

**PUBLICATIONS
UPDATE**

Operating System/3 (OS/3)

Basic Data Management

Programmer Reference

UP-8159 Rev. 3-A

This Library Memo announces the release and availability of Updating Package A to "SPERRY UNIVAC Operating System/3 (OS/3) Basic Data Management Programmer Reference", UP-8159 Rev. 3.

This update for the 8.0 release provides new information on the file lock feature. All other changes are corrections or expanded descriptions applicable to features present in basic data management prior to the 8.0 release.

Copies of Updating Package A are now available for requisitioning. Either the updating package only or the complete manual with the updating package may be requisitioned by your local Sperry Univac representative. To receive only the updating package, order UP-8159 Rev. 3-A. To receive the complete manual, order UP-8159 Rev. 3.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
<p>Mailing Lists BZ, CZ and MZ</p>	<p>Mailing Lists A00, A01, 18, 18U, 19, 19U, 20, 20U, 21, 21U, 75, 75U, 76, and 76U (Package A to UP-8159 Rev. 3, 39 pages plus Memo)</p>	<p>Library Memo for UP-8159 Rev. 3-A</p> <hr/> <p>RELEASE DATE: September, 1982</p>



PUBLICATIONS REVISION
Operating System/3 (OS/3)
Basic Data Management Programmer Reference
UP-8159 Rev. 3

This Library Memo announces the release and availability of "SPERRY UNIVAC[®] Operating System/3 (OS/3) Basic Data Management Programmer Reference", UP-8159 Rev. 3.

Changes for this revision include the following:

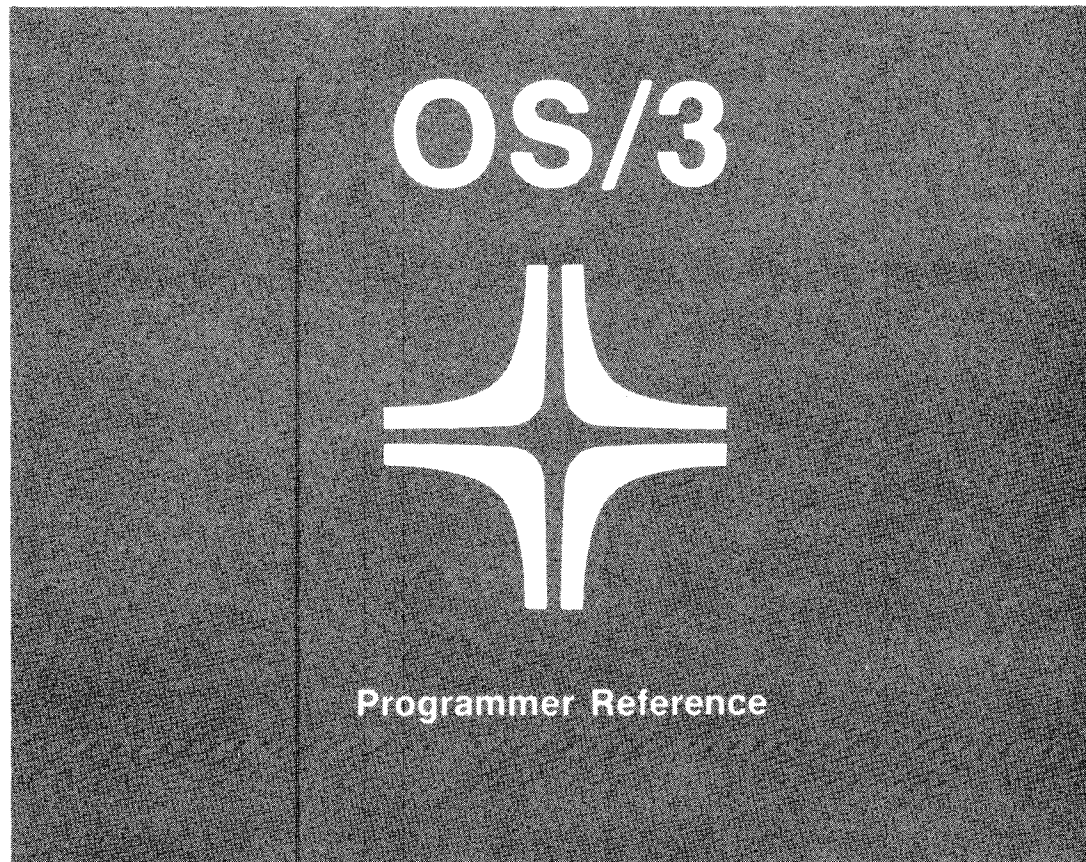
- Reference to consolidated data management is included in the preface.
- The *ALL parameter has been added to all CLOSE macros.
- All mention of "logical IOCS configuration" and configuring your own processing modules are deleted, including references to CDIO, PRIO, MTIO, SAMIO, etc.
- Information about the Sperry Univac 8424 and 8425 Disc Subsystems and the UNISERVO 10 and 14 Magnetic Tape Subsystems has been included.
- Appendix D includes information on IRAM and MIRAM format labels.

Additional copies may be ordered by you; local Sperry Univac representative.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS:
Mailing Lists BZ, CZ and MZ	Mailing Lists 18, 18U, 19, 19U, 20, 20U, 21, 21U, 75, 75U, 76 and 76U (Covers and 398 pages)	Library Memo
		RELEASE DATE: October, 1980



Basic Data Management



Environment: 90/25, 30, 30B, 40 Systems

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry Univac representative.

Sperry Univac reserves the right to modify or revise the content of this document. No contractual obligation by Sperry Univac regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry Univac.

Sperry Univac is a division of the Sperry Corporation.

FASTRAND, SPERRY UNIVAC, UNISCOPE, UNISERVO, and UNIVAC are registered trademarks of the Sperry Corporation. ESCORT, PAGEWRITER, PIXIE, and UNIS are additional trademarks of the Sperry Corporation.

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

PAGE STATUS SUMMARY

ISSUE: Update A -- UP-8159 Rev. 3

RELEASE LEVEL: 8.0 Forward

Part/Section	Page Number	Update Level	Part/Section	Page Number	Update Level	Part/Section	Page Number	Update Level
Cover/Disclaimer		Orig.	Section 9	Title Page 1 thru 12	Orig. Orig.	Appendix J	Title Page 1 thru 12	Orig. Orig.
PSS	1	A	Section 10	Title Page 1 2 3 4 5	Orig. Orig. A Orig. A Orig.	Appendix K	Title Page 1 thru 4	Orig. Orig.
Preface	1 thru 3	Orig.	Section 11	Title Page 1 2 3 4 5	Orig. Orig. A Orig. A Orig.	Appendix L	Title Page 1, 2	A A
Contents	1 thru 8 9 10 11	Orig. A Orig. A	Appendix A	Title Page 1 thru 25	Orig. Orig.	Glossary	1 thru 6 7 8 thru 19 20 21 thru 57	Orig. A Orig. A Orig.
Section 1	Title Page 1 thru 3 4 5	Orig. Orig. A Orig.	Appendix B	Title Page 1 thru 7	Orig. Orig.	User Comment Sheet		
Section 2	Title Page 1 thru 11	Orig. Orig.	Appendix C	Title Page 1 thru 9	Orig. Orig.			
Section 3	Title Page 1 thru 9	Orig. Orig.	Appendix D	Title Page 1 thru 33	Orig. Orig.			
Section 4	Title Page 1 thru 16	Orig. Orig.	Appendix E	Title Page 1 thru 22	Orig. Orig.			
Section 5	Title Page 1 thru 11 12, 13 14, 15	Orig. Orig. A Orig.	Appendix F	Title Page 1 thru 9	Orig. Orig.			
Section 6	Title Page 1 thru 13 14, 15 16, 17	Orig. Orig. A Orig.	Appendix G	Title Page 1 thru 16	Orig. Orig.			
Section 7	Title Page 1 thru 27 28 29, 30 31 32 thru 34	Orig. Orig. A Orig. A Orig.	Appendix H	Title Page 1	Orig. Orig.			
Section 8	Title Page 1 thru 19 20 21 22 23	Orig. Orig. A Orig. A Orig.	Appendix I	Title Page 1 thru 9	Orig. Orig.			

All the technical changes are denoted by an arrow (→) in the margin. A downward pointing arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (↑) is found. A horizontal arrow (→) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.



Preface

This programmer reference manual (PRM) is one in a series designed to be used as quick-reference documents by programmers familiar with the SPERRY UNIVAC Operating System/3 (OS/3). The subject of this particular manual is OS/3 basic data management. Addressed to the experienced BAL programmer, it describes the declarative macro instructions and associated keyword parameters by which he defines his files to OS/3 basic data management, as well as the imperative macros he uses to process them. It also describes the procedures for configuring basic data management processing modules and the methods available for assembling, linking, or sharing these in a multiprogramming environment.

The information presented here is limited to facts; no introductory information or examples of use are provided. Information of these types is presented in two other OS/3 manuals; an introduction to basic data management, UP-8031 (current version); and a basic data management user guide, UP-8068 (current version).

For systems with interactive facilities, an additional series of manuals is provided to instruct and guide the programmer in the use of OS/3 consolidated data management. These are:

- Introduction to consolidated data management, UP-8824 (current version)
- Consolidated data management concepts and facilities, UP-8825 (current version)
- Consolidated data management macro language user guide/programmer reference, UP-8826 (current version)

In general, any further references to the term *data management* in this user guide imply basic data management.

Information in this manual is presented as follows:

- Section 1. Introduction

Describes the layout and use of this manual, OS/3 data management concepts, and the statement conventions for presenting and coding OS/3 data management declarative and imperative macro instructions.

- Section 2. Card File Data Management

Describes the imperative macro instructions and the DTFCDD declarative macro instruction of the card sequential access method (card SAM). This section also includes the minor modifications to card processing procedures involved when the 8413 diskette is used.

- Section 3. Printer File Data Management

Describes the imperative macro instructions and the DTFPR declarative macro instruction of the printer sequential access method (printer SAM).

- Section 4. Magnetic Tape File Data Management

Describes the imperative macros and the DTFMT declarative macro of the magnetic tape sequential access method (magnetic tape SAM).

- Section 5. Disc Sequential Access Method

Describes the imperative macros and the DTFSD declarative macro of the disc sequential access method (disc SAM).

- Section 6. Direct Access Method

Describes the imperative macros and the DTFDA declarative macro of the direct access method for disc (DAM).

- Section 7. Nonindexed Access Method for Disc

Describes the imperative macros and the DTFNI and DPCA declarative macros of the nonindexed disc file processor system.

- Section 8. Indexed Sequential Access Method and Alternate Sequential Access Method

Describes the imperative macros and the DTFIS declarative macro of the disc indexed sequential access method (ISAM) and the alternate sequential access method (ASAM).

- Section 9. Paper Tape Data Management

Describes the imperative macros and the DTFPT declarative macro of the OS/3 data management system for paper tape files.

- Section 10. Indexed Random Access Method

Describes the DTFIR declarative macro of the indexed random access method (IRAM) for disc files.

- Section 11. Multiple Indexed Random Access Method

Describes the DTFMI declarative macro of the multiple indexed random access method (MIRAM) for disc files.

- Appendix A. Tables Cited in Sections 2 Through 10

Provides tabular information required to support the macro instruction descriptions presented in the foregoing sections.

- Appendix B. Error and Exception Handling

Provides information on OS/3 data management error and exception handling procedures and on error/status flags set in *filename*, a programmer-addressable field in each DTF.

- Appendix C. Peripherals

Summarizes operational characteristics of peripheral hardware devices relevant to data management.

- Appendix D. Disc File Labels

Provides information on the layout and content of the volume label and the seven types of format label contained in the disc volume table of contents (VTOC). Also describes optional user header and trailer labels.

- Appendix E. Tape File Labels

Describes the layout and content of the system standard labels for ASCII and EBCDIC tape volumes and files.

- Appendix F. Nonstandard DTF Keyword Parameters

Provides checklists of variant forms of DTF keyword parameters accepted in OS/3 data management.

- Appendix G. File and Record Formats

Describes file and record formats for all OS/3 data management files.

- Appendix H. Lase Factor Calculation

Provides formulas for calculating the lase factor specified for record interlace operations on sequentially processed disc files under the nonindexed file processor system.

- Appendix I. Code Correspondences

Provides a table cross-correlating the ASCII, EBCDIC, and Hollerith punched card codes and describes the compressed and column binary card codes.

- Appendix J. Magnetic Tape Formats and File Conventions

Describes the organization of magnetic tape files and reels according to the four OS/3 conventions for ASCII and EBCDIC files.

- Appendix K. Load Code and Vertical Format Buffers

Explains how to specify codes for the load code buffer and vertical format buffer of the printer subsystems supported by OS/3 data management.

- Appendix L. User File Lock Facility

Describes the user's capability, exercised through data management and job control, of controlling the extent to which data management files may be shared among tasks in a multiprogramming environment.

- Glossary

Defines, in a strict alphabetic ordering, certain data management terms having special meaning in OS/3. Also includes, in the same alphabetic ordering, descriptions of OS/3 data management concepts or programming considerations which are inappropriate for the macro instruction descriptions or the appendixes.



Contents

PAGE STATUS SUMMARY

PREFACE

CONTENTS

1. INTRODUCTION

OS/3 DATA MANAGEMENT CONCEPTS	1-1
File Definition	1-1
File Processing	1-2
OS/3 Access Methods	1-2
Programming Considerations	1-2
Error and Exception Handling	1-3
Record Interface	1-3
STATEMENT CONVENTIONS	1-3
LAYOUT AND USE OF THIS MANUAL	1-4

2. CARD FILE DATA MANAGEMENT

CLOSE	2-1
CNTRL	2-2
GET	2-3
OPEN	2-4
PUT	2-5
DTFCD	2-6

3. PRINTER FILE DATA MANAGEMENT

CLOSE	3-1
CNTRL	3-2
OPEN	3-3
PRTOV	3-4
PUT	3-5
DTFPR	3-6

4. MAGNETIC TAPE FILE DATA MANAGEMENT

CLOSE	4-1
CNTRL	4-2
FEOV	4-3
GET	4-4
LBRET	4-5
OPEN	4-6
PUT	4-7
RELSE	4-8
SETF	4-9
TRUNC	4-10
DTFMT	4-11

5. DISC SEQUENTIAL ACCESS METHOD

CLOSE	5-1
CNTRL	5-2
FEOV	5-3
GET	5-4
LBRET	5-5

OPEN	5-6
PUT	5-7
RELSE	5-8
SETF	5-9
TRUNC	5-10
DTFSD	5-11
6. DIRECT ACCESS METHOD	
CLOSE	6-1
CNTRL	6-2
LBRET	6-3
OPEN	6-4
READ, ID	6-5
READ, KEY	6-6
WAITF	6-7
WRITE, AFTER	6-8
WRITE, AFTER, EOF	6-9
WRITE, ID	6-10
WRITE, KEY	6-11
WRITE, RZERO	6-12
DTFDA	6-13
7. NONINDEXED ACCESS METHOD FOR DISC	
CLOSE	7-1
CNTRL	7-2
FEOV	7-3
GET	7-4
LBRET	7-5

NOTE	7-6
OPEN	7-7
POINT	7-8
POINTS	7-9
PUT	7-10
READ, ID	7-11
READ, KEY	7-12
RELSE	7-13
SETF	7-14
SETP	7-15
SETS	7-16
TRUNC	7-17
WAITF	7-18
WRITE, AFTER	7-19
WRITE, AFTER, EOF	7-20
WRITE, ID	7-21
WRITE, KEY	7-22
WRITE, RZERO	7-23
DPCA	7-24
DTFNI	7-27
8. INDEXED SEQUENTIAL ACCESS METHOD AND ALTERNATE SEQUENTIAL ACCESS METHOD	
ADD	8-1
CLOSE	8-2
ENDFL	8-3
ESETL	8-4
GET	8-5

OPEN	8-6
PUT	8-7
READ, ID	8-8
READ, KEY	8-9
SETFL	8-10
SETL, BOF	8-11
SETL, GKEY	8-12
SETL, ID	8-13
SETL, KEY	8-14
UPDT	8-15
WAITF	8-16
WRITE, KEY	8-17
WRITE, NEWKEY	8-18
DTFIS	8-19
 9. PAPER TAPE DATA MANAGEMENT	
CLOSE	9-1
GET	9-2
OPEN	9-3
PUT	9-4
DTFPT	9-5
 10. INDEXED RANDOM ACCESS METHOD	
DTFIR	10-1
 11. MULTIPLE INDEXED RANDOM ACCESS METHOD	
DTFMI	11-1

APPENDIXES**A. TABLES CITED IN SECTIONS 2 THROUGH 11****B. ERROR AND EXCEPTION HANDLING**

GENERAL	B-1
RETURN OF CONTROL	B-1
ERROR FLAGGING PROCEDURES	B-1
filenameC	B-1
Summary of Data Management Error Processing for Paper Tape Files	B-4
Recovery from Parity Errors in Input Paper Tape Files	B-6
SYSTEM ERROR MESSAGES	B-7
Data Management Error Messages	B-7

C. PERIPHERALS**D. DISC FILE LABELS**

GENERAL	D-1
VOLUME INFORMATION GROUP	D-1
VOL1 Label	D-3
Disc Format 4 Label	D-4
Disc Format 5 Label	D-8
Disc Format 6 Label	D-9
Disc Format 0 Label	D-12
FILE INFORMATION GROUP	D-13
Disc Format 1 Label	D-15
Disc Format 2 Label	D-20
Disc Format 3 Label	D-27
OPTIONAL USER STANDARD LABELS	D-29
User Header Labels	D-29
User Trailer Labels	D-30
DISKETTE FILE LABEL	D-31

E. TAPE FILE LABELS

OS/3 SYSTEM STANDARD LABELS FOR MAGNETIC TAPE	E-1
---	-----

SYSTEM STANDARD TAPE LABEL GROUPS	E-1
Volume Label Group	E-1
File Header Label Group	E-3
First File Header Label (HDR1)	E-3
Second File Header Label (HDR2)	E-5
File Trailer Label Group	E-7
Standard User Header and Trailer Labels	E-11
ASCII STANDARD MAGNETIC TAPE LABELS	E-12

F. NONSTANDARD DTF KEYWORD PARAMETERS

DTFCD KEYWORDS ACCEPTED, BUT NOT IMPLEMENTED	F-1
SPERRY UNIVAC 9200/9300 Systems	F-1
SPERRY UNIVAC OS/4 System	F-1
SPERRY UNIVAC OS/7 System	F-1
IBM 360/20 System	F-1
Other IBM S/360 Systems	F-1
ALTERNATE DTFCD KEYWORD SPELLINGS	F-2
DTFPR KEYWORDS ACCEPTED, BUT NOT IMPLEMENTED	F-3
SPERRY UNIVAC 9200/9300 Systems	F-3
SPERRY UNIVAC OS/4 System	F-3
SPERRY UNIVAC OS/7 System	F-3
IBM 360/20 System	F-3
Other IBM S/360 Systems	F-3
ALTERNATE DTFPR KEYWORD SPELLINGS	F-4
NONSTANDARD DTFMT KEYWORD PARAMETERS	F-5
NONSTANDARD DTFDA, DTFSD, DTFNI, AND DPCA KEYWORD PARAMETERS	F-6
NONSTANDARD DTFIS KEYWORD PARAMETERS	F-7
NONSTANDARD DTFIR KEYWORD PARAMETERS	F-7
NONSTANDARD DTFMI KEYWORD PARAMETERS	F-8
DTFPT KEYWORD COMPATIBILITY WITH OS/4	F-8
DTFPT KEYWORD COMPATIBILITY WITH 9200/9300	F-8
DTFPT KEYWORD COMPATIBILITY WITH IBM S/360 DOS	F-9

G. FILE AND RECORD FORMATS

H. LACE FACTOR CALCULATION

TWO-STEP CALCULATION	H-1
Determining Time Frame	H-1
Programming Considerations	H-1
Device-Independence	H-1

I. CODE CORRESPONDENCES

GENERAL	I-1
EBCDIC/ASCII/HOLLERITH CORRESPONDENCE	I-1
Hollerith Punched Card Code	I-1
EBCDIC	I-1
ASCII	I-1
OTHER CARD CODES	I-7
Compressed Card Code	I-7
Column Binary (Image) Code	I-8
DATA CONVERSION	I-8

J. MAGNETIC TAPE FORMATS AND FILE CONVENTIONS

TAPE VOLUME AND FILE ORGANIZATION	J-1
EBCDIC Standard Volume Organization	J-1
EBCDIC Nonstandard Volume Organization	J-5
EBCDIC Unlabeled Volume Organization	J-7
ASCII Standard Volume Organizations	J-8
END-OF-FILE AND END-OF-VOLUME COINCIDENCE	J-11

K. LOAD CODE AND VERTICAL FORMAT BUFFERS

LOAD CODE AND VERTICAL FORMAT BUFFERS	K-1
LOAD CODE BUFFER INTERCHANGEABILITY	K-1
LCB STATEMENT SPECIFICATION	K-1
LCB Specification for the 0773 and 0778 Printers	K-1
LCB Specification for the 0770 and 0776 Printers	K-2
LCB Specification for the 0768 Printer	K-2
VERTICAL FORMAT BUFFER INTERCHANGEABILITY	K-2
VFB STATEMENT SPECIFICATION	K-2
Specifying Home Paper Position	K-2
Specifying Forms Overflow Position	K-4
Specifying Special Forms	K-4
Paper Tape Loop, 0768 Printer	K-4

L. USER FILE LOCK FEATURE

USING THE FILE LOCK FEATURE	L-1
INDICATING WHICH FILES ARE LOCKABLE	L-1
SETTING FILE LOCKS FOR DATA FILES IN BAL PROGRAMS	L-1
SETTING FILE LOCKS FOR DATA FILES IN NON-BAL PROGRAMS	L-2
FILE LOCK FEATURE SUMMARY	L-2

GLOSSARY**USER COMMENT SHEET****FIGURES**

D-1. VTOC Volume Information Label Group	D-2
D-2. VTOC VOL1 Label	D-3
D-3. Disc Format 4 Label	D-5
D-4. Disc Format 5 Label	D-8
D-5. Disc Format 6 Label	D-11
D-6. Disc Format 0 Label	D-12
D-7. File Information Group Label Chain	D-14
D-8. Disc Format 1 Label	D-15
D-9. Disc Format 2 Label, Nonindexed Files (DTFSD, DTFDA, DTFNI)	D-20
D-10. ISAM (DTFIS) File Information Area, Disc Format 2 Label	D-21
D-11. IRAM/MIRAM File Information Area, Disc Format 2 Label	D-22
D-12. Library File Information Area, Disc Format 2 Label	D-22
D-13. Disc Format 3 Label	D-27
D-14. Optional User Standard Header Label	D-29
D-15. Optional User Standard Trailer Label	D-30
D-16. Diskette File Label Layout	D-31
E-1. Tape Volume 1 (VOL1) Label Format for an EBCDIC Volume	E-2
E-2. First File Header Label (HDR1) Format for an EBCDIC Tape Volume	E-4
E-3. Second File Header Label (HDR2) Format for an EBCDIC Tape Volume	E-6
E-4. Tape File EOF1 and EOVI Label Formats	E-8
E-5. Tape File EOF2 and EOVI Label Formats	E-10
E-6. Optional User Header and Trailer Label Format, ASCII and Standard-Labeled EBCDIC Tape Files	E-12
E-7. Volume Header Label (VOL1) for an ASCII Magnetic Tape Volume	E-13
E-8. First File Header Label (HDR1) for an ASCII Magnetic Tape Volume	E-15
E-9. Second File Header Label (HDR2) for an ASCII Magnetic Tape Volume	E-17
E-10. First End-of-File or End-of-Volume Label (EOF1/EOV1) for an ASCII Magnetic Tape Volume	E-19
E-11. Second End-of-File or End-of-Volume Label (EOF2/EOV2) for an ASCII Magnetic Tape Volume	E-21

