXCEPTION STATUS PROCESSING ****
         CLI   CD$SCOD,X'00'  IF END OF FILE CONDITION
         BE    EOF            GO TO EOF PROCESSING ROUTINE
           .                  IF NECESSARY,CHECK FOR EXCEPTION CONDITIONS
           .                  OTHER THAN EOF AND TAPE
           .                  USER LABEL PROCESSING REQUIRED
**** EXCEPTION STATUS PROCESSING-TAPE USER LABELS ****
         BAL   Ry,ULABEL     GO TO TAPE USER LABEL ROUTINE
         B     STATUS        CHECK STATUS
           .
           .
           .
```

```
ULABEL   EQU   *
           .                 PRE-I/O PROCESSING
           .
           .
         DMLAB (1),WORK,IO   READ/WRITE LABEL
         BCR   4,Ry          IF ERROR GO TO STATUS ROUTINE       Repeat for each label
         BC    2,LABEND      IF EXCEPTION,GO TO LABEL
           .                 PROCESSING TERMINATION ROUTINE
           .                 POST-I/O PROCESSING
           .
LABEND   DMLAB (1),WORK,END  TERMINATE LABEL PROCESSING
         BR    Ry            CHECK STATUS
WORK     DS    CL80          LABEL WORKAREA
           .
           .
           .
ERROR    EQU   *             ERROR PROCESSING ROUTINE
           .
           .
           .
EOF      EQU   *             EOF   PROCESSING ROUTINE
           .
           .
           .
```

A.2.8. Function Link (CD$LINK)

This field is for system use only.

A.2.9. Function Control 2 (CD$FCTL2)

This field is for system use only.

A.2.10. Function Control 3 (CD$FCTL3)

This field is for system use only.

A.2.11. Device Independent Completion Status (CD$CS)

This field is used to pass device independent information (common for all devices) back to the program from data management.

The format of this field for completion status depends on whether an OPEN or non-OPEN imperative macroinstruction was issued and whether or not the function was successfully completed.

Successful Completion for OPEN Imperative Macroinstructions

The format of the CD$CS field when a successful OPEN imperative macroinstruction is issued is shown in Figure A-2.

BYTE	0	1	2	3
20	flags (CD$OFLG)		record size (CD$ORCSZ)	
24	block/buffer size (CD$OBFSZ)		undefined	
28	device type (CD$ODEV)	reserved		

Figure A-2. Format of CD$CS Field for Successful Completion - OPEN Imperative Macroinstruction

As you can see in Figure A-2, there are four subfields. These are:

- Flags (CD$OFLG)

  This subfield is used to specify the type of file that was opened. The following indicators are used:

  - CD$FIXBK

    Indicates fixed-blocked record format

  - CD$VARBK

    Indicates variable-blocked record format

  - CD$FIXUN

    Indicates fixed-unblocked record format

  - CD$VARUN

    Indicates variable-unblocked record format

  - CD$UNDEF

    Indicates undefined record format

- Record Size (CD$ORCSZ)

  This field contains the record size specification generated at open time. This is the size of the work area required to hold the record. For variable-length records, this reflects the maximum record size (including the 4-byte record header).

- Block/Buffer Size (CD$OBFSZ)

  This field contains the I/O buffer size generated at open time. This is the minimum size for the I/O buffer. It includes any necessary overhead for variable-length records and padding.

- Device Type (CD$ODEV)

  This field specifies the device type. The DSECT defines the values that are associated with the various devices.

Successful Completion for Non-OPEN Imperative Macroinstructions

The format of the CD$CS field is undefined when a successful non-OPEN imperative macro is issued for all types of files.

Unsuccessful Completion for All Imperative Macroinstructions

The format of the CD$CS field for an unsuccessful imperative macroinstruction is shown in Figure A-3.

<table>
<tr><th>BYTE</th><th>0</th><th>1</th><th>2</th><th>3</th></tr>
<tr><td>20</td><td colspan="4">error flags
(CD$FNMC)</td></tr>
<tr><td>24</td><td colspan="2">reserved</td><td>error code
(CD$ECOD)</td><td>error subcode
(CD$SCOD)</td></tr>
<tr><td>28</td><td colspan="4">undefined</td></tr>
</table>

Figure A-3. Format of CD$CS Field for Unsuccessful Completion - Imperative Macroinstruction

As you can see in Figure A-3, there are three subfields. These are:

- Error Flags (CD$FNMC)

  This is a 4-byte field that contains error flags; that is, the bit settings in the various bytes in this field indicate the error or exception that has occurred. These flags are used for debugging purposes. It is suggested that you use the error code field (CD$ECOD) and the error subcode field (CD$SCOD) if your program needs to check for specific error or exception conditions.

- Error Code (CD$ECOD) and Error Subcode (CD$SCOD)

  This is a 2-byte field. The first byte contains the error code. If the error needs further definition, the second byte contains the error subcode. If the error code does not have a subcode associated with it, the error subcode will be X'00'. The error code and error subcode (if applicable) are included in the data management error message that is displayed. All data management error messages begin with DMxx, where xx is the error code and those that have error subcodes end with TYPE=nn where nn is the error subcode. A complete listing of the error messages and their subcodes can be found in the System Mesages Reference Manual, UP-8076.

Note that the following error codes do not have corresponding messages:

X'31'
Indicates record not found for a random function (DMINP or DMSEL macroinstruction) issued for a disk/diskette (MIRAM) file, or a data set label diskette file. There are no subcodes associated with this error code.

X'34'
Indicates exception status. See A.2.7 for a description of the error subcodes.

If your program needs to check for specific error or exception conditions, it is suggested that you interrogate the error code field (CD$ECOD), and, if applicable, interrogate the error subcode field (CD$SCOD).

A.2.12. Device Dependent Completion Status (CD$DDINF)

The format of this field for completion status depends on the type of file that is involved, whether or not an OPEN or non-OPEN imperative macroinstruction was issued, and whether or not the function was successfully completed. Tape and workstation files are the only file types for which there is a device dependent completion status.

Successful Completion for OPEN Imperative Macroinstruction - Workstation File

Figure A-4 shows the format of the CD$DDINF field when a workstation file is successfully opened.

BYTE	0	1	2	3
32	reserved		subelement 1 (CW$SUB1)	subelement 2 (CW$SUB2)
36	number of lines on screen (CW$LS)	number of bytes on line (CW$BL)	number of bytes jon full screen (CW$SS)	
40	number of volumes (CW$NOV)		reserved	

Figure A-4. Format of CD$DDINF Field for Successful Completion - OPEN Imperative Macroinstruction for Workstation File

As you can see in Figure A-4, there are six subfields. The first field, CW$SUB1, is cleared to X'00' when the file is opened. The second field, CW$SUB2, is set to X'00' if no workstations are connected and it is set to X'01' if one or more workstations are connected. The third field, CW$LS, specifies the length of the workstation screen. The fourth field, CW$BL, specifies the width (number of bytes) for the workstation screen. The fifth field, CW$SS, specifies the screen size. The sixth field, CW$NOV, specifies the number of volumes.

Successful Completion for Non-OPEN Imperative Macroinstruction - Workstation File

Figure A-5 shows the format of the CD$DDINF field when a non-OPEN macroinstruction is successfully completed for a workstation file.

BYTE	0	1	2	3
32	reserved		subelement 1 (CW$SUB1)	subelement 2 (CW$SUB2)
36	number of bytes read (CW$BTI)		remaining bytes available on screen (CW$BAOS)	
40	number of volumes after connect/free (CW$NOV)		reserved	

Figure A-5. Format of CD$DDINF Field for Successful Completion - Non-OPEN Imperative Macroinstruction for Workstation File

There are five subfields in Figure A-5. The first field, CW$SUB1, should always be X'00'. The second field, CW$SUB2, is used to specify the workstation to which the imperative macroinstruction is directed (see 4.4). When status is returned, this field specifies the workstation that is reporting status. The third field, CW$BTI, specifies the number of bytes read from the workstation screen. This field is used only when a DMINP imperative macroinstruction has been successfully completed. The fourth field, CW$BAOS, specifies the number of character positions remaining on the screen. This field is used to determine the number of records that can be blocked to the screen size specification when processing fixed-blocked (FIXBLK) or fixed-unblocked (FIXUNB) output records. The fifth field, CW$NOV, shows the number of volumes after a workstation is connected or freed.

Exception Status for Non-OPEN Imperative Macroinstruction - Workstation File

Figure A-6 shows the format of the CD$DDINF field when exception status is reported for a non-OPEN macroinstruction.

BYTE	0	1	2	3
32	flags (CW$FLG)		subelement 1 (CW$SUB1)	subelement 2 (CW$SUB2)
36	number of bytes read (CW$BTI)		remaining bytes available on screen (CW$BAOS)	
40	number of volumes after connect/free (CW$NOV)		reserved	

Figure A-6. Format of CD$DDINF Field for Exception Status - Non-OPEN Imperative Macroinstruction for Workstation File

There are six subfields. The first subfield, CW$FLG, is used to specify the exception that has occurred. The following indicators are used:

- CW$F15

  Indicates that function key 15 has been activated by the operator.

- CW$FKY

  Indicates that a function key has been activated by the operator.

- CW$CON

  Indicates that connect has occurred.

- CW$FRE

  Indicates that free has occurred.

- CW$TRUNC

  Indicates that data has been truncated.

- CW$DTA

  Indicates that data is being transferred with function keys.

The remaining subfields (CW$SUB1, CW$SUB2, CW$BTI, CW$BAOS, and CW$NOV) provide the same information in this case as that they provide for successful completion. See "Successful Completion for Non-Open Imperative Macroinstruction-Workststion File" in this section for details.

Exception Status for User Label Processing Required Condition or Successful Completion for the DMLAB (IO) Macroinstruction

Figure A-7 shows the format of the CD$DDINF field when exception status due to a user label processing required condition is reported or successful completion of the DMLAB (IO) macroinstruction occurs.

BYTE	0	1	2	3
32	reserved	label code (CT$LCODE)	undefined	
36	undefined			
40	undefined			

Figure A-7. Format of CD$DDINF Field for Exception Status for User Label Processing Required Condition or Successful DMLAB (IO) Macroinstruction

The subfield, CT$LCODE, indicates when user label processing is being performed. The alphabetic character O in this field indicates that the user label processing is being performed at file or volume open; V indicates at volume close; and F indicates at file close.

Appendix B
Error Flags

When an error or exception occurs, data management will make appropriate entries in the CD$FNMC (error flags) field of the CDIB (see Figure A-4). The significance of the bits in the error flags are shown in Table B-1. It is suggested that these flags be used for debugging purposes.

Table B-1. Significance of Bits in the Error Flags

Bit	MIRAM Disk	Tape	Reader Punch	Printer	Workstation	Diskette
			Byte 0			
0	Last block on track accessed	Reserved	Record size invalid	Line truncated	Undefined	Undefined
1	Invalid ID	Reserved	Reserved	Invalid control character	Undefined	Reserved
2	Invalid control structure	Invalid control structure	Validity check	Character mismatch	Invalid control structure	Invalid control structure
3	Hardware error	Hardware error	Hardware error	Hardware error	Hardware error	Hardware error
4	Error found in OPEN	Error found in OPEN	Error found in OPEN	Error found in OPEN	Error found in OPEN	Error found in OPEN
5	Error found in CLOSE	Error found in CLOSE	Error found in CLOSE	Error found in CLOSE	Error found in CLOSE	Error found in CLOSE
6	Invalid macro sequence	Invalid macro sequence	Invalid macro sequence	Invalid macro sequence	Invalid macro sequence	Invalid macro sequence
7	File lock error	Reserved	Reserved	Record size invalid	WAITF required	Reserved

continued

Table B-1. Significance of Bits in the Error Flags (cont.)

Bit	MIRAM Disk	Tape	Reader Punch	Printer	Workstation	Diskette
Byte 1						
0	I/O completed	I/O completed	I/O completed	I/O completed	I/O completed	I/O completed
1	Unrecoverable error	Unrecoverable error	Unrecoverable error	Unrecoverable error	Unrecoverable error	Unrecoverable error
2	Unique unit error	Unique unit error	Unique unit error	Unique unit error	Unique unit	Unique unit
3	Record not found	Reserved	Reserved	Reserved	Reserved	Record not found
4	Unit exception	Unit exception	Unit exception	Unit exception	Unit exception	Unit exception
5	Wrong length found	Reserved	Reserved	Reserved	Reserved	Wrong length found
6	End of track	Reserved	Reserved	Reserved	Reserved	End of track
7	End of cylinder	Reserved	Reserved	Reserved	Reserved	End of cylinder
Byte 2						
0	Command reject	Command reject	Command reject	Command reject	Command reject	Command reject
2	Intervention required	Intervention required	Intervention required	Intervention required	Intervention required	Intervention required
2	Output parity check	BUS out check	BUS out check	BUS out check	BUS out check	Output parity check
3	Equipment check	Equipment check	Card jam	Equipment check	Equipment check	Equipment check
4	Data check	Data check	Data check	Data check	Data check	Data check
5	Overrun	Overrun	Overrun	Overrun	Reserved	Overrun
6	STOP state	Word count zero	STOP state	STOP state	BUS in check	STOP state
7	Device check	Data converter check	Device check	Device check	Program alert	Data check

continued

Table B-1. Significance of Bits in the Error Flags (cont.)

Bit	MIRAM Disk	Tape	Reader Punch	Printer	Workstation	Diskette
			Byte 3			
0	Invalid record size	Invalid record size	Reserved	Reserved	Reserved	Invalid record size
1	Logical end of file	Logical end of file	Logical end of file	Forms overflow	Logical end of file	Logical end of file
2	Logical end of volume	Logical end of volume	Reserved	Reserved	Reserved	File space exhausted
3	Processing inhibited	Wrong length error	Reserved	Reserved	Reserved	Reserved
4	Invalid index condition	Reserved	Reserved	Reserved	Reserved	Reserved
5	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
6	Duplicate key	Reserved	Reserved	Reserved	Reserved	Reserved
7	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Appendix C
Labels for Disk and Diskette Files

C.1. General

This appendix describes the system standard labels for disk and format label diskette files that are supported by consolidated data management. Files within a disk or diskette volume may be stored in various locations; consequently, a directory of where the files are stored is required. This directory is called the volume table of contents (VTOC). In addition, each file requires various standard labels to describe the properties of the files and the volumes on which they reside.

The system standard labels include the volume label (VOL1) and seven types of format labels. These labels fall into two distinct groups:

- Volume Information Group
  - VOL1 label
  - Format 4 label
  - Format 5 label
  - Format 6 label
  - Format 0 label
- File Information Group
  - Format 1 label
  - Format 2 label
  - Format 3 label

The VOL1 label has a length of 84 bytes; all format labels are 140 bytes long.

The volume information group labels and the file information group labels are discussed in C.2 and C.3, respectively.

This appendix also describes the system standard labels for data set label diskette files. These labels are discussed in C.4.

C.2. Volume Information Group

The volume information group, comprising the VOL1 label and the format 4, 5, 6, and 0 labels, identifies the volume and defines the VTOC, the status of the VTOC, the available space within the volume, and the device-dependent characteristics of the volume on which the group resides.

Standard linkages maintained within the group are shown in Figure C-1. The VOL1 label, normally the first label in the group to be referenced, is written at cylinder 0, track 0, record 3 on each volume. The VOL1 label identifies the volume and contains a link to the format 4 label. The format 4 label defines the extent area occupied by the VTOC and the device-dependent characteristics of the volume; it also supplies a link to the first format 0 label.

The format 0 label describes format label records not in use, and is linked to the next available format 0 label.

The format 4 label also supplies a link to the first format 6 label, which defines space available within the extent areas of files sharing extents (split cylinder allocation). If more format 6 labels are required, they are linked in the same manner as format 0 labels. The format 6 label and its link from the format 4 label are present only if split-cylinder allocation has taken place.

The first format 5 label immediately follows the format 4 label, supplying an implied linkage. The format 5 label defines unused space on the volume in terms of full cylinders. Successive format 5 labels, if required, are linked one to another. The VTOC extent, as specified in the format 4 label, supplies an additional linkage because it is this area that must be searched in order to access the file information groups.

C.2.1. VOL1 Label

As each disk volume enters the system, it is given a unique identification code or volume serial number and the rudiments of a VTOC. The volume serial number and the address of the VTOC are placed in the VOL1 label.

The VOL1 label, identified by a key field and label identification field containing VOL1, is written by the disk initialization routine at cylinder 0, head 0, record 3.

The VOL1 label is the standard volume label. All reference to the VTOC of a given volume is made by first obtaining the VOL1 label, verifying the volume serial number, and, because the location of the VTOC may vary from volume to volume, using the VTOC address contained in the VOL1 label to locate the VTOC itself.

The format of the VOL1 label is shown in Figure C-2; Table C-1 summarizes its contents.

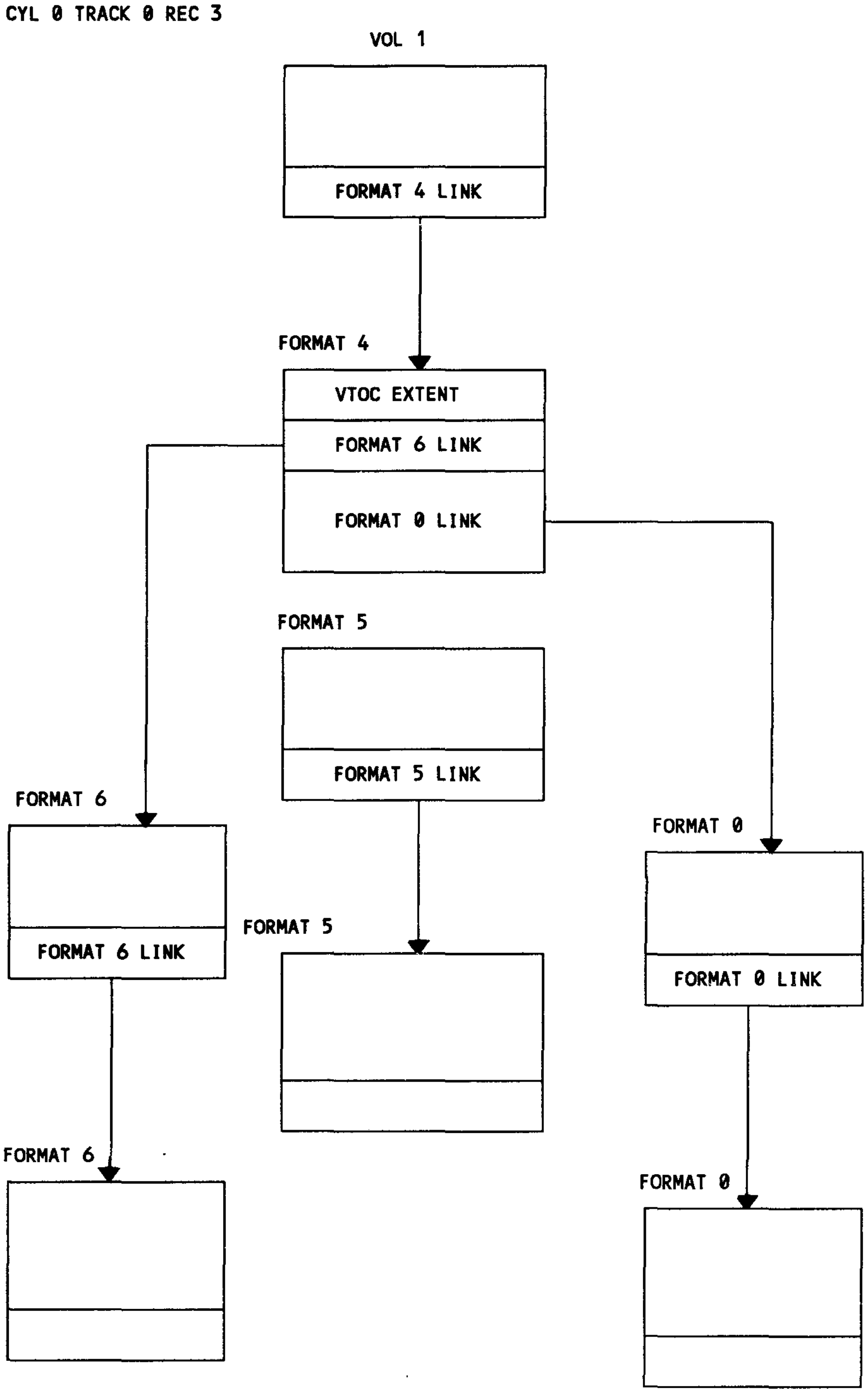


Figure C-1. VTOC Volume Information Label Group

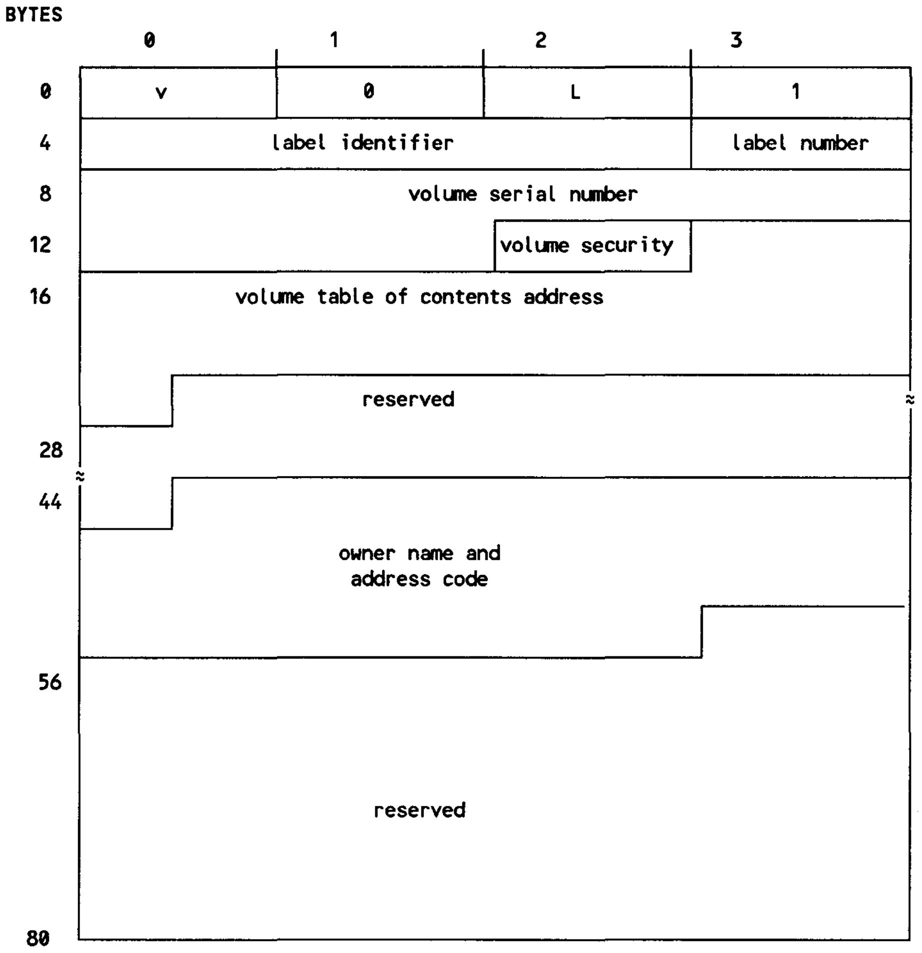


Figure C-2. VTOC VOL1 Label

Table C-1. Contents of VOL1 Label

Label	Initialized	Bytes	Code	Description
DL$VL	Disk prep	0-3	EBCDIC	Key - contains VOL1.
DL$VL1		4-6		Label identifier - VOL.
		7		Label number - always 1.
DL$VSN		8-13		Volume serial number - a unique code assigned to a disk pack when it enters the system. The same code should appear visually on the disk pack for operator identification.
DL$VSB	Disk prep	14	Binary	Volume security - reserved for future use.
DL$VTC		15-24	Discontinuous binary*	VTOC address - This field is used to point to the format 4 label, which starts the VTOC. The address is in the form cchhrr in bytes 15 through 19. Bytes 20 through 24 are zero.
		25-44		Reserved
DL$ONR	Disk prep	45-54	EBCDIC	Optional owner name and address code - an installation-supplied user identifier.
		55-83		Reserved

* For discontinuous binary, each subfield is treated as a distinct binary entity. In the notation cchhr, each different letter represents a subfield.

C.2.2. Disk Format 4 Label (VTOC)

The format 4 label describes the VTOC itself and is the first record of the VTOC. In addition to describing the area occupied by the VTOC and its current status, the format 4 label contains information on all device-dependent characteristics of the volume on which it resides. An indicator in the format 4 label states whether the format 5 label contains valid information.

The format 4 label is written by the disk initialization routine at the disk address specified in the VOL1 label. Only one format 4 label may exist on a given volume.

The address of the first available label record (a format 0 label) is placed in the format 4 label for use by disk space management. An additional linkage is created and maintained by disk space management that specifies the first active format 6 label and is used only during split-cylinder allocation of data files. Figure C-3 shows the format 4 label; Table C-2 summarizes its contents.

BYTES	0	1	2	3
	key field			
44	format ID	last active format 1		
48			available file label records	
52	highest alternate track			
56	number of alternate tracks		VTOC indicators	number of extents
60	reserved		device size	
64	device size		track length	
68	record overhead			flag
72	tolerance		labels	blocks
	pointer to format 0 label			
80		reserved		
100	pointer to format 6 label			
104		VTOC extent		
112	reserved			
136				

Figure C-3. Disk Format 4 Label

Table C-2. Contents of Disk Format 4 Label

Label	Initialized by	Bytes	Code	Description
DL$KY4	Disk prep	0-43	Hexadecimal	Key field - Each byte of this field contains 04_{16}.
DL$ID4	Disk prep	44	EBCDIC	Format ID - always 4 for format 4 label.
DL$LF4	Space management	45-49	Discontinuous binary	Last active format 1 - the address, in the form cchhr, used for a search on filename.
DL$AF4	Disk prep Space	50-51	Binary	Available file label records - number of unusual records in the VTOC
DL$HA4	Disk prep Space	52-55	Discontinuous binary	Highest alternate track - address, in the form cchh, of alternate tracks set aside in case of bad tracks.
DL$AT4	Disk prep	56-57	Binary	Number of alterate tracks.
DL$VI4	Space management	58		VTOC indicators - Bit Contents Meaning 0 1 A format 5 label, if present, contains information. 1-7 0 Unused
DL$XC4	Disk prep	59	Binary	Number of extents - contains 01_{16} to indicate the one extent in the VTOC.
	Disk prep	60-61		Reserved
DL$DS4	Disk prep	62-65		Device size - indicates the number of cylinders and the number of heads per cylinder on the device, in the form cchh.
DL$TL4	Disk prep	66-67		Track length - number of available bytes on a track, exclusive of home address and record 0.
DL$RO4	Disk prep	68-70		Record overhead - ILK describes overhead bytes on track, where I is for a keyed record which is not the last on track, L is for a keyed record which is the last on track, and K is a decrement which is applied to records which have no key.

continued

Table C-2. Contents of Disk Format 4 Label (cont.)

Label	Initialized by	Bytes	Code	Description
DL$FG4	Disk prep	71	Binary	Flag - Bit Meaning 0-5 Reserved 6,7 Device-dependent characteristics
DL$TO4	Disk prep	72-73		Tolerance - a device-dependent factor which is used to calculate effective record lengths for that device.
DL$LT4	Disk prep	74		Labels per track - a device-dependent factor specifying the number of 140-byte labels possible in a VTOC track.
DL$BK4	Disk prep	75		Blocks per track - a device-dependent factor specifying the number of directory blocks of a partitioned file which can be written on a track.
DL$F04	Disk prep	76-80	Discontinuous binary	Format 0 address in the form cchh - points to the first available format 0 record in the VTOC.
		81-99		Reserved
DL$F64	Space management	100-104		Format 6 address in the form cchhr - points to the first format 6 label created by space management.
DL$VX4	Disk prep	105-114		VTOC extent - describes the extent occupied by the VTOC itself. The format of this field is identical to the fields describing the extent in the format 1 and 3 labels.
		115-139		Reserved

C.2.3. Disk Format 5 Label

The format 5 label is the second record in the VTOC and is used to maintain control of the available extents in the volume at any time. The format 5 label is initialized by the disk initialization routine and maintained by the disk space management routine. Each format 5 label may define up to 26 available extents. Format 5 labels may be linked together should more than one become necessary. Figure C-4 shows the format 5 label; Table C-3 summarizes its contents.

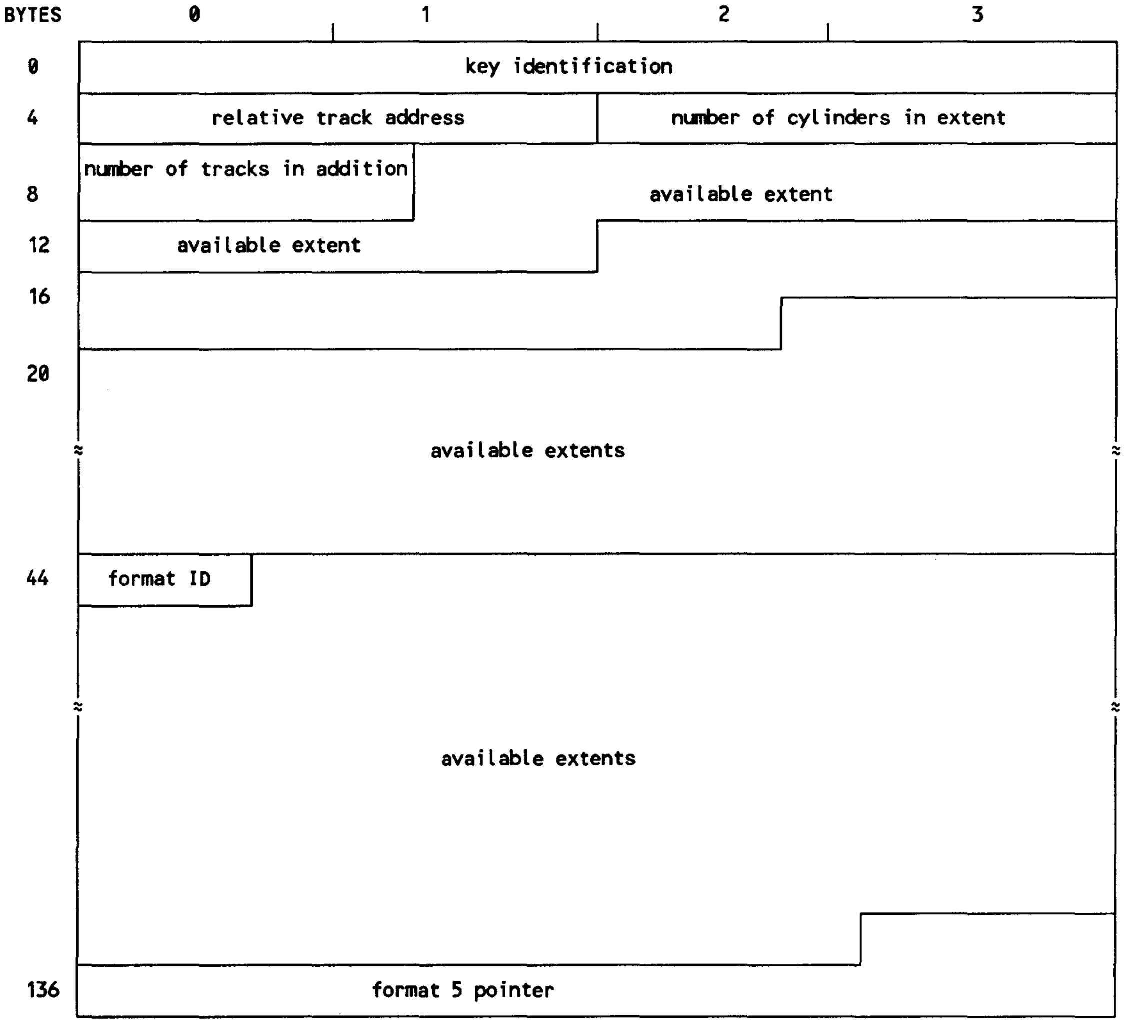


Figure C-4. Disk Format 5 Label

Table C-3. Contents of Disk Format 5 Label

Label	Initialized by	Bytes	Code	Description
DL$ID5	Disk prep	0-3	Hexadecimal	Key identification - Each byte of this field contains 05_{16}.
DL$XT5	Disk prep	4-5	Discontinuous binary	Relative track address - start of extent.
DL$XC5	Disk prep	6-7	Binary	Number of cylinders in extent.
DL$XE5	Disk prep	8	Binary	Number of tracks in extent in addition to the cylinders.
	Space management	9-13		Available extent - describes another extent in fields with the same format as bytes 4 through 8.
	Space management	14-43		Six more available extents.
DL$FI5	Disk prep	44	EBCDIC	Format ID - always 5, format 5 label.
DL$XS5	Space	45-134		Eighteen more available extents.
DL$CP5	Space management	135-139	Discontinuous binary	Pointer - indicates the address of another format 5 label, in the form cchhr. Binary 0 if no further label.

C.2.4. Disk Format 6 Label

The format 6 label is used to control split-cylinder allocation. Each format 6 label contains a code that identifies all member files sharing the same extent area. Each member file is allocated from 1 to n tracks within each cylinder allocated to the set, where n is the number of tracks per cylinder, minus one. Additionally, a head pool is maintained that specifies all tracks not currently allocated and available for use by new members of the same split-cylinder set. A format 6 label is created for each split-cylinder set defined.