G-1.	Card Punch (Output File) Record Formats	G-1
G-2.	Diskette (Output File) Record Formats	G-2
G-3.	Printer Record Formats	G-3
G-4.	Fixed-Length Physical Record Formats, Nonindexed Disc Files without Keys	G-4
G-5.	Variable-Length Physical Record Formats, Nonindexed Disc Files without Keys	G-5
G-6.	Keyed Fixed- and Variable-Length Physical Record Formats, Nonindexed Disc Files	G-6
G-7.	Organization of a DTFNI Disc File into Partitions and Subfiles	G-7
G-8.	Fixed-Length ISAM and ASAM Records, with and without Keys	G-8
G-9.	Variable-Length ISAM and ASAM Records, with and without Keys	G-9
G-10.	Layout of ISAM Data Blocks (Prime or Overflow) on Disc, Each Containing Two Logical Records	G-10
G-11.	OS/3 ISAM File Structure	G-11
G-12.	Record and Block Formats for Magnetic Tape Files, ASCII and EBCDIC	G-12
G-13.	IRAM Data Record Formats	G-15
G-14.	MIRAM Data Record Formats	G-16
I-1.	Compressed Card Code	I-7
I-2.	Column Binary (Image) Card Code	I-8
J-1.	Reel Organization for EBCDIC Standard-Labeled Tape Volumes, Containing a Single File	J-2
J-2.	Reel Organization for EBCDIC Standard-Labeled Tape Volume: Multifile Volume with End-of-File Condition	J-3
J-3.	Reel Organization for EBCDIC Standard-Labeled Tape Volumes: Multifile Volumes with End-of-Volume Condition	J-4
J-4.	Reel Organization for EBCDIC Nonstandard Volume Containing a Single File	J-5
J-5.	Reel Organization for EBCDIC Nonstandard Multifile Volume	J-6
J-6.	Reel Organization for Unlabeled EBCDIC Volumes	J-7
J-7.	Label Configuration, ASCII Single File, Single-Volume and Multivolume Sets	J-8
J-8.	Label Configuration, ASCII Multifile, Single-Volume Set	J-9
J-9.	Label Configuration, ASCII Multifile, Multivolume Set	J-10
J-10.	Label Configuration Options, ASCII Multifile, Multivolume Set, When End-of-Volume and End-of-File Coincide	J-12

TABLES

A-1.	Use of IOREG Keyword Parameter for Sequential Processing of Block or Unblocked Card, Tape, Printer, or Disc Files	A-1
A-2.	Sizes of DTFCD File Tables	A-2
A-3.	Summary of DTFCD Keyword Parameters	A-2
A-4.	Device-Independent Control Character Codes	A-4
A-5.	Device Skip Code Table	A-6
A-6.	Sizes of DTFPR File Table	A-7
A-7.	Summary of DTFPR Keyword Parameters	A-7
A-8.	Overflow and Home Paper Control Character Codes	A-8
A-9.	Summary of Imperative Macros Used with Magnetic Tape SAM	A-8
A-10.	Summary of DTFMT Keyword Parameters	A-9
A-11.	Effects of Job Control VOL and LBL Statements on Data Management OPEN Transient, Standard-Labeled Tape Files	A-11
A-12.	Summary of DTFSD Keyword Parameters	A-13
A-13.	Summary of DTFDA Keyword Parameters	A-14
A-14.	Summary of DTFNI and DPCA Keyword Parameters	A-16

A-15.	Sizes of DTFIS File Table	A-18
A-16.	Summary of DTFIS Keyword Parameters	A-18
A-17.	Summary of DTFPT Keyword Parameters	A-20
A-18.	Relative Disc Address (ID) Returned to IDLOC Field by WAITF Macro after a READ or WRITE Macro Is Issued to a Direct Access File	A-22
A-19.	Summary of DTFIR Keyword Parameters	A-23
A-20.	Summary of DTFMI Keyword Parameters	A-24
B-1.	Significance of Bits in <i>filenameC</i>	B-2
B-2.	Significance of Bits in <i>filenameC</i> , DTFPT Declarative Macro	B-4
C-1.	SPERRY UNIVAC Card Reader Subsystem Characteristics	C-1
C-2.	SPERRY UNIVAC Card Punch Subsystem Characteristics	C-2
C-3.	SPERRY UNIVAC Printer Subsystem Characteristics	C-3
C-4.	SPERRY UNIVAC Disc Subsystem Characteristics	C-7
C-5.	UNISERVO Subsystem Characteristics	C-8
C-6.	SPERRY UNIVAC 0920 Paper Tape Subsystem Characteristics	C-9
D-1.	Contents of VOL1 Label	D-4
D-2.	Contents of Disc Format 4 Label	D-6
D-3.	Contents of Disc Format 5 Label	D-9
D-4.	Contents of Disc Format 6 Label	D-10
D-5.	Contents of Disc Format 0 Label	D-13
D-6.	Contents of Disc Format 1 Label	D-16
D-7.	Contents of Disc Format 2 Label	D-23
D-8.	Contents of Indexed File Information Area, Disc Format 2 Label	D-25
D-9.	Contents of IRAM/MIRAM File Information Area, Disc Format 2 Label	D-25
D-10.	Contents of Library Information Area, Disc Format 2 Label	D-26
D-11.	Contents of Disc Format 3 Label	D-28
D-12.	Diskette File Label Description	D-32
E-1.	Tape Volume (VOL1) Label Format, Field Description for an EBCDIC Volume	E-3
E-2.	First File Header Label (HDR1), Field Description	E-5
E-3.	Second File Header Label (HDR2), Field Description	E-7
E-4.	Tape File EOF1 and EOVI Labels, Field Description	E-9
E-5.	Tape File EOF2 and EOVI Labels, Field Description	E-11
E-6.	Optional User Header and Trailer Labels, Field Description for ASCII and Standard-Labeled EBCDIC Tape Files	E-11
E-7.	Volume Header Label (VOL1), Field Description for an ASCII Volume	E-14
E-8.	First File Header Label (HDR1), Field Description for an ASCII Volume	E-16
E-9.	Second File Header Label (HDR2), Field Description for an ASCII Volume	E-18
E-10.	First End-of-File or End-of-Volume Label (EOF1/EOV1), Field Description for an ASCII Volume	E-20
E-11.	Second End-of-File or End-of-Volume Label (EOF2/EOV2), Field Description for an ASCII Volume	E-22
I-1.	Cross-Reference Table: EBCDIC/ASCII/Hollerith	I-2
K-1.	VFB Statement Specification and Interchangeability	K-3
L-1.	File Lock Summary	L-2





1. Introduction



OS/3 DATA MANAGEMENT CONCEPTS

The SPERRY UNIVAC Operating System/3 (OS/3) data management system is a logical input/output control system (logical IOCS) that simplifies the input and output of data on card readers, card punches, printers, magnetic tape units, and disc storage systems for the basic assembly language (BAL) programmer. OS/3 data management enables him to deal with his files by logical name, rather than by directly addressing the peripheral devices on which they are stored. In common with other system programs, such as the linkage editor and the language processors, OS/3 data management communicates with SAT (the resident systems access technique of the supervisor) for access to the physical input/output control system (physical IOCS) of OS/3.

File Definition

The components of OS/3 data management include file-processing modules, transient routines, and declarative and imperative macro instructions. The BAL programmer describes his files to OS/3 data management by means of *define the file* declarative macro instructions (DTFs). He uses their keyword parameters to describe certain file characteristics and to inform data management of the file-processing techniques to be used on the file. These DTF declaratives establish DTF file control tables, maintained by data management to record file characteristics and certain of the results of file processing. Certain DTF specifications can be changed at run time via the data definition job control statement (DD). The DTF's may have the following forms:

- DTFCD
Define a card file. (Section 2)
- DTFPR
Defines a printer file. (Section 3)
- DTFMT
Defines a magnetic tape file. (Section 4)
- DTFSD
Defines a sequential disc file. (Section 5)
- DTFDA
Defines a direct access disc file. (Section 6)
- DTFNI
Defines a nonindexed disc file. (Section 7)
- DPCA
Defines a partition of a nonindexed disc file. (Section 7)
- DTFIS
Defines an indexed sequential or alternate sequential access method disc file. (Section 8)

- DTFPT
Defines a paper tape file. (Section 9)
- DTFIR
Defines sequential and random indexed or nonindexed IRAM files. (Section 10)
- DTFMI
Defines sequential and random indexed or nonindexed MIRAM files. (Section 11)

File Processing

In addition to his own coding, the BAL programmer processes his files with using imperative macro instructions for each of the file definitions associated with OS/3 data management access methods. These macros are supported by variously configurable logical IOCS processors which the programmer may include or invoke with his program. At an installation using multiprogramming, however, these processors may be incorporated in shared, reentrant data management modules which he may use simultaneously in common with a number of other users.

OS/3 Access Methods

OS/3 data management offers the following access methods:

- Unit record sequential access methods for card and printer (card SAM, printer SAM)
- Magnetic tape sequential access method (tape SAM)
- Indexed sequential access method for disc (ISAM)
- Alternate sequential access method for disc (ASAM)
- Sequential access method for disc (disc SAM)
- Direct access method for disc (DAM)
- Nonindexed access method for disc (no acronym)
- Sequential access method for paper tape files (no acronym)
- Indexable random access method for disc (IRAM)
- Multiple indexed random access method for disc (MIRAM)

Programming Considerations

Certain considerations are common to all programming with OS/3 data management. For example, by using the OS/3 job control language, all OS/3 users employ the same conventions for designating existing files and new files in the control stream. The data management user, in creating a file table in a data area for each file, uses the appropriate DTF declarative macro to describe the file and to provide the addresses of buffers and work spaces. He must also either indicate that he will handle all returns inline or provide the address of his routine for

accepting control when errors or exceptions occur. He uses regular BAL instructions for this error/exception routine and to reserve sufficient buffer and work space in main storage. In his use of the general registers, the BAL programmer generally avoids registers 0, 1, 14, 15 — these are loaded by the data management imperatives before registers are saved. Each program requires a 72-byte save area for general registers; the programmer may provide this in the DTF, or he may provide it in register 13 as he enters each data management imperative.

Error and Exception Handling

OS/3 data management imperative macros always return control to the user program (whether or not an error or exception has been detected) and set a reply field in the DTF to indicate the result of processing or the nature of the error or exception in every case. If the imperative macro function is successfully executed, control is always returned inline to the instruction next following the macro call. If the user provides the address of an error/exception-handling routine in his DTF, data management transfers control to that address on all occurrences of error or exception; otherwise, control returns to the user inline. Register 14 always contains the inline return address. Because data management interprets a zero value in the DTF error field of the DTF file table as the nonexistence of an error routine, users must not locate their error routine at location 0 relative to the load module. File extend errors are always due to inability to acquire buffer output space. File extend procedures now minimize loss of output data to one record.

Record Interlace

In OS/3 data management, record interlace is a data-mapping and tuning technique that enhances throughput in sequential operations with disc files by reducing the effect of rotational delay on record input and output. Before creating his disc files, the data management user calculates and specifies in his DTF a *lace factor*, by which data management and SAT adjust the physical interval between logically sequential records on disc. Calculation of the lace factor is described in Appendix I; the objective is to arrange the records on each track so that they may be retrieved as fast as the processing program is capable of handling them. The resulting physical arrangement of records is a permanent characteristic of the file, and all programs subsequently accessing the file must specify the same lace factor used to create it. The lace factor is strongly program dependent; the device-dependent aspects of the calculation are automatically handled by SAT and OS/3 data management. The record interlace technique is not available in ISAM or ASAM.

STATEMENT CONVENTIONS

- Positional parameters must be coded in the specified order in the operand field and must be separated by commas. Except for trailing parameters, each positional parameter not coded must be represented by a comma.
- A keyword parameter consists of an alphanumeric character string immediately followed by an equal sign and one completion or specification. Keyword parameters may be coded in any order in the operand field; commas are required only to separate parameters. A comma must not be coded in column 16 of a continuation line nor follow the last keyword in a string.
- Capital letters, commas, equal signs, and parentheses must be coded exactly as shown in format statements. Exceptions are those capitalized acronyms which are part of generic terms representing information to be supplied by the user.
- Lowercase letters and words are generic terms representing information to be supplied by the user. For readability, these lowercase terms may contain hyphens and capitalized acronyms, but the capitals and hyphens are never coded.

- Braces enclose mandatory entries, one of which must be chosen and coded by the programmer. Braces themselves are never coded.
- Brackets enclose optional entries that may be omitted or coded, depending on program requirements. Certain entries enclosed in brackets are mandatory under specified conditions. Braces within brackets signify that one of the enclosed entries must be chosen if the optional parameter is chosen. Brackets themselves are never coded.
- A shaded background printed in the format delineation behind one of the listed entries of an optional parameter indicates the default specification supplied by OS/3 data management when the programmer does not specify the parameter. When omission of an optional parameter causes data management to perform some operation other than supplying a default specification, this action is explained in the parameter description.
- An ellipsis indicates the omission from the format delineation of a variable number of entries which may be coded serially by the programmer. The ellipsis itself is never coded.
- The word *unused*, printed in the label or operand field of a format delineation, indicates that no entry is to be made in this field and that the statement will be rejected as invalid by OS/3 data management if an entry is coded.
- The word *ignored*, printed in the label or operand field of a format delineation, indicates that no entry is to be made in this field and that any entry coded will be ignored by OS/3 data management.

LAYOUT AND USE OF THIS MANUAL

→ This programmer reference (PR) manual is laid out to facilitate its use as quick-reference aid to the experienced BAL programmer. Following this generalized section, 10 separate sections (Section 2 through 11) provide essentially the same treatment for the 10 access methods of OS/3 data management. In each section, format delineation of the logical IOCS declarative macro precedes the delineations of the imperative macros pertaining to the access method. Each section concludes with format delineations for DTF declarative macros and descriptions of the pertinent keyword parameters.

The imperative macros within each section are ordered alphabetically, as are the keyword parameters for the declarative macros. Macro mnemonics, printed in large type at the top, are carried over from page to page to serve as quick location aids within each section. Beneath each macro call, the mnemonic for the logical IOCS processor which supports it is printed in parentheses.

Self-contained — in that every macro is presented in full in each section where it belongs — these sections are nevertheless not complete descriptions of the access methods. For textual or tabular information supporting the macro instruction descriptions, the BAL programmer is referred to the appendixes at the end of the manual. Specific cross-references are made in the text to both tables and appendixes, to each of which the table of contents is also a guide. There is no index.

The Glossary defines or describes italicized terms appearing in other parts of the manual (except for those italicized only for emphasis). The Glossary is intended as a supplementary reference to the rest of the manual for the experienced BAL programmer who needs to locate information in a hurry and does not find it in the macro descriptions. To become familiar with the Glossary's content and structure beforehand will be helpful to him.

Glossary entries are of two kinds, presented with cross-references in one alphabetic order. A small number of entries represent basic definitions of terms having special meaning in OS/3 data management; very few fundamental terms have been carried over from the user guide glossary, in order to make this one more serviceable to the experienced BAL programmer. The majority of the entries are of a kind not found in the user guide: these represent programming information that supports the macro descriptions but was inappropriate to

reproduce among them. Most italicized terms in the macro descriptions point to entries in the Glossary; this scheme leaves the format delineations uncluttered and most useful for quick lookup, at the same time giving the reader assurance that he can readily locate related information.

Although the Glossary is authoritative for OS/3 data management, it has a special purpose and is not to be considered a controlled or standard thesaurus of data management terms. For a standardized work of that type, the reader is referred to the systems programming vocabulary for information processing, UD1-1604 (current version).

This PRM contains all technical information required by the experienced BAL programmer from the OS/3 data management user guide, UP-8068 (current version), which he should consult when he wishes to review any subject or function in depth.



2. Card File Data Management



CLOSE

Function:

Initiates the termination procedures for a card SAM file.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	CLOSE	$\left\{ \begin{array}{l} \text{filename-1, [..., filename-n]} \\ (1) \\ 1 \\ *ALL \end{array} \right\}$

Parameters:

filename

Is the label of the corresponding DTFCD macro instruction in the user's program.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction when a single file is to be terminated.

***ALL**

Specifies that all files open in the job step are to be closed.



CNTRL

Function:

Specifies output *stacker selection* on the *0604 card punch*. The keyword parameter CONTROL must be specified in the DTFCD declarative macro when the CNTRL imperative macro is to be issued to the file.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	CNTRL	$\left\{ \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} ,SS \left[, \left\{ \begin{array}{c} 1 \\ 2 \end{array} \right\} \right]$

Parameters:

filename

Is the label of the corresponding DTFCD macro instruction in the program.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction.

SS

Specifies stacker selection on the 0604 card punch.

1

Specifies the normal stacker.

2

Specifies the select (error) stacker.

If the third positional parameter is omitted, specification of the select stacker is assumed. If this parameter is specified, but is not specified as 1 or 2, specification of the normal stacker is assumed, and an error flag appears in the user's program listing.

GET

Function:

Makes the next logical record available, either in the current I/O area or in the work area specified in the second positional parameter. More than one I/O or work area may be used, each GET macro specifying a different work area if desired. See Table A—1 for multiple work area and I/O area considerations, and the uses of the IOREG keyword parameter in the DTFCD declarative macro.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	GET	$\left\{ \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} \left[, \left\{ \begin{array}{c} \text{workarea} \\ (0) \\ 0 \end{array} \right\} \right]$

Parameters:

filename

Is the label of the corresponding DTFCD macro instruction in the program.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction.

workarea

Is the label of an area into which the current record is moved for processing.

(0) or 0

Indicates that register 0 has been preloaded with address of a work area.

Omission of the second parameter indicates the user has chosen processing either by means of an index register (IOREG keyword parameter) or by directly accessing the data relative to the name of the I/O area. When it is specified, WORKA=YES must also be specified in the DTFCD declarative macro.

OPEN

Function:

Calls a transient routine to perform certain validation checks and initiate file processing. A check is made to determine whether all the necessary keyword parameters defining the file have been supplied and whether all parameters supplied are in valid form. The device allocation performed by the job control program is determined.

For input files, the first data record is not available until a GET macro instruction is issued. For output files, no data is written; however, the data area is made available for use.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	OPEN	{ filename-1 [, ..., filename-n] } (1) 1

Parameters:

filename

Is the label of the corresponding DTFCD macro instruction in the program. The *filename* may have a maximum of seven characters; the maximum number of file names is 16.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction, when a single file is to be opened.

PUT

Function:

Delivers an output record to data management either in the current output area or in the work area specified in the second positional parameter. Data management delivers records singly to the card punch but clears neither area.

When data management receives a record in the work area, it moves it to the current output area and frees the work area. When punching a record containing an odd number of characters, data management places a nonpunching character in the I/O area immediately after the user-supplied data.

See Glossary for notes on outputting variable and undefined *cardpunch records*, and Table A-1 for multiple I/O and work area considerations and use of the IOREG keyword parameter in the DTFCD declarative macro.

Format:

LABEL	△ OPERATION △	OPERAND	
[name]	PUT	{ filename } (1) 1	[{ workarea }] (0) 0

Parameters:

filename

Is the label of the corresponding DTFCD macro instruction in the program.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction.

workarea

Is the label of the work area from which the record may be obtained.

(0) or 0

Indicates that register 0 has been preloaded with the address of the work area.

Omission of the second positional parameter indicates the user has chosen processing either by means of an index register (IOREG keyword parameter) or by directly accessing the data relative to the name of the I/O area. When the second positional parameter is specified, WORKA=YES must also be specified in the DTFCD declarative macro.

DTFCD

Function:

Defines punched card files or diskette files, and creates a file table of varying length; see Table A—2. Keyword parameters are summarized in Table A—3; variant forms in Appendix F.

Format:

LABEL	△ OPERATION △	OPERAND
filename	DTFCD	[ASCII=YES] [,AUE=YES] [,BLKSIZE=n] [,CONTROL=YES] [,CRDERR=RETRY] [,EOFADDR=symbol] [,ERROR=symbol] ,IOAREA1=symbol [,IOAREA2=symbol] [,IOREG=(r)] [,ITBL=symbol] [,MODE= { BINARY CC STD TRANS }] [,OPTION=YES] [,ORLP=YES] [,OTBL=symbol] [,OUBLKSZ=n] [,RECFORM= { FIXUNB UNDEF VARUNB }] [,RECSIZE= { (r) n }] [,SAVAREA=symbol] [,STUB= { 51 66 }] [,TYPEFLE= { INPUT OUTPUT CMBND }] [,WORKA=YES]

DTFCD

Parameters:

ASCII=YES

Specifies processing in *American National Standard Code for Information Interchange (ASCII)*.

For input files, this parameter is specified if card data is to be translated into ASCII for internal processing and storage.

For output files, this parameter is specified if internal processing is in ASCII and output in *Hollerith punched card code* is desired.

MODE=STD must be specified if this parameter is supplied.

AUE=YES

Inhibits data management error processing when *unique unit errors* (validity check errors) are detected on nonbinary input card files. If specified, and the operator replies "I" or "U" to the physical IOCS validity check message, data management does not branch to the user's ERROR routine.

If omitted, and operator replies "U", data management sets *unique unit error* flag (byte 0, bit 2) in *filenameC* and branches to user's error routine, issuing error message DM30 (VALIDITY CHECK ERROR). Refer to Appendix B.

Data management ignores the AUE keyword parameter if 8413 diskette files are used.

BLKSIZE=n

Specifies the length of the I/O area in bytes. If the records in a file are variable length, *n* specifies the maximum size for records.

The user can specify BLKSIZE=1 to BLKSIZE=96 to read from 1 to 96 columns of a 96-column card.

A user program which specifies BLKSIZE=1 to BLKSIZE=80 can read from 1 to 80 columns of 96-column cards. A program which has a BLKSIZE of 1 to 80 and which has been used with 80-column cards, in any mode except binary, can read 96-column cards with no program changes.

A program which specifies BLKSIZE=1 to BLKSIZE=96 to the DTFCD proc call can be used to read up to 96 columns of a 96-column card. Such a program can also read up to 80 columns of 80-column cards. The DMCS will blank out (insert the appropriate blank character based on data mode) bytes in the user's I/O and work areas for columns beyond 80. At OPEN time, DMCS checks for 80-column cards being read with BLKSIZE from 81 to 96. If such a condition exists, a message is issued to the operator with a required reply.

If omitted, the block size (80 or 160) is determined from the MODE, RECFORM, or STUB keyword parameter.

CONTROL=YES

Specifies that user's program will issue one or more CNTRL macros to control *stacker selection* on the *0604 card punch*; used only for output or *combined card files*.

DTFCD

If CNTRL macro instructions are issued for files to be output on the *0605 card punch*, which does not have program control over the stacker, the CNTRL macro instructions are ignored. The CNTRL macro instruction does not apply to input files. If CONTROL=YES is specified on an input file, the parameter is ignored, and a diagnostic message is printed in the macro expansion. This parameter also is ignored when the 8413 diskette is used.

CRDERR=RETRY

Specifies that error recovery should be attempted for a card punch (hole count) error on the 0604 and 0605 combined read/punch files or on the 0604 card punch output files with stacker selection. If error recovery is not successful, data management returns control to the user's error routine, if he has provided one; otherwise, control returns to him inline. Except for combined files, the image of the error card is returned to *filenameS*.

If omitted, recovery is not attempted and data management returns control as stated.

Whether the CRDERR keyword is specified or not, on detecting a card punch error, data management sets the *hardware error* flag (bit 3, byte 0) in *filenameC* and issues error message DM23. After six successive attempts to punch a card have failed, data management sets the *hardware error* bit in *filenameC* and also places the image of the erroneous card in *filenameS* of the DTFCD file table. *FilenameS* which your program may address in the same way as *filenameC*, is an 80-byte field for all modes except the binary (image) mode, and a 160-byte field for that mode. If the error card is in a combined file, *filenameS* will not contain the error card image.

Error cards are automatically selected into the error select stacker of the 0604 card punch in all cases. Error recovery is possible with 0604 and 0605 read/punch card files and the 0604 card output files with stacker selection.

If the 8413 diskette is used, the CRDERR=RETRY parameter is ignored.

EOFADDR=symbol

Specifies the address to which control is transferred when the *end-of-data (/*)* job control statement is sensed. This keyword parameter is required for all input and combined files.

ERROR=symbol

Specifies the address of the user's error-handling routine, to which control is transferred when a fatal hardware or detectable logical error occurs on a file.

If not specified, errors return inline.

IOAREA1=symbol

Specifies the address of an I/O area that each input or output file must have reserved for its individual use. Keyword parameter IOAREA1 specifies the input area for a combined file; IOAREA2 must also be used to specify the combined file's output area. I/O areas must contain an even number of bytes for data to be punched or read; when STUB=51 is specified, the I/O area must contain at least 52 bytes. The first data byte (character read or to be punched) must be aligned on a half-word boundary. The length of the area is specified by the keyword parameter BLKSIZE.

IOAREA2=symbol

May specify a secondary I/O area for standby processing; must be used to specify the output area for a combined file. I/O areas must provide for an even number of data bytes. The first data byte must be aligned on a half-word boundary.

DTFCD**IOREG=(r)**

Specifies the number of the general register (2 through 12) used to reference current data. If SAVAREA is specified, register 13 may be used for IOREG. If a work area is not required, this keyword parameter must be specified when there are two I/O areas. This parameter may not be specified if either a work area or a combined file is specified through the DTFCD macro instruction. Do not specify WORKA if this parameter is specified. Refer to Table A-1.

ITBL=symbol

Specifies the address of the 256-byte *translation table* in the user's program when records in an input or combined file are to be translated on input. If MODE=TRANS is specified in a DTFCD macro for such files in these circumstances, the keyword parameter ITBL must also be specified.

MODE=BINARY

Used for cards read on the card reader in *column binary* mode or for cards read or punched on the card punch in column binary (image) mode. An I/O area of 160 bytes is required for one 80-column card.

The binary mode is not available for 96-column cards.

This parameter is ignored if the 8413 diskette is used.

MODE=CC

On the card punch, this form must be specified for cards read or punched in *compressed code*. An I/O area of 80 bytes is required for one 80-column card.

MODE=STD

Should be specified for cards to be read or punched in *EBCDIC*. This form of the MODE keyword parameter must be specified if ASCII=YES is specified. If no MODE keyword is supplied, this option is assumed.

MODE=TRANS

Specified to have cards read in *EBCDIC* and translated by the user's ITBL *translation table*, or translated by his OTBL translation table and then punched in EBCDIC. Requires specification of the ITBL or OTBL keyword.

OPTION=YES

Specifies an optional file: one which will not invariably be required for every execution of the user's program.

When the OPTION keyword parameter is used, *optional file processing* is performed by data management:

- if the OPT positional parameter is included in the DVC job control statement and the device is not available at execution time; or
- when no device is assigned to the file by job control statements (i.e., no DVC-LFD device assignment set).

DTFCD

If OPTION=YES is not specified, and one of the foregoing conditions occurs, the file is not opened. Data management branches to the user's error routine, if he has supplied one, or to the normal return point in his program if he has not. He will not be able to perform further processing on the file.

ORLP=YES

Must be specified to process *combined files* in *overlap* mode when using a card read/punch unit with the prepunch read station feature installed.

In the overlap mode, each GET and PUT macro instruction advances one card. Without overlap, the unit advances one card on PUT macro instructions only.

OTBL=symbol

Specifies the address of the 256-byte translation table in the user's program when records in an output or combined file are to be translated on output. A *translation table* is required if MODE=TRANS is specified.

OUBLKSZ=n

Specifies the length (in bytes) of the secondary I/O area (IOAREA2) for a combined file. If OUBLKSZ is omitted, the size of the output block is assumed to be the same length as BLKSIZE.

RECFORM=FIXUNB

Specifies fixed-length, unblocked records. No other record format specification may be used for input or *combined card files*. Data management assumes RECFORM=FIXUNB has been specified if the RECFORM keyword is omitted from the DTFCD, regardless of the TYPEFLE keyword specification.

RECFORM=UNDEF

Used for undefined records in output files only. The RECSIZE keyword parameter must be specified when this option is used. If the 8413 diskette is used, this parameter specification is ignored.

RECFORM=VARUNB

Used for variable-length, unblocked records in output files only.

RECSIZE=(r)

Specified only for output files, and only with undefined *card punch record* format; *r* indicates the number (2 through 12) of the general register that holds the length of the output record. The record size must be entered into the RECSIZE register before the PUT macro instruction is issued. If SAVAREA is specified, register 13 may be used for the RECSIZE register.

RECSIZE=n

Specifies the record size in bytes used in conjunction with the BLKSIZE parameter value. Data management uses both values to invoke multisector I/O operations in processing diskette files.

SAVAREA=symbol

Specifies the label of a 72-byte register save area, aligned on a full-word boundary.

If SAVAREA is not specified, register 13 must be loaded with the address of a 72-byte register save area, aligned on a full-word boundary, before any imperative macros are issued.

DTFCD**STUB=51**

The *stub card read feature* is to be used for cards containing 51 columns; the I/O area must not be less than 52 bytes.

STUB=66

The stub card read feature is to be used for cards containing 66 columns.

The block size supplied (BLKSIZE=n) must be less than or equal to the number of columns in the card. Used with 0716, 0717, and 0719 card readers, the stub read feature must be installed.

If omitted, standard 80-column cards are assumed. This parameter is ignored when the 8413 diskette is used.

TYPEFLE=INPUT

Describes an input file only. This option is assumed if this keyword parameter is not specified.

TYPEFLE=OUTPUT

Describes an output file only.

TYPEFLE=CMBND

Describes a *combined card file*, for which the *read/punch* feature is to be used.

WORKA=YES

Specified if I/O records are to be processed in a work area rather than in the I/O area. The address of the current work area must be specified with each GET or PUT macro instruction. If this keyword parameter is specified, the keyword parameter IOREG must not be specified. Refer to Table A—1.



3. Printer File Data Management



CLOSE

Function:

The CLOSE macro instruction is issued when all the data in a file has been processed. This macro instruction calls a transient routine which checks for errors that may have occurred in the final output operation and then prevents further access to the file.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	CLOSE	{ filename-1 [, ..., filename-n] } (1) 1 *ALL

Parameters:

filename

Is the label of the corresponding DTFPR macro instruction in the user's program. The maximum number of file names is 16.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction when a single file is to be terminated.

***ALL**

Specifies that all files open in the job step are to be closed.

CNTRL

Function:

The CNTRL macro instruction controls printer forms spacing and skipping. Forms motion can occur before, after, or both before and after a line is printed.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	CNTRL	{ filename } { SK } [,m] [,n] { (1) } { 1 }

Parameters:

filename

Is the label of the corresponding DTFPR macro instruction in the program.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction.

SK

Indicates that forms skipping is desired.

SP

Indicates that forms spacing is desired.

m

Specifies either the number of lines (0 through 15) for immediate spacing, or the channel code (1 through 15) for immediate skipping. See Tables A—4 and A—5. The PUT macro performs spacing after printing. The number of lines spaced is determined by the PRAD keyword parameter (default causes advance of one line). Immediate CNTRL spacing is in addition to any spacing performed on a previous PUT macro instruction. It should be noted that forms overflow processing can be initiated after CNTRL processing (immediate or delayed) is performed.

n

Specifies either the number of lines (0 through 15) for delayed spacing, or the channel code (1 through 15) for delayed skipping. See Tables A—4 and A—5.

OPEN

Function:

Calls a transient routine to perform certain validation checks and initiate file processing. A check is made to determine whether all the necessary keyword parameters defining the file have been supplied and whether all parameters supplied are in valid form. The device allocation performed by the job control program is determined, and the I/O register (if any is specified) is set up.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	OPEN	{ filename-1 [, ..., filename-n] } (1) 1 *ALL

Parameters:

filename

Is the label of the corresponding DTFPR macro instruction in the program; *filename* may have a maximum of seven characters. The maximum number of file names is 16.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction when a single file is to be opened.

PRTOV

Function:

The PRTOV macro instruction specifies the action to be taken when a *forms overflow* condition occurs.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	PRTOV	{ filename } [, { 9 }] [, { overflowname }] (1) 1 (0) 0

Parameters:

filename

Is the label of the corresponding DTFPR macro instruction in the program.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction.

9

Indicates that forms overflow is to be indicated by channel 9.

12

Indicates that forms overflow is to be indicated by channel 12 (0770 printer only).

If the second positional parameter is omitted, channel 9 is assumed for the 0768 printer, channel 12 is assumed for the 0776 printer, and channel 1 is assumed for the 0773 and 0778 printers. Channel 1 is the only overflow code recognized by the 0773 and 0778 printers.

overflowname

Is the label of the user's overflow routine. When overflow occurs and this option is specified, data management transfers control to this address.

(0) or 0

Indicates that register 0 has been preloaded with the address of the user's overflow routine.

If the third positional parameter is omitted, data management provides an automatic skip to *home paper*.

PUT

Function:

Delivers an output record to data management in either the current output area or a user-specified work area. Refer to Table A—1. This output record may be a line to be printed or, if the record comprises only a *control character code*, causes carriage movement alone.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	PUT	$\left\{ \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} \left[\cdot \left\{ \begin{array}{c} \text{workarea} \\ (0) \\ 0 \end{array} \right\} \right]$

Parameters:

filename

Is the label of the corresponding DTFPR macro instruction in the program.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction.

workarea

Is the label of the work area from which the record may be obtained.

(0) or 0

Indicates that register 0 has been preloaded with the address of the work area.

Omission of the second positional parameter indicates that the user has chosen processing either by means of a register (IOREG keyword parameter) or by directly accessing the data relative to the name of the I/O area.

When the second positional parameter is specified, WORKA=YES must also be specified in the DTFPR declarative macro.

DTFPR

Function:

→ Defines printer files and creates a file table of varying length (Table A—6). Keyword parameters are summarized in Table A—7; variant forms in Appendix F. Appendix K treats specification of the *load code buffer* and *vertical format buffer*.

Format:

LABEL	Δ OPERATION Δ	OPERAND
filename	DTFPR	[BLKSIZE=n] [,CONTROL=YES] [,CTLCHR=DI] [,ERROR=symbol] ,IOAREA1=symbol [,IOAREA2=symbol] [,IOREG=(r)] [,OPTION=YES] [,PRAD=n] [,PRINTOV= { SKIP } { symbol } { YES }] [,RECFORM= { FIXUNB } { UNDEF } { VARUNB }] [,RECSIZE=(r)] [,SAVAREA=symbol] [,UCS= { OFF } { ON }] [,WORKA=YES]

Parameters:

BLKSIZE=n

Specifies the length of the I/O area in bytes. If the record format is variable length or undefined, *n* specifies the length of the longest block, including block size and record size bytes for variable-length unblocked records. (See Appendix G.)

If omitted, the block size (120, 121, 128, or 129) is determined from the CTLCHR and RECFORM keyword parameters. The minimum block size is two bytes; maximums depend upon installation of *print position options* and printer type.

With RECFORM=FIXUNB specified and CTLCHR not specified, block size is set to 120. With RECFORM=VARUNB and CTLCHR specified, block size is set to 129.

DTFPR**CONTROL=YES**

Specified if spacing or skipping lines on the printer is controlled by the user's program through the CNTRL macro instruction.

The CONTROL keyword parameter and the CTLCHR keyword parameter are mutually exclusive. If they are both used in the same DTFPR, an error flag appears in the output listing and the control specification is ignored.

CTLCHR=DI

Specifies the device-independent, 2-digit, hexadecimal *control character* code (listed in Table A-4) which is to be used with data records.

The use of device-independent characters allows a single character for each function to be used with any printer, even if the hardware opcode varies with printer type. With this set of characters, some substitutions have to be made to compensate for printer device characteristics (Table A-8).

ERROR=symbol

Specifies the address of a special error handling routine to which the user may have control transferred when a fatal hardware or detectable logic error occurs on his file. Information concerning reasons for the error is contained in *filenameC*.

IOAREA1=symbol

Defines the address of the I/O area. Each input or output file must have an area reserved for its individual use. The I/O area must be aligned so that the first byte of data (the character to be printed in column 1) is on a half-word boundary. Refer to Table A-1, Note 2.

IOAREA2=symbol

Provides a second I/O area to allow overlapped processing and to speed I/O operations. Refer to Table A-1, Note 1.

The same conditions that apply for IOAREA1 also apply for IOAREA2. If IOAREA2 is specified and WORKA is not, the IOREG keyword must be specified.

IOREG=(r)

Specified when a general register (2 through 12) is used to reference current data. If SAVAREA is specified, register 13 is also available. The register must be specified if two output areas are used and records are not to be processed in a work area. Refer to Table A-1.

The IOREG and WORKA keyword parameters are mutually exclusive. If both are specified, the WORKA keyword parameter is ignored, and a diagnostic appears in the DTF listing.

DTFPR

OPTION=YES

Allows the user to specify an *optional file*, one which he anticipates is not invariably required to be printed every time the program is executed.

When the OPTION keyword parameter is used, the PUT, CNTRL, and PRTOV imperative macro instructions are disabled:

- if an OPT positional parameter is included in the DVC job control statement and the device is not available at execution time; or
- when no device is assigned to the file by user's job control statements (i.e., no DVC-LFD sequence).

If the OPTION keyword parameter is used under these conditions, the occurrence of a PUT, CNTRL, or PRTOV macro instruction results in a branch back to the user's program, and no I/O is performed.

If the OPTION keyword parameter is not specified and one of the two previously stated conditions exists, the file is not opened; data management sets the *error in OPEN* flag in *filenameC* and branches to the user's error routine. If the user has not provided an error routine, control returns inline.

PRAD=n

Allows the user to specify a standard form advance of 1 to 15 lines. The form advance takes place after the line is printed. If the PRAD and CTLCHR keyword parameters are not specified, PRAD=1 is assumed; if both are specified, the control character will determine the line advancement.

A delayed CNTRL macro instruction, which spaces or skips lines after printing, overrides the PRAD keyword parameter specification for one print operation only.

PRINTOV=SKIP

Specifies an automatic skip to the *home paper* position when the *forms overflow* code is detected during a space or print-and-space command.

PRINTOV=symbol

Specifies that control is transferred to the user's overflow routine, where *symbol* is the address of the routine. When this option is specified, the form will not be automatically advanced to the home paper position.

PRINTOV=YES

Specifies that the PRTOV imperative macro instruction is used in the program to control overflow detection and actions.

RECFORM=FIXUNB

Fixed-length records for print files are assumed by data management when this keyword parameter is omitted.

RECFORM=UNDEF

Used for undefined records. The user must specify the RECSIZE keyword parameter when this format is used.

DTFPR**RECFORM=VARUNB**

Used for variable-length, unblocked records. The user places record length, as a binary value, in the first two bytes of the record size field.

RECSIZE=(r)

For output files with undefined record format, specifies the number of the general register that holds the size of the output record. The record size must be entered into this general register before the PUT macro instruction is issued.

SAVAREA=symbol

Specifies the address of a 72-byte labeled save area (aligned on a full-word boundary) for the contents of general registers.

This keyword parameter should be specified for each DTF in a program. Only one register save area is needed per program.

If this keyword is not present in a DTF, data management assumes that register 13 has been loaded with the address of a 72-byte save area, aligned on a full-word boundary.

UCS=OFF

Specifies that *character mismatches* are to be ignored by the program. All unprintable characters are printed as the nonprinting code (NP) in the *load code buffer*. When the standard load code is used, a blank (40_{16}) is printed. This option is assumed if the UCS keyword is not specified.

UCS=ON

Specifies that the operator is to be notified of *character mismatches*. If an error routine has been provided, control will be transferred to that routine and the registers restored. If no error routine has been provided, a message will be issued, and control will return to the program as if no error had occurred.

WORKA=YES

Specifies use of a work area for preparation of output records. The address of the current work area must be specified with each PUT macro instruction, using the multiple-parameter form of the macro.

The WORKA and IOREG keyword parameters are mutually exclusive; if both are specified, the WORKA keyword is ignored and a diagnostic appears in the DTFPR macro expansion. Refer to Table A-1.



4. Magnetic Tape File Data Management



CLOSE

Function:

Initiates termination procedures for one or more files.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	CLOSE.	{ filename-1 [, ..., filename-n] } (1) 1 *ALL

Parameters:

filename

Is the symbolic address (label) in the user's program of the DTFMT declarative macro. Each *filename* parameter contains from one to seven alphanumeric characters; the first character is alphabetic. The notations *filename-1*, *filename-n*, represent a series of files that may be specified. The user may have 1 or as many as 16 files terminated by the same CLOSE macro.

(1) or 1

Indicates that the user has preloaded general register 1 with the address of the DTFMT file table. This form is used only when there is but one file to terminate.

***ALL**

Specifies that all files open in the job step are to be closed.

CNTRL

Function:

Issues control commands to the magnetic tape subsystem.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	CNTRL	{ filename } (1) 1 ,code

Parameters:

filename

Is the label of the corresponding DTFMT declarative macro instruction in the user's program. Parameter *filename* may contain a maximum of seven characters.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction.

code

Is the mnemonic code of the command to be performed:

<u>Code</u>	<u>Command</u>
BSF	Backspace to tape mark.
BSR	Backspace to interrecord gap.
ERG	Erase gap (writes 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.

FEOV

Function:

Initiates *end-of-volume procedures* on an input or output tape file when the user wishes to discontinue processing the current volume and begin processing the next.

To *extend* a multivolume output file automatically, the user may issue a PUT macro to the file after the FEOV macro that follows the end of his processing for the last of the volumes he initially specified to job control. Refer to the Glossary entry *extend* (6).

Format:

LABEL	△ OPERATION △	OPERAND
[name]	FEOV	{ filename (1) 1 }

Parameters:

filename

Is the label of the corresponding DTFMT declarative macro instruction in the user's program. Parameter *filename* may contain a maximum of seven characters.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction.

GET

Function:

Causes the next logical record in an input file to become available to the user, either in the current I/O area or in a user-specified work area.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	GET	$\left\{ \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} \left[\left\{ \begin{array}{c} \text{workarea} \\ (0) \\ 0 \end{array} \right\} \right]$

Parameters:

filename

Is the label of the corresponding DTFMT declarative macro instruction in the user's program. Parameter *filename* may contain a maximum of seven characters.

(1) or 1

Indicates that register 1 has been preloaded with the address of the declarative macro instruction.

workarea

Is the label of an area into which the current record is to be moved for processing. The label *workarea* may contain a maximum of seven characters. The work area itself does not require specific alignment.

(0) or 0

Indicates that register 0 has been preloaded with the address of a work area. If the second positional parameter is specified, the WORKA keyword parameter must be specified in the DTFMT declarative macro. Omission of the second positional parameter indicates that the user has chosen to process either by means of an IOREG register or by directly accessing his data relative to the name of the I/O area.

LBRET

Function:

Issued in the user's label processing routine to write or read optional user header or trailer labels and to transfer control to tape SAM. Is the only macro the user may issue in this routine. The address of his *tape user label processing* routine is specified with the LABADDR keyword in the DTFMT declarative macro.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	LBRET	{ 1 } { 2 }

Parameters:

- 1
Informs tape SAM that the user has completed label processing. For an input file, no more user labels are read; for an output file, a tape mark is written. Control returns to the OPEN or CLOSE transient in effect, not to the address of the user's LABADDR routine.
- 2
Reads another user label from an input file, or writes to an output file the label just generated by the user's LABADDR routine; control returns to the user's LABADDR routine, at the instruction immediately following the LBRET 2 macro call.

OPEN

Function:

Calls a transient routine to make certain validation checks and to initialize a DTFMT file for processing. Checks whether all required keyword parameters have been supplied and whether all keywords specified are in valid form. Determines what device allocation has been performed by job control and accesses the job control stream to obtain necessary label and volume information; inspects and validates labels in the file according to specification of FILABL keyword. Transfers control to the user's LABADDR routine for processing UHL. Refer to Table A—11 for a summary of the effects of job control statements on the OPEN transient.

Format:

LABEL	△ OPERATION △	OPERAND
→ [name]	OPEN	{ filename-1 [, ..., filename-n] } (1) 1

Parameters:

filename

Is the symbolic address (label) in the user's program of the DTFMT declarative macro. Each *filename* parameter contains from one to seven alphanumeric characters; the first character is alphabetic.

→ The notations *filename-1, ..., filename-n*, represent a series of files that may be specified; the user may have 1 or as many as 16 files opened by the same macro.

(1) or 1

Indicates that the user has preloaded general register 1 with the address of the DTFMT file table. This form is specified only when there is not more than one file to open.

PUT

Function:

Delivers an output record to tape SAM either in the current I/O area or in a user-specified work area. Once delivered, the record is no longer available to the user. Data management handles blocking automatically from the work area, but the user must block variable-length records constructed in the I/O area. Refer to VARBLD keyword parameter, TRUNC imperative macro, and Table A—1.

If the user issues a PUT macro after *end-of-volume* is detected (or forced, with the FEOV macro) on the last volume specified for a multivolume file, he may cause data management to *extend* the file automatically. Refer to the Glossary entry *extend* (6).

Format:

LABEL	△ OPERATION △	OPERAND
[name]	PUT	$\left\{ \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} \left[\left\{ \begin{array}{c} \text{workarea} \\ (0) \\ 0 \end{array} \right\} \right]$

Parameters:

filename

Is the label of the DTFMT declarative macro instruction that defines the output tape file.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFMT file table.

workarea

Is the label of the work area from which tape SAM may obtain the record.

(0) or 0

Indicates that the user has preloaded register 0 with the address of the work area.

Omission of the second positional parameter indicates that the user has chosen to reference the current record either by means of an index register (IOREG keyword parameter) or by directly accessing the data relative to the name assigned to IOAREA1. When the work area is used, WORKA=YES must be specified in the DTFMT declarative macro.

RELSE

Function:

Skips over the remaining records in the current block of an input DTFMT file. The next GET imperative macro makes the first record of the succeeding block available to the user.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	RELSE	{ filename } (1) 1

Parameters:

filename

Is the label in the user's program of the DTFMT declarative macro instruction.

(1) or 1

Indicates that the user has preloaded general register 1 with the address of the DTFMT file table.

SETF

Function:

Alters *file processing mode* for a DTFMT file for which TYPEFLE=INOUT has been specified. Issued before issuing the OPEN macro to a closed file; not issued to an open file.

Format:

LABEL	Δ OPERATION Δ	OPERAND	
[name]	SETF	$\left\{ \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\}$	$\left\{ \begin{array}{c} \text{INPUT} \\ \text{OUTPUT} \end{array} \right\}$

Parameters:

filename

Is the label of the DTFMT declarative macro instruction that defines the in/out magnetic tape file.

(1) or 1

Indicates that the user has preloaded general register 1 with the address of the DTFMT file table.

INPUT

Specifies that the file processing mode for this in/out file is to be changed to input when it is opened.

OUTPUT

Specifies that the file processing mode for this in/out file is to be changed to output when it is opened.

TRUNC

Function:

Writes a short block of data to an output DTFMT file. Must be issued when the user determines from the VARBLD register that the current variable-length record will not fit into *residual space* remaining in the current I/O area. May also be used with fixed-length blocked records. Resets the IOREG index register to point to the address of the next available I/O area. The subsequent PUT macro starts off the next block with the current record. Refer to Table A—1.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	TRUNC	{ filename } (1) 1

Parameters:

filename

Is the label of the DTFMT declarative macro defining the output tape file.

(1) or 1

Indicates that the user has preloaded general register 1 with the address of the DTFMT file table.

DTFMT

Function:

Defines magnetic tape files that may be processed with the imperative macros. Refer to Table A—9. Creates a 258-byte DTFMT file table, unvarying in size and aligned on a double-word boundary. Table A—10 summarizes the DTFMT keyword parameters; their variant forms are given in Appendix F.

Format:

LABEL	△ OPERATION △	OPERAND
filename	DTFMT	[ASCII=YES] [,BKNO=YES] [,BLKSIZE=n] [,BUFOFF=n] [,CKPTREC=YES] [,CLRW= { NORWD } { RWD }] [,EOFADDR=symbol] [,ERROPT= { IGNORE } { SKIP }] [,ERROR=symbol] [,FILABL= { NO { NSTD { STD } }] ,IOAREA1=symbol [,IOAREA2=symbol] [,IOREG=(r)] [,LABADDR=symbol] [,LENCHK=YES] [,OPRW=NORWD] [,OPTION=YES] [,READ= { BACK { FORWARD }] [,RECFORM= { FIXBLK { FIXUNB { UNDEF { VARBLK { VARUNB } }] [,RECSIZE= { n { (r) }] [,REWIND= { NORWD { UNLOAD }] [,SAVAREA=symbol] [,TPMARK=NO] [,TYPEFLE= { INOUT { INPUT { OUTPUT }] [,VARBLD=(r)] [,WORKA=YES]

DTFMT

Parameters:

ASCII=YES

Required when *ASCII* files are to be processed; the user must ensure that automatic or specific inclusion of the 516-byte *ASCII/EBCDIC translation table* module provided by OS/3 is accomplished at link time. If specified and non-ASCII labels or data are present on tape, data management issues error message DM55 and branches to user's error routine. If omitted, data management assumes EBCDIC processing mode; if ASCII labels are present on tape, the same error processing results. Refer to Appendix B.

ASCII files may not be processed on magnetic tape units fitted with 7-track read/write heads. Refer to Appendix C.

BKNO=YES

Required to cause data management to accept 3-byte *block number* prefixes on EBCDIC input or output files (or 1-byte block sequence indicators on ASCII files). If specified, user must reserve a 4-byte storage area, fullword-aligned, preceding each I/O buffer used. Optional user labels must be 83 bytes long in EBCDIC files and 81 bytes long in ASCII files to ensure correct processing.

BLKSIZE=n

Specifies the length of the I/O buffer in bytes, where *n* is this decimal number. The minimum block size is 18 bytes; buffer length must accommodate user labels, if these are to be processed, and the size of the largest block of variable-length, blocked records. If no BLKSIZE specification is made, and data management assumes a 256-byte block length. Maximum block size for the UNISERVO VI-C and UNISERVO 10 Magnetic Tape Subsystems is 8191 bytes; the maximum block size set by OS/3 for other magnetic tape subsystems supported is 65,535 bytes. The user's BLKSIZE specification must not exceed these limits and must be four bytes less than the maximum for the device if *block numbering* is specified.

For optional user labels in a magnetic tape file residing on a 9-track tape device or on a 7-track device with the CONVERT feature *off*, the I/O area must be at least 80 bytes long. For a 7-track device with the CONVERT feature *on*, the block size specification must be a multiple of 3 and not less than 81 bytes. Refer to Appendix C.

BUFOFF=n

For ASCII files only; *n* specifies the length of a block prefix in bytes. The value of *n* may range from 0 through 99. For input files, only the value 4 is significant and is used when variable-length records are retrieved. For output files, the value of *n* should be 0 for undefined records and for fixed-length records, blocked or unblocked. It may be 0 or 4 for variable-length records, blocked or unblocked. If the BUFOFF keyword is specified, the ASCII keyword must also be specified; ignored for EBCDIC files. Data management assumes BUFOFF=0 if this keyword is omitted.