The format 6 label is created and maintained by the disk space management routines. Each label contains the disk address of the format 3 label that defines the extents allocated to that split member set. The disk address of the first format 6 label is maintained in the format 4 label. If more that one format 6 label is required, they are linked together.

Note that no extent information is maintained in the format 1 label of a split-cylinder file and that all members of a split-cylinder set share a common format 3 label. Figure C-5 shows the format 6 label; Table C-4 summarizes its contents.

BYTES	0	1	2	3
0	key identification			device high head
4	split set identifier	common format 3		
8	disk address	head available	high head available	
12	9 additional head pool entries			
28			primary member format 1 address	
32	primary member format 1 address			
	19 additional format 1 label disk address entries			
108			reserved	
132			format ID	
136	format 6 pointer			

Figure C-5. Disk Format 6 Label

Table C-4. Contents of Disk Format 6 Label

Label	Initialized by	Bytes	Code	Description
DL$ID6	Space management	0-2	Hexadecimal	Key identification- always 060606_{16}.
DL$HH6		3		Device high head.
DL$SET		4-5		Set identifier - identifies each member file of the split-cylinder set.
DL$IDF36		6-9	Discontinuous binary	Disk address of the format 3 label binary shared by all member files.
DL$LHA6		10	Hexadecimal	Low head available in the specified extent areas.
DL$HHA6		11	Hexadecimal	High head available in the specified extent areas.
	Space management	12-29	Hexadecimal	Nine additional entries for low and high available head.
DL$IDF16		30-33	Hexadecimal	Format 1 disk address of primary member in the form ccrh. 19 additional split set format 1 disk disk address entries in the same format as bytes 30-33.
		34-109	Hexadecimal	Format 1 label disk addresses of up to 19 dditional members of the split-cylinder set in the same format as bytes 30-33.
		110-133		Reserved
DL$FI6		134	Hexadecimal	Format identification X'F6'.
DL$CP6		135-139	Discontinuous binary	Pointer to next format 6 label, in the form cchhr.

C.2.5. Disk Format 0 Label

The format 0 label is used to identify label records in the VTOC not currently in use.

Format 0 records are initialized by the disk initialization routine. The address of the first format 0 is placed in the format 4 label, and each format 0 label is linked to the next. The remainder of the label is filled with binary zeros. Figure C-6 shows the format 0 label; Table C-5 summarizes its contents.

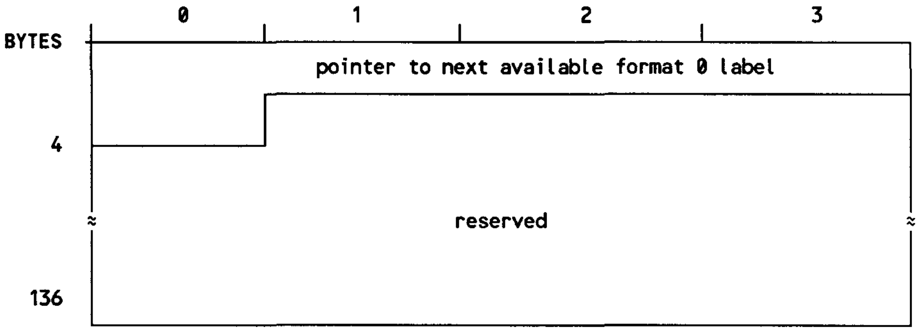


Figure C-6. Disk Format 0 Label

Table C-5. Contents of Disk Format 0 Label

Label	Initialized	Bytes	Code	Description
	Disk prep	0-4	Discontinuous binary	Disk address in the form cchhr of the next available format 0 label.
		5-139	Binary zero	Reserved

C.3. File Information Group

The file information group (Figure C-7) is composed of the format 1, format 2, and format 3 labels. The format 1 label is normally the first referenced label of the group. It is obtained by executing a key search for file ID in the VTOC extent defined in the format 4 label.

The format 1 label defines the characteristics of the file and may define up to three extents occupied by the file. The format 1 label is linked to the format 2 label, which is used to further define the file. These two labels are present for each file in the volume.

The format 3 label is used to define the extent area occupied by the file and is an optional label, except that it exists for all files created when using split-cylinder allocation. For all other files, the format 3 label exists only if the file occupies more than three separate extent areas.

C.3.1. Disk Format 1 Label

A format 1 label exists for each file in a volume. As many as three extents of a file may be described in the format 1 label, provided that the file is not a member of a split-cylinder set.

The format 1 label is initialized by the disk space management routines. It is maintained by both the space management and data management routines. The format 1 label contains a pointer to the format 2 label. Figure C-8 shows the format 1 label; Table C-6 summarizes its contents.

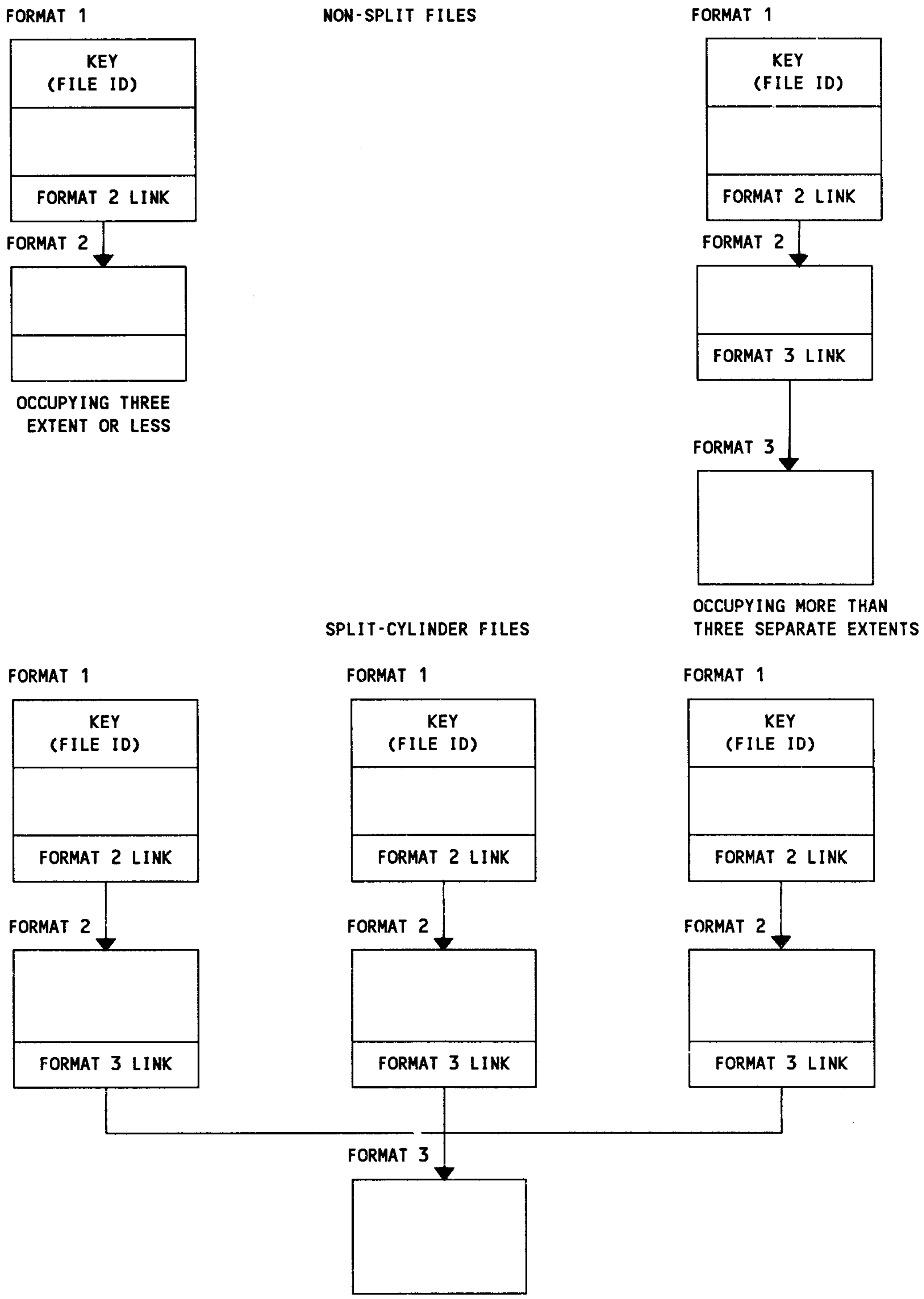


Figure C-7. File Information Group Label Chain

BYTES	0	1	2	3
0	file ID (key field 44 bytes)			
44	format ID	file serial number		
48				volume sequence no.
52	volume sequence no.	creation date		
56	expiration date			extent count
60	option codes	number of PCAs	file type	
64	PCA 1 block size		reserved	
68	PCA 1 record format	PCA 2 specifications		
72				
76	PCA 3 specifications			
80	PCA 4 specifications			
84	PCA 5 specifications			
88		PCA 6 specifications		
92				
96	PCA 7 specifications			data set indicator
100	reserved		secondary allocation increment	file low head
104	file high head	extent type	extent sequence no.	lower limit
108	lower limit			upper limit
112	upper limit			
116	second extent (10 bytes)			
128	third extent (10 bytes)			
136	format 2 pointer			

Figure C-8. Disk Format 1 Label

Table C-6. Contents of Disk Format 1 Label

Label	Initialized by	Bytes	Code	Description
DL$KEY1	Space management	0-43	EBCDIC	File identifier - Each file must have a unique 1- to 44-byte name in this key field, the first six bytes of which may be a lock ID. A search of the VTOC is made on this name.
DL$ID1		44	EBCDIC	Format identifier - always 1, for format 1 label.
DL$FS1	Data management	45-50	EBCDIC	File serial number - identifies the volume on which the file starts, is a 6-digit alphanumeric number, and is the same as the volume serial number of the volume on which the file starts. The first volume of a file is defined by the first job control DVC statement in the device assignment set for the file.
DL$VS1		51-52	Binary	Volume sequence number - indicates the number of this volume relative to the first volume in the file. The first volume of a file is defined by the first job control DVC statement in the device assignment set for the file.
DL$CD1	Space management	53-55	Discontinuous binary	Creation date - format is ydd (year-day-day), where y is 0 to 99, and dd is 1 to 366.
DL$ED1		56-58	Discontinuous binary	Expiration date - the date when the file may be deleted. Format is the same as the creation date.
DL$XC1		59	Binary	Extent count - specifies the number of extents currently constituting the file, or portions of it, on this volume.

continued

Table C-6. Contents of Disk Format 1 Label (cont.)

Label	Initialized by	Bytes	Code	Description
DL$OC1	Space management and data management	60	Binary	Option codes Bit Content Meaning 0 1 Read allow 1 1 Write allow 2 1 New file 3 1 Partitions cylinder aligned 4 1 User labels 5 1 Track alignment extension 6 1 Single mount 7 1 Track extend indicator
DL$PC1	Data management	61	Binary	PCA count - number of partitions that constitute the file.
DL$FT1	Data management	62	Hexadecimal	File type Hexadecimal Code Meaning 90 MIRAM 02 SAT 00 Undefined
DL$FT1	Data management	63*	Hexadecimal	File Type Bit Meaning 0 Reserved 1 MIRAM characteristics 2-7 Reserved

continued

Table C-6. Contents of Disk Format 1 Label (cont.)

Label	Initialized by	Bytes	Code	Description
DL$BL1	Data management	64-65	Binary	Reserved for PCA1 block length - size of fixed-length blocks or maximum size of variable-length blocks.
DL$RL1	Data management	66-67	Binary	Reserved.
DL$RF1	Data management	68	Binary	Reserved for PCA1 record format Bit Content Meaning 0,1 Reserved 2 0 Records have no keys. 1 Records have keys. 3 Reserved 4 Reserved 5 1 Always set 6 Reserved 7 1 Records are interlaced.
	Data management	69-73	Discontinuous binary	Partition 2 descriptor; block size, and record format for partition 2.
	Data management	74-78	Discontinuous binary	Partition 3 descriptor for SAT. Binary zeros for MIRAM.
	Data management	79-83	Discontinuous binary	Partition 4 descriptor for SAT. Binary zeros for MIRAM.
	Data management	84-88	Discontinuous binary	Partition 5 descriptor for SAT. Binary zeros for MIRAM.
	Data management	89-93	Discontinuous binary	Partition 6 descriptor for SAT. Binary zeros for MIRAM.
	Data management	94-98	Discontinuous binary	Partition 7 descriptor for SAT. Binary zeros for MIRAM.
DL$DS1	Space management	99	Binary	Data set indicators - reserved for future use.

continued

Table C-6. Contents of Disk Format 1 Label (cont.)

Label	Initialized by	Bytes	Code	Description
DL$KL1	Data management	100-101	Binary	Reserved
DL$SA1		102	Binary	Secondary allocation increment - the number of cylinders of disk storage to be requested for each dynamic extention of the file.
DL$LH1		103	Hexadecimal	File low head - split-cylinder allocation.
DL$HH1		104	Hexadecimal	File high head - split-cylinder allocation.
DL$XT1		105	Hexadecimal	Extent type indicator - Code Meaning 00 No valid extent described 90 MIRAM 02 SAT FF Job Control
DL$XS1		106	Binary	Extent sequence number - relative number of extents in multiple-extent volume.
DL$XL1		107-110	Discontinuous binary	Lower limit - the address specifying the start of the extent, in the form cchh.
DL$XU1		111-114	Discontinuous binary	Upper limit - the address specifying the end of the extent, in the form cchh.
		115-124		Second extent - same format as described for bytes 105 through 114.
		125-134		Third extent - same format as second extent.
DL$CP1	Space management	135-139	Discontinuous binary	Continuation pointer - the address of a format 2 label. The address is in the form cchhr.

*Byte 63 is meaningless unless byte 62 = X'90'.

C.3.2. Disk Format 2 Label

The format 2 label is used as an extension to the format 1 label to further describe the file.

For MIRAM files, bytes 13 through 43 are used to carry index and file characteristic information. For SAT library files, bytes 32 through 47 are used to carry information on the library text and directory; bytes 13 through 31 contain binary zeros.

The format 2 label is initialized by space management and maintained by data management. The label is always present and is linked from the format 1 label. The link field in the format 2 label points to a format 3 label, if used. This pointer is present for all split-cylinder files and for nonsplit-cylinder files requiring more than three extents. If it is not present, the field is filled with binary zeros. Figure C-9 shows the format 2 label; Table C-7 summarizes its contents. The format of the MIRAM file information area is shown in Figure C-10, and the contents of this area are listed in Table C-8. The format of the library file information area is shown in Figure C-11, and the contents of this area are listed in Table C-9.

<table>
<tr><th>BYTES</th><th>0</th><th>1</th><th>2</th><th>3</th></tr>
<tr><td>0</td><td>key ID</td><td>reserved</td><td>key length or lace factor</td><td>reserved</td></tr>
<tr><td>4</td><td colspan="3">EOD ID</td><td>reserved</td></tr>
<tr><td>8</td><td>key length or lace factor</td><td>reserved</td><td colspan="2">EOD ID</td></tr>
<tr><td>12</td><td>EOD ID</td><td colspan="3" rowspan="2">(For SAT library, see Figures C-9 and C-11.
For MIRAM, see Figures C-8 and C-10.)</td></tr>
<tr><td>40</td><td></td></tr>
<tr><td>44</td><td colspan="2">reserved</td><td colspan="2">blocks/track, PCA1</td></tr>
<tr><td>48</td><td>PCA1 ID
0 2</td><td>relative track addr-1
3 15</td><td colspan="2">tracks
16 31</td></tr>
<tr><td>52</td><td>PCA2 ID
0 2</td><td>relative track addr-2
3 15</td><td colspan="2">tracks
16 31</td></tr>
<tr><td></td><td colspan="4">≈ ≈</td></tr>
<tr><td>128</td><td colspan="2">track per cylinder</td><td colspan="2">file low head number</td></tr>
<tr><td>132</td><td colspan="2">relative extent count</td><td>flags</td><td></td></tr>
<tr><td>136</td><td colspan="4">format 3 pointer</td></tr>
</table>

NOTE:

Bytes 0 through 12 of the disk format 2 label are the same for MIRAM and SAT library files.

Figure C-9. Disk Format 2 Label

Table C-7. Contents of Disk Format 2 Label

Label	Initialized by	Bytes	Code	Description
DL$SID2	Space management	0	Hexadecimal	Key identification X'02'
DL$SPC2	Space management	1		Reserved
DL$SLF2		2		Key length or lace factor.
DL$SLA2		3		Reserved; for the library file information area, contains binary zeros.
DL$SED2	Data management	4-6 7-12	Binary	End of data ID - relative block address plus 1 of the last blocks written into the partition. A 6-byte partition descriptor entry in the same form as bytes 1-6.
	Data management	13-43	Hexadecimal	For MIRAM file information area (Table C-8 and Figure C-10). For SAT library files, see library file information area (Table C-9 and Figure C-11).