CKPTREC=YES

Specifies that *checkpoint* blocks are to be bypassed on input EBCDIC files. If this keyword is omitted, checkpoint records are read. Ignored if specified for output files. Not valid for ASCII files.

CLRW=NORWD

Specifies that tape is not to be rewound when the file is closed.

DTFMT**CLRW=RWD**

Specifies that tape is to be rewound without interlock and not unloaded when the file is closed.

If the CLRW keyword is not specified, the tape is rewound with interlock and unloaded when the file is closed. If the CLRW keyword is specified, the REWIND keyword must not also be specified.

EOFADDR=symbol

Specifies address of user's routine for processing *end-of-data*, to which data management transfers control on sensing the tape mark that follows the last block of input data.

ERROPT=IGNORE

Specifies that the user will process an input or output block, on which a parity error occurs, as if no error has occurred.

ERROPT=SKIP

Specifies that an input block on which a parity error occurs is to be bypassed and not made available to the user. An output block on which a parity error occurs is ignored as if it were written correctly.

ERROR=symbol

Specifies the address of the user's error handling routine, to which data management transfers control when an error is detected. If the ERROR keyword is omitted, errors cause control to be returned to the user inline, following the imperative macro that initiated the error condition. Refer to Appendix B.

FILABL=NO

Specifies that the EBCDIC file is unlabeled; invalid for an ASCII file.

FILABL=NSTD

Specifies that an EBCDIC file contains nonstandard labels; invalid for an ASCII file. The user must also specify the LABADDR keyword parameter to create or check labels. If nonstandard labels are present on an input file, but are not to be checked, the user omits the LABADDR keyword parameter; in this event, a tape mark must precede his data blocks.

FILABL=STD

Specifies that the ASCII or EBCDIC file contains standard labels (described in Appendix F). If his file contains optional *tape user labels*, the user must specify the address of a label processing routine with the LABADDR keyword. User labels are processed in the I/O area whose address is contained in register 1.

IOAREA1=symbol

Always required; specifies the address of the I/O area for this file. The I/O area must be half-word aligned. User labels are processed in the I/O area whose address is contained in register 1. When the user specifies the LABADDR keyword, the length of the I/O area must not be less than 80 bytes.

IOAREA2=symbol

Specifies the address of an optional secondary I/O area for this file; subject to the same requirements as the IOAREA1 buffer. If the IOAREA2 keyword is specified, either the WORKA or the IOREG keyword must also be specified.

DTFMT

IOREG=(r)

Specifies the number general register to be used as the index register. Value of *r* which must be enclosed in parentheses, may range from 2 through 12, and general register 13 may also be specified if the SAVAREA keyword is specified. Required when two I/O areas are used, unless the user also specifies the WORKA keyword. Required when records are blocked, or when read-backward processing of variable-length or undefined records is desired.

LABADDR=symbol

Specifies the address of the user's *tape label processing* routine for optional user labels, standard or nonstandard. Required for output files if user labels are to be created, and for input files if they are to be read. If user labels on an input file are to be bypassed, the user omits the LABADDR keyword. When this keyword is specified, the length of the I/O area must not be less than 80 bytes; user labels are processed in the I/O area whose address is contained in register 1.

LENCHK=YES

Specifies, for ASCII input files, that data management is to check the block length specified in the *buffer offset field* of variable-length records against their physical record length. Requires that the user also specify the ASCII and BUFOFF keywords. If a discrepancy is detected, data management issues error message DM25 and branches to the user's error routine. Refer to Appendix B. For ASCII output files, the block length will be inserted in the block prefix of variable-length records only if BUFOFF=4 is specified.

OPRW=NORWD

Specifies that reels of this file are not to be rewound before labels are checked during the processing of an OPEN macro instruction. Omission causes reels to be rewound at open time. Data management assumes OPRW=NORWD is specified if the user specifies READ=BACK. If the OPRW keyword is specified, the REWIND keyword must not be specified.

OPTION=YES

Specifies that this input or output file is an optional file that may not be present for every execution of the user's program. If the OPTION keyword is specified for an input file that is not allocated to a device with necessary job control statements, the OPEN transient transfers control to user's EOFADDR routine after execution of the first GET imperative macro. The user should close the optional file. For an output file, the mechanism of the PUT macro is disabled. If the OPTION keyword is not specified and the file is not allocated to a device at open time, the OPEN transient transfers control to the user's error handling routine or to him inline (Appendix B). Refer also to the OPT keyword parameter of the DVC job control statement.

READ=BACK

Specifies that this input file is to be read backward; invalid for a multivolume file. If specified, data management assumes that the OPRW=NORWD keyword is also specified. When using this parameter, ensure that your blocksize specification accommodates the largest block on your tape.

READ=FORWARD

Specifies that this input file is to be read forward.

RECFORM=FIXBLK

Specifies that records are fixed-length and blocked. The user must specify the RECSIZE keyword and either the IOREG or the WORKA keyword.

DTFMT**RECFORM=FIXUNB**

Specifies that records are fixed-length and unblocked. The RECSIZE keyword is not specified.

RECFORM=UNDEF

Specifies that the record format is undefined. The RECSIZE keyword is required for both output and input files.

RECFORM=VARBLK

Specifies that records are variable-length and blocked. The user must also specify either the IOREG keyword or the WORKA keyword. Refer to the VARBLD keyword and the TRUNC imperative macro.

RECFORM=VARUNB

Specifies that records are variable-length and unblocked. For read-backward processing, the user must also specify either the IOREG or the WORKA keyword parameter.

RECSIZE=n

Required for fixed-length, blocked records; *n* specifies the decimal number of bytes in records.

For variable-length records, data management expects to find record size contained in the first two bytes of each record; if omitted for fixed-length, unblocked records, data management assumes that record size equals block size.

RECSIZE=(r)

Required for both input and output files containing undefined records. The completion, *r*, must be enclosed in parentheses and specifies the number of the general register into which the user loads the block length of each undefined record. Registers 2 through 12 are always available for specification as the RECSIZE register; register 13 may also be used if the SAVAREA keyword is specified.

REWIND=NORWD

Specifies that the reel is not to be rewound when the file is opened, and that the file is to be positioned between two tape marks when it is closed. If specified, neither the CLRW nor the OPRW keywords should be specified. If omitted, and neither the CLRW nor the OPRW keyword is specified, tape SAM rewinds the reel to load point on open and rewinds with interlock when the file is closed or when *end-of-volume* condition is reached.

REWIND=UNLOAD

Specifies that reel is to be rewound to load point when file is opened, and that it is to be rewound with interlock for unloading when file is closed or when an *end-of-volume* condition is encountered.

SAVAREA=symbol

Specifies the address of a 72-byte labeled save area for the contents of general registers, full-word aligned. If this keyword is omitted, data management assumes that the user has preloaded register 13 with the address of a 72-byte labeled save area, full-word aligned, before issuing an imperative macro to the file.

DTFMT

TPMARK=NO

For nonstandard labeled or unlabeled output files, specifies that data management is not to write the tape mark that normally separates header labels from data. Ignored for input files. If specified for a nonstandard labeled output file, differentiating between labels and data on input is the user's responsibility. Not specified to bypass labels on a nonstandard labeled input file nor if READ=BACK is specified for unlabeled or nonstandard labeled files.

TYPEFLE=INOUT

Specifies that this file is used for either input or output. Data management opens this file for output if the user specifies READ=FORWARD or omits the READ keyword parameter. If the user specifies READ=BACK, the file is opened for input. File processing mode may be altered to input or output by issuing the SETF imperative macro to the file before opening it.

TYPEFLE=INPUT

Specifies that this file is an input file, to be read.

TYPEFLE=OUTPUT

Specifies that this file is an output file, to be written. If OUTPUT is specified, READ=BACK should not be specified.

VARBLD=(r)

Required for output files with variable-length, blocked records that the user processes in an I/O area without specifying a work area. The completion, *r*, which must be enclosed in parentheses, specifies the number of the general register that data management loads with the number of bytes of *residual space* in the current I/O area. Refer to the TRUNC imperative macro.

WORKA=YES

Specifies that the user is processing records in a work area, rather than in an I/O area. The user must specify the address of the current work area with each GET or PUT macro issued. If the WORKA keyword is specified, the IOREG keyword must not be specified.

5. Disc Sequential Access Method

CLOSE

Function:

Initiates termination procedures for a DTFSD file.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	CLOSE	{ filename-1, [..., filename-n] } (1) 1 *ALL

Parameters:

filename

Is the label of the DTFSD declarative macro in the user's program; there may be 1 or as many as 16 files named.

(1) or 1

Specifies that register 1 has been preloaded with the address of the DTFSD file table when there is a single file to terminate.

***ALL**

Specifies that all files open in the job step are to be closed.

CNTRL

Function:

Issues a seek command to position the disc head to the current track of a DTFSD file.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	CNTRL	{ filename (1) 1 } ,SEEK

Parameters:

filename

Is the label of the DTFSD declarative macro in the user's program.

(1) or 1

Specifies that register 1 has been preloaded with the address of the DTFSD file table.

SEEK

Issues a seek command to the disc head, positioning it to the current track of the DTFSD file.

FEOV

Function:

Initiates *end-of-volume* procedures on the current volume of a DTFSD file. The next GET or PUT macro continues processing on the next sequential volume.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	FEOV	{ filename } (1) 1

Parameters:**filename**

Is the label of the corresponding DTFSD macro instruction in the user's program.

(1) or 1

Indicates that register 1 has been preloaded with the address of the DTFSD file table.

GET

Function:

Makes the next logical record of a DTFSD input file available to the user in the current I/O area, or in a work area specified by the user. Refer to Table A—1.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	GET	$\left\{ \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} \left[, \left\{ \begin{array}{c} \text{workarea} \\ (0) \\ 0 \end{array} \right\} \right]$

Parameters:

filename

Specifies the label of the DTFSD declarative macro instruction.

(1) or 1

Indicates that register 1 has been preloaded with the address of the DTFSD file table.

workarea

Is the label of an area into which data management moves the current record for the user to process. A different work area may be used for each GET macro.

(0) or 0

Indicates that register 0 has been preloaded with the address of a work area.

The second positional parameter is omitted when the user is not processing in a work area. If it is specified, WORKA=YES must be specified in the DTFSD declarative macro.

LBRET

Function:

Issued in the user's *label processing* routine to create, retrieve, or retrieve and update optional user header or trailer labels and to transfer control. Is the only macro the user may issue in this routine. The address of the label processing routine is specified with the LABADDR keyword parameter in the DTFSD declarative macro.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	LBRET	{ 1 } { 2 } { 3 }

Parameters:

- 1
Returns control to the user program at the instruction next after the OPEN or CLOSE macro call. Does not read or write a label.
- 2
Reads or writes one label, depending on the type of processing specified by the TYPEFLE keyword parameter or specification of the SETF imperative macro, and returns control to the user inline, at the next instruction after the LBRET 2 macro call.
- 3
Writes back to disc the label just read (or read and updated) and returns control to the user inline, at the next instruction after the LBRET 3 macro call. TYPEFLE=INOUT must be specified in the DTFSD declarative macro, or file processing direction must be reset to update with the SETF imperative.

TYPEFLE=INOUT must be specified in your DTF declarative macro instruction, or the file processing direction must be reset for update processing with the SETF macro instruction.

OPEN

Function:

Calls a transient routine to make certain validation checks and initialize a DTFSD file for processing. Checks whether all required keyword parameters have been supplied and whether all keywords specified are in valid form. Validates or creates system standard labels and user header labels if present. Transfers control to the user's LABADDR routine for processing UHL.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	OPEN	{ filename-1 [, ..., filename-n] (1) 1 }

Parameters:

filename

Is the label of one or more corresponding DTFSD declarative macro instructions in the user's program. The user may have as many as 16 files opened.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFSD file table; used when a single file is to be initialized.

PUT

Function:

Issued to create, extend, or update DTFSD files. Delivers an output record to data management either in the current I/O area or in a user-specified work area. Once delivered, the record is no longer available to the user. Data management handles blocking automatically from the work area, but the user must block variable-length records constructed in the I/O area. Refer to VARBLD keyword parameter, TRUNC imperative macro, and Table A—1.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	PUT	{ filename } [{ workarea }] (1) (0) 1 0

Parameters:

filename

Is the label of the DTFSD declarative macro instruction that defines the output file.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFSD file table.

workarea

Is the label of the work area from which the output record may be obtained.

(0) or 0

Indicates that register 0 has been preloaded with the address of the work area.

Omission of the second positional parameter indicates that the user has chosen to reference the current record either by means of an index register (IOREG keyword parameter) or by directly accessing the data relative to the name assigned to IOAREA1. The latter method is used only for unblocked records processed in a single I/O area. When the work area is used, WORKA=YES must be specified in the DTFSD declarative macro.

RELSE

Function:

Skips over remaining records in the current block of an input DTFSD file. The next GET imperative macro makes the first record of the succeeding block available to the user.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	RELSE	{ filename } (1) 1

Parameters:**filename**

Is the label in the user's program of the DTFSD declarative macro instruction.

(1) or 1

Indicates that the user has preloaded general register 1 with the address of the DTFSD file table.

SETF

Function:

Issued between the CLOSE and OPEN imperative macros to change the *file processing mode* for a DTFSD file for which TYPEFLE=INOUT has been specified.

Format:

LABEL	Δ OPERATION Δ	OPERAND	
[name]	SETF	{ filename (1) 1 }	{ INPUT OUTPUT UPDATE }

Parameters:

filename

Is the label of the DTFSD macro instruction which defines the INOUT file.

(1) or 1

Indicates that register 1 has been preloaded with the address of the DTFSD file table describing the INOUT file.

INPUT

Indicates that the INOUT file is to be set for input processing, without updating.

OUTPUT

Indicates that the INOUT file is to be set for output processing.

UPDATE

Indicates that the INOUT file is to be set for input processing, with updating.

TRUNC

Function:

Writes a short block of variable-length records to an output DTFSD file. Issued when the user determines from the VARBLD register that the current record will not fit into *residual space* remaining in the current I/O area. Resets IOREG index register to point to the address of the next available I/O area. The subsequent PUT macro starts off the next block with the current record. Refer to Table A—1.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	TRUNC	{ filename } (1) 1

Parameters:

filename

Is the label of the corresponding DTFSD macro instruction in the user's program.

(1) or 1

Indicates that register 1 has been preloaded with the address of the DTFSD file table.

DTFSD

Function:

Defines sequential disc files and creates a 242-byte file table. DTFSD keyword parameters are summarized in Table A-12; variant forms in Appendix F. ←

Format:

LABEL	△ OPERATION △	OPERAND
filename	DTFSD	<pre> [ACCESS= { EXC EXCR SRD SRDO }] BLKSIZE=n ,EOFADDR=symbol [,ERROPT= { IGNORE } SKIP] [,ERROR=symbol ,IOAREA1=symbol [,IOAREA2=symbol] [,IOREG=(r)] [,LABADDR=symbol] [,LACE=n] [,LOCK=NO] [,OPTION=YES] [,RECFORM= { FIXBLK FIXUNB VARBLK VARUNB }] [,RECSIZE=n] [,SAVAREA=symbol] [,TRLBL=YES] [,TYPEFLE= { INOUT INPUT OUTPUT }] [,UPDATE=YES] [,VARBLD=(r)] [,VERIFY=YES] [,WORKA=YES] </pre>

DTFSD

Parameters:

ACCESS=EXC

Specifies the exclusive read/update/add use of the file, no other jobs can access the file while it is being used.

NOTE:

This specification is equivalent to the default value for the LOCK parameter; that is, LOCK=NO is omitted.

ACCESS=EXCR

Specifies read/update/add use of the file and also allows other jobs to read from the file while it is being used.

ACCESS=SRD

Specifies that only the read function is allowed for the file and allows other jobs read/update/add use of the file.

ACCESS=SRDO

Specifies that only the read function is allowed for the file and also allows other jobs to read from the file. Writing to the file is not allowed from either the job associated with this DTF or from other jobs.

The ACCESS parameter should not be specified on the same DTF with a LOCK parameter; however, if both the ACCESS and LOCK parameters are specified on the same DTF, the ACCESS parameter specification overrides the LOCK parameter specification.

BLKSIZE=n

Specifies the length of a *block*, where *n* is the maximum number of bytes to be transferred into or out of main storage by a single access. The value of *n* includes the number of bytes of data and, for variable-length record formats, must also accommodate the largest logical record or block, including the additional bytes required for the *BDW* and the *RDWs*. The block size specified by the BLKSIZE keyword may not exceed the *disc track size* of the subsystem on which the file resides.

EOFADDR=symbol

Specifies the address of a routine the user has coded to handle end-of-data for a DTFSD input file, including extension beyond the end-of-data record, where *symbol* is the symbolic address to which data management transfers control on sensing end-of-data.

ERROPT=IGNORE

Specifies that data management is to make the current block or record, in which a parity error is detected, available to the user in the I/O area as if no parity error had occurred.

ERROPT=SKIP

Specifies that data management is to bypass or skip over an input block or logical record containing a parity error which it does not make available to the user for processing. For output records, data management ignores the block or record as if it were written correctly.

DTFSD**ERROR=symbol**

Specifies the address of the user's error handling routine, to which data management transfers control upon detecting an error/status condition, where *symbol* is the symbolic address of this routine.

If the ERROR keyword is omitted, data management returns control to the user's program inline, at the next instruction after the imperative macro that initiated transfer of control to his error routine.

IOAREA1=symbol

Specifies the location of the I/O area, where *symbol* (label) is the address. Must always be half-word aligned. Length of the I/O area is the length of the space reserved for it with the BAL *define storage* (DS) statement and may equal or exceed the BLKSIZE specification; it must never be less. For output files, an additional 8-byte area must be reserved immediately preceding the I/O area for data management use; this is not included in the length of the I/O area, nor in the BLKSIZE specification. For device-independence, or when the file resides on a fixed-sector disc, the length of the I/O area should be a multiple of 256 bytes.

IOAREA2=symbol

Specifies the location of an optional secondary I/O area, where *symbol* (label) is the address of the area. Subject to same requirements as IOAREA1.

IOREG=(r)

Specifies the general register to be used as an index register to reference current data for I/O operations, where *r* is the number of the register. Registers 2 through 12 are available, plus register 13 when the SAVAREA keyword is specified. The IOREG keyword must be specified for DTFSD files when records are blocked or two I/O areas are used. For input files, data management loads the index register with the address of the next available record or block. For output files, data management loads the register with the address of the next available I/O area; the register is reset by the issue of a TRUNC macro. Refer to Table A—1.

LABADDR=symbol

Specifies the address of the user's routine for processing optional user standard header and trailer labels, where *symbol* (label) is the address. The user's label processing routine may issue the LBRET imperative macro, but no other. If the user will be processing UTL on closing the file, he must also specify the TRLBL keyword parameter in the DTFSD declarative macro.

LACE=n

Specifies the *lace factor*, a decimal integer, for data management use in applying the record *interlace* technique to sequentially processed input or output files defined by the DTFSD declarative macro. Is ignored when the KEYLEN keyword parameter is specified. Appendix H contains formulas for calculating the lace factor.

LOCK=NO

Is equivalent to specifying ACCESS=SRDO. It should not be used in the same DTF as the ACCESS keyword parameter. If it is, the ACCESS keyword parameter will override it.

Omitting both LOCK=NO and the ACCESS keyword parameter is equivalent to specifying ACCESS=EXC.



DTFSD

OPTION=YES

Specifies that the sequentially processed input file defined by this DTFSD declarative macro is an optional file: one that the user anticipates will not be invariably present for every program execution. When specified for a file not allocated to a device by job control DVC statement, transfers control to the user's EOFADDR routine on first issue of the GET macro.

RECFORM=FIXBLK

Specifies that records are fixed length and blocked.

RECFORM=FIXUNB

Specifies that records are fixed length and unblocked. Data management assumes this format if the user omits the RECFORM keyword parameter from the DTF.

RECFORM=VARBLK

Specifies that records are variable length and blocked.

RECFORM=VARUNB

Specifies that records are variable length and unblocked.

RECSIZE=*n*

Specifies the length of logical records in blocked files, where *n* is the number of bytes in the record.

Data management assumes that record size equals block size (BLKSIZE keyword parameter) for fixed-length, unblocked records and expects to find record size in the *RDW* of variable-length records.

SAVAREA=*symbol*

Specifies the address of a 72-byte labeled save area for the contents of general registers, full-word aligned, where *symbol* (label) is the address. If used, must be specified in the DTF for each file the user's program will process; however, only one such save area is required per program.

If omitted, data management assumes that, before issuing an imperative macro to the file, the user has preloaded register 13 with the address of a 72-byte save area, aligned on a full-word boundary.

TRLBL=YES

Specifies that the user will process UTL when he issues the CLOSE imperative macro to this DTFSD file. The user must also specify the LABADDR keyword parameter.

If omitted, the user's label processing routine does not receive control when the file is closed.

TYPEFLE=INOUT

Specifies a file which may be used for either input or output. Opened initially for output; use SETF imperative macro for changing *file processing mode*.

DTFSD**TYPEFLE=INPUT**

Specifies a read-only DTFSD file; data management reads and checks standard labels for this file. The user may not issue an output function to this file unless he has also specified UPDATE=YES in the DTF.

TYPEFLE=OUTPUT

Specifies a DTFSD file that is to be written; data management writes standard labels for this file. This keyword controls not only the mode of *label processing* to be employed, but also the mode of record processing.

UPDATE=YES

Used with the DTFSD macro instruction defining sequentially processed files, when the user has specified TYPEFLE=INPUT or TYPEFLE=INOUT for these files. When used, specifies that sequential output function (PUT macro) may be issued to update data records in file. Unrelated to label processing.

If omitted from DTF for a sequentially processed input file (TYPEFLE=INPUT), the user may not issue an output function to the file.

VARBLD=(r)

Specifies a general register into which data management loads the number of bytes of *residual space* remaining in the I/O area after each execution of a PUT macro instruction to a sequentially processed output file containing blocked, variable-length records, where *r* is the number of the register. Value of *r* may range from 2 through 12, but register 13 may also be used if the SAVAREA keyword has been specified. Refer to Table A—1.

VERIFY=YES

Specifies that data management is to conduct a parity check of output records after writing them to disc. Parity check verification necessarily increases execution time for the PUT macro.

If omitted, no verification is performed; however, data management may detect an output parity check error by other means. The user may direct data management to take certain actions with the ERROPT keyword parameter.

WORKA=YES

Specifies to data management that the user will be processing input or output records sequentially in a work area and not in the I/O area. The user does not specify the IOREG keyword parameter when he specifies the WORKA keyword. The address of the work area is specified with each issue of the GET and PUT macros.



6. Direct Access Method



CLOSE

Function:

Initiates termination procedures for a DTFDA file.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	CLOSE	{ filename-1, [..., filename-n] } (1) 1 *ALL

Parameters:

filename

Is the label of the DTFDA declarative macro in the user's program; there may be 1 or as many as 16 files named.

(1) or 1

Specifies that the user has preloaded register 1 with the address of the DTFDA file table. Used when there is only one file to terminate.

***ALL**

Specifies that all files open in the job step are to be closed.

CNTRL

Function:

Issues a seek command to position the disc head to a new relative track address of a DTFDA file. The user loads the address into the location specified by the SEEKADR keyword of the DTFDA declarative macro.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	CNTRL	{ filename } (1) 1 } ,SEEK

Parameters:

filename

Is the label of the DTFDA declarative macro in the user's program.

(1) or 1

Specifies that register 1 has been preloaded with the address of the DTFDA file table.

SEEK

Issues a seek command to the disc head, positioning it to the new relative track address preloaded by the user into the location specified by the SEEKADR keyword of the DTFDA macro.

LBRET

Function:

Issued in the user's *label processing* routine to create, retrieve, or retrieve and update optional user header or trailer labels and to transfer control. Is the only macro the user may issue in this routine. The address of the label processing routine is specified with the LABADDR keyword in the DTFDA declarative macro.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	LBRET	$\left. \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \right\}$

Parameters:

1

Returns control to the user's program at the next instruction after the OPEN or CLOSE macro call. Does not read or write a label.

2

Reads or writes one label, depending on the type of label processing specified by the TYPEFLE keyword parameter, and returns control to the user inline, at the next instruction after the LBRET 2 macro call.

3

Writes back to disc the label just read (or read and updated) and returns control to the user inline, at the next instruction after the LBRET 3 macro call.

OPEN

Function:

Calls a transient routine to make certain validation checks and to initialize a DTFDA file for processing. Checks whether all required keyword parameters have been supplied and whether all keywords specified are in valid form. Validates or creates system standard labels and user header labels if present. Transfers control to the user's LABADDR routine for processing UHL.

Format:

LABEL	Δ OPERATION Δ	OPERAND
→ [name]	OPEN	{ filename-1 [, ..., filename-n] } (1) 1

Parameters:

filename

Is the label of one or more corresponding DTFDA declarative macro instructions in the user's program. The user may have 1 or as many as 16 files opened.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFDA file table; used only when a single file is to be initialized.

READ, ID

Function:

Reads into the I/O area the key (if a key exists) and data portion of the block from a DTFDA input file whose *relative disc address* the user has preloaded into the location specified by the SEEKADR keyword parameter of the DTF. The relative disc address specified must be greater than zero and in the form specified by the RELATIVE keyword. Requires specification of READID=YES in the DTFDA declarative macro.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	READ	{ filename } (1) 1 , ID

Parameters:

filename

Is the label in the user's program of the DTFDA declarative macro that defines the randomly processed file from which the user is retrieving data.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFDA file table.

ID

Specifies that a search is to be made for a block with an ID matching the relative disc address the user has presented to data management via the field specified in the SEEKADR keyword parameter of the DTFDA macro.

READ,KEY

Function:

Reads into the I/O area the key and data portion of the block from an input DTFDA file whose key the user has presented to data management in the location specified by the KEYARG keyword parameter of the DTFDA declarative macro. The data portion of the record is found in the I/O area at a displacement equal to the key length. The user specifies the starting point for *search-on-key* by loading the initial *relative disc address* into the location specified by the SEEKADR keyword, in the form specified by the RELATIVE keyword. Search is restricted to the initial track unless the user specifies SRCHM=YES in the DTFDA macro, in which case search continues to the end of the cylinder. Requires specification of READKEY, KEYARG, and KEYLEN parameters in the DTFDA macro.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	READ	{ filename } (1) 1 , KEY

Parameters:

filename

Is the label of the DTFDA declarative macro in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFDA file table.

KEY

Specifies that data management is to search for a block with a key matching the key the user has loaded into the location specified by the KEYARG keyword. Search begins at the relative address preloaded into the location specified by the SEEKADR keyword and continues to the end of the track, unless the user has specified SRCHM=YES.

WAITF

Function:

Checks the completion of data transfer initiated by the READ or WRITE macro issued to a DTFDA input or output file and sets the appropriate error or status bits in *filenameC*. It is also the function of the WAITF macro to set the address of the record written, or of the next available record, in the location specified by the IDLOC keyword parameter. This is done after the WAITF return; the form in which this relative address is returned is governed by the specification of the RELATIVE keyword parameter. Refer to Tale A-18.

The user must issue the WAITF macro following each READ or WRITE macro, before issuing a CNTRL or another READ or WRITE macro to the same DTFDA file. If omitted when required, data management sets the *WAITF required* error flag (byte 0, bit 7) and the *invalid macro sequence* error flag (byte 0, bit 6) in *filenameC* and issues error message DM14. Refer to Appendix B. Not required after CNTRL macro.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	WAITF	{ filename } (1) 1

Parameters:

filename

Is the label of the DTFDA declarative macro in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFDA file table.

WRITE,AFTER

Function:

Writes the current block from main storage as the next sequential block on the current track of a DTFDA file. If a file resides on a *variable-sector disc*, the remainder of the track is overwritten with binary 0's. Does not clear remainder of the track on a *fixed-sector disc*. Requires specification of the AFTER keyword parameter in the DTFDA declarative macro.

If the current block, when written, occupies the last position on the track, the following WAITF macro sets the *last block on track accessed* flag (byte 0, bit 0) in *filenameC*.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	WRITE	$\left\{ \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} ,\text{AFTER}$

Parameters:

filename

Is the label in the user's program of the DTFDA declarative macro.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFDA file table.

AFTER

Specifies that the current block in main storage is to be written as the next sequential block on the current track of the DTFDA file.

WRITE,AFTER,EOF

Function:

Does not write a block to disc. Records the *relative disc address* of the current block, plus 1, as a *logical end-of-file* (EOF) pointer in the DTFDA file table; data management records this in the disc *format 2 label* when the file is closed. Refer to Appendix D. Requires specification of the AFTER keyword parameter in the DTFDA declarative macro.

The WRITE,AFTER,EOF macro is not required to be issued in programs written for OS/3, as data management automatically records logical EOF on file close.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	WRITE	{ filename } (1) 1 } ,AFTER, EOF

Parameters:

filename

Is the label in the user's program of the DTFDA declarative macro.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFDA file table.

AFTER

Specifies that the relative block address of the next block position in physical sequence on the current track of this DTFDA file is to be recorded as logical EOF.

EOF

Records the *relative disc address* of the current block, plus one, as the *logical end-of-file* (EOF).

WRITE, ID

Function:

Issued to create or update a DTFDA file, with or without keys. Writes either the data portion only, or both the key and the data portion, of a block in a preformatted DTFDA file. The user loads the *relative disc address* of the block into the field specified by the SEEKADR keyword parameter of the DTFDA declarative macro; this disc address must be greater than zero. Also requires specification of WRITEID=YES in the DTFDA macro.

Searches the DTFDA file for a block with matching ID and writes or overwrites the first such block found. To update the key, the user specifies the length of the key actually on disc with the KEYLEN keyword parameter of the DTFDA macro; only the data portion of a block is written if the KEYLEN keyword is omitted.

If a search is unsuccessful, data management sets the *record not found* status flag (byte 1, bit 3) in *filenameC*. If lengths of the update key or data portions do not match those on disc, data management sets the *wrong length found* error flag (byte 1, bit 5). Refer to Appendix B.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	WRITE	{ filename } (1) 1 , ID

Parameters:

filename

Is the label in the user's program of the DTFDA declarative macro.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFDA file table.

ID

Specifies a search of the DTFDA file for a block whose ID matches that provided by the user in the location specified by the SEEKADR keyword parameter.

WRITE,KEY

Function:

Rewrites, to the original disc location in the DTFDA file, the key and data portions of the block just retrieved with the READ,KEY imperative macro. Requires specification of WRITEKEY=YES in the DTFDA declarative macro. Does not conduct a search. Not used to create a file.

Consecutive issues of the WRITE,KEY imperative macro without intervening issue of the READ,KEY macro constitute a sequence error. Data management sets the *invalid macro sequence* error flag (byte 0, bit 6) in *filenameC* and issues error message DM14. Refer to Appendix B.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	WRITE	{ filename } (1) 1 , KEY

Parameters:

filename

Is the label in the user's program of the DTFDA declarative macro.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFDA file table.

KEY

Specifies that the block just read by a READ,KEY macro is to be rewritten to its original location on disc in a DTFDA file.

WRITE,RZERO

Function:

Positions the relative block address to a new track of the DTFDA file, specified by the user in the location given by the SEEKADR keyword parameter, and initializes this track. Subsequent issue of the WRITE,AFTER imperative macro writes a block beginning at the first position on this track. Requires specification of AFTER=YES in the DTFDA declarative macro. Although the WRITE,RZERO macro does not write a block, the WAITF macro must be issued before issuing another imperative macro to the DTFDA file.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	WRITE	{ filename } (1) 1 , RZERO

Parameters:

filename

Is the label in the user's program of the DTFDA declarative macro.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFDA file table.

RZERO

Positions the relative block address to a new track of the DTFDA file, specified by the user in a location specified by the SEEKADR keyword parameter, and initializes this track.

DTFDA

Function:

Defines the direct access files and creates a 242-byte file table. DTFDA keyword parameters are summarized in Table A—13; their variant forms in Appendix F. ←

Format:

LABEL	Δ OPERATION Δ	OPERAND
filename	DTFDA	<pre> [ACCESS= { EXC EXCR SRD SRDO }] [AFTER=YES] ,BLKSIZE=n [,ERROR=symbol] [,IDLOC=symbol] ,IOAREA1=symbol [,KEYARG=symbol] [,KEYLEN=n] [,LABADDR=symbol] [,LACE=n] [,LOCK=NO] [,READID=YES] [,READKEY=YES] [,RECFORM= { FIXUNB VARUNB }] [,RELATIVE= { R T }] [,SAVAREA=symbol] ,SEEKADR=symbol [,SRCHM=YES] [,TRLBL=YES] [,TYPEFLE= { INPUT OUTPUT }] [,VERIFY=YES] [,WRITEID=YES] [,WRITEKEY=YES] </pre>

DTFDA

Parameters:

ACCESS=EXC

Specifies the exclusive read/update/add use of the file. No other jobs can access the file while it is being used.

NOTE:

This specification is equivalent to the default value for the LOCK parameter; that is, LOCK=NO is omitted.

ACCESS=EXCR

Specifies read/update/add use of the file and also allows other jobs to read from the file while it is being used.

ACCESS=SRD

Specifies that only the read function is allowed for the file and allows other jobs read/update/add use of the file.

ACCESS=SRDO

Specifies that only the read function is allowed for the file and also allows other jobs to read from the file. Writing to the file is not allowed from either the job associated with this DTF or from other jobs.

The ACCESS parameter should not be specified on the same DTF with a LOCK parameter; however, if both the ACCESS and LOCK parameters are specified on the same DTF, the ACCESS parameter specification overrides the LOCK parameter specification.

AFTER=YES

Specifies if a subsequent WRITE macro instruction contains an AFTER or an RZERO positional parameter. This keyword parameter is used only when creating or adding records, or when repositioning the disc head to a new track. The WRITE, ID macro may not be issued to a file for file creation if the AFTER keyword is specified in the DTF.

BLKSIZE=n

Specifies the length of a *block*, where *n* is the maximum number of bytes to be transferred into or out of main storage by a single access. The value of *n* includes the number of bytes of data and, for keyed blocks, the length of keys. For the variable-length record format, *n* must also accommodate the largest logical record, including the additional bytes required for the *BDW* and the *RDW*. The block size specified by the BLKSIZE keyword may not exceed the *disc track size* of the subsystem on which the file resides.

ERROR=symbol

Specifies the address of the user's error handling routine, to which data management transfers control upon detecting an error/status condition, where *symbol* is the symbolic address of this routine.

DTFDA

If specification of this optional keyword parameter is omitted, data management returns control to user's program inline, at the next instruction after the imperative macro that initiated transfer of control to his error routine.

IDLOC=symbol

Specifies the field to which data management returns the *relative disc address* (ID) after the execution of a WAITF macro, where *symbol* (label) is the address of the field. Data management assumes that size and format of the field are the same as specified in the SEEKADR keyword parameter. The form in which ID is returned is governed by specification of the RELATIVE keyword parameter. The record whose ID is returned is governed by the positional parameter used with the READ or WRITE macro. Refer to Table A—18.

IOAREA1=symbol

Specifies the location of the I/O area, where *symbol* (label) is the address. Must always be half-word aligned. The length of the I/O area is the length of the space reserved for it with the BAL *define storage* (DS) statement and may equal or exceed the BLKSIZE specification; it must never be less. For output files, an additional 8-byte area must be reserved immediately preceding the I/O area for data management use; this is not included in the length of the I/O area, nor in the BLKSIZE specification. For device-independence, or when the file resides on a fixed-sector disc, the length of the I/O area should be a multiple of 256 bytes.

KEYARG=symbol

Specifies that a search is to be made for a record having a key identical to a *search key* or argument the user provides to data management, where *symbol* (label) is the field in the user's program containing this argument. The KEYARG keyword parameter is required for DTFDA files when READ,KEY or WRITE,KEY imperative macros are issued. When the user specifies the KEYARG keyword parameter, he must also specify the length of the key, using the KEYLEN keyword parameter.

KEYLEN=n

Specifies the length of keys in DTFDA files, where *n* is the number of bytes in the keys. All keys in the same file must have the same length; the *key length* may range from 3 bytes minimum to 255 bytes maximum.

LABADDR=symbol

Specifies the address of the user's routine for processing optional standard user header and trailer labels, where *symbol* (label) is the address.

The user's *label processing* routine may issue the LBRET imperative macro only. If the user is processing UTL on closing the file, he must also specify the TRLBL keyword parameter in the DTFDA macro.

LACE=n

Specifies the *lace factor*, a decimal integer, for data management use in applying the record interlace technique to sequentially processed input or output files defined by the DTFDA macro. Is ignored when the KEYLEN keyword parameter is specified.

LOCK=NO

Is equivalent to specifying ACCESS=SRDO. It should not be used in the same DTF as the ACCESS keyword parameter. If it is, the ACCESS keyword parameter will override it.

Omitting both LOCK=NO and the ACCESS keyword parameter is equivalent to specifying ACCESS=EXC.



DTFDA

READID=YES

Specifies to data management that the user will issue a READ,ID imperative macro to the DTFDA file defined by this declarative macro. The IDLOC and SEEKADR keyword parameters are also required.

READKEY=YES

Specifies to data management that the user will issue a READ,KEY imperative macro to the DTFDA file defined by this declarative macro. The SEEKADR, KEYARG, and KEYLEN keyword parameters are also required.

RECFORM=FIXUNB

Specifies that records are fixed length and unblocked. Data management assumes this format if the user omits the RECFORM keyword parameter from the DTFDA macro.

RECFORM=VARUNB

Specifies that records are variable length and unblocked.

RELATIVE=R

Specifies that the *relative disc address* (ID) given in the location defined by the SEEKADR keyword parameter is a relative record address.

RELATIVE=T

Specifies that the *relative disc address* (ID) given in the location defined by the SEEKADR keyword parameter is a relative track address.

SAVAREA=symbol

Specifies the address of a 72-byte labeled save area for the contents of general registers, full-word aligned, where *symbol* (label) is the address. If used, must be specified in the DTF for each file the user program will process; however, only one such save area is required per program.

If omitted, data management assumes that the user has preloaded register 13 with the address of a 72-byte save area, aligned on a full-word boundary.

When data management is used as a shared code element, register 13 must point to a register save area. This save area is used by the SEXTRN processor and (if the SAVAREA keyword parameter has not been specified) by data management. When the SAVAREA keyword parameter is used, the SEXTRN processor must be supplied with the address of a save area in register 13. This address may be the same area specified by the SAVAREA keyword parameter.

SEEKADR=symbol

Specifies the 4-byte location in the user's program into which he will load the *relative disc address* for use in processing direct access files with the READ,ID; READ,KEY; WRITE,ID; WRITE,RZERO; and CNTRL imperative macros. Required for DTFDA files. The form in which the address is loaded is governed by the user's specification of the RELATIVE keyword parameter.

SRCHM=YES

Specifies that a *search-on-key* issued to DTFDA file (READ,KEY) is to be extended beyond the current track to the end of the cylinder.

DTFDA**TRLBL=YES**

Specifies that the user will process UTL when he issues the CLOSE imperative macro to the DTFDA file defined by the DTF. The user must also specify the LABADDR keyword parameter.

TYPEFLE=INPUT

Specifies that data management will read and check standard labels for this DTFDA file. If the TYPEFLE keyword is omitted, data management assumes that TYPEFLE=INPUT has been specified.

TYPEFLE=OUTPUT

Specifies that data management will write standard labels for this DTFDA file.

VERIFY=YES

Specifies that data management is to conduct a parity check of output records after writing them to disc. Parity check verification necessarily increases execution time for the WRITE macro.

If omitted, no verification is performed; however, data management may detect an output parity check error by other means.

WRITEID=YES

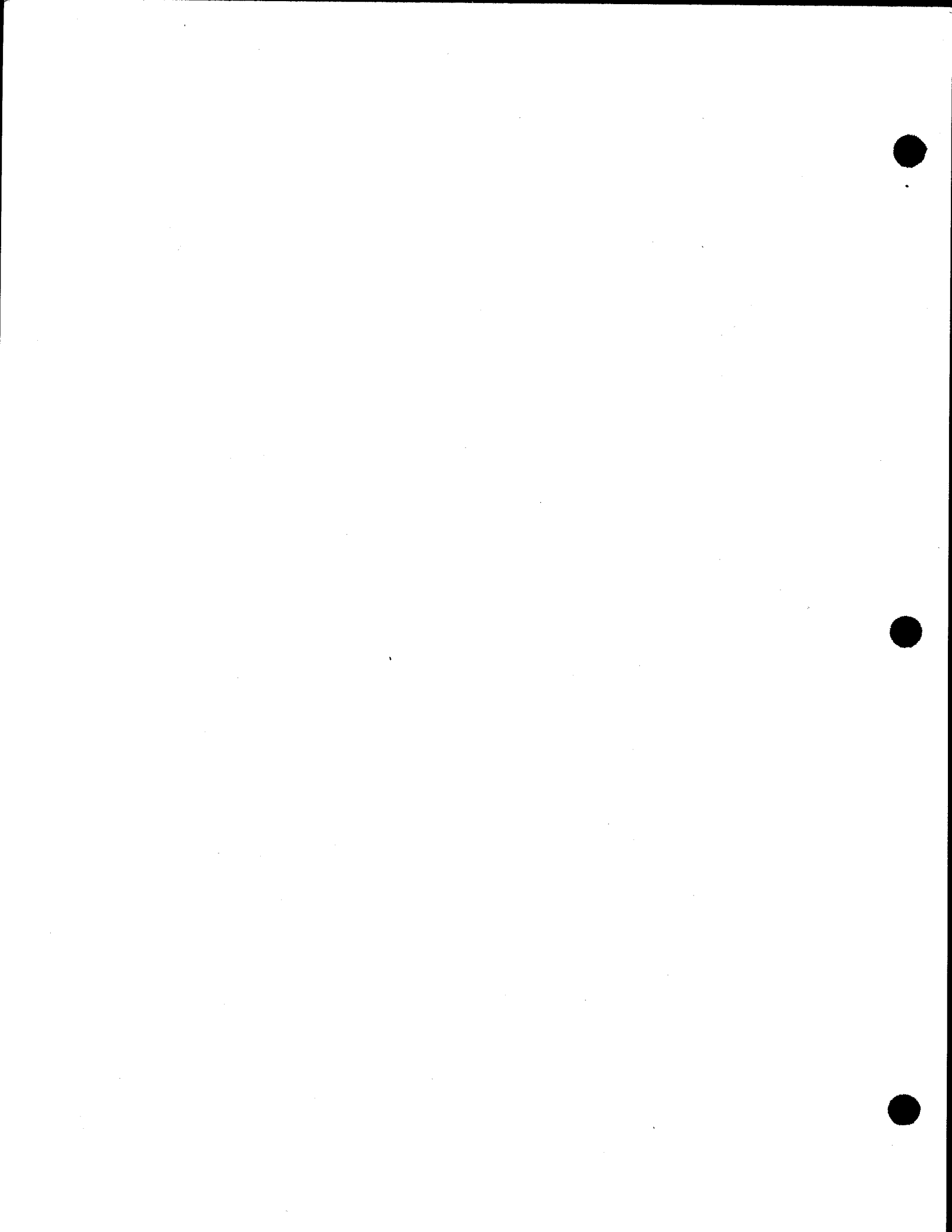
Specifies that the user will issue a WRITE,ID macro to the randomly processed file defined by this DTFDA declarative macro to locate an output record by its relative disc address (ID). The IDLOC and SEEKADR keyword parameters must also be specified.

WRITEKEY=YES

Specifies that the user will issue a WRITE,KEY imperative macro to the randomly processed file defined by this DTFDA declarative macro to rewrite a record just retrieved by the READ,KEY macro. The SEEKADR, KEYARG, and KEYLEN keyword parameters must be specified also.



7. Nonindexed Access Method for Disc



CLOSE

Function:

Initiates termination procedures for all nonindexed disc file types.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	CLOSE	{ filename-1 [...,filename-n] (1) 1 *ALL }

Parameters:

filename

Is the label of the DTF declarative macro in the user's program; there may be 1 or as many as 16 files named.

(1) or 1

Specifies that the user has preloaded register 1 with the address of the DTF file table. Used when there is only one file to terminate.

***ALL**

Specifies that all files open in the job step are to be closed.

CNTRL

Function:

Issues a seek command to position the disc head to the current track of a DTFSD file, or a new relative track address in a DTFDA or DTFNI file. The user loads the address into the location specified by the SEEKADR keyword parameter of the DTF.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	CNTRL	{filename} { (1) } { 1 } ,SEEK

Parameters:

filename

Is the address in the user's program of the DTF declarative macro.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTF file table.

SEEK

Issues a seek command to position the disc head to the current track of a DTFSD file, or to a new relative track address of a DTFDA or DTFNI file. The user preloads the address into the location specified by the SEEKADR keyword parameter.

FEOV

Function:

Initiates *end of volume* (EOV) procedures on the current volume of a DTFSD file; the next GET or PUT imperative macro continues processing on the next sequential volume. Not issued to DTFDA or DTFNI files.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	FEOV	{ filename (1) 1 }

Parameters:

filename

Is the label of the corresponding DTFSD macro instruction in the program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFSD file table.

GET

Function:

Makes the next logical record of a DTFSD file, or of a sequentially processed DTFNI file, available to the user in the current I/O area, or in a work area specified by the user. Automatically deblocks blocked records, making them available one at a time. Not issued to DTFDA files. Refer to Table A-1.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	GET	{filename} [{workarea}] { (1) } [{ (0) }] { 1 } [{ 0 }]

Parameters:**filename**

Specifies the label of the DTFSD or sequentially processed DTFNI declarative macro instruction.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTF file table.

workarea

Is the label of an area into which data management moves the current record for the user to process. (A different work area may be used for each GET macro.)

(0) or 0

Indicates that the user has preloaded register 0 with the address of a work area.

The user omits the second positional parameter when he is not processing in a work area. If it is specified, WORKA=YES must be specified in the DTF.

LBRET**Function:**

Issued in the user's *label processing* routine to create, retrieve, or retrieve and update optional user header or trailer labels and to transfer control. Is the only macro the user may issue in this routine. The address of the label processing routine is specified with the **LABADDR** keyword in the DTF declarative macro. Labels are not maintained below the file level.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	LBRET	{ 1 } { 2 } { 3 }

Parameters:

- 1 Returns control to the user's program at the next instruction after the OPEN or CLOSE macro call. Does not read or write a label.
- 2 Reads or writes one label, depending on the type of label processing specified by the TYPEFLE keyword parameter, and returns control to user inline, at the next instruction after the LBRET 2 macro call.
- 3 Writes back to disc the label just read (or read and updated) and returns control to the user inline, at the next instruction after the LBRET 3 macro call.

TYPEFLE=INOUT must be specified in your DTF declarative macro instruction or the file processing direction must be reset for update processing with the SETF macro instruction.

NOTE

Function:

Returns the partition-relative address of the current block or record to *filenameB* of a DTFNI file table or to *partitionnameB* of a DPCA *partition control appendage*. (Refer to Glossary entry for *filenameB*.) Issued following a GET, READ, PUT, or WRITE imperative macro.

The user must have pre-positioned the file to the desired partition with the SETP imperative macro, and to the proper subfile with the SETS macro.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	NOTE	{ filename } (1) 1

Parameters:

filename

Is the label in the user's program of the corresponding DTFNI macro instruction. The partition name is never used.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFNI file table.

OPEN**Function:**

Calls a transient routine to make certain validation checks and initialize any nonindexed disc file for processing. Initializes all partitions of a multipartitioned DTFNI file, marking the first partition (PCA1) active. If the IOREG keyword parameter has been specified in the DTFNI declarative macro, sets the index register to point to PCA1.

Checks whether all required keyword parameters have been supplied and whether all keywords specified are in valid form. Validates or creates system standard labels and user header labels if present. Transfers control to the user's LABADDR routine for processing UHL.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	OPEN	{ filename-1[,...filename-n] (1) 1 }

Parameters:**filename**

Is the label of one or more corresponding DTF declarative macro instructions in the user's program. The user may have 1 or as many as 16 files opened.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTF file table; used only when a single file is to be initialized.

POINT

Function:

Randomly positions a sequentially processed DTFNI file to a relative block address specified by the user. Not issued to DTFDA or DTFSD files.

Data management modifies the current partition-relative block address and block displacement maintained in the DTFNI file table or the DPCA partition control appendage; subsequent sequential processing continues from the new address. If the user issues random functions to the file, however, data management modifies the address by the relative track or record address supplied by the user in the field specified by the SEEKADR keyword parameter of the DTFNI declarative macro, and the POINT imperative macro is not effective.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	POINT	{ filename } , { address-field } { (1) } , { (0) } { 1 } , { 0 }

Parameters:

filename

Is the label in the user's program of the corresponding DTFNI macro instruction.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFNI file table.

address-field

Is the label of a 6-byte field (Obbbdd) in the user's program containing the relative block address and displacement of the record within the block. The relative block address portion (Obbb) is right-justified in the first four bytes, and the record displacement (dd) is right-justified in the second two.

(0) or 0

Indicates that the user has preloaded register 0 with the address of the 6-byte address field.

POINTS

Function:

Initializes the relative block address to the first block of the current partition of a multipartitioned DTFNI file. Not issued to DTFSD or DTFDA files. To reset the current relative block address to that of a new partition, the user must preselect the new partition by issuing the SETP imperative macro.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	POINTS	{ filename } { (1) } { 1 }

Parameters:

filename

Is the label of the corresponding DTFNI macro instruction in the user's program. The partition name is never used.

(1) or 1

Indicates that register 1 has been preloaded with the address of the DTFNI file table.