		44-45		Unused (binary zero) for all but indexed files; reserved for indexed files.
DL$SBPT2		46-47		Blocks per track - the number of blocks per track in the first or only partition of the file.
DL$SXAR2	Data management	48-127	Discontinuous binary	Logical extent table area - These entries are 4 bytes in length and specify PCA ID in 3 bit, starting relative track address in 13 bits, and number of tracks in that address. From one to twenty 4-byte logical extent entries may be placed in this 80-byte area. Each 4-byte entry has the following format: Bit Meaning 0-2 The high order three bits of the logical extent identify the partition to which it is assigned. (This value may be from 1 to 7.)

continued

Table C-7. Contents of Disk Format 2 Label (cont.)

Label	Initialized by	Bytes	Code	Description
DL$SXAR2	Data management	48-127		Bit Meaning 3-15 The next 13 bits indicate the relative track address of the logical extent. 16-31 If the first bit (bit 16) of the track field is set on, a value of 8192 must be added to the relative track address to indicate the the relative track address of the logical extent. If bit 17 is set on, a value of 16384 must be added to indicate the relative track address. The remaining 14 bits indicate the number of tracks contained in the extent.
DL$TPC2		128-129	Hexadecimal	Tracks per cylinder for this file.
DL$FLH2		130-131		File low head - the lowest head number in the assigned cylinders accessible for this file.
DL$SXCT2		132-133		Number of relative extents contained in this label.
DL$SFL2		134		Flags Bit Content Meaning 0 1 Extended Format 2 label active. (Bytes 140-255 are the extended label, which does not apply to 8430/33 disks.) 1-4 Reserved 5 1 Library lace adjustment, type 2 6 1 Library lace adjustment, type 3 7 1 9400 SAT compatible
DL$SCID2	Space management	135-139	Discontinuous binary	The disk address, in the form cchhr, of the format 3 label (if required) asssociated with this file.

continued

Table C-7. Contents of Disk Format 2 Label (cont.)

Label	Initialized by	Bytes	Code	Description
DL$ACTIV	Data management	140	Discontinuous Binary	Extensions Flags Bit Content Meaning 0 1 Format 2 label not updated. 1-8 Unused
DL$ACDTE	Space management	141-143	Discontinuous binary	Access date - the date of the last access to this file. Format is ydd(year-day-day) where y is 0-99 and dd is 1-366.
DL$ACTME	Space management	144-147	Discontinuous binary	Accesstime - the time in milliseconds of the last access to this file.
DL$UPDTE	Space management	148-150	Discontinuous binary	Last modification date - the date of the last modification to this file. Same format as DL$ACDTE.
DL$UPTME	Space management	151-154	Discontinuous binary	Last modification time - the time of the last modification to this file. Same format as DL$ACTME.
DL$UPID	Space management	155-162	EBCDIC	Jobname/Logon-id - the jobname or logon-id identifying the user program that last modified this file.
DL$UPACC	Space management	163	Binary	Access code - the file share access code used in the last modification of this file.
DL$PUPD	Data management	164-166	Discontinuous Binary	Previous modification date. Same format as DL$ACDTE.
DL$PUPT	Data management	167-170	Discontinuous Binary	Previous modification time. Same format as DL$ACTME

continued

Table C-7. Contents of Disk Format 2 Label (cont.)

Label	Initialized by	Bytes	Code	Description
DL$PUPID	Data management	171-178	EBCDIC	Previous modification jobname/logon-id. Same format as DL$UPID.
DL$PUPAC	Data management	179	Binary	Previous modification access code. Same format as DL$UPACC.
DL$SORC	Data management	180-183	Discontinuous Binary	Sector offset and record count - the sector offset value for file recovery and the record count of the file from the last INIT. Used for restore processing.
DL$DELRC	Data management	184-187	Discontinuous Binary	Deleted record count - a count of the number of deleted records in the MIRAM file.
DL$LETA	Data management	188-227	Discontinuous Binary	Logical extent table information from last file initialization. Used when // DD RESTORE=YES is specified.
DL$DRDTE	Data management	228-230	Discontinuous Binary	Date of last backup of this file by DMPRST. Same format as DL$ACDTE.
DL$DRTME	Data management	231-234	Discontinuous Binary	Time of last backup of this file by DMPRST. Same format as DL$ACTME.
	Data management	235-255		Unused.

BYTES		13	14	15
		key location for key 1		key length for key 1
16	used for IRAM processing or key characteristic flag byte (MIRAM)	descriptor for key 2		
20	descriptor for key 2	descriptor for key 3		
24	descriptor for key 3	descriptor for key 4		
28	descriptor for key 4	descriptor for key 5		
32	descriptor for key 5	sector offset	number of records in file, plus 1	
36	number of records in file, plus 1	record size		fine-level index block size in sectors
40	number of index levels	last fine-level index block		

NOTE:

See Figure C-9 for the format of bytes 0 through 12.

Figure C-10. Disk Format 2 Label, MIRAM File Information Area

BYTES	28	29	30	31
28	directory partition lace factor			
32	directory partition lace adjustment factor			
36	text partition lace factor			
40	text partition lace adjustment factor			
44	number of library blocks per track			

NOTE:

In the format 2 label for SAT library files, bytes 13-27 are reserved and contain binary zeros. See Figure C-9 for the format of bytes 0 through 12.

Figure C-11. Disk Format 2 Label, SAT Library File Information Area

Table C-8. Contents of Disk Format 2 Label, IRAM/MIRAM File Information Area

Label	Initialized by	Bytes	Code	Description
DL$XILOC	Data	13-14	Hexadecimal	Key location for key 1
		15		Key length for key 1
		16	Hexadecimal (IRAM characteristics)	Used for IRAM characteristics file index processing
			Binary MIRAM characteristics)	For MIRAM files, byte 16 contains key 1 contains key 1 characteristics: This Bit On — Means Bit 0 — Duplicates allowed for this key Bit 1 — Key change allowed for this key during update Bit 2-4 — Unused

continued

Table C-8. Contents of Disk Format 2 Label, IRAM/MIRAM File Information Area (cont.)

Label	Initialized by	Bytes	Code	Description
				This Bit On — Means *Bit 5 — Index-only records permitted in this file *Bit 6 — Variable length record format *Bit 7 — Record control byte (RCB) present
		17-20	Discontinuous	Descriptor for key 2 (binary zeros for IRAM characteristics)
		21-24		Descriptor for key 3 (binary zeros for IRAM characteristics)
		25-28		Descriptor for key 4 (binary zeros for IRAM characteristics)
		29-32		Descriptor for key 5 (binary zeros for IRAM characteristics)
DL$MARSO		33	Hexadecimal	Sector offset for files created with recovery
DL$COUTR		34-36		Number of records in file (plus 1)
DL$REC		37-38		Record size
DL$CSIZ		39		Fine-level index block size in sectors
DL$CLEV		40		Number of index levels
DL$FAB		41-43		Relative block-number of last block of the fine-level index

* These bit positions are unused in the descriptors for keys 2 through 5.

NOTE:

Descriptions pertaining to IRAM files also apply to MIRAM files with IRAM characteristics.

Table C-9.Contents of Disk Format 2 Label, SAT Library Information Area

Label	Initialized by	Bytes	Code	Description
		0-27		Bytes 0 through 12 are the same as described in Table C-7; bytes 13 through 27 are reserved and contain binary zeros.
DL$DIRL2	Data management	28-31		Hardware-adjusted lace factor for the directory partition
DL$DIRF2	Data management	32-35		Rotational adjustment for director lace factor
DL$TXTL2	Data management	36-39		Hardware-adjusted lace factor for the library text partition
DL$TXTF2	Data management	40-43		Rotational adjustment factor for the library text
-	Data management	44-47		Number of library blocks per track

C.3.3. Disk Format 3 Label

The format 3 label is used to maintain extent information for the file. For split-cylinder files, a format 3 label is always present. For files not using split-cylinder allocation, a format 3 label for the file exists only when more than three extents are required. The format 3 label is initialized and maintained by the disk space management routines. The format 3 label, when required, is always linked from a format 2 label. Figure C-12 shows the format 3 label; Table C-10 summarizes its contents.

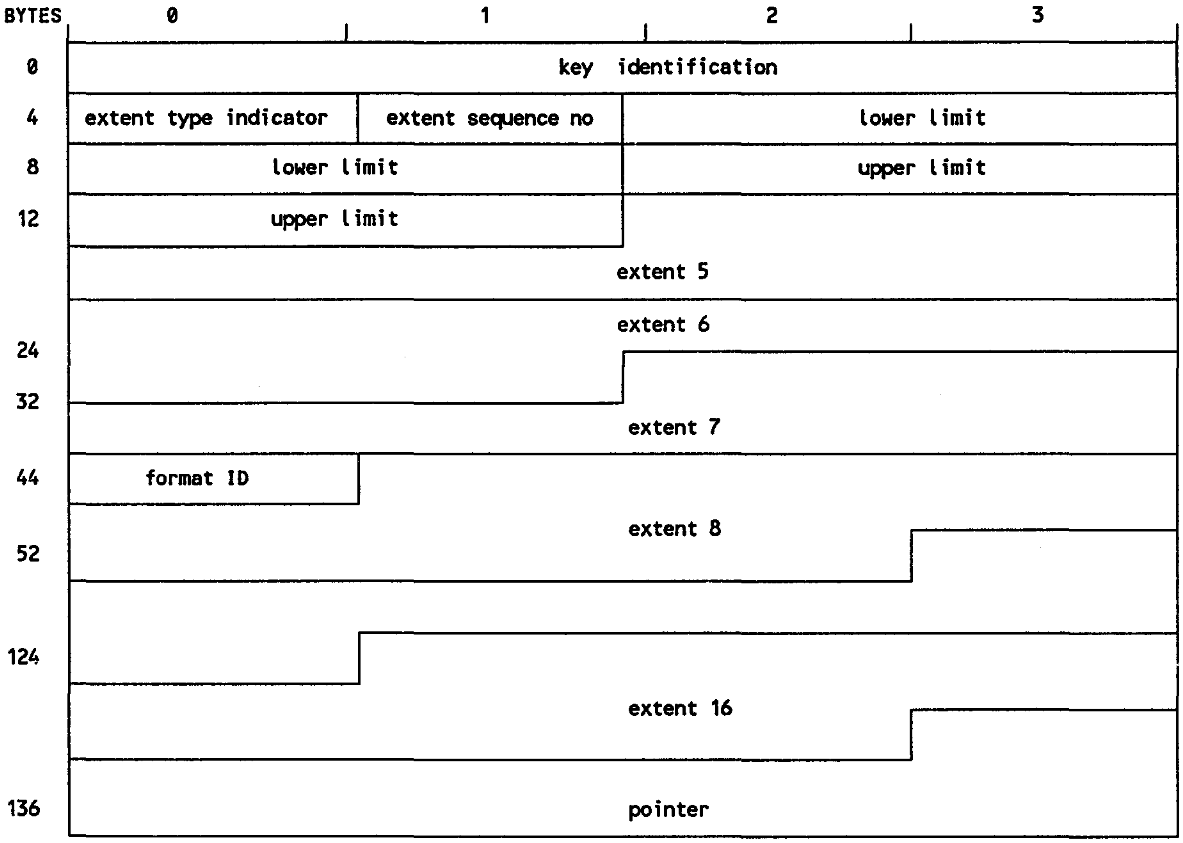


Figure C-12. Disk Format 3 Label

Table C-10. Contents of Disk Format 3 Label

Label	Initialized by	Bytes	Code	Description
DL$ID3	Space management	0-3	Hexadecimal	Key identification - each byte contains 03_{16}.
DL$XT3		4	Hexadecimal	Extent type indicator - Code Meaning 00 No valid extent described 01 Prime data area
DL$SN3		5	Binary	Extent sequence number - relative number of extents in this volume of the file.
DL$XL3		6-9	Discontinuous binary	Lower limit - starting track address of the extent, in the form cchh.
DL$XU3		10-13	Discontinuous binary	Upper limit - terminating track address of the extent, in the form cchh.
		14-23		Extent 5 - same format as described for bytes 4 through 13 for this extent.
		24-43		Extents 6 and 7.
DL$FI3		44	EBCDIC	Format identifier - always 3, for format 3 label.
		45-134		Extents 8 and 16.
DL$CP3		135-139	Discontinuous binary	Pointer - address of next format 3 label, in the form cchhr. Binary zero if no further label.

C.4. Diskette Data Set Labels

Each diskette data set volume contains two types of labels: a VOL1 label that identifies the volume and the individual file labels that identify the characteristics of each file on the volume.

C.4.1. Diskette Data Set VOL1 Label

Figure C-13 illustrates the format of the VOL1 label. Table C-11 explains the contents of each field in the VOL1 label.

C.4.2. Diskette Data Set File Label

Figure C-14 illustrates the diskette data set file label format. Table C-12 explains the contents of each field in the diskette file label.

BYTES	0	1	2	3
0	V	O	L	1
4	volume serial number			
8			volume accessibility	
12	reserved			
20				
24	system code			
36		owner name and address code		
40				
48				
52	reserved			
68				volume surface indicator
72	extent arrangement indicator	reserved		physical sector length
76	physical sector sequence code		reserved	label standard version
80				
124				

Figure C-13. Diskette Data Set VOL1 Label

Table C-11. Diskette Data Set VOL1 Label Description

Field	Byte Position	Description
Label ID	0-3	Contains the 4-byte label identifier VOL1.
Volume serial number	4-9	A unique user assigned code to a diskette when it enters the system. The same code should appear visually on the diskette cover for operator identification.
Volume accessibility	10	Reserved for future use.
Reserved	11-23	Reserved
System code	24-36	Reserved for future use.
Owner name and address code	37-50	An optional installation-supplied user identifier.
Reserved	51-70	Reserved

Volume surface indicator	71	Indicates the number of recording surfaces on the diskette and the recording density.
		Blank = 1 surface, single-density.
		2 = 2 surfaces, single-density.
		M = 2 surfaces, double-density.

Extent arrangement indicator	72	Blank indicates that are no special constraints on the arrangement of extents, data set file labels, or unallocated space on this diskette. P indicates that the extents must begin at cylinder 1, head 0, sector 1. It also indicates that the data set file labels must begin at cylinder 0, head 0, sector 8; must be in the same sequence as the extents they describe; and all unallocated space must follow the last data extent on the volume.
Reserved	73-74	Reserved
Physical sector length	75	Indicates the physical sector length. Blank = 128 bytes 1 = 256 bytes 2 = 512 bytes

continued

Table C-11. Diskette Data Set VOL1 Label Description (cont.)

Field	Byte Position	Description
Physical sector	76-77	Blank or 1 indicates that the sectors are physically sequential. Otherwise, this field is used as an increment to determine the next physical sector.
Reserved	78	Reserved
Label standard version	79	U or W indicates standard labels are on this diskette.
	80-127	If volume surface indicator field contains a blank or 2, this field is padded with zeros. If volume surface indicator field contains M, this field is padded with spaces.

BYTES	0	1	2	3
0	label ID			
4	reserved	file identifier		
≈				≈
	not used			
20	block length			record attribute
24	beginning of extent (BOE)			
28		physical record length		
32	end of extent (EOE)			record/block format
36	bypass indicator	file security	write protect	exchange type indicator
40	multivolume indicator	volume sequence number		
44	file creation data			
48		record length		
52		not used		
56			reserved	
60	reserved			
64	file expiration date			
68	verify/copy indicator	file organization		
72	end of data (EOD)			
76				reserved
80				
≈				≈
124				

Figure C-14. Diskette Data Set File Label

Table C-12. Diskette Data Set File Label Description

Field	Byte Position	Description
Label ID	0-3	Contains 4-byte label identifier HDR followed by the number 1
Reserved	4	Reserved
File identifier	5-21	Names user files and is from 1 to 17 characters. First character must be alphabetic and no blanks are allowed. Duplicate names on the same diskette are not allowed.
Block length	22-26	Indicates the record size as follows: Character Meaning △△△△ Record size will be obtained from the RIB. 1-512 Actual record size; for example, 80.
Record attribute	27	Blank; indicates unblocked-unspanned records. B indicates blocked-unspanned records. R indicates blocked-spanned records.
Beginning of extent (BOE)	28-32	Identifies the address of the first sector of the file by cylinder number (pos. 28-29), head (pos. 30), and sector number (pos. 31-32).
Physical record length	33	Indicates physical record length in bytes. Blank indicates 128, 1 indicates 256, and 2 indicates 512.
End of extend (EOE)	34-38	Indicates address of the last sector for this file and uses the same format as BOE.
Record/block format	39	This field must be blank.
Bypass indicator	40	Indicates a file to be skipped during exchange or copy operations when transmitting or transferring files on the volume. If position 40 is blank, the file is transferred; if B, the file is not transferred.
File security	41	Blank indicates the file can be accessed. Nonblank indicates restricted access. When nonblank, the volume accessibility indicator in the volume label (track 00, sector 07) must also be nonblank.