PUT

Function:

Issued to create, extend, or update DTFSD files or sequentially processed files defined by the DTFNI declarative macro. Not issued to DTFDA files. Delivers an output record to data management either in the current output area or in a user-specified work area. Once delivered, the record is no longer available to the user. Data management handles blocking automatically from the work area, but the user must block variable-length records constructed in the I/O area. Refer to the VARBLD keyword parameter, to the TRUNC imperative macro, and to Table A-1.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	PUT	$\left\{ \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} \left[\begin{array}{c} \text{workarea} \\ (0) \\ 0 \end{array} \right]$

Parameters:

filename

Is the label of the corresponding DTFSD or DTFNI macro instruction which defines the output file.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTF file table.

workarea

Is the label of the work area from which the output record may be obtained.

(0) or 0

Indicates that register 0 has been preloaded with the address of the work area.

Omission of the second positional parameter indicates that the user has chosen to reference the current record either by means of a register (IOREG keyword parameter) or by directly accessing the data relative to the name assigned to IOAREA1. The latter method is used only for unblocked records processed in a single I/O area. When a work area is used, WORKA=YES must be specified in the DTF declarative macro.

READ, ID**Function:**

Reads into the I/O area the key (if a key exists) and data portion of the block from a DTFDA or DTFNI input file, containing the record whose *relative disc address* the user has preloaded into the location specified by the SEEKADR keyword parameter of the DTF. The relative disc address specified must be greater than zero and in the form specified by the RELATIVE keyword parameter. Requires specification of READID=YES in the DTF declarative macro. Not issued to DTFSD files.

When the logical record whose relative disc address has been supplied via the SEEKADR field lies within a block of records retrieved, the following WAITF macro returns its displacement into the block to *filenameD*. The desired record and subsequent records within the block may be retrieved and placed in work areas by issuing successive GET imperative macros.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	READ	$\left. \begin{array}{c} \text{(filename)} \\ (1) \\ 1 \end{array} \right\} ,ID$

Parameters:**filename**

Is the label in the user's program of the DTF declarative macro which defines the randomly processed file from which the user is retrieving records.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTF file table.

ID

Specifies that a search is to be made for a record with an ID matching the disc address the user has presented to data management via the field specified in the SEEKADR keyword parameter of the DTF macro.

READ,KEY

Function:

Reads into the I/O area the key and data portion of the block from an input DTFDA or DTFNI file whose key the user has presented to data management in the location specified by the KEYARG keyword parameter of the DTF declarative macro. The data portion in the I/O area begins at a displacement equal to the key length. The user specifies the starting point for *search-on-key* by loading the initial *relative disc address* into the location specified by the SEEKADR keyword in the form specified by the RELATIVE keyword. The search is restricted to the initial track unless the user specifies SRCHM=YES in the declarative macro, in which case the search continues to the end of the cylinder. Requires specification of KEYARG,KEYLEN, and READKEY parameters in the DTF. Not issued to DTFSD files.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	READ	{ filename } (1) 1 },KEY

Parameters:

filename

Is the label of the DTFDA or DTFNI declarative macro in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTF file table.

KEY

Specifies that data management is to search for a record with a key matching the key the user has loaded into the location specified by the KEYARG keyword. The search begins at the relative address preloaded into the location specified by the SEEKADR keyword and continues to the end of the track, unless the user has specified SRCHM=YES.

RELSE

Function:

Skips over the remaining records in the current block of an input DTFSD or sequentially processed DTFNI file. The next GET imperative macro issued makes the first record of the next block available to the user. Not issued to DTFDA files.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	RELSE	{ filename } (1) 1

Parameters:

filename

Is the label in the user's program of the DTFSD or DTFNI declarative macro defining the file.

(1) or 1

Indicates that the user has preloaded general register 1 with the address of the DTFSD or DTFNI file table.

SETF

Function:

Issued between the CLOSE and OPEN imperative macros to change the *file processing mode* for a sequentially processed DTFNI file for which TYPEFILE=INOUT has been specified.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	SETF	{filename} (1) 1

Parameters:

filename

Is the label of the DTFNI macro instruction which defines the INOUT file.

(1) or 1

Indicates that register 1 has been preloaded with the address of the DTFNI file table describing the INOUT file.

INPUT

Indicates that the INOUT file is to be set for input processing, without updating.

OUTPUT

Indicates that the INOUT file is to be set for output processing.

UPDATE

Indicates that the INOUT file is to be set for input processing, with updating.

SETP

Function:

Selects for subsequent processing a new partition of a multipartitioned DTFNI file. Subsequent processing continues on this partition until the user issues another SETP macro; each SETP macro modifies the current partition address that data management maintains in the DTFNI file table.

If the user has specified the IOREG keyword parameter in the DTFNI declarative macro, the SETP macro loads the IOREG register for the partition accessed.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	SETP	{ filename } , { partition-name } (1) (0) 1 0

Parameters:

filename

Is the label of the DTFNI declarative macro that describes the already-opened file of which *partition-name* is a part.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFNI file table.

partition-name

Is the label within the DTFNI file table of the *partition control appendage* (PCA 1—7) which denotes the partition the user wants to access.

(0) or 0

Indicates that the user has preloaded register 0 with the address of the partition control appendage.

SETS

Function:

Selects, for subsequent processing, a *subfile* of a partition of a DTFNI file; must be issued to mark the end of a subfile. The user must preselect the appropriate partition of a multipartitioned file with the SETP imperative macro. Required specification of the SUBFILE keyword parameter in the DTFNI and DPCA declarative macros involved. The user may create as many as 71 subfiles in each partition; they must be created serially, but may be accessed at random for subsequent processing. Not issued to DTFDA or DTFSD files.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	SETS	$\left. \begin{array}{c} \text{(filename)} \\ (1) \\ 1 \end{array} \right\}, \left. \begin{array}{c} \text{(subfile-no.)} \\ (0) \\ 0 \end{array} \right\}$

Parameters:

filename

Is the label of the DTFNI macro instruction describing the file to which the subdivided partition belongs.

(1) or 1

Indicates that register 1 has been preloaded with the address of the DTF file table.

subfile-no.

Is the decimal integer number of the subfile (1 through 71) to be referenced.

(0) or 0

Indicates that register 0 has been preloaded with the subfile number.

TRUNC**Function:**

Writes a short block of variable-length records to an output DTFSD or sequentially processed DTFNI file. Issued when the user determines from the VARBLD register that the current record will not fit into *residual space* remaining in the current I/O area. Resets the IOREG index register to point to the address of the next available I/O area. The subsequent PUT macro starts off the next block with the current record. Not issued to DTFDA files; must not be issued to files containing fixed, blocked record format (RECFORM=FIXBLK). Refer to Table A—1.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	TRUNC	{ filename } { (1) } { 1 }

Parameters:**filename**

Is the label of the corresponding DTFSD or DTFNI macro instruction in the user's program.

(1) or 1

Indicates that register 1 has been preloaded with the address of the DTF file table.

WAITF

Function:

Checks completion of the data transfer initiated by the READ or WRITE macro issued to a DTFDA or a randomly processed DTFNI input or output file and sets the appropriate error or status bits in *filenameC*. It is also the function of the WAITF macro to set the address of the record written, or of the next available record, in the location specified by the IDLOC keyword parameter. This is done after the WAITF return; the form in which this *relative disc address* is returned is governed by the specification of the RELATIVE keyword parameter. Refer to Table A—18.

The user must issue the WAITF macro following each READ or WRITE macro, before issuing a CNTRL or another READ or WRITE macro to the same DTFDA file. If the WAITF macro is omitted when required, data management sets the WAITF macro to the same DTFDA file. If the WAITF macro is omitted when required, data management sets the *WAITF required* error flag (byte 0, bit 7) and the *invalid macro sequence* error flag (byte 0, bit 6) in *filenameC* and issues error message DM14. Refer to Appendix B. Not issued to sequentially processed DTFNI or DTFSD files. Not required after CNTRL macro.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	WAITF	{ filename } (1) 1

Parameters:

filename

Is the label of the DTFDA or DTFNI declarative macro in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFDA or DTFNI file table.

WRITE,AFTER

Function:

Writes the current block from main storage as the next sequential block on the current track of a DTFDA or randomly processed DTFNI file. If the file resides on a *variable-sector disc*, the remainder of the track is overwritten with binary 0's. Does not clear remainder of the track on a *fixed-sector disc*. Requires specification of the AFTER keyword parameter in the declarative macro.

If the current block, when written, occupies the last position on the track, the following WAITF macro sets the *last block on track accessed* flag (byte 0, bit 0) in *filenameC*.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	WRITE	{ filename } (1) 1 } ,AFTER

Parameters:

filename

Is the label in the user's program of the DTFDA or DTFNI declarative macro.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFDA or DTFNI file table.

AFTER

Specifies that the current block in main storage is to be written as the next sequential block on the current track of the DTFDA or DTFNI file.

WRITE,AFTER,EOF

Function:

Does not write a block to disc. Records the *relative disc address* of the current block, plus 1, as a *logical end-of-file* (EOF) pointer in the DTFDA or DTFNI file table. Data management records this in the disc *format 2 label* when the file is closed; refer to Appendix D. Requires specification of the AFTER keyword parameter in the DTFDA or DTFNI declarative macro. Not issued to DTFSD files.

The WRITE,AFTER,EOF macro is not required to be issued in programs written for OS/3, as data management automatically records logical EOF on file close.

Format:

LABEL	△OPERATION △	OPERAND
[name]	WRITE	{ filename } (1) 1 } ,AFTER,EOF

Parameters:

filename

Is the label in the user's program of the DTFDA or DTFNI declarative macro.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTF file table.

AFTER

Specifies that the relative block address of the next block position in physical sequence on the current track of this DTFDA or DTFNI file is to be recorded as logical EOF.

EOF

Marks the *relative disc address* of the current block, plus 1, as the *logical end-of-file (EOF)*.

WRITE, ID

Function:

Issued to create or update a DTFDA or randomly processed DTFNI file, with or without keys. Writes either the data portion only, or both the key and the data portion, of a block in a DTFDA or a randomly processed DTFNI file. The user loads the *relative disc address* of the block into the field specified by the SEEKADR keyword parameter of the declarative macro; this disc address must be greater than zero. Also requires specification of WRITEID=YES in the declarative macro. Not issued to DTFSD files.

Searches the file for a block with matching ID and writes or overwrites the first such block found. To update a key, the user specifies the length of the key actually on disc with the KEYLEN keyword parameter of the declarative macro; only the data portion of a block is written if the KEYLEN keyword is omitted.

If the search is unsuccessful, data management sets the *record not found* status flag (byte 1, bit 3) in *filenameC*. If lengths of the update key or data portions do not match those on disc, data management sets *wrong length found* error flag (byte 1, bit 5). Refer to Appendix B.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	WRITE	{ filename } (1) 1 ,ID

Parameters:

filename

Is the label in the user's program of the DTFDA or DTFNI declarative macro.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTF file table.

ID

Specifies a search of the DTFDA or DTFNI file for a block whose ID matches that provided by the user in the location specified by the SEEKADR keyword parameter.

WRITE,KEY

Function:

Rewrites, to the original disc location in the DTFDA or DTFNI file, the key and data portions of the block just retrieved with the READ,KEY imperative macro. Requires specification of WRITEKEY=YES in the declarative macro. Does not conduct a search. Not issued to DTFSD files. Not issued to create a file.

Consecutive issues of the WRITE,KEY imperative macro without intervening issue of the READ,KEY macro constitute a sequence error. Data management sets the *invalid macro sequence* error flag (byte 0, bit 6) in *filenameC* and issues error message DM14. Refer to Appendix B.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	WRITE	$\left. \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} ,\text{KEY}$

Parameters:

filename

Is the label in the user's program of the DTFDA or DTFNI declarative macro.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTF file table.

KEY

Specifies that the block just read by a READ,KEY macro is to be rewritten to its original location on the disc in a DTFDA or DTFNI file.

WRITE,RZERO**Function:**

Positions the disc access arm to a new track of the DTFDA or DTFNI file specified by the user in the location given by the SEEKADR keyword parameter, and initializes this track. Subsequent issue of the WRITE,AFTER imperative macro writes a block beginning at the first position on this track. Requires specification of AFTER=YES in the DTFDA or DTFNI declarative macro. Although the WRITE,RZERO macro does not write a block, the WAITF macro must be issued before issuing another imperative macro to the file. Not issued to DTFSD files.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	WRITE	{ filename } (1) 1 } ,RZERO

Parameters:**filename**

Is the label in the user's program of the DTFDA or DTFNI declarative macro.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFDA or DTFNI table.

RZERO

Positions the disc access arm to the new track of the DTFDA or DTFNI file, specified by the user in the location specified by the SEEKADR keyword parameter, and initializes this track.

DPCA

Function:

Defines the second and each subsequent partition of a multipartitioned DTFNI file. A maximum of seven partitions may be defined in all. Each DPCA macro creates an 82-byte *partition control appendage* to the DTFNI file table.

The label of the DPCA macro is the symbolic label of the partition specified by the user in the corresponding PCA keyword parameter of the parent DTFNI declarative macro.

DPCA and DTFNI keyword parameters are summarized in Table A—14; their variant forms in Appendix F.

Format:

LABEL	△ OPERATION △	OPERAND
partition-name	DPCA	BLKSIZE=n [,EOFADDR=symbol] ,IOAREA1=symbol [,IOAREA2=symbol] [,IOREG=(r)] [,KEYARG=symbol] [,KEYLEN=n] [,LACE=n] [,RECFORM= { FIXBLK FIXUNB VARBLK VARUNB }] [,RECSIZE=n] [,SIZE=n] [,SUBFILE=YES] [,UOS=n] [,VARBLD=(r)] [,WORKA=YES]

Parameters:

BLKSIZE=n

Specifies the length of a *block*, where *n* is the maximum number of bytes to be transferred into or out of main storage by a single access. The value of *n* includes the number of bytes of data and, for keyed records, the length of keys. For variable-length record formats, *n* must also accommodate the largest logical record or block, including the additional bytes required for the *BDW* and the *RDWs*. The block size specified by the BLKSIZE keyword may not exceed the *disc track size* of the subsystem on which the file resides. Required in the DPCA for each partition of a DTFNI file.

EOFADDR=symbol

Specifies the address of a routine coded to handle end-of-data for this partition of a sequentially processed DTFNI input file, where *symbol* is the symbolic address to which data management transfers control on sensing end-of-data. If omitted, data management transfers control to the address specified by the EOFADDR keyword parameter of the parent DTFNI macro.

DPCA

IOAREA1=*symbol*

Specifies the location of the I/O area, where *symbol* (label) is the address. Must always be half-word aligned. Length of the I/O area is the length of the space reserved for it with the BAL *define storage* (DS) statement and may equal or exceed the BLKSIZE specification; it must never be less. For output files, an additional 8-byte area must be reserved immediately preceding the I/O area for data management use; this is not included in the length of the I/O area, nor in the BLKSIZE specification. For device-independence, or when the file resides on a fixed-sector disc, the length of the I/O area should be a multiple of 256 bytes. Required in the DPCA for each partition of a DTFNI file.

IOAREA2=*symbol*

Specifies the location of an optional secondary I/O area for the sequentially processed file partition defined by this DPCA macro, where *symbol* (label) is the address of the secondary I/O area. If specified, the area is subject to the same considerations as noted for the area defined by the IOAREA1 keyword parameter.

IOREG=(*r*)

Specifies the general register to be used as an index register to reference current data for I/O operations, where *r* is the number of the general register. Registers 2 through 12 are available, and register 13 is also available when the user specifies the SAVAREA keyword parameter. The IOREG keyword parameter may not be specified when the user specifies a work area with the WORKA keyword parameter; it is required for DTFNI file partitions processed randomly with two I/O areas and for partitions processed sequentially when records are blocked or when two I/O areas are used. Refer to Table A-1.

KEYARG=*symbol*

Specifies that a search is to be made for a record having a key identical to a search argument the user provides to data management, where *symbol* (label) is the field in his program containing this argument. The KEYARG keyword parameter is required for a randomly processed file partition when READ,KEY or WRITE,KEY imperative macros will be issued. When the user specifies the KEYARG keyword parameter, he must also specify the length of the key, using the KEYLEN keyword parameter.

KEYLEN=*n*

Specifies the length of keys in the file partition defined by this DPCA macro, where *n* is the number of bytes in the keys. All keys in the same partition must have the same length; the key length may range from 3 bytes minimum to 255 bytes maximum. Required only when the key length is not the same as specified in the parent DTFNI macro.

LACE=*n*

Specifies the *lace factor*, a decimal integer, for data management use in applying the record *interlace* technique to this sequentially processed partition. Is ignored when the KEYLEN keyword parameter is specified. Refer to Appendix I.

RECFORM=FIXBLK

Specifies that records are fixed length and blocked.

RECFORM=FIXUNB

Specifies that records are fixed length and unblocked. Data management assumes this format if the user omits the RECFORM keyword parameter from the DPCA macro.

DPCA

RECFORM=VARBLK

Specifies that records are variable length and blocked. Not supported for DTFDA files.

RECFORM=VARUNB

Specifies that records are variable length and unblocked.

RECSIZE=*n*

Specifies the length of logical records in blocked files, where *n* is the number of bytes in the record. Data management assumes that record size equals block size (BLKSIZE keyword parameter) for fixed-length, unblocked records and expects to find record size in RDW of variable-length records.

SIZE=*n*

Specifies the percentage of total file allocation to be initially assigned by data management to the partition being defined by this DPCA declarative macro.

If omitted from the DPCA macro, data management makes a 1 percent allocation.

SUBFILE=YES

Specifies that data management is to support *subfiles* in the partition defined by this DPCA declarative macro. A maximum of 71 subfiles may be created in each partition; the user accesses these serially by issuing a SETS imperative macro to the file, having previously selected the correct partition with a SETP imperative macro.

UOS=*n*

Specifies, as the *unit of store*, the percentage of secondary disc storage allocation for the file that data management is to suballocate to the partition being defined each time it requires more space. The value of *n*, which is the specified percentage, may not exceed 100. Secondary storage allocation is specified in the EXT job control statement in the *device assignment set* for the file.

If omitted, or if a zero secondary storage allocation is specified in the EXT job control statement, no extension will be made.

VARBLD=(*r*)

Specifies a general register into which data management loads the number of bytes remaining in the I/O area after each execution of a PUT macro instruction to a sequentially processed partition of an output file containing blocked, variable-length records, where *r* is the number of the register. Value of *r* may range from 2 through 12, but register 13 also may be used if the SAVAREA keyword has also been specified. Refer to Table A—1.

WORKA=YES

Specifies to data management that the user will be processing input or output records sequentially in a work area and not in the I/O area. Specified for sequentially processed partitions of DTFNI files defined by the DPCA macro. The user does not specify the IOREG keyword when he specifies the WORKA keyword. The address of each work area is specified with the multiple-parameter forms of the GET and PUT macros. Refer to Table A—1.

DTFNI

Function:

Defines a nonindexed disc file to be processed sequentially, randomly, or by a combination of sequential and direct access techniques. Creates a 242-byte file table that includes provision for the first partition (PCA1). Table A—14 summarizes the DTFNI and DPCA keyword parameters; their variant forms are listed in Appendix F.

Format:

LABEL	△ OPERATION △	OPERAND
filename	DTFNI	<pre> [ACCESS= { EXC EXCR SRD SRDO }] [AFTER=YES] ,BLKSIZE=n [,EOFADDR=symbol] [,ERROPT= { IGNORE SKIP }] [,ERROR=symbol] [,IDLOC=symbol] ,IOAREA1=symbol [,IOAREA2=symbol] [,IOREG=(r)] [,KEYARG=symbol] [,KEYLEN=n] [,LABADDR=symbol] [,LACE=n] [,LOCK=NO] [,OPTION=YES] [[,PCA1=symbol ,..., [PCA7=symbol]]] [,READID=YES] [,READKEY=YES] [,RECFORM= { FIXBLK FIXUNB VARBLK VARUNB }] [,RECSIZE=n] [,RELATIVE= { R T }] </pre>

DTFNI

LABEL	△ OPERATION △	OPERAND
	<p>DTFNI (cont)</p>	<p>[,SAVAREA=symbol] ,SEEKADR=symbol [,SIZE=n] [,SRCHM=YES] [,SUBFILE=YES] [,TRLBL=YES] [,TYPEFLE= { INOUT INPUT OUTPUT }] [,UOS=n] [,UPDATE=YES] [,VARBLD=(r)] [,VERIFY=YES] [,WORKA=YES] [,WRITEID=YES] [,WRITEKEY=YES]</p>

Parameters:

ACCESS=EXC

Specifies the exclusive read/update/add use of the file. No other jobs can access the file while it is being used.



NOTE:

This specification is equivalent to the default value for the LOCK parameter; that is, LOCK=NO is omitted.



ACCESS=EXCR

Specifies read/update/add use of the file and also allows other jobs to read from the file while it is being used.

ACCESS=SRD

Specifies that only the read function is allowed for the file and allows other jobs read/update/add use of the file.

ACCESS=SRDO

Specifies that only the read function is allowed for the file and also allows other jobs to read from the file. Writing to the file is not allowed from either the job associated with this DTF or from other jobs.

DTFNI

The ACCESS parameter should not be specified on the same DTF with a LOCK parameter; however, if both the ACCESS and LOCK parameters are specified on the same DTF, the ACCESS parameter specification overrides the LOCK parameter specification.

AFTER=YES

Specified if the user issues a WRITE macro instruction containing an AFTER or an RZERO positional parameter. This keyword parameter used only when creating or adding records, or when repositioning the disc head to a new track. The WRITE, ID macro may not be issued to a file for file creation if the AFTER keyword is specified in the DTF.

BLKSIZE=n

Specifies the length of a *block*, where *n* is the maximum number of bytes to be transferred into or out of main storage by a single access. The value of *n* includes the number of bytes of data and, for keyed records, the length of keys. For variable-length record formats, *n* must also accommodate the largest logical record or block, including the additional bytes required for the *BDW* and the *RDWs*. The block size specified by the BLKSIZE keyword may not exceed the *disc track size* of the subsystem on which the file resides.

EOFADDR=symbol

Specifies the address of a routine the user has coded to handle end-of-data for a sequentially processed DTFNI input file, where *symbol* is the symbolic address to which data management transfers control on sensing the end of data. Required for sequentially processed DTFNI input files.

ERROPT=IGNORE

Specifies that data management is to make the current block or record (in which a parity error is detected) available to the user in the I/O area, as if no parity error had occurred.

ERROPT=SKIP

Specifies that data management is to bypass or skip over an input block or logical record containing a parity error, which it does not make available to the user for processing. For output records, data management ignores the block or record as if it were written correctly.

ERROR=symbol

Specifies the address of the user's error handling routine, to which data management transfers control upon detecting an error/status condition, where *symbol* is the symbolic address of this routine.

If this optional keyword parameter is omitted, data management returns control to the user's program inline, at the next instruction after the imperative macro which initiated transfer of control to his error routine.

IDLOC=symbol

Specifies the 4-byte field to which data management returns the *relative disc address* (ID) after the execution of a READ or WRITE macro, where *symbol* (label) is the address of the field. Data management assumes that size and format of the field are the same as specified in the SEEKADR keyword parameter. The form in which ID is returned is governed by specification of the RELATIVE keyword parameter; the record whose ID is returned is governed by the positional parameter used with the READ or WRITE macro. Refer to Table A-18. Optional; used for randomly processed DTFNI files only.

DTFNI

IOAREA1=symbol

Specifies the location of the I/O area, where *symbol* (label) is the address. Must always be half-word aligned. Length of the I/O area is the length of the space reserved for it with the BAL *define storage* (DS) statement and may equal or exceed the BLKSIZE specification; it must never be less. For output files, an additional 8-byte area must be reserved immediately preceding the I/O area for data management use; this is not included in the length of the I/O area, nor in the BLKSIZE specification. For device independence, or when the file resides on a fixed-sector disc, the length of the I/O area should be a multiple of 256 bytes.

IOAREA2=symbol

Specifies the location of an optional secondary I/O area for a sequentially processed file defined by the DTFNI macro, where *symbol* (label) is the address of the secondary I/O area. If specified, the area is subject to the same considerations as noted for the area defined by the IOAREA1 keyword parameter.

IOREG=(r)

Specifies the general register to be used as an index register to reference current data for I/O operations, where *r* is the number of the general register. Registers 2 through 12 are available, and register 13 is also available when the user specifies the SAVAREA keyword parameter. The IOREG keyword parameter may not be specified when the user specifies a work area with the WORKA keyword parameter; it is required for DTFNI files processed randomly with two I/O areas and for DTFNI files processed sequentially when records are blocked or when two I/O areas are used. Refer to Table A-1.

KEYARG=symbol

Specifies that a search is to be made for a record having a key identical to a search argument the user provides to data management, where *symbol* (label) is the field in the user's program containing this argument. The KEYARG keyword parameter is required for randomly processed DTFNI files when READ,KEY or WRITE,KEY imperative macros will be issued. When the user specifies the KEYARG keyword parameter, he must also specify the length of the key, using the KEYLEN keyword parameter.