continued

Table C-12. Diskette Data Set File Label Description (cont.)

Field	Byte Position	Description
Write protect	42	Blank indicates both reading and writing allowed. P indicates only read activities allowed.
Exchange type indicator	43	Blank indicates file can be used for basic exchange. Nonblank indicates additional label checking is needed to exchange the file.
Multivolume indicator	44	Character Meaning blank File on 1 diskette C File continued on next diskette L Last diskette on which file resides
Volume sequence number	45-46	Indicates volume sequence number in multivolume set (01-99). Blanks indicate no volume sequence checking performed on this device.
File creation date	47-52	Indicates the creation date of the file by year (yy), month (mm), and day (dd). Blanks indicates nonsignificance of creation date.
Record length	53-56	Defines record length. Character Meaning ΔΔΔΔ = Record length equals block length (position 22)
Offset to next record space	57-61	Not used
Reserved	62-65	Reserved
File expiration date	66-71	Contains date of deletion for a file. (Format is the same as creation date in position 47-52). Character Meaning ΔΔΔΔΔΔ = File date expired 999999 = File date never expires

continued

Table C-12. Diskette Data Set File Label Description (cont.)

Field	Byte Position	Description
Verify/copy indicator	72	Character Meaning blank File is created V File is verified C File data successfully transferred to another medium
File organization	73	Character Meaning Δ or S Sequential file D Nonsequential file If position 43 contains an H, this field must contain a blank.
End of data (EOD)	74-78	Identifies address of next unused sector within the file extent using BOE format. - If EOD equals BOE, the extent contains a null file. - If EOD equals address of next block beyond file extent (unblocked records), entire extent was used. - If blocked records are used, EOD must be used with offset to next record space (position 57-61) to find the actual EOD.
Reserved	79	Reserved
	80-127	Padded with binary zeros

Appendix D
Magnetic Tape Labels

D.1. General

This appendix describes the system standard labels for magnetic tape files and volumes (reels) that are supported by consolidated data management. Magnetic tapes may be labeled or unlabeled, and a labeled tape may contain either nonstandard or system standard labels.

D.2. System Standard Tape Labels

All standard tape labels, including optional UHL and UTL, are in blocks of 80 bytes. There are five tape label groups: three are required, two are optional.

- Volume label group
- File header label group
- User header label group (optional)
- File trailer label group
- User trailer label group (optional)

D.2.1. Volume Label Group

The volume label group comprises a single volume label, VOL1. The VOL1 label identifies the tape reel and its owner, and it is used to check that the proper reel is mounted. When a tape is first used at an installation, its volume serial number (VSN) and other volume information (shown in Figure D-1) are usually recorded on it magnetically with the TPREP utility routine. The volume serial number should also be written on the exterior of the reel for visual identification to the operator.

As an alternative to using TPREP, if you want to prep the volumes of a standard labeled file dynamically, as a preliminary part of the job step in which you create the file, you specify the parameter (PREP) in the VOL job control statement in the device assignment set for the file, also specifying a unique VSN. Data management then preps the volumes from information you supply on the associated VOL and LBL statements.

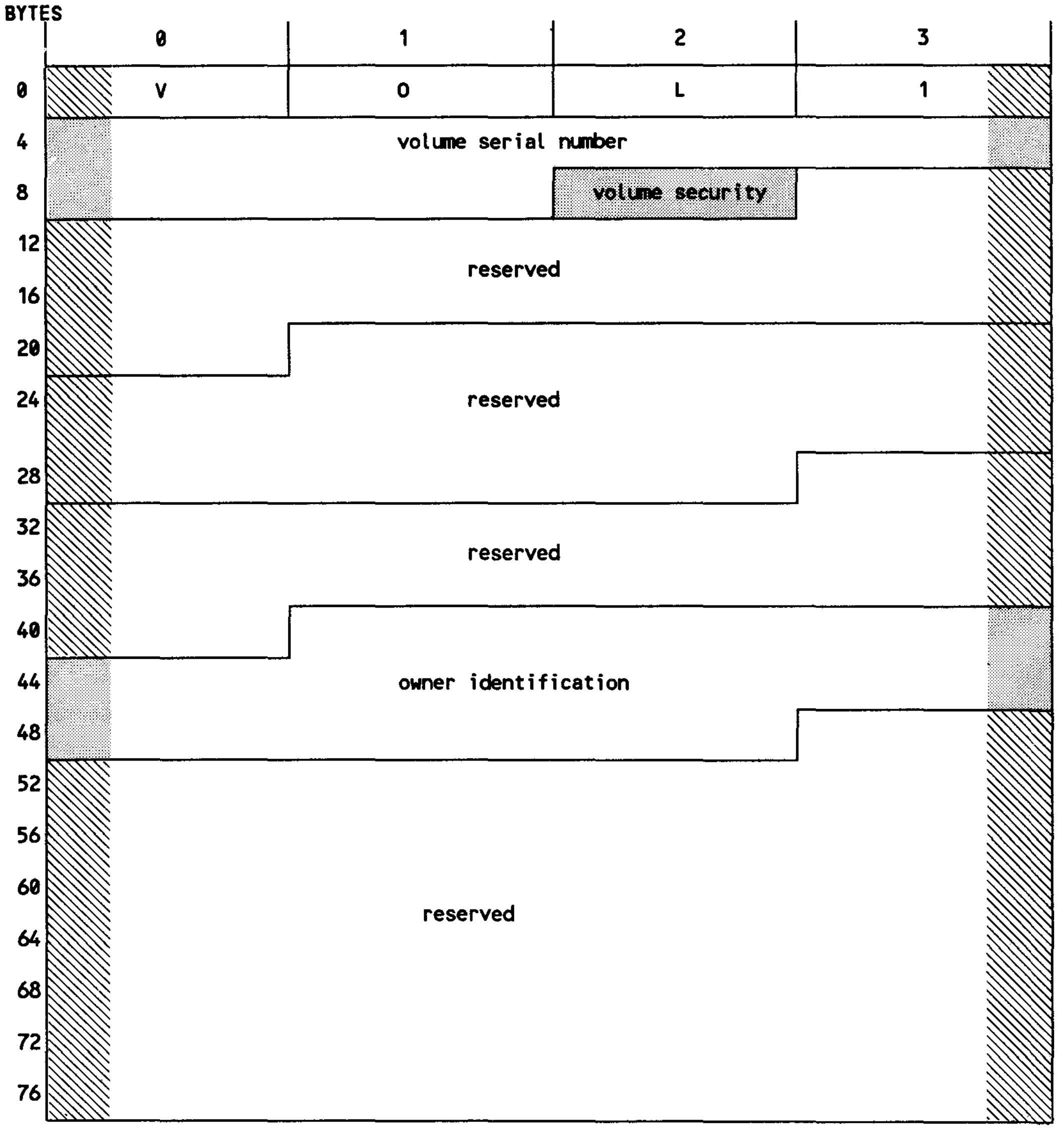
When you open an output tape file, its open-and-rewind options are executed first and then the tape is checked to see if it is at the load point. If it is at the load point, data management reads the VOL1 label (if it is not in the prep mode) and, checking the VSN, saves this for use in writing or reading the file header labels (HDR1 and HDR2). It then positions your tape so that the volume labels are not destroyed, and no further volume processing is performed.

If the output tape is not at the load point after the open-and-rewind options are executed, it is assumed that it is positioned between the two ending tape marks of the previous file, or just prior to the HDR1 label of an existing file. In either case, no volume label checking or creation is performed.

When an input tape is opened, the open and rewind options are executed first and then the tape is checked to see whether it is at the load point. If it is, the VOL1 label is read and the VSN is used to check the file serial number in the appropriate file header or trailer label. The tape is then positioned to the proper file header or trailer label, as specified in the file sequence number field of the associated LBL job control statement, and no further volume label processing is performed. If the input tape is not at the load point after your open-and-rewind options are executed, it is assumed that the tape is positioned between the two ending tape marks of a previously created file or just prior to the HDR1 label of an existing file. In either case, no further volume label processing is performed.

If you have specified that an input tape is to be read backwards and you encounter any volume label when you are closing the file, that label is bypassed.

Figure D-1 shows the format of the VOL1 label and its fields are described in Table D-1.



LEGEND:

▨ Generated by data management or reserved for system expansion.

░ Written by data management from user-supplied data.

Figure D-1. Tape Volume (VOL1) Label Format for an EBCDIC Volume

Table D-1. Tape Volume (VOL1) Label Format, Field Description for an EBCDIC Volume

Field	Initialized	Bytes	Code	Description
Label identifier	Tape prep	0-2	EBCDIC	Contains VOL to indicate that this is a volume label
Label number	Tape prep	3	EBCDIC	Always 1 for the initial volume label
Volume serial number	Tape prep	4-9	EBCDIC	Unique identification number assigned to this volume by your installation. Data management expects 1 to 6 alphanumeric characters, the first of which is alphabetic.
Volume security	Data management	10	EBCDIC	Reserved for future by installations requiring security at the reel level. Currently blank.
Reserved	-	11-20	EBCDIC	Contains blanks (40_{16})
Reserved	-	21-30	EBCDIC	Contains blanks (40_{16})
Reserved	-	31-40	EBCDIC	Contains blanks (40_{16})
Owner	Tape prep	41-50	EBCDIC	Unique identification of the owner of the reel; any combination of alphanumerics
Reserved	-	51-79	EBCDIC	Contains blanks (40_{16})

D.2.2. File Header Label Group

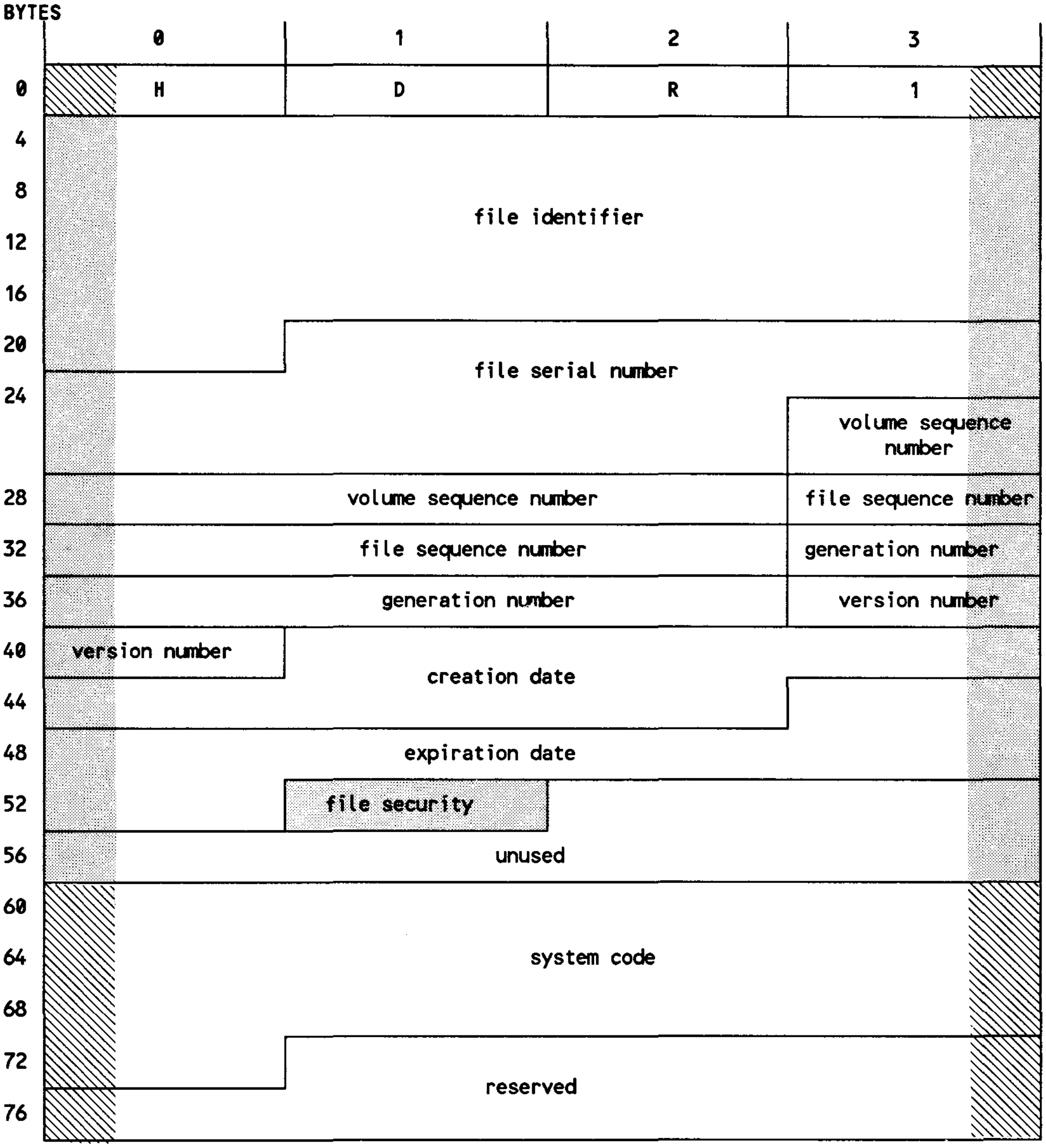
The file header label group comprises two labels, HDR1 and HDR2, described in the following subsections.

First File Header Label (HDR1)

The first file header label (HDR1), which identifies the file, is written at the beginning of each file. The HDR1 label is required and has the fixed format shown in Figure D-2; its fields are described in Table D-2. All fields in the HDR1 label may be specified in the job control stream.

For input tapes, all fields up to and including the system code field in the HDR1 label are checked at file open against the values you specify in the LBL job control statement, unless you have specified read-backward processing. If you have specified read-backward processing, the HDR1 label is bypassed. For multifile input volumes, you should specify the file sequence number in the LBL job control statement to ensure proper tape positioning.

For output files, the tape must be positioned properly before you may open the files. On file open, the expiration date in the HDR1 label is checked against the current or actual calendar date to determine whether the associated file has expired. If it has not, data management issues an error message, and it is not possible to write to the file. You should mount the correct volume and rerun.



LEGEND:

▨ Generated by data management or reserved for system expansion.

▒ Written by data management from user-supplied data.

Figure D-2. First File Header (HDR1) Format for an EBCDIC Tape Volume

Table D-2. First File Header Label (HDR1), Field Description

Field	Bytes	Description
Label identifier	0-2	Contains HDR to indicate a file header label
Label number	3	Always 1
File identifier	4-20	A 17-byte configuration that uniquely identifies the file. It may contain embedded blanks and is left-justified in the field if fewer than 17 bytes are specified.
File serial number	21-26	The same as the VSN of the VOL1 label for the first reel of a file or a group of multifile reels.
Volume sequence number	27-30	The position of the current reel with respect to the first reel on which the file begins.
File sequence number	31-34	The position of this file with respect to the first file in the group.
Generation number	35-38	The generation number of the file (0000-9999).
Version number of generation	39-40	The version number of a particular generation of a file
Creation date	41-46	The date the file was created, expressed in the form yyddd and right-justified. The leftmost position is blank.
Expiration date	47-52	The date the file may be written over or used as scratch, in the same form as the creation date.
File security indicator	53	Reserved for file security indicator. Indicates whether additional qualifications must be met before a user program may have access to the file. 0 = No additional qualifications are required. 1 = Additional qualifications are required.
Unused	54-59	Unused field, containing EBCDIC zeros.
System code	60-72	Reserved for system code, the unique identification of the operating system that produced the file
Reserved	73-79	Reserved field, containing blanks (40_{16})

Second File Header Label (HDR2)

The HDR2 label acts as an extension of the HDR1 label and is required in standard labeled files. Figure D-3 shows the format of the HDR2 label and Table D-3 describes its fields.

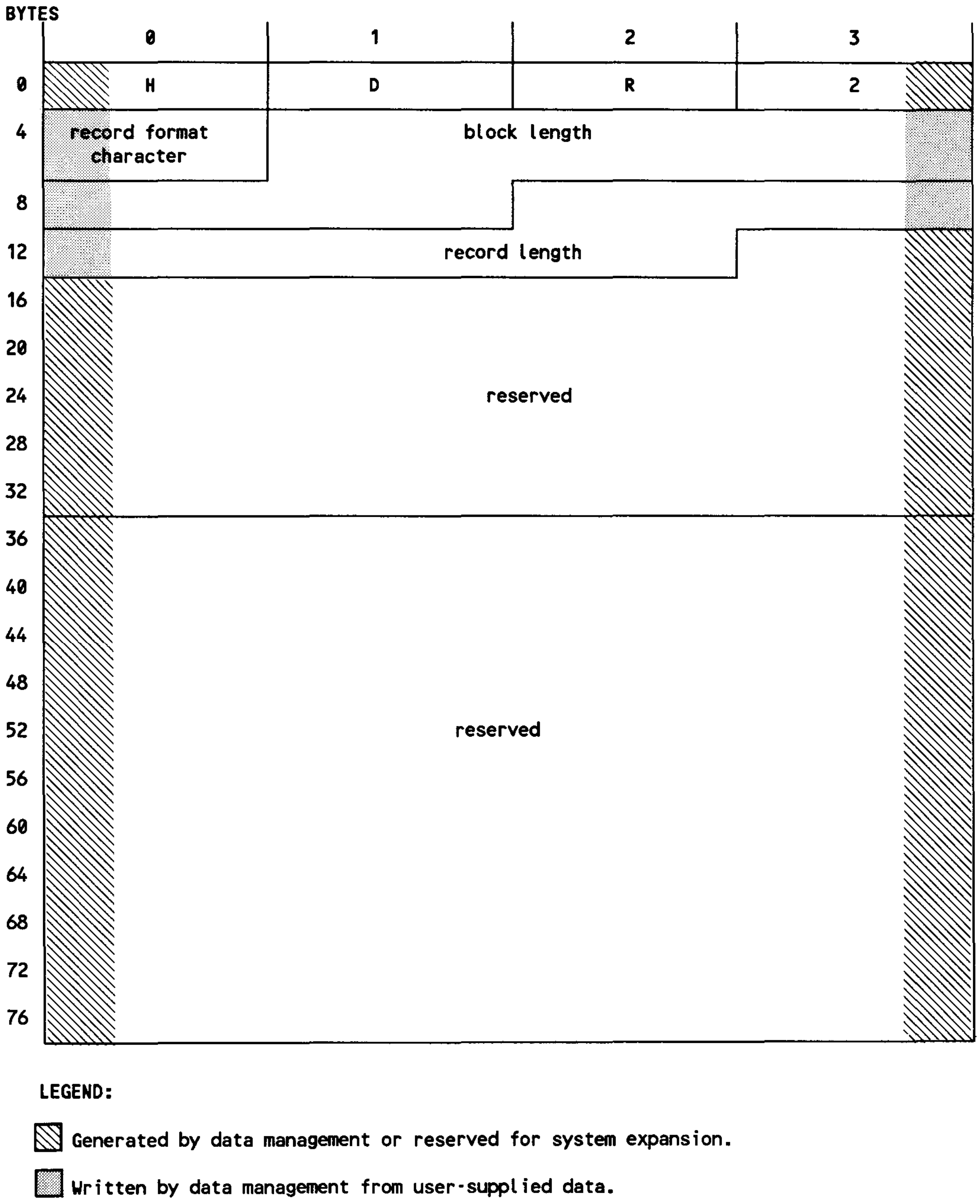


Figure D-3. Second File Header Label (HDR2) Format for an EBCDIC Tape Volume

Table D-3. Second File Header Label (HDR2), Field Description

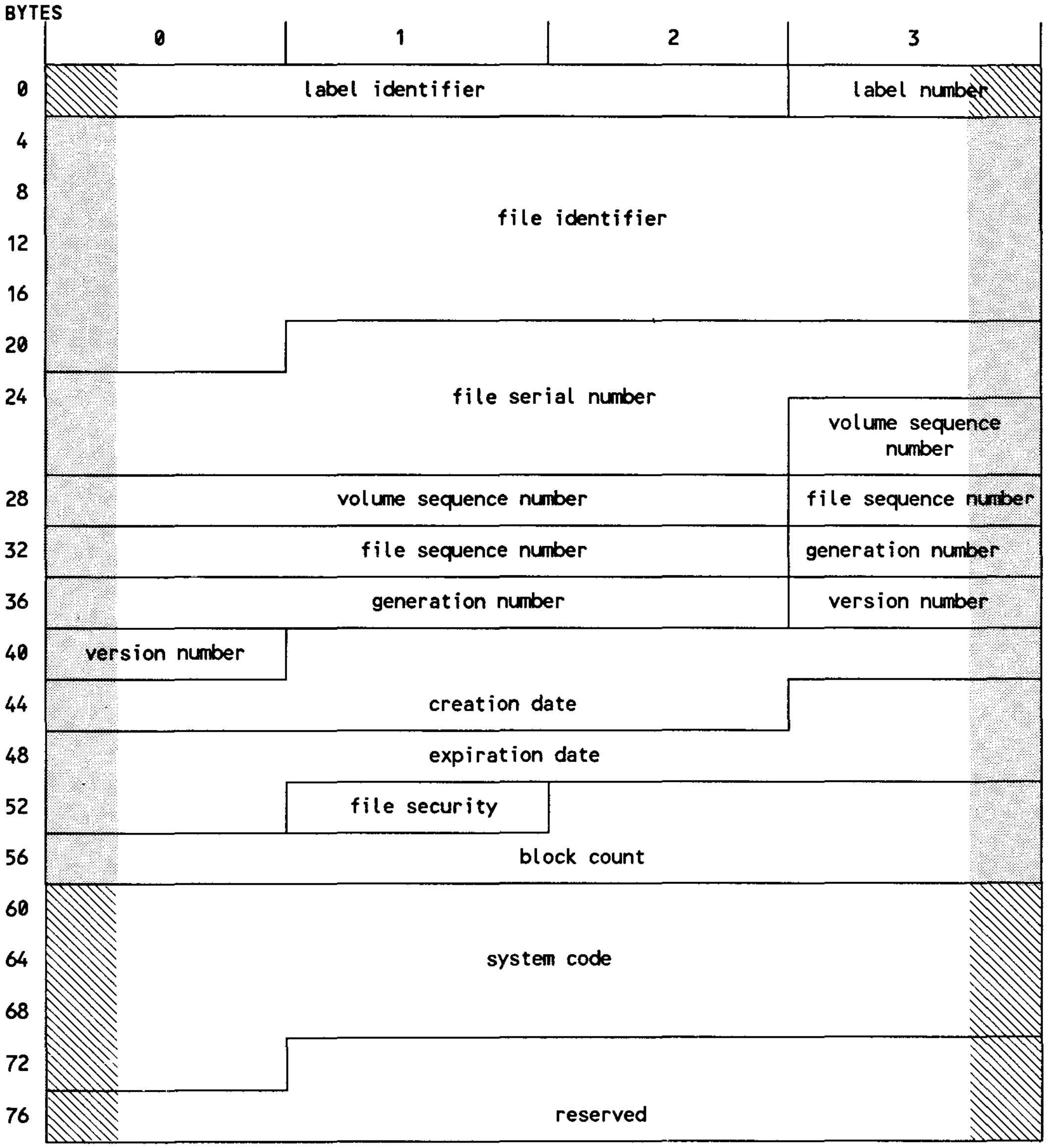
Field	Bytes	Description
Label identifier	0-2	Contains HDR to indicate a file header label
Label number	3	Always 2
Record format character	4	Character Meaning D Variable-length, with length fields specified in decimal F Fixed-length U Undefined V Variable-length, with length fields specified in binary
Block length	5-9	Five EBCDIC characters specifying the maximum number of characters per block.
Record length	10-14	Five EBCDIC characters specifying the record length for fixed-length records. For any other record format, this field contains zeros.
Reserved	15-35	Reserved for future system use.
Reserved	36-79	Reserved for future system use.

D.2.3. File Trailer Label Group

The file trailer label group comprises either of two pairs of labels, depending on whether the reel contains an end-of-file or an end-of-volume condition. In the first condition, the first label of the pair is the EOF1 label, in a format identical to the HDR1 label; the second label is the EOF2 label. Its format is identical to the HDR2 label. In the end-of-volume condition, these labels are the EOV1 and EOV2 labels; again, the formats of these labels are identical to their counterparts in the file header label group, HDR1 and HDR2.

When you open an input file and read-backward is specified, the fields in the EOF1 or EOV1 label are checked against the values you have specified in the LBL job control statement. This processing is similar to that of the HDR1 label.

Figure D-4 shows the format of the EOF1 or EOV1 label and Table D-4 summarizes the contents of its fields. Similarly, Figure D-5 and Table D-5 describe the EOF2 or EOV2 label.



LEGEND:

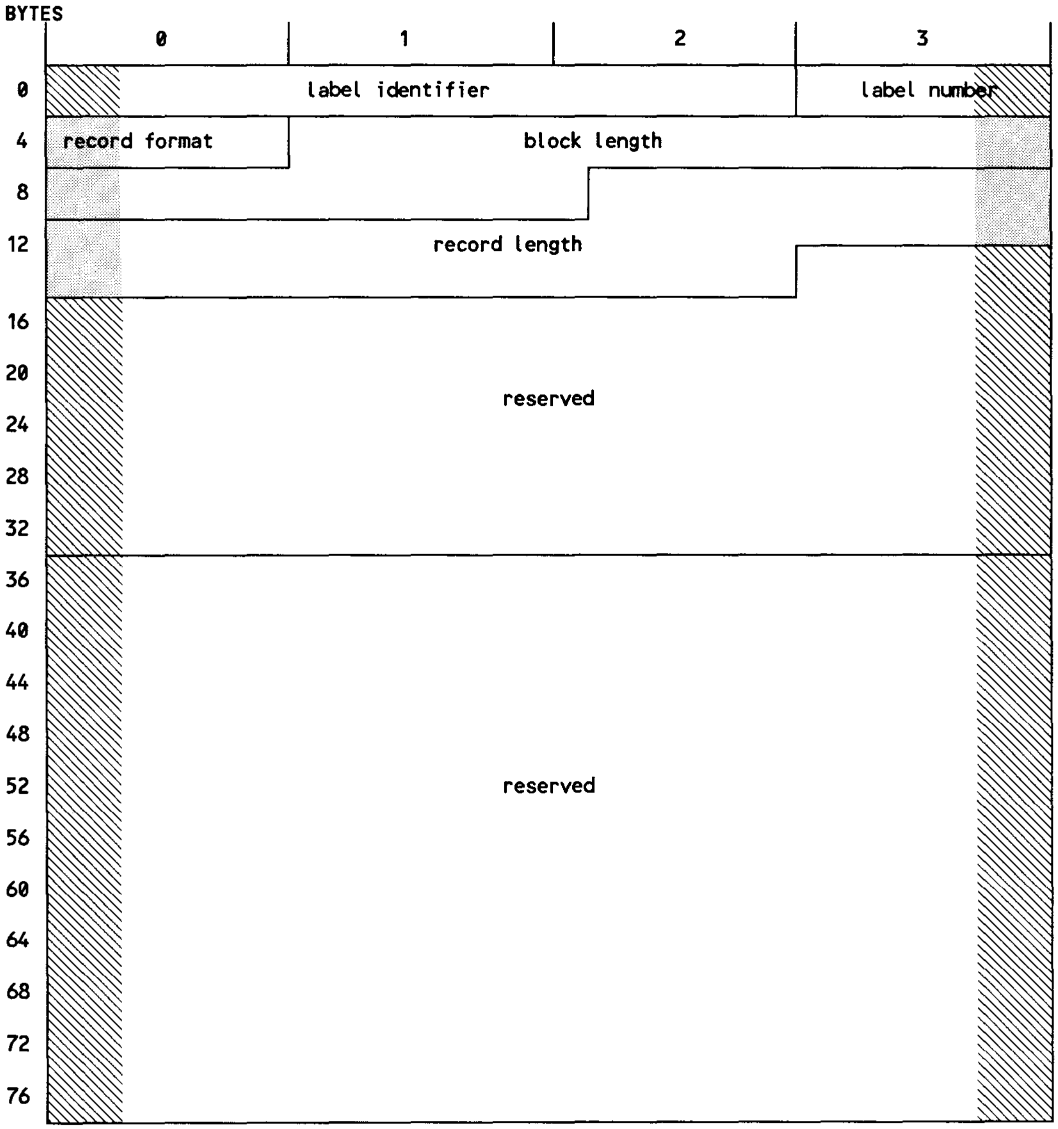
▨ Generated by data management or reserved for system expansion

░ Written by data management from user-supplied data

Figure D-4. Tape File EOF1 and EOV1 Label Formats

Table D-4. Tape File EOF1 and EOV1 Labels, Field Description

Field	Bytes	Description
Label identifier	0-2	Indicates that this is a file trailer label; contains EOF for an end-of-file label or EOV for an end-of-volume label.
Label number	3	Always 1
File identifier	4-20	A 17-byte configuration that uniquely identifies the file. It may contain embedded blanks and is left-justified in the field if fewer than 17 bytes are specified.
File serial	21-26	The same as the VSN of the VOL1 label for the first reel of a file or a group of multifile reels
Volume sequence number	27-30	The position of the current reel with respect to the first reel on which the file begins
File sequence number	31-34	The position of this file with respect to the first file in the group
Generation number	35-38	The generation number of the file (0000-9999)
Version number of generation	39-40	The version number of a particular generation of a file
Creation date	41-46	The date the file was created, expressed in the form yyddd and right-justified. The leftmost position is blank.
Expiration date	47-52	The date the file may be written over or used as scratch, in the same form as the creation date
File security indicator	53	Reserved for file security indicator. Indicates whether additional qualifications must be met before a user program may have access to the file. 0 = No additional qualifications are required. 1 = Additional qualifications are required.
Block count	54-59	In the first file trailer label, indicates the number of data blocks either in this file of a multifile reel, or on the current reel of a multivolume file. Data management checks the block count for input files or writes the count for output files.
System code	60-72	Reserved for system code, the unique identification of the operating system that produced the file
Reserved	73-79	Reserved field, containing blanks (40_{16})



LEGEND:

▨ Generated by data management or reserved for system expansion.

░ Written by data management from user-supplied data.

Figure D-5. Tape File EOF2 and EOV2 Label Formats

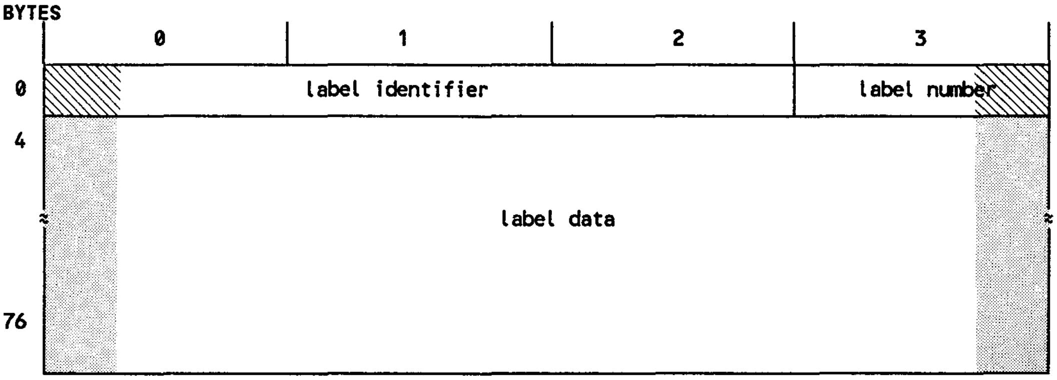
Table D-5. Tape File EOF2 and EOV2 Labels, Field Description

Field	Bytes	Description
Label identifier	0-2	Indicates that this is a file trailer label; contains EOF for an end-of-file label, or EOV for an end-of-volume label
Label number	3	Always 2
Record format character	4	Character Meaning D Variable-length, with length fields specified in decimal F Fixed-length U Undefined V Variable-length, with length fields specified in binary
Block length	5-9	Five EBCDIC characters specifying the maximum number of characters per block
Record length	10-14	Five EBCDIC characters specifying the record length for fixed-length records. For any other record format, this field contains zeros.
Reserved	15-35	Reserved for future system use
Reserved	36-79	Reserved for future system use

D.2.4. Standard User Header and Trailer Labels

The use of the standard UHL and UTL in a standard labeled file is entirely optional. You may have one or as many as eight UHLs, or none at all. The same applies to the UTL. If they are used, they must be in the 80-byte format shown in Figure D-6 and the fields must have the contents shown in Table D-6.

If you have specified block numbering, the optional user labels must be 83 bytes long for EBCDIC files and 81 bytes long for ASCII input files to ensure correct positioning.



LEGEND:

▒ These bytes are written by the user's label routine.

▨ These bytes are written by data management. (See note to Table D-6.)

Figure D-6. Optional User Header and Trailer Label Format, ASCII and Standard Labeled EBCDIC Tape Files

Table D-6. Optional User Header and Trailer Labels, Field Description for Standard Labeled Tape Files

Field	Bytes	Description
Label identifier	0-2	Contains UHL for user header label; UTL for user trailer label
Label number	3	Ranges from 1 through 8. (See note.)
Label data	4-79	Contains 76 bytes of user-specified information

NOTE:

For ASCII files, the label number is not written by data management; it is the user's responsibility. Also there is no limit on the range; the user may have any number of user labels he wants.

D.3. ASCII Standard Magnetic Tape Labels

The figures and tables that follow describe the labels written (and expected to be read) by consolidated data management for ASCII files. Note the very small differences between the foregoing EBCDIC labels and these ASCII labels, which conform to American National Standard Magnetic Tape Labels for Information Interchange, X3.27-1969.