KEYLEN=n

Specifies the length of all keys in a nonpartitioned DTFNI file, or the length of the keys in the first partition (PCA1) only of a multipartitioned file; *n* is the number of bytes in each key. All keys in the same partition, or in the same nonpartitioned DTFNI file must have the same length. The key length may range from 3 bytes minimum to 255 bytes maximum.

LABADDR=symbol

Specifies the address of the user's routine for processing optional standard user header and trailer labels, where *symbol* (label) is the address.

The user's label processing routine may issue the LBRET imperative macro, but no other. If he is processing UTL on closing the file, the user must also specify the TRLBL keyword parameter in the DTF.

LACE=n

Specifies the *lace factor*, a decimal integer, for data management use in applying the record interlace technique to sequentially processed input or output files defined by the DTFNI macro. Is ignored when the KEYLEN keyword parameter is specified. Refer to Appendix H.

DTFNI**LOCK=NO**

Is equivalent to specifying ACCESS=SRDO. It should not be used in the same DTF as the ACCESS keyword parameter. If it is, the ACCESS keyword parameter will override it.

Omitting both LOCK=NO and the ACCESS keyword parameter, is equivalent to specifying ACCESS=EXC.

OPTION=YES

Specifies that the sequentially processed input or output file defined by this DTFNI macro is an optional file: one the user anticipates will not invariably be present for every program execution. When specified for a file not allocated to a device by the job control DVC statement, transfers control to the user's EOFADDR routine on the first issue of the GET macro. When specified for a sequential file, issue of a direct access imperative macro (READ or WRITE) causes transfer of control to the user's error routine or to him inline.

PCA1=symbol,...,PCA7=symbol

Specifies the address of each *partition* of a multipartitioned DTFNI file, where *symbol* (label) is the address. The PCA keyword is not used for nonpartitioned DTFNI files. Partitions must be specified in unbroken sequence. The symbolic address is used as the label of the corresponding DPCA declarative macro defining the second and each subsequent partition.

READID=YES

Specifies to data management that the user will issue a READ,ID imperative macro to the DTFNI file defined by this declarative macro.

READKEY=YES

Specifies to data management that the user will issue a READ,KEY imperative macro to the DTFNI file defined by this declarative macro. The SEEKADR, KEYARG, and KEYLEN keyword parameters are also required.

RECFORM=FIXBLK

Specifies that records are fixed length and blocked. Requires specification of the RECSIZE keyword parameter.

RECFORM=FIXUNB

Specifies that records are fixed length and unblocked. Data management assumes this format if the user omits the RECFORM keyword parameter from the DTF.

RECFORM=VARBLK

Specifies that records are variable length and blocked.

RECFORM=VARUNB

Specifies that records are variable length and unblocked.

DTFNI

RECSIZE=*n*

Required when records in a DTFNI file are fixed length and blocked. Specifies the length of each logical record, where *n* is the length in bytes. Not required for other record formats: data management assumes that record size equals block size (BLKSIZE keyword parameter) for fixed-length, unblocked records, and it expects to find the record size in the RDW of each variable-length record.

RELATIVE=R

Specifies that the *relative disc address* (ID) given in the location defined by the SEEKADR keyword parameter is a relative record address.

RELATIVE=T

Specifies that the *relative disc address* (ID) given in the location defined by the SEEKADR keyword parameter is a relative track address.

SAVAREA=*symbol*

Specifies the address of a 72-byte labeled save area for the contents of general registers, full-word aligned, where *symbol* (label) is the address. If used, must be specified in the DTF for each file the user's program will process; however, only one such save area is required per program.

SEEKADR=*symbol*

Specifies the 4-byte location in the user's program into which he loads the *relative disc address* for use in processing direct access files with the READ, ID; READ, KEY; WRITE, ID; WRITE, RZERO; and CNTRL imperative macros. Required for randomly processed files defined by the DTFNI macro. The form in which the address is loaded is governed by the user's specification of the RELATIVE keyword parameter.

SIZE=*n*

Specifies the percentage of total file allocation to be initially assigned by data management to the partition being defined by this DTFNI declarative macro (i.e., PCA1). Not used with nonpartitioned DTFNI files.

If omitted from the DTFNI macro, data management assumes that SIZE=1 has been specified and assigns 1 percent to PCA1.

SRCHM=YES

Specifies that a *search on key* issued to a randomly processed DTFNI file (READ, KEY) is to be extended beyond the current track to the end of the cylinder.

SUBFILE=YES

Specifies that data management is to support subfiles in the file partition defined by this DTFNI declarative macro (i.e., PCA1). A maximum of 71 subfiles may be created in each partition; the user accesses these serially by issuing a SETS imperative macro to the file, having previously selected the correct partition with a SETP imperative macro.

DTFNI**TRLBL=YES**

Specifies that the user will process UTL when he issues the CLOSE imperative macro to terminate this DTFNI file. The user must also specify the LABADDR keyword parameter.

TYPEFLE=INOUT

Specifies a file which may be used for either input or output.

TYPEFLE=INPUT

Specifies a read-only DTFNI file; data management will read and check standard labels for this file. The user may not issue a sequential output function to this file unless he has also specified UPDATE=YES in the DTF. Data management assumes TYPEFLE=INPUT has been specified when the user omits the TYPEFLE keyword parameter.

TYPEFLE=OUTPUT

Specifies a write-only DTFNI file. Data management will write standard labels for this file.

The user may not issue a sequential input function to this file unless he closes it, resets its file processing direction to *input* with the SETF imperative macro, and reopens the file.

UOS=n

Specifies, as the *unit of store*, the percentage of secondary disc storage allocation for the file that data management is to suballocate to the partition being defined each time it requires more space. The value of *n*, which is the specified percentage, may not exceed 100. Secondary storage allocation is specified in the EXT job control statement in the device assignment set for the file.

Used with the DTFNI declarative macro to provide suballocation for the first (or only) partition of a nonindexed file.

If omitted, or if a zero secondary storage allocation is specified in the EXT job control statement, no extension will be made.

UPDATE=YES

Used, with the DTFNI declarative macro defining a sequentially processed file, only when the user has specified TYPEFLE=INPUT or TYPEFLE=INOUT. When used, specifies that the sequential output function (PUT macro) may be issued to update data records in the file. Unrelated to label processing.

If omitted from DTF for a sequentially processed input file (TYPEFLE=INPUT), the user may not issue an output function to the file.

VARBLD=(r)

Specifies a general register into which data management loads the number of bytes of *residual space* remaining in the I/O area after each execution of a PUT macro instruction to a sequentially processed DTFNI output file containing blocked, variable-length records, where *r* is the number of the register. The value of *r* may range from 2 through 12, but register 13 may also be used if the SAVAREA keyword has also been specified. Refer to Table A-1.

DTFNI

VERIFY=YES

Specifies that data management is to conduct a parity check of output records after writing them to disc. Parity check verification necessarily increases execution time for the PUT or WRITE macro.

If omitted, no verification is performed; however, data management may detect an output parity check error by other means. The user may direct data management to take certain actions with the ERROPT keyword parameter.

WORKA=YES

Specifies to data management that the user will be processing input or output records sequentially in a work area and not in the I/O area. Specified for sequentially processed files defined by the DTFNI macro. The user does not specify the IOREG keyword parameter when he specifies the WORKA keyword. The address of the work area is specified with each issue of the GET and PUT macro. Refer to Table A—1.

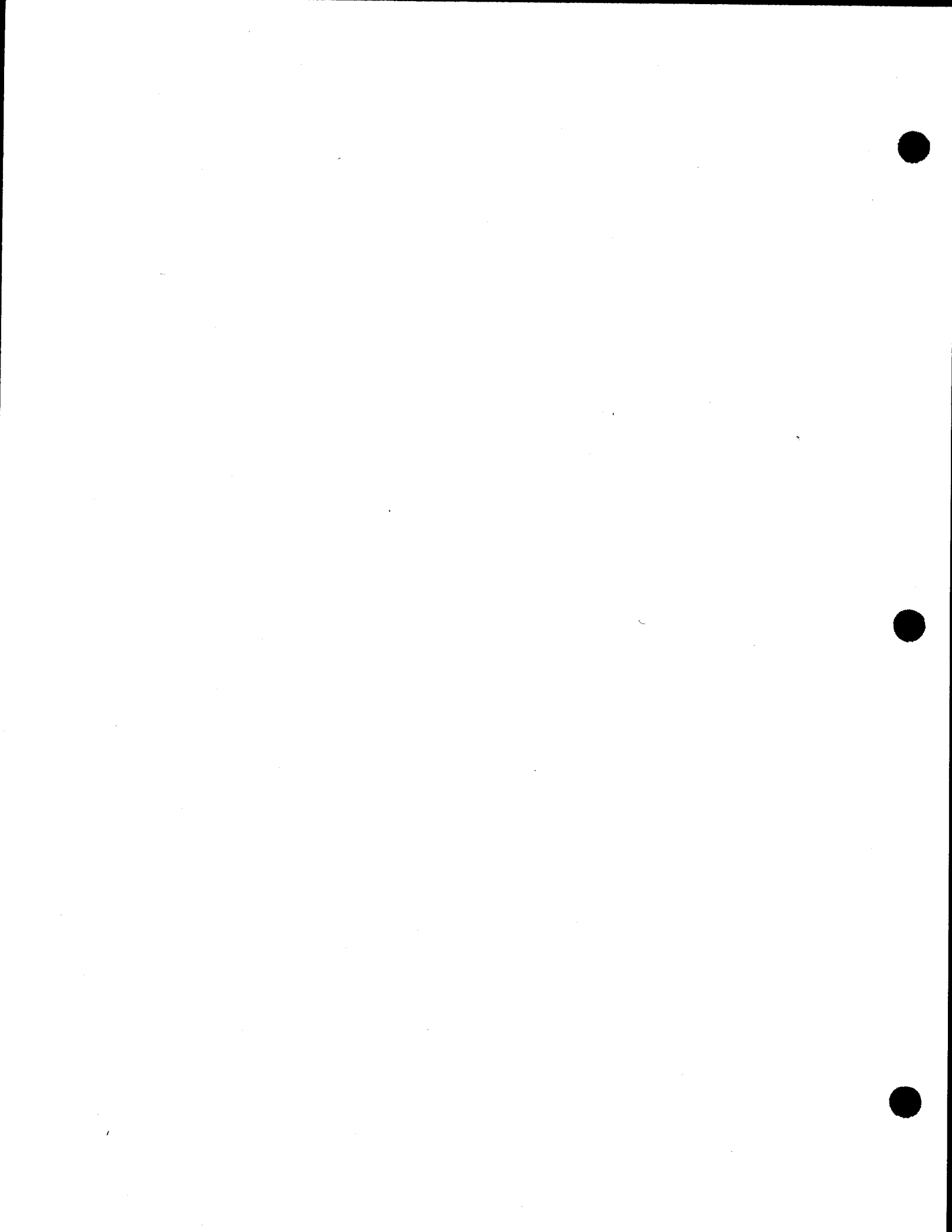
WRITEID=YES

Specifies that the user will issue a WRITE,ID macro to the randomly processed file defined by this DTFNI declarative macro to locate an output record by its relative disc address (ID).

WRITEKEY=YES

Specifies that the user will issue a WRITE,KEY imperative macro to the randomly processed file defined by this DTFNI declarative macro to rewrite a record just retrieved by the READ,KEY macro.

**8. Indexed Sequential Access Method and
Alternate Sequential Access Method**



ADD

Function:

Inserts a new record to the *overflow* area in an existing ISAM or ASAM file. The user must have preloaded the record, in logical record format, into the working storage area. Requires prior allocation of the cylinder overflow area during load (PCYLOFL keyword parameter). Is equivalent to the WRITE,NEWKEY macro, which is also accepted. Requires specification of IOROUT=ADD or IOROUT=ADDRTR in the DTFIS declarative macro. Requires issue of the WAITF macro.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	ADD	{ filename (1) 1 }

Parameters:

filename

Is the label of the corresponding DTFIS declarative macro in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

CLOSE

Function:

Initiates termination procedures for an ASAM or ISAM file. Once the file is closed, no data management macro instructions may be executed for the file until it is reopened. The user may access *filename* to ascertain the number of bytes required to hold the entire ISAM index in main storage.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	CLOSE	{ filename-1 [, ..., filename-n] (1) 1 *ALL }

Parameters:

filename

Is the label of the corresponding DTFIS declarative macro instruction in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

***ALL**

Specifies that all files currently open in the job step are to be processed.

ENDFL

Function:

Terminates *load* or *extending* functions for an ASAM or ISAM file and writes the final block to disc. Tests the file parameters and processes the index, if required for an ISAM file.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	ENDFL	{filename} (1) 1

Parameters:

filename

Is the label of the corresponding DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

ESETL

Function:

Terminates a retrieval sequence initiated by a SETL imperative macro on an ASAM or ISAM file. Rewrites any updated logical records that have not yet been rewritten. The user may initiate a new retrieval sequence with the SETL macro after the execution of the ESETL macro.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	ESETL	{filename} (1) 1

Parameters:

filename

Is the label of the corresponding DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded the address of the DTFIS file table into register 1.

GET

Function:

Makes the next logical record in the sequence available to the user, either in the current I/O area or in a work area specified by the user. If the block containing the next logical record is not already in main storage, reads the new block into the I/O area. If the GET macro requires the reading of a new block, and a PUT macro has been executed for any logical record in the previous block, rewrites the previous block, as updated, back to disc before reading the new block. Must be part of a retrieval sequence initiated by a SETL macro.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	GET	{ filename } [{ workname }] (1) (0) 1 0

Parameters:

filename

Is the label of the DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

workname

Is the label of the work area into which the record is to be transferred.

(0) or 0

Indicates that register 0 has been preloaded with the address of the work area into which the record is to be transferred.

The second positional parameter is omitted if WORKS=NO is specified in the DTFIS declarative macro.

OPEN

Function:

Initializes an ASAM or ISAM file before other imperative macros can be issued. Processes standard labels and validates extent information. Creates or checks disc *format 1* and *format 2* labels, and checks *volume 1* (VOL1) label.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	OPEN	{ filename-1[,...,filename-n] (1) 1 }

Parameters:

filename

Is the label of the corresponding DTFIS declarative macro instruction in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

PUT

Function:

Rewrites to disc an updated ASAM or ISAM logical record last retrieved by a GET macro instruction, which it must follow in a retrieval sequence initiated by a SETL macro. Not issued unless the last record retrieved by a GET macro has been updated by the user. Sets an indicator in the DTFIS file table that causes the block containing the updated record to be rewritten when all logical records in the current block have been processed and the next block is to be accessed.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	PUT	{ filename } [{ workname }] { (1) } [{ (0) }] { 1 } [{ 0 }]

Parameters:

filename

Is the label of the DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS macro instruction.

workname

Is the label of the work area from which the record is to be transferred.

(0) or 0

Indicates that the user has preloaded register 0 with the address of the work area from which the record is to be transferred.

The second positional parameter is not specified if WORKS=NO has been specified in the DTFIS macro instruction.

READ, ID

Function:

Initiates direct retrieval of a single logical record from an ASAM or ISAM file. Before issuing the READ, ID macro, the user must have preloaded the field specified by the KEYARG keyword parameter of the DTFIS declarative macro with the file-relative disc address of the record to be retrieved. The user must issue a WAITF imperative macro with the file-relative disc address of the record to be retrieved. The user must issue a WAITF imperative macro before accessing the record retrieved. He may not retrieve the *dummy record* at file start.

If the user specifies the use of a work area with the WORK1 keyword, the READ, ID macro reads the block into the I/O area and moves the logical record from there into the designated work area. If the user specifies the use of an I/O area instead, the READ, ID macro loads the general register specified by the IOREG keyword with the address of the first character of the logical record. The relative disc address from which the record has been retrieved is available in *filenameG*.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	READ	$\left. \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} ,ID$

Parameters:

filename

Is the label of the corresponding DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

ID

Indicates that random retrieval by record location is performed.

READ,KEY

Function:

Initiates direct retrieval of a single logical record from an ISAM file; invalid for an ASAM file. Before issuing the READ,KEY macro, the user must preload the field specified by the KEYARG keyword parameter of the DTFIS declarative macro with the key of the record to be retrieved. The user must issue a WAITF macro before accessing the retrieved record. He may not retrieve the *dummy record* at file start.

If the user specifies the use of a work area with the WORK1 keyword, the READ,KEY macro reads the block into the I/O area and moves the logical record from there into the designated work area. If the user specifies the use of an I/O area, the READ,KEY macro loads the general register specified by the IOREG keyword with the address of the first character of the logical record. The relative disc address from which the record has been retrieved is available in *filenameG*.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	READ	{ filename (1) 1 },KEY

Parameters:

filename

Is the label of the corresponding DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

KEY

Indicates that random retrieval by *search on key* is performed.

SETFL

Function:

Prepares an ASAM or ISAM file for *loading* or *extending* by setting controls in the DTFIS file table and, for an ISAM file, in the indexes on disc.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	SETFL	{ filename } (1) 1

Parameters:

filename

Is the label of the DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

SETL,BOF**Function:**

Initiates a retrieval sequence from an ASAM or ISAM file, specifying that the sequence is to start with the first logical record in the file. The WAITF macro is not required; retrieval may begin when the user receives control from the SETL,BOF macro.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	SETL	{ filename (1) 1 },BOF

Parameters:**filename**

Is the label of the DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

BOF

Indicates that the retrieval sequence is to begin with the first logical record of the file.

SETL,GKEY

Function:

Initiates a retrieval sequence from an ISAM file; invalid for an ASAM file. Specifies that the sequence is to start with the first logical record whose key is greater than or equal to the value that the user preloads into the field specified by the KEYARG keyword parameter of the DTFIS declarative macro.

When this starting point cannot be satisfied, data management sets the *record not found* error flag (byte 1, bit 3) in *filenameC* and branches to the user's error routine or returns control to him inline. The user's program should invariably check for this condition. If the user preloads the KEYARG field with a key of all binary 0's, data management prepares to give him the record beyond the *dummy record*; no error condition results.

The WAITF macro is not required; retrieval may begin when the user receives control from the SETL,GKEY macro.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	SETL	$\left. \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} \text{,GKEY}$

Parameters:

filename

Is the label of the DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

GKEY

Indicates that the retrieval sequence is to start with the first logical record whose key is greater than or equal to the value in the area equated to the KEYARG keyword parameter of the applicable DTFIS macro instruction.

SETL, ID**Function:**

Initiates a retrieval sequence from an ASAM or ISAM file; specifies that the sequence is to begin at the location that the user preloads into the field specified by the KEYARG keyword parameter of the DTFIS declarative macro instruction. He may not preload the address of the *dummy record* at file start.

When this starting point cannot be satisfied, data management sets the *record not found* error flag (byte 1, bit 3) in *filenameC* and transfers control to the user's error routine or to him inline. The user's program should invariably check for this condition. If he specifies the address of the dummy, data management returns error message DM24.

The WAITF macro is not required; retrieval may begin when the user receives control from the SETL, ID macro.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	SETL	{ filename } (1) 1 } ID

Parameters:**filename**

Is the label of the corresponding DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

ID

Indicates that the retrieval sequence begins at the location given in the KEYARG keyword parameter in the applicable DTFIS macro instruction.

SETL,KEY

(ISSQS3)

Function:

Initializes a retrieval sequence from an ISAM file; invalid for an ASAM file. Specifies that the sequence is to begin with the logical record whose key the user has preloaded into the field specified by the KEYARG keyword parameter of the DTFIS declarative macro.

When this starting point cannot be specified, or if the user preloads the KEYARG field with the all-zero key of the *dummy record* at file start, data management sets the *record not found* error flag (byte 1, bit 3) in *filenameC* and transfers control to the user's error routine or to him inline, issuing error message DM32.

The WAITF macro is not required; retrieval may begin when the user receives control from the SETL,KEY macro.

LABEL	Δ OPERATION Δ	OPERAND
[name]	SETL	{ filename } (1) 1 } ,KEY

Parameters:

filename

Is the label of the corresponding DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

KEY

Indicates that the area specified by the KEYARG keyword parameter in the DTFIS macro instruction holds the key of the first logical record to be retrieved.

UPDT

Function:

Initiates rewriting, to its original disc location in an ASAM or ISAM file, an updated logical record last retrieved by a READ imperative macro. Is not issued unless the user has updated the retrieved record. It is equivalent to the WRITE,KEY macro for updating randomly processed files.

The user must issue the WAITF imperative macro following the UPDT macro to ensure completion of the rewrite function before issuing another imperative macro to his file. When the WAITF macro returns control to him, the block containing the updated record will have been rewritten to disc, or an exception/error flag will have been set in *filenameC*.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	UPDT	{ filename } (1) 1

Parameters:

filename

Is the label of the DTFIS declarative macro instruction in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

WAITF

Function:

Ensures the completion of record transfer between disc and main storage; required following all random processing functions issued to ASAM and ISAM files before the user issues another record processing imperative macro. Any error or exceptional status conditions detected during the execution of the WAITF macro are reflected in *filenameC* of the DTFIS file table.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	WAITF	{ filename } (1) 1

Parameters:

filename

Is the label of the DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

WRITE,KEY**Function:**

Initiates rewriting, to its original disc location in an ASAM or ISAM file, an updated logical record last retrieved by a READ,ID or READ,KEY macro. Not issued if the user does not update the record retrieved. The user must also issue the WAITF macro if the WRITE,KEY macro is issued.

When the user updates the logical record in a work area, data management moves the updated record from the work area to the correct location in the I/O area and rewrites the block. If the user is processing in the I/O area, no move is made; the block and record are rewritten to the disc from there.

When the following WAITF macro returns control to the user, the block containing the updated record will have been rewritten to disc, or an exception/error flag will have been set in *filenameC*.

The WRITE,KEY macro is equivalent to the UPDT macro for updating randomly processed files.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[name]	WRITE	$\left. \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} \text{,KEY}$

Parameters:**filename**

Is the label of the DTFIS file table in the user's program.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFIS file table.

KEY

Indicates that the last record retrieved by a READ,KEY or READ,ID macro instruction is to be rewritten to the file.

WRITE,NEWKEY

Function:

Writes a logical record to the *prime data* area of an ASAM or ISAM file that is being *loaded* or *extended*, or *adds* a record to the *overflow* area. Prior to issuing the WRITE,NEWKEY macro, the user must preload the logical record into the work area. Requires specification of IOROUT=LOAD, IOROUT=ADD, or IOROUT=ADDRTR in the DTFIS declarative macro. Issue of the WAITF imperative macro is not required for load. Equivalent to the ADD macro for writing a new record to the *overflow* area.

Format:

LABEL	△ OPERATION △	OPERAND
[name]	WRITE	{filename} (1) 1 } ,NEWKEY

Parameters:

filename

Is the label of the DTFIS declarative macro instruction in the user's program.

(1) or 1

Indicates that register 1 has been preloaded with the address of the DTFIS file table.

NEWKEY

Indicates that a new record is to be written into an ASAM or ISAM file.

DTFIS

Function:

Defines an *indexed sequential access method* (ISAM) or *alternate sequential access method* (ASAM) file, and establishes a file table of varying size (Table A—15). Table A—16 summarizes the DTFIS keyword parameters; their variant forms are listed in Appendix F.

Format:

LABEL	△ OPERATION △	OPERAND
filename	DTFIS	<pre> [ACCESS= { EXC EXCR SADD SRD SRDO SUPD }] BLKSIZE=n [,ERROR=symbol] [,INDAREA=symbol] [,INDEXED=NO] [,INDSIZE=n] ,IOAREA1=symbol [,IOAREA2=symbol] [,IOREG=(r)] [,IOROUT= { ADD ADDRTR LOAD RETRVE }] [,KEYARG=symbol] [,KEYLEN=n] [,KEYLOC=n] [,LOCK=NO] [,PCYLOFL=nn] [,RECFORM= { FIXBLK VARBLK }] [,RECSIZE=n] [,SAVAREA=symbol] [,TYPEFLE= { RANDOM RANSEQ SEQNTL }] [,UPDATE=NO] [,VERIFY=YES] [,WORKS=NO] [,WORK1=symbol] </pre>

DTFIS

Parameters:

ACCESS=EXC

Specifies the exclusive read/update/add use of the file. No other jobs can access the file while it is being used.

NOTE:

This specification is equivalent to the default value for the LOCK parameter; that is, LOCK=NO is omitted.

ACCESS=EXCR

Specifies read/update/add use of the file and also allows other jobs to read from the file while it is being used.

ACCESS=SADD

Specifies read/update/add use of the file and also allows other jobs read/update use of the file.

ACCESS=SRD

Specifies that only the read function is allowed for the file and allows other jobs read/update/add use of the file.

ACCESS=SRDO

Specifies that only the read function is allowed for the file and also allows other jobs to read from the file. Writing to the file is not allowed from either the job associated with this DTF or from other jobs.

ACCESS=SUPD

Specifies only read/update use for the file and allows other jobs read/update/add use of the file.