Consolidated data management writes and processes the following ASCII standard labels: VOL1, HDR1, HDR2, EOV1, EOV2, EOF1, and EOF2. Although data management also provides for input and output processing of optional UHLs and UTLs (their forms are identical in ASCII to those described in D.2.4), it does not provide for processing optional user volume labels (UVLs) on output. If present on ASCII input tapes, UVLs are accepted, but bypassed and ignored.

D.3.1. ASCII Character Code and Processing

During input and output processing of ASCII magnetic tape files, data management uses the character code specified by American National Standard Code for Information Interchange, X3.4-1977, performing appropriate translations (EBCDIC to ASCII, or vice versa) so that your program always processes in EBCDIC.

Output Processing of Labels in ASCII Magnetic Tape Files

When you specify an ASCII output magnetic tape file, data management writes out all system labels in ASCII. Just as you must present your data to data management in EBCDIC, so also must you present your optional UHL and UTL label data in EBCDIC. Data management translates these into ASCII before writing them out to tape. It is your responsiblity to write out the label number.

Input Processing of Labels in ASCII Magnetic Tape Files

When reading input magnetic tape files coded in ASCII, data management assumes that these comply fully with American National Standard X3.27-1969, and that there is no mixture of character codes. Any exception may result in incorrect processing. Before passing your data and your optional UHLs or UTLs to you, data management translates these into the form it expects to receive before output: your program receives data and labels in EBCDIC.

D.3.2. OS/3 Processing of Certain Fields in ASCII Tape Labels

The format and content of ASCII magnetic tape labels are depicted in Figures D-7 through D-11 and Tables D-7 through D-11. The following subsections describe the way OS/3 processes certain of the label fields; further notes appear in the tables.

Accessibility Field

The accessibility field occurs in the VOL1, HDR1, EOF1, and EOV1 labels. During read-forward input, a *space* (2/0) encountered in this field of both the VOL1 and HDR1 labels allows processing to continue, whereas a nonspace in either field causes the issuance of an error message to the operator's console; processing may not continue unless the operator responds with the override option in effect at your installation. (For read-backward input, the procedure is identical, the fields interrogated being those in the EOV1 and EOF1 labels.)

There is no option to write a nonspace character in the accessibility field of any system label on output ASCII tapes; data management always creates this field as space.

Label Standard Level Field

The label standard level field occurs only in the VOL1 label. When writing output ASCII tapes, data management writes 1 in this field to indicate that the tape is in full compliance with American National Standard X3.27-1969. On input tapes, a 1 in this field ensures correct processing; tapes created under later versions of the standard may also be accepted, but you cannot be assured of correct processing.

Expiration Date Field

The expiration date field occurs in the HDR1, EOF1, and EOV1 labels. If you attempt to write over an unexpired file in an ASCII tape (one whose expiration date is later than today's date), data management issues an error message to the operator console. You are not able to continue processing unless the operator responds with the overrride option in effect at your installation.

In a multiple ASCII volume, you should not give a file an expiration date later than any other file already recorded on that tape; otherwise, the additional protection you intend is not effective. The expiration date of the first file on a volume is the latest date on which any part of the volume is protected from being overwritten.

System Code

In writing output tapes, data management writes OS3, left-justified in the 13-byte system code field. The remainder of the field is filled with spaces. The field is ignored on input.

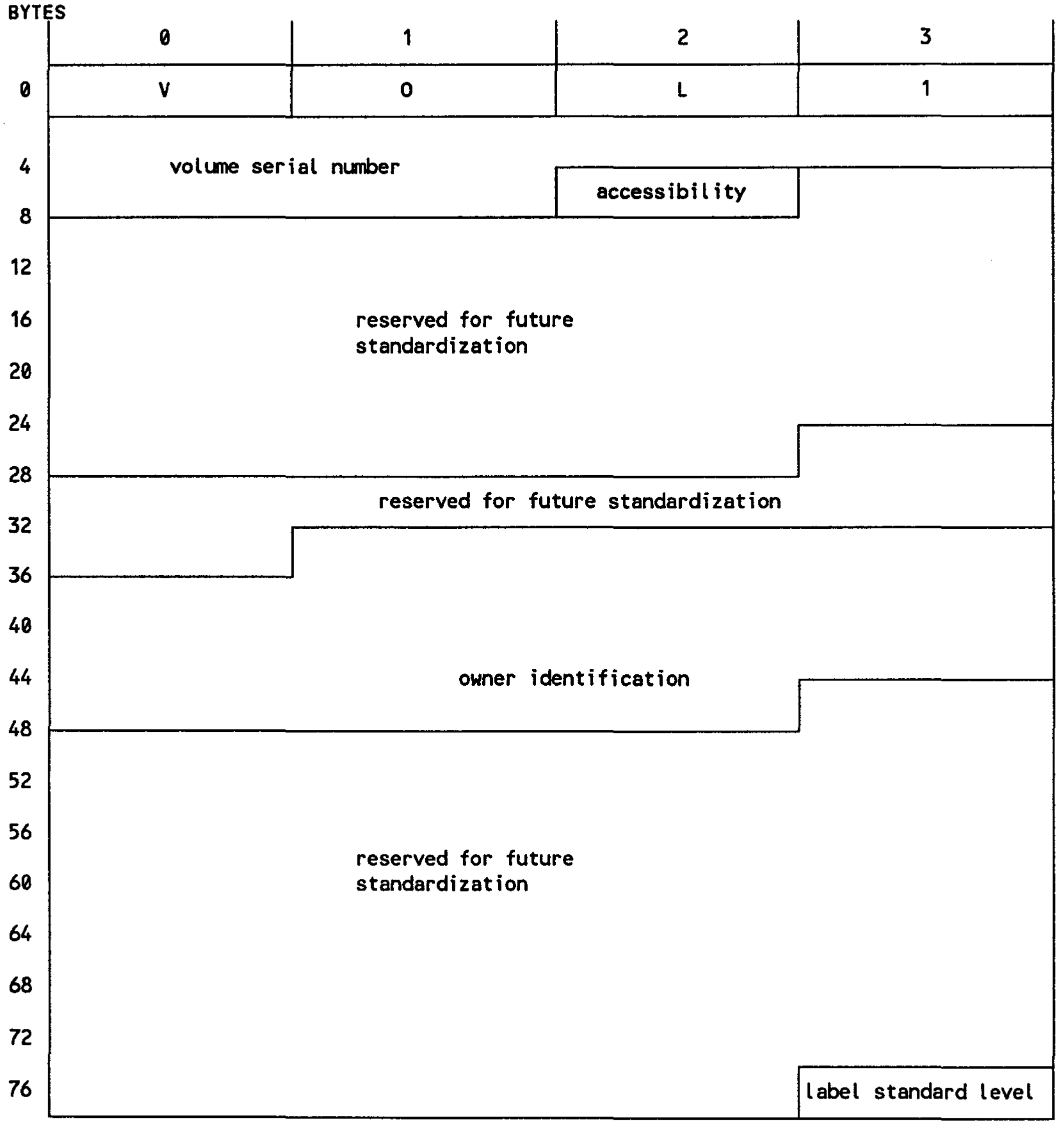


Figure D-7. Volume Header Label (VOL1) for an ASCII Magnetic Tape Volume

Table D-7. Volume Header Label (VOL1), Field Description for an ASCII Tape Volume

Field	Bytes	Description
Label identifier	0-2	Must be "VOL"
Label number	3	Must be "1"
Volume serial number	4-9	Six "a" characters permanently assigned by the owner to identify this physical volume (that is, this reel of magnetic tape). (See Note 1.)
Accessibility	10	An "a" character that indicates any restrictions on who may have access to the information in this volume. A "space" means unlimited access; any other character means special handling, in the manner agreed upon between the interchange parties. (See Notes 1 and 2.)
Reserved	11-30	Reserved for future standardization. Must be "spaces". (See Note 2.)
Reserved	31-36	Reserved for future standardization. Must be "spaces". (See Note 2.)
Owner identification	37-50	Any "a" characters, identifying the owner of the physical volume. (See Note 1.)
Reserved	51-78	Reserved for future standardization. Must be "spaces". (See Note 2.)
Label standard level	79	"1" means that the labels and data formats on this volume conform to the requirements of American National Standard X3.27-1969. "Space" means that the labels and data formats on this volume require the agreement of the interchange parties. (See Note 2.)

NOTES:

1. An "a" character is any of the characters occupying the center four columns of ASCII (American National Standard Code for Information Interchange) except position 5/15 and those positions where there is a provision for alternative graphic representation.

2. "Space" is the normally nonprinting graphic character occupying position 2/0 of ASCII.

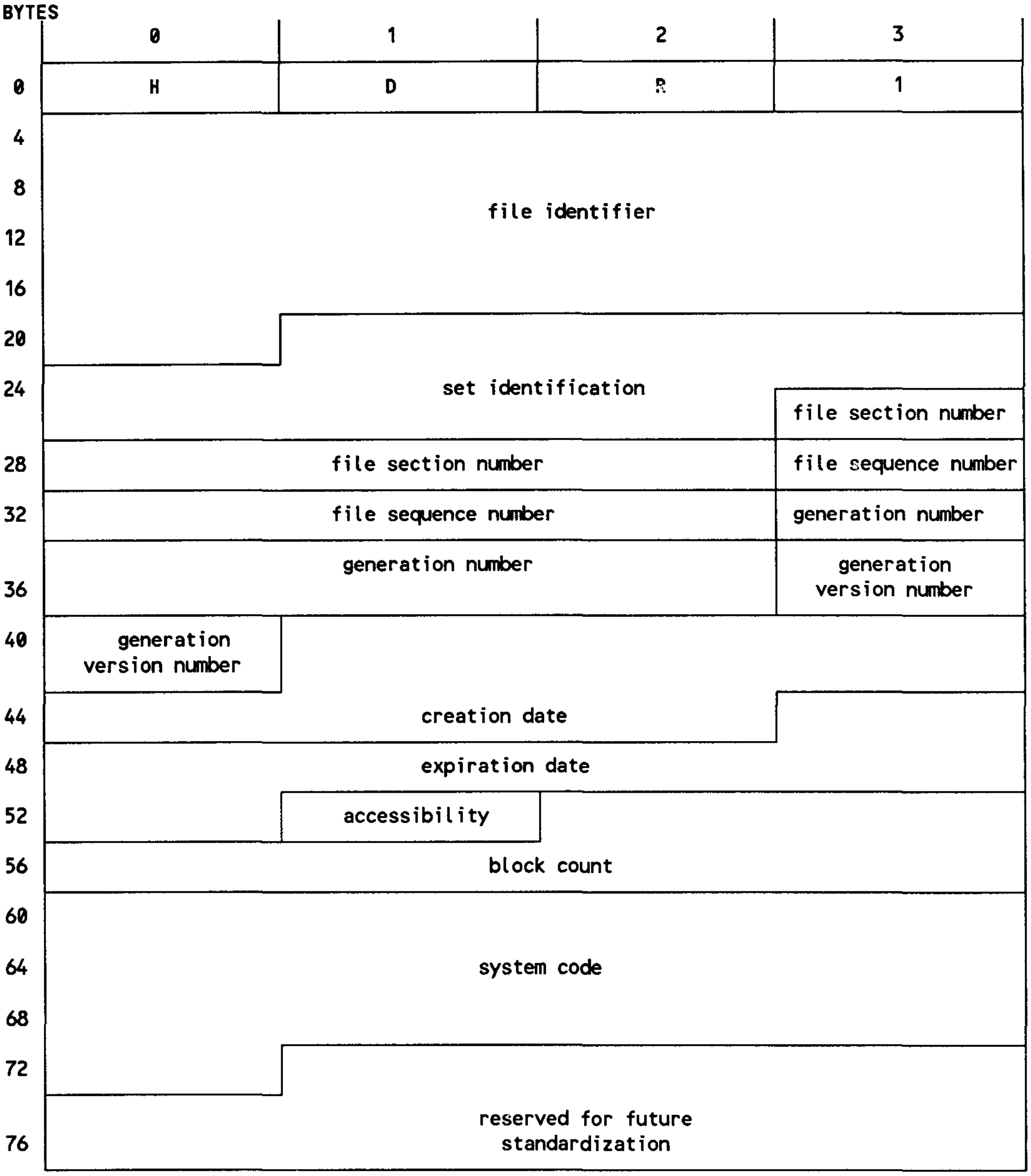


Figure D-8. First File Header Label (HDR1) for an ASCII Tape Volume

Table D-8. First File Header Label (HDR1), Field Description for an ASCII Tape Volume

Field	Bytes	Description
Label identifier	0-2	Must be "HDR"
Label number	3	Must be "1"
File identifier	4-20	Any "a" characters agreed on between originator and recipient. (See Note 1.)
Set identification	21-26	Any "a" characters to identify the set of files of which this is one. This identification must be the same for all files of a multifile set. (See Note 1.)
File section number	27-30	The file section number of the first header label of each file is "0001". This applies both to the first or only file on a volume and to subsequent files on a multifile volume. This field is incremented by one on each subsequent volume of the file.
File sequence number	31-34	Four "n" characters denoting the sequence (that is, 0001, 0002, etc.) of files within the volume or set of volumes. In all the labels for a given file, this field contains the same number. (See Note 3.)
Generation number (optional)	35-38	Four "n" characters denoting the current stage in the succession of one file generation by the next. When a file is first created, its generation number is 0001. (See Notes 3 and 4.)
Generation version number (optional)	39-40	Two "n" characters distinguishing successive iterations of the same generation. The generation version number of the first attempt to produce a file is 00. (See Notes 3 and 4.)
Creation date	41-46	A "space" followed by two "n" characters for the year, followed by three "n" characters for the day (001 to 366) within the year. (See Notes 2 and 3.)
Expiration date	47-52	Same format as creation date field. This file is regarded as "expired" when today's date is equal to, or later than, the date given in this field. When this condition is satisfied, the remainder of this volume may be overwritten. To be effective on multifile volumes, therefore, the expiration date of a file must be less than, or equal to, the expiration date of all previous files on the volume.
Accessibility	53	An "a" character that indicates any restrictions on who may have access to the information in this file. A "space" means unlimited access; any other character means special handling, in a manner agreed upon between the interchange parties. (See Note 1.)

continued

Table D-8. First File Header Label (HDR1), Field Description for an ASCII Tape Volume (cont.)

Field	Bytes	Description
Block count	54-59	Must be "zeros"
System code (optional)	60-72	Thirteen "a" characters identifying the operating system that recorded this file. (See Note 1.)
Reserved	73-79	Reserved for future standardization; must contain "spaces".

NOTES:

1. An "a" character is any of the characters occupying the center four columns of ASCII (depicted in American National Standard X3.4-1977), except for position 5/15 and those positions where there is provision for alternative graphic representation.

2. A "space" is the normally nonprinting graphic character occupying position 2/0 in ASCII.

3. An "n" character is any ASCII numeric digit, from 0 through 9.

4. "Optional", when used to describe a field in these ASCII labels, means that the field may, but need not, contain the information described. If an optional field does not contain the designated information, it must contain "spaces".

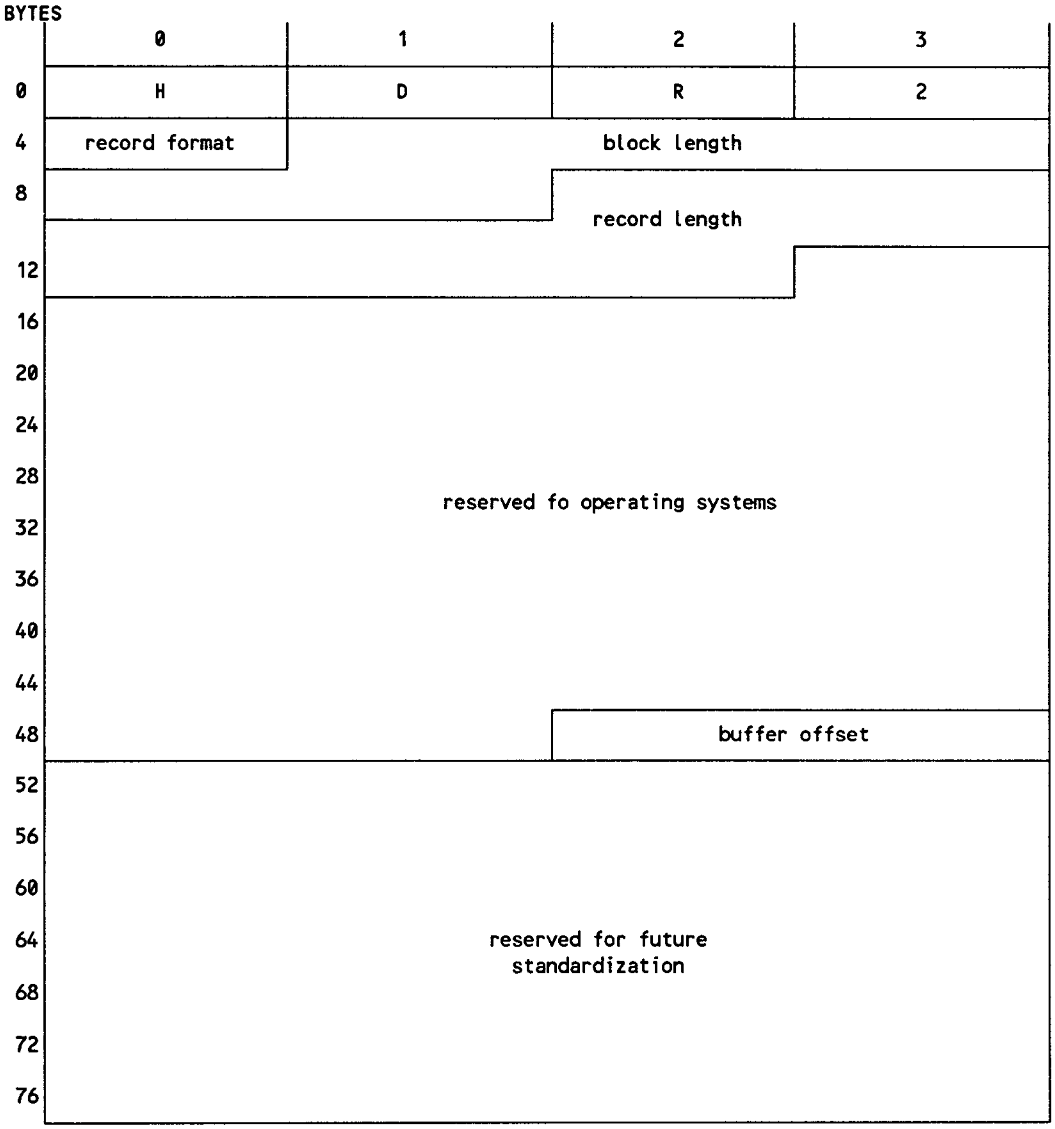


Figure D-9. Second File Header Label (HDR2) for an ASCII Tape Volume

Table D-9. Second File Header Label (HDR2), Field Description for an ASCII Tape Volume

Field	Bytes	Description
Label identifier	0-2	Must be "HDR"
Label number	3	Must be "2"
Record format	4	Character Meaning F Fixed length D Variable length, with the number of characters in the record specified in decimal V Variable length, with the number of characters specified in binary. (See Note.) U Undefined
Block length	5-9	Five "n" characters specifying the maximum number of characters per block. (See Note 3, Table D-8.)
Record length	10-14	Five "n" characters specifying: if "record format" is F, record length; if D or V, maximum record length including any count fields; if U, then undefined. (See Note.)
Reserved	15-49	Currently "spaces"
Buffer offset (optional)	50-51	Two "n" characters specifying the length in characters of any additional field inserted before a data block - for example, block length. This length is included in the block length. (See Notes 3 and 4, Table D-8.)
Reserved	52-79	Reserved for future standardization; must contain "spaces". (See Note 2, Table D-8.)

NOTE:

Consolidated data management does not support the ASCII "V-format" record.

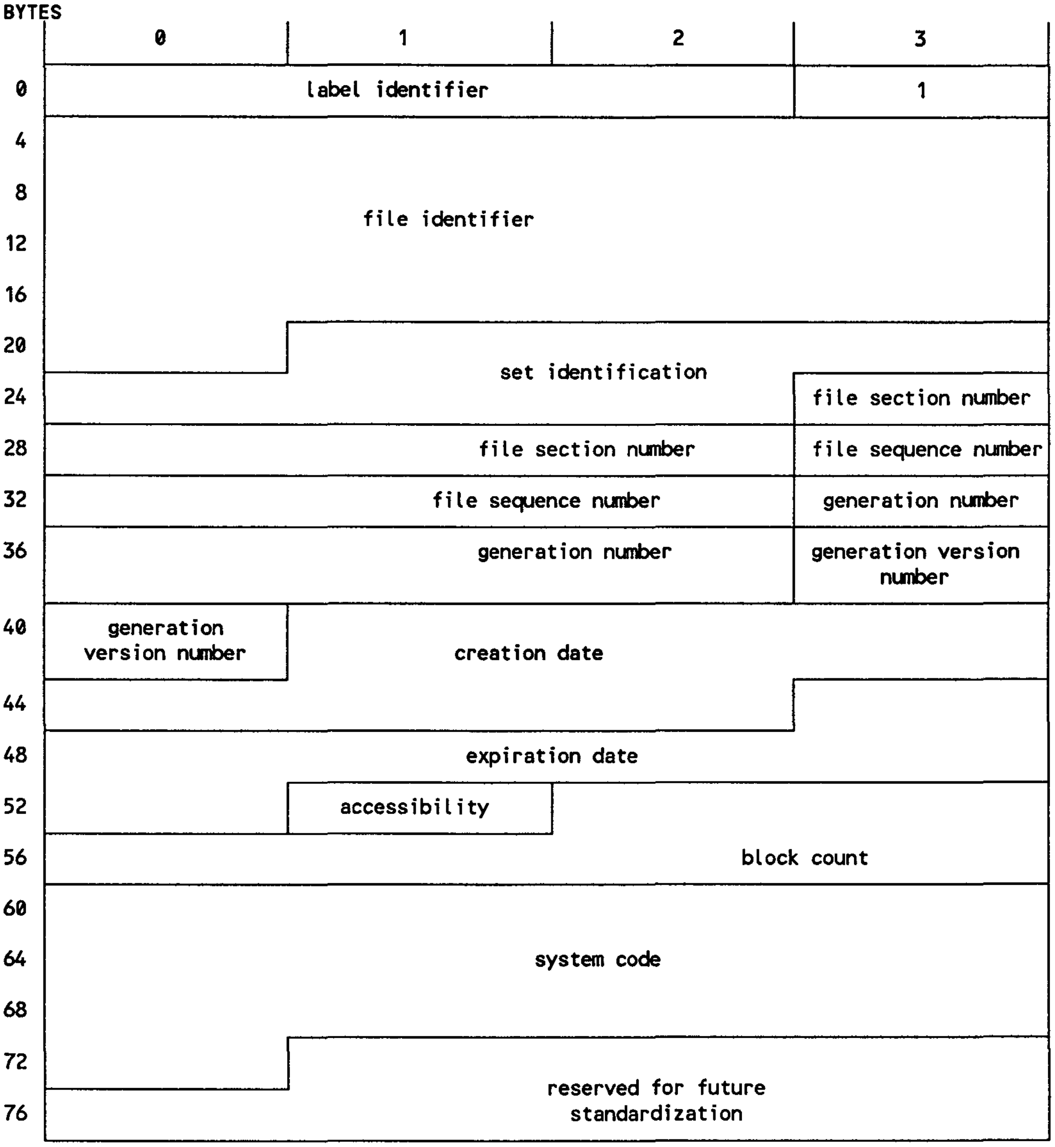


Figure D-10. First End-of-File or End-of-Volume Label (EOF1/EOV1) for an ASCII Tape Volume

Table D-10. First End-of-File or End-of-Volume Label (EOF1/EOV1) Field Description for an ASCII Tape Volume

Field	Bytes	Description
Label identifier	0-2	Contains "EOF" if an end-of-file label; "EOV" for end-of-volume label.
Label number	3	Must be "1"
File identifier	4-20	Any "a" characters agreed on between originator and recipient. (See Note 1, Table D-8.)
Set identification	21-26	Any "a" characters to identify the set of files of which this is one. This identification must be the same for all files of a multifile set. (See Note 1, Table D-8.)
File section number	27-30	The file section number of the first header label of each file is "0001". This applies both to the first or only file on a volume and to subsequent files on a multifile volume. This field is incremented by one on each subsequent volume of the file.
File sequence number	31-34	Four "n" characters denoting the sequence (that is, 0001, 0002, etc.) of files within the volume or set of volumes. In all the labels for a given file, this field will contain the same number. (See Note 3, Table D-8.)
Generation number (optional)	35-38	Four "n" characters denoting the current stage in the succession of one file generation by the next. When a file is first created, its generation number is 0001. (See Notes 3 and 4, Table D-8.)
Generation version number (optional)	39-40	Two "n" characters distinguishing successive iterations of the same generation. The generation version number of the first attempt to produce a file is 00. (See Notes 3 and 4, Table D-8.)
Creation date	41-46	A "space" followed by two "n" characters for the year, followed by three "n" characters for the day (001 to 365) within the year. (See Notes 2 and 3, Table D-8.)
Expiration date	47-52	Same format as creation date field. This file is regarded as "expired" when today's date is equal to, or later than, the date given in this field. When this condition is satisfied, the remainder of this volume may be overwritten. To be effective on multifile volumes, therefore, the expiration date of a file must be less than, or equal to, the expiration date of all previous files on the volume.

continued

Table D-10. First End-of-File or End-of-Volume Label (EOF1/EOV1) Field Description for an ASCII Tape Volume (cont.)

Field	Bytes	Description
Accessibility	53	An "a" character that indicates any restrictions on who may have access to the information in this file. A "space" means unlimited access; any other character means special handling, in a manner agreed upon between the interchange parties. (See Notes 1 and 2, Table D-8.)
Block count	54-59	Number of data blocks in the file or volume.
System code (optional)	60-72	Thirteen "a" characters identifying the operating system that recorded this file. (See Note 1, Table D-8.)
Reserved	73-79	Reserved for future standardization; must be "spaces". (See Notes 3, Table D-8.)

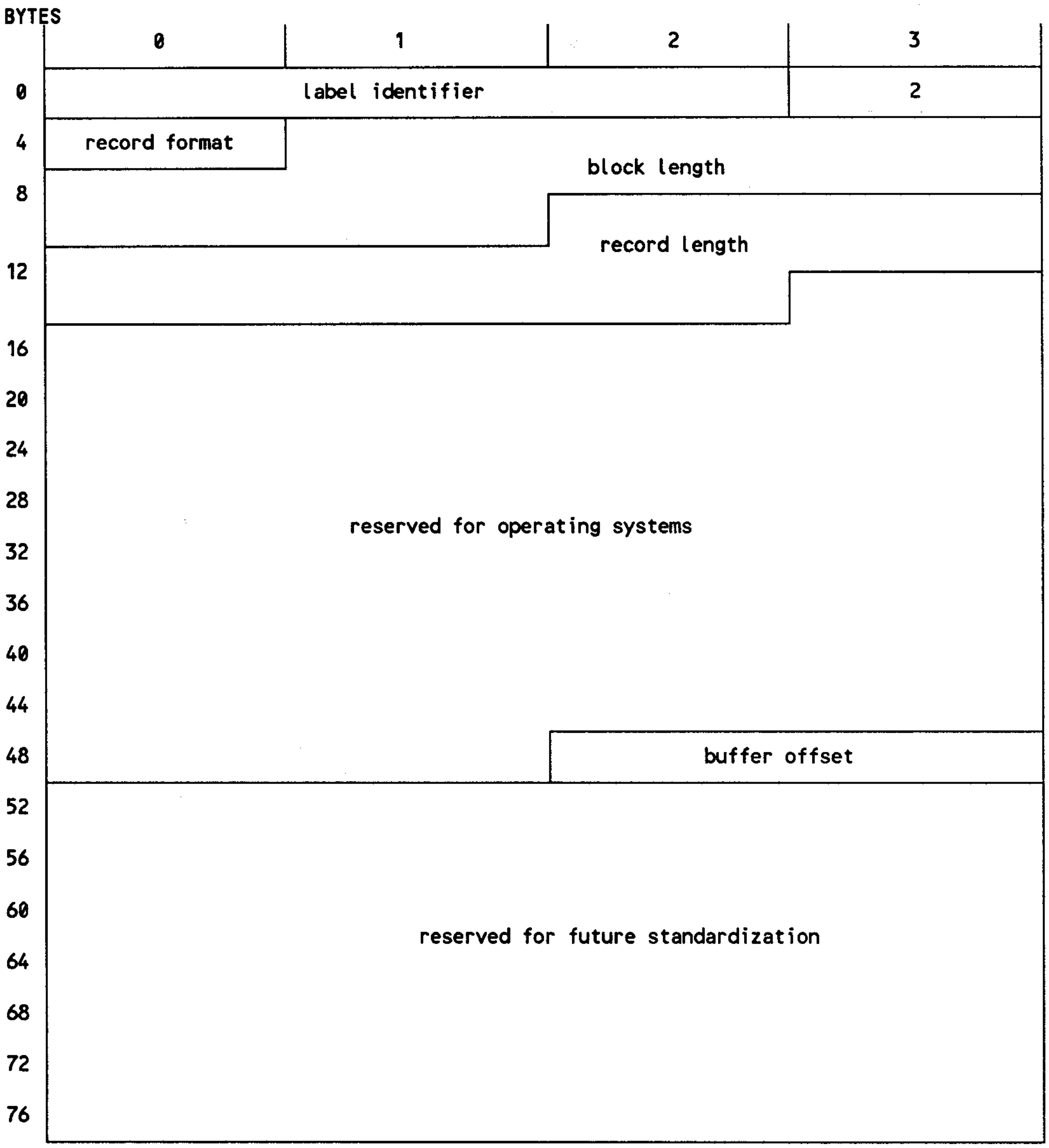


Figure D-11. Second End-of-File or End-of-Volume Label (EOF2/EOV2) for an ASCII Tape Volume

Table D-11. Second End-of-File or End-of-Volume Label (EOF2/EOV2), Field Description for an ASCII Tape Volume

Field	Bytes	Description
Label identifier	0-2	Contains "EOF" if an end-of-file label; "EOV" for end-of-volume.
Label number	3	Must be "2"
Record format	4	Character Meaning F Fixed length D Variable length, with the number of characters in the record specified in decimal V Variable length, with the number of characters specified in binary. (See Note 1, Table D-9.) U Undefined
Block length	5-9	Five "n" characters specifying the maximum number of characters per block. (See Note 3, Table D-8.)
Record length	10-14	Five "n" characters specifying: if "record format" is F, record length; if D or V, maximum record length including any count fields; if U, then undefined. (See Note 1, Table D-9.)
Reserved	15-49	Currently "spaces". (See Note 2, Table D-8.)
Buffer offset (optional)	50-51	Two "n" characters specifying the length in characters of any additional field inserted before a data block - for example, block length. This length is included in the block length. (See Notes 3 and 4, Table D-8.)
Reserved	52-59	Reserved for future standardization; must be "spaces". (See Note 2, Table D-8.)

Index

NOTES

NOTES

NOTES

Help Us To Help You

Publication Title

Form Number Date

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

□ Addition □ Deletion □ Revision □ Error

Comments

Name

Title Company

Address (Street, City, State, Zip)

Telephone Number

Help Us To Help You

Publication Title

Form Number Date

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

□ Addition □ Deletion □ Revision □ Error

Comments

Name

Title Company

Address (Street, City, State, Zip)

Telephone Number

Help Us To Help You

Publication Title

Form Number Date

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

□ Addition □ Deletion □ Revision □ Error

Comments

Name

Title Company

Address (Street, City, State, Zip)

Telephone Number

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

First Class Permit No. 21 Blue Bell, PA

Postage Will Be Paid By Addressee

Unisys Corporation
OS/3 Systems Product Information Development
PO Box 500 - E5-114
Blue Bell, PA 19422-9990

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

First Class Permit No. 21 Blue Bell, PA

Postage Will Be Paid By Addressee

Unisys Corporation
OS/3 Systems Product Information Development
PO Box 500 - E5-114
Blue Bell, PA 19422-9990

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

First Class Permit No. 21 Blue Bell, PA

Postage Will Be Paid By Addressee

Unisys Corporation
OS/3 Systems Product Information Development
PO Box 500 - E5-114
Blue Bell, PA 19422-9990

Help Us To Help You

Publication Title

Form Number   Date

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

□ Addition   □ Deletion   □ Revision   □ Error

Comments

Name

Title   Company

Address (Street, City, State, Zip)

Telephone Number

Help Us To Help You

Publication Title

Form Number   Date

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

□ Addition   □ Deletion   □ Revision   □ Error

Comments

Name

Title   Company

Address (Street, City, State, Zip)

Telephone Number

Help Us To Help You

Publication Title

Form Number   Date

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

□ Addition   □ Deletion   □ Revision   □ Error

Comments

Name

Title   Company

Address (Street, City, State, Zip)

Telephone Number

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

First Class Permit No. 21 Blue Bell, PA

Postage Will Be Paid By Addressee

Unisys Corporation
OS/3 Systems Product Information Development
PO Box 500 - E5-114
Blue Bell, PA 19422-9990

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

First Class Permit No. 21 Blue Bell, PA

Postage Will Be Paid By Addressee

Unisys Corporation
OS/3 Systems Product Information Development
PO Box 500 - E5-114
Blue Bell, PA 19422-9990

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

First Class Permit No. 21 Blue Bell, PA

Postage Will Be Paid By Addressee

Unisys Corporation
OS/3 Systems Product Information Development
PO Box 500 - E5-114
Blue Bell, PA 19422-9990