The ACCESS parameter should not be specified on the same DTF with a LOCK parameter; however, if both the ACCESS and LOCK parameters are specified on the same DTF, the ACCESS parameter specification overrides the LOCK parameter specification.

BLKSIZE=n

Specifies the size of the data blocks in the ASAM or ISAM file, where *n* is the size, in bytes. This keyword is always required. Size may not exceed the *disc track size* for the subsystem on which the file resides, nor the length of the storage area defined for the I/O buffer. Note, when 8415 discs are used, ISAM requires at least 2 blocks of space per cylinder to be reserved for overflow.

ERROR=symbol

Specifies the address of the user's error-handling routine, to which data management transfers control for all conditions of error or exception to exact performance of the requested function. When data management transfers control, *filenameC* contains information on the reasons for the error. Refer to Appendix B.

If omitted, control returns to the user inline.

INDAREA=symbol

Specifies the location in main storage in which ISAM builds index blocks during *load* operations, and into which it loads as much as will fit of the *top index* when it opens the file for random processing; *symbol* (address) is the location. Must be half-word-aligned. Required for *load* operations; optional for others. The length of the area is specified by the INDSIZE keyword. Not used for ASAM files.

DTFIS**INDEXED=NO**

Specifies that the user has elected the option to create an *alternate sequential access method* (ASAM) file and to reference its records by relative addresses, rather than by keys. Data management does not construct the ISAM index; certain key-based processing functions are inoperative.

If omitted, the ISAM index structure is provided.

INDSIZE=n

Specifies length of index area in main storage (INDAREA keyword), where *n* is the length, in bytes. For *load* operations, the length must be at least 256 bytes. For random processing, length is optional. Not used for ASAM files. Refer to *filenameS*.

IOAREA1=symbol

Required to specify the location of the input/output area, where *symbol* (address) is the location. Must be half-word aligned. Space is not reserved with the BLKSIZE keyword, but with the BAL *define storage* (DS) statement; length must equal to or be greater than data block size, and it must not be less than 256 bytes. For device independence, length should be a multiple of 256 bytes.

IOAREA2=symbol

Specifies the location of optional additional input/output area, where *symbol* (address) is the location. Must also be half-word aligned and of the same size as the required area specified by the IOAREA1 keyword.

IOREG=(r)

Required to specify the general register to be used to point to the current I/O area when the user is not referencing records in the work areas, where *r* is the number of the general register. Registers 2 through 12 are available, and register 13 is also available if the user has specified the SAVAREA keyword.

IOROUT=ADD

Specifies that new records are to be *added* to a file. The user may not specify the ADD form of this keyword unless he has allocated an overflow area with the PCYLOFL keyword parameter.

IOROUT=ADDRTR

Specifies that new records are to be added to the file and that records will also be retrieved and *updated* randomly or sequentially. The user may not specify the ADDRTR form of this keyword unless he has allocated an overflow area with the PCYLOFL keyword.

IOROUT=LOAD

Specifies that either a new file is to be *loaded* or an existing file is to be *extended*. Data management assumes that the LOAD form has been specified if the user omits the IOROUT keyword parameter.

IOROUT=RETRVE

Specifies that ASAM or ISAM records are to be retrieved or *updated* randomly or sequentially.

KEYARG=symbol

Specifies the field in the user's program where he will place addresses or keys to effect retrieval of ASAM or ISAM records, where *symbol* (address) is the location of this field. The length of the KEYARG area is five bytes when used for relative addressing and equal to key length (KEYLEN keyword parameter) otherwise. May be omitted when the user will not retrieve records, or when he is retrieving records with the SETL, BOF macro. The user may not retrieve the *dummy record* at file start.

DTFIS

KEYLEN=*n*

Specifies the length of keys in an ASAM or ISAM file, where *n* is the length in bytes. All keys in an ISAM or ASAM file must have the same length; the minimum length is 3 bytes and the maximum is 253. Required for ISAM files; optional for ASAM. If specified for an ASAM file, causes data management to check the sequence of keys during a record load.

KEYLOC=*n*

Specifies the number of bytes that *precede* the key of an ASAM or ISAM record, where *n* is this number. The location of the key field must be the same within all records of the file. If the key begins in the first byte of the record, KEYLOC=0 should be specified.

If the KEYLOC keyword is omitted, ISAM assumes that the user has specified a value of *zero* for fixed records or 2 for variable records.

LOCK=NO

Is equivalent to specifying ACCESS=SRDO. It should not be used in the same DTF as the ACCESS keyword parameter. If it is, the ACCESS keyword parameter will override it.

Omitting both LOCK=NO and the ACCESS keyword parameter is equivalent to specifying ACCESS=EXC.

PCYLOFL=*nn*

Specifies the percentage of each cylinder of an ASAM or ISAM file that data management is to reserve for *overflow*, where *nn* is the percent. The value of *nn* may range from 00 through 80. If the user specifies PCYLOFL=00, records presented later for insertion in an *add* operation will be rejected.

If omitted, data management assumes that the user has specified PCYLOFL=00.

RECFORM=FIXBLK

Specifies that records are fixed length, blocked. The user must also specify the RECSIZE keyword.

RECFORM=VARBLK

Specifies that records are variable length, blocked. The user does not specify the RECSIZE keyword.

If the RECFORM keyword is omitted, data management assumes that the user has specified RECFORM=FIXBLK, and he must specify the RECSIZE keyword parameter.

RECSIZE=*n*

Specifies the length of fixed records, where *n* is this length, measured in bytes. Required for fixed-length records only; must be specified if the user omits the RECFORM keyword. Not used for variable records.

SAVAREA=*symbol*

Specifies the address of a 72-byte labeled save area for the contents of general registers, full-word aligned, where *symbol* (label) is the address. Used only when register 13 is not loaded with the save area address before each issue of imperative macros to the file.

DTFIS

If omitted, data management assumes that the user has preloaded register 13 with the address of a save area before issuing each imperative.

TYPEFLE=RANDOM

Specifies that random (direct) retrieval operations are to be performed.

TYPEFLE=РАНSEQ

Specifies that both random and sequential retrieval will be performed.

TYPEFLE=SEQNTL

Specifies that sequential retrieval operations will be performed.

If the TYPEFLE keyword is omitted, data management assumes that the user has specified TYPEFLE=SEQNTL. Not used unless records are to be retrieved. IOROUT=RETRVE or IOROUT=ADDRTR must also be specified.

UPDATE=NO

Specifies that data management is to flag a subsequent issue of the UPDT, PUT, or WRITE macro as an *invalid macro* error (byte 0, bit 6 of *filenameC*); no further reference to the file, other than to close it, is possible.

If omitted, the possibility of inadvertent updating of the file is not forestalled.

VERIFY=YES

Specifies that data management is to check parity of output records after they have been written to disc. Necessarily increases execution time for UPDT, PUT, and WRITE macros by about one rotation period per block. If bad parity is detected, data management sets the *output parity check* flag (byte 2, bit 2) in *filenameC* and transfers control to the user's error routine or to him inline.

If omitted, no output parity verification will be done.

WORKS=NO

When IOREG keyword is also specified for sequential retrieval, indicates that the user will process records in the current input buffer.

If omitted and the IOREG keyword is not specified, data management expects to transfer sequentially retrieved records (one at a time) to the work area specified as an operand of each GET macro issued.

WORK1=symbol

Provides the location of a record work area, where *symbol* (address) is the location. Required for *load* and *add* functions (when IOROUT=LOAD, IOROUT=ADD, or IOROUT=ADDRTR has been specified). Also required for random retrieval unless the user has specified the IOREG keyword parameter. Is ignored for sequential retrieval.



9. Paper Tape Data Management



CLOSE

Function:

Initiates termination procedures for a paper tape file.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	CLOSE	{ filename-1 [...,filename-n] } (1) 1 *ALL

Parameters:

filename

Is the label of the DTFTP declarative macro defining the paper tape file whose processing is to be terminated. As many as 16 files may be closed with the same macro; they need not all be paper tape files.

(1) or 1

Indicates that user has preloaded register 1 with the address of the declarative macro; used when a single file is to be terminated.

***ALL**

Specifies that all files open in the job step are to be closed.

GET

Function:

Makes next logical record from an input paper tape file available to the user in the current I/O area or (if the multiple-parameter form is used) in the specified work area. Before the record is made available to the user, data management has removed all *shift* or *delete characters* and translated all data requiring translation.

Format:

LABEL	△ OPERATION △	OPERAND
[symbol]	GET	{ filename } [{ workarea }] { (1) } [{ (0) }] 1 0

Parameters:

filename

Is the label in the user's program of the DTFPT declarative macro defining the input paper tape file from which user is retrieving records.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFPT file table.

workarea

Is the symbolic address (label) of the user's work area to which data management is to move the record.

(0) or 0

Indicates that the user has preloaded register 0 with the address of the work area to which data management is to move the record.

When the user specifies the multiple-parameter form of the GET macro, he must also specify the WORKA keyword parameter in the DTFPT declarative macro defining the file. The address of the work area may be different for each GET macro.

OPEN

Function:

Initializes a paper tape file for input or output processing. Data management assumes that the first character of the first record of an input file containing shifted codes is either a "figure" or the *letter shift character*.

Format:

LABEL	Δ OPERATION Δ	OPERAND
[symbol]	OPEN	{ filename-1 [, ..., filename-n] } (1) 1

Parameters:

filename

Is the label in the user's program of the DTFPT declarative macro defining the paper tape file that is to be initialized.

As many as 16 files may be initialized with one OPEN macro; they need not all be paper tape files.

(1) or 1

Indicates that the user has preloaded register 1 with the label of the single file that is to be initialized.

PUT

Function:

Punches an output record from the current I/O area to paper tape. If the user supplies the multiple-parameter form of the PUT macro, he forms his data record in the specified work area, from which data management moves it to the I/O area.

Before punching an output record to a paper tape file that contains *shifted codes*, data management inserts *letter* and *figure shift codes* and translates data as necessary; these actions are performed in the I/O area. In punching the first record to such an output file, data management automatically punches the *letter shift character* as the first code on tape if the first data character is not a "figure".

Data management automatically punches the *end-of-record stop* character after each undefined record. (Refer to the EORCHAR keyword parameter of the DTFPT declarative macro.)

Format:

LABEL	△ OPERATION △	OPERAND
[symbol]	PUT	$\left\{ \begin{array}{c} \text{filename} \\ (1) \\ 1 \end{array} \right\} \left[\begin{array}{c} \text{workarea} \\ (0) \\ 0 \end{array} \right]$

Parameters:

filename

Is the label in the user's program of the DTFPT declarative macro defining the output paper tape file.

(1) or 1

Indicates that the user has preloaded register 1 with the address of the DTFPT file table.

workarea

Is the label in the user's program of the work area from which data management is to move the record to the I/O area before punching it to paper tape.

(0) or 0

Indicates that the user has preloaded register 0 with the address of the work area.

When the user specifies the multiple-parameter form of the PUT macro, he must also specify the WORKA keyword parameter in the DTFPT declarative macro defining the file. The address of the work area may be different for each PUT macro.

DTFPT

Function:

Defines an input or output paper tape file and creates a 215-byte file table. DTFPT keyword parameters are summarized in Table A-17; their variant forms in Appendix F. ←

Format:

LABEL	△ OPERATION △	OPERAND
filename	DTFPT	BLKSIZE=n [,EOFADDR=symbol] [,EORCHAR=expression] [,ERROR=symbol] [,FSCAN=symbol] [,FTRANS=symbol] ,IOAREA1=symbol [,IOAREA2=symbol] [,IOREG=(r)] [,LSCAN=symbol] [,LTRANS=symbol] [,MODE= { BINARY } { STD }] [,OPTION=YES] [,OVBLKSZ=n] [,RECFORM= { FIXUNB } { UNDEF }] [,RECSIZE=(r)] [,SAVAREA=symbol] [,SCAN=symbol] [,TRANS=symbol] [,TYPEFLE= { INPUT } { OUTPUT }] [,WORKA=YES]

DTFPT

Parameters:

BLKSIZE=*n*

Required for all paper tape files. Specifies maximum size, in bytes, of largest logical record; *n* is a decimal number in the range 1 through 4095.

If omitted or improperly specified, the file is not opened and may not be processed. Refer to Appendix B.

The BLKSIZE specification does not include the extra bytes required for *letter* and *figure shift codes*, but must include one byte for the *end-of-record stop* character when record format is undefined. For output files (and for input files processed in binary mode), the BLKSIZE specification must also accommodate any record gap characters to be punched after the record. However, for character mode (MODE=STD) input files, no allowance is made for characters serving as an *interrecord gap*.

When user specifies *double-buffering* via work areas (WORKA keyword), the length of each work area must not be less than the BLKSIZE specification.

EOFADDR=*symbol*

Required for all input files; *symbol* specifies the label of the user's mandatory routine to handle the end-of-tape condition.

Data management branches automatically to this routine when end-of-tape is sensed and, in the course of *optional file processing*, when user issues a GET imperative macro to an optional input file. The user's EOFADDR routine may be assembled separately from the DTFPT declarative macro.

If the user is reading multiple strips of paper tape and desires to branch (after each strip) to a routine that reads and processes tape, his EOFADDR routine should contain a branch to the address contained in register 14; that address indicates the end-of-tape condition. If he issues imperative macros in his EOFADDR routine before he takes this branch, he must store and restore the contents of this register, preserving the return address it contained when this routine was entered.

EORCHAR=*expression*

Required for output paper tape files containing undefined records to specify the *end-of-record stop* character that data management punches following each record to serve as a delimiter. Here, *expression* is either an expression of self-defining terms or a symbol that is equated to an expression of self-defining terms. The end-of-record stop character may be any code that can be punched in the number of *levels* on the paper tape in use and is independent of shift status; one byte for it must be included in the user's BLKSIZE specification.

ERROR=*symbol*

Specifies the label of the user's error routine, to which data management transfers control on detecting software, hardware, or logical error conditions. If the user does not specify the label of an error routine (which may be assembled separately from the DTFPT declarative macro), data management expects him to test for errors inline and take appropriate action. Control in this situation returns to user inline, at the next instruction after the imperative macro during whose execution the error is detected.

When the error return is made to the user, or control is transferred to his ERROR routine, data management has set one or more error flags in a 1-byte addressable field of the DTFPT file table, designated *filenameC*, and has issued an error message to the log. Refer to Table B-2 for a summary of data management error processing for paper tape files.

DTFPT**FSCAN=symbol**

Specifies the label of the user's figure *scan table*; required to produce an output paper tape file in character mode (MODE=STD) that contains *shifted codes*. The LSCAN keyword must also be specified.

Each position in the scan table corresponds to a code or bit-pattern that may be presented in the user's output data to be punched. The FSCAN table contains the *letter shift character* in each position that corresponds to a "letter" code, and hexadecimal 00 in all other positions. It may be assembled separately from the DTFPT macro.

Data management uses the FSCAN table to identify the letter shift code and those characters to be treated as "figures". Before each group of these (unless they are the first characters of the first record), it punches the *figure shift code* on tape.

Both the FSCAN and LSCAN keywords must be specified and only for character mode files (MODE=STD). If one is specified without the other, a diagnostic message appears in the DTFPT assembly listing, and the specified keyword is ignored. If either of these keywords is specified for binary mode files, another diagnostic is issued in the assembly and the keyword is ignored.

FTRANS=symbol

Specifies the label of the user's figure *translation table* for an input file containing shifted codes, processed in character mode (MODE=STD). The translation table is used to relate hole-patterns punched on tape after the *figure shift code* to the 8-bit codes that are used in main storage to represent them. These codes, beginning with the null character (hexadecimal 00) and extending through hexadecimal 1F, 3F, or 7F (depending on the number of levels on the tape in use) may represent numerics or any characters the user desires.

To process shifted codes for input files, the FTRANS, LTRANS, and SCAN keywords must be specified. Only paper tape files processed in character mode (MODE=STD) may contain shifted codes. If the FTRANS keyword is specified for binary mode files, it is ignored. It must not be specified with TRANS keyword; if both are specified, both are ignored. A diagnostic message is listed in the DTFPT macro expansion in either case. The FTRANS table may be assembled separately from the DTFPT macro.

IOAREA1=symbol

Required for all files; *symbol* specifies the label of an area reserved for use as a primary I/O area. At least one I/O area must be specified.

If omitted, data management will not open the file and the user cannot process it. Refer to Table B-2 for the resultant data management error processing.

The user specifies the length of the I/O area with a *define storage* (DS) or *define constant* (DC) statement elsewhere in his program. This length must accommodate the largest logical record (including one byte for the *end-of-record stop* character if undefined record format is specified), excluding shift characters. Unless the user has specified the OVBLKSZ keyword (when I/O area length must not be less than the OVBLKSZ specification), the length of the I/O area must not be less than the BLKSIZE specification.

DTFPT

Because an EXTRN is generated for *symbol*, the coding that defines it may be assembled separately from the DTFPT macro call.

IOAREA2=*symbol*

Optional for all files; *symbol* is the label of a secondary I/O area established by the user to speed processing through *double-buffering*. Length considerations are the same as for the primary I/O area.

If the user specifies the IOAREA2 keyword and does not specify a work area with his GET or PUT macros, he must then specify an I/O register and use it to reference his data. Refer to the IOREG keyword parameter.

IOREG=(*r*)

Specifies the general register to be used as an I/O register to access current data, where *r* is the number of the register and must be encoded within parentheses. Registers 2 through 12 are always available for use as the I/O register, as is register 13 if the user specifies the SAVAREA keyword.

The IOREG keyword must be specified, and used to access current data, when the user specifies the IOAREA2 keyword and does not specify one or more work areas with his GET or PUT macros.

For output files, data management loads the IOREG register to point to the I/O area where the user must move or create the record to be punched, using this register, before he issues the PUT macro. (The IOREG register is loaded by data management on file open and after each PUT macro is issued).

For input files, data management loads the IOREG register after each GET macro is executed, to point to the I/O area where the current record is located.

The IOREG keyword should not be specified if the user has specified *double-buffering* via work areas by specifying the WORKA keyword. When both the IOREG and the WORKA keywords are specified, a diagnostic message appears in the DTFPT assembly listing, and the WORKA keyword is ignored.

LSCAN=*symbol*

Required, with the FSCAN keyword, to produce a punched paper tape with *shifted codes* in character mode (MODE=STD), where *symbol* is the label of a *letter scan table*.

Each position in the scan table corresponds to a code or bit-pattern that the user may present in his output data to be punched. The LSCAN table contains the *figure shift code* in each position that corresponds to a "figure" code, and hexadecimal 00 in all others. Like its reciprocal (the FSCAN table), the LSCAN table may be assembled separately from the DTFPT macro.

Data management uses the LSCAN table to identify the figure shift code and those characters it is to treat as "letters". Before each group of these, it punches the *letter shift code* on tape.

The LSCAN and the FSCAN keywords must be specified together, but only for character mode files (MODE=STD). Refer to the FSCAN keyword description for data management's action if misspecified.

DTFPT

LTRANS=symbol

To process *shifted codes* for input paper tape files in character mode (MODE=STD), the LTRANS keyword must be specified, together with the FTRANS and SCAN keywords. Here, *symbol* is the label of a *translation table* coded by the user for those characters that follow the *letter shift* code on tape.

The LTRANS table, like the FTRANS table, may be assembled separately from the DTFPT macro call. It is used to relate those hole-patterns that occur after the *letter shift* code on tape to the 8-bit patterns that are used in main storage to represent them. These codes may be used for alphabetic characters or any others the user desires.

The LTRANS keyword should not be specified for binary mode files, nor with the TRANS keyword. If specified when MODE=BINARY is specified, it is ignored; both keywords are ignored if the TRANS and the LTRANS keywords are specified together. A diagnostic message is listed in the DTFPT macro expansion in either case.

MODE=BINARY

Specifies that the *0920 paper tape subsystem* is to be operated in binary mode for processing the input or output file defined by this DTFPT declarative macro.

In binary mode, only 8-level (8-track) paper tapes may be processed. All eight bits of data are read into each byte in the I/O area from the reader; on output, all eight bits of data in each byte in the I/O area are punched. The *program connector* board is bypassed in the binary mode; it cannot be used to generate or suppress parity checking nor to specify a hardware *delete* or *end-of-record stop* character.

Input and output data may be translated in binary mode (TRANS keyword), but not shifted. Software *delete* characters may be specified via a *scan table* (SCAN keyword). *Null characters* (except for those used in the *paper tape leader*, before the first non-null character on tape) are transferred to main storage; the user must program to deal with these on input. Because undefined record processing of input files depends on hardware recognition of the *end-of-record stop* character, which cannot be specified to the paper tape subsystem in binary mode, undefined record format may not be used: only fixed, unblocked record format is specifiable in binary mode.

MODE=STD

Specifies that the *0920 paper tape subsystem* is to be operated in the character, or nonbinary, mode for processing the input or output file defined by this DTFPT macro.

In character mode, the user may read or punch paper tapes with from five to seven data levels, or tracks. Using the *program connector board*, the user may specify odd or even parity generation or checking, or he may suppress the generation or checking of parity signals. He may also wire the program connector board so that the hardware recognizes and deletes one *delete* character and recognizes the *end-of-record stop* character that delimits undefined records. Either record format may be specified; data may be shifted and translated. Null characters in input files are not transferred to main storage.

DTFPT

OPTION=YES

Specifies that the input or output paper tape file defined by this DTFPT declarative macro is an optional file: one that is not invariably required to be processed for every run of the user's program.

Two conditions result in *optional file processing* if this keyword is specified:

- If the user specifies the OPT positional parameter on the job control DVC statement for the file and the device is not available at execution time.
- If no DVC-LFD job control *device assignment set* is provided for the file

If the OPTION keyword is omitted, but one of the foregoing occurs, data management error processing results at file open time. Refer to Table B—2.

OVBLKSZ=n

Specifies the use of *over-sized I/O buffers*; optional and recommended for increased efficiency in processing input or output paper tape files containing fixed, unblocked records with *shifted codes*. Here, *n*, a decimal number in the range 2 through 4095, specifies the length of the buffers in bytes; the value of *n* must be at least one byte larger than the BLKSIZE specification.

When specified, main storage areas reserved for I/O buffers must not be smaller than the OVBLKSZ specification, but work areas need not be larger than BLKSIZE specification.

OVBLKSZ may not be specified if RECFORM=FIXUNB is not specified, nor if records do not contain shifted codes.

If omitted, records are processed in I/O buffers within areas limited in length to the BLKSIZE specification.

RECFORM=FIXUNB

Specifies that the record format is fixed and unblocked. Only this format may be used in binary mode. The user specifies record length with the BLKSIZE keyword.

RECFORM=UNDEF

Specifies the undefined record format; this format may be used only in character mode (MODE=STD). Undefined records may vary in length from record to record and are delimited by an *end-of-record stop* character on paper tape. The length of each current record is defined by the RECSIZE register.

If specified when MODE=BINARY is specified, this keyword is ignored and RECFORM=FIXUNB is assumed; a diagnostic appears in the DTFPT macro expansion.

DTFPT

RECSIZE=(r)

Specifies the general register to be used to refer to record length, where *r* is the number of this register and must be coded within parentheses. Required for output files with undefined records; the user must place the length of each undefined record in this register before he issues each PUT macro. Optional for input files with undefined records; if specified, data management places the length of the undefined records in this register before returning to the user from executing the GET macro. Record length for undefined records ranges from 1 through 4094 and does not include the *end-of-record stop* character nor shift characters. Registers 2 through 12 are always available for assignment as the RECSIZE register, as is register 13 if the user has specified the SAVAREA keyword.

SAVAREA=symbol

Specifies the label of a 72-byte, fullword-aligned area in main storage in which data management saves the contents of general registers while executing imperative macros issued to this file.

If omitted, the user must load register 13 with the address of a general register *save area*, 72-bytes long and fullword-aligned, before issuing any imperative macro to his file.

SCAN=symbol

Specifies the label of a shift code *scan table*; required, along with the FTRANS and LTRANS keywords, for processing input files with *shifted codes*. This keyword may also be specified to provide the label of a scan table that is used solely to delete characters from an input record in either binary or character mode (MODE=STD).

Each position in the SCAN table corresponds to a code or bit-pattern that may be read into the user's data area. Data management uses the content of the table to identify the *figure shift character*, the *letter shift character*, and one or more software *delete* characters. The nonzero entries that may be used in the SCAN table are the following three hexadecimal codes; all other entries must be hexadecimal 00:

- | | |
|----|---|
| 04 | Placed in the byte of the table that corresponds to the <i>figure shift</i> code. |
| 08 | Placed in the byte of the table that corresponds to the <i>letter shift</i> code. |
| 0C | Placed in those bytes of the table that correspond to one or more characters that are to be deleted by data management. |

When the sole purpose of the SCAN table is to identify software delete characters, the only entries the table may contain are hexadecimal 00 and 0C. If hexadecimal 04 and 08 are entered in the table to identify the shift codes, then the FTRANS and LTRANS keywords must also be specified.

DTFPT

The SCAN keyword is ignored if it is specified when the TRANS keyword is specified in company with the FTRANS or LTRANS keyword.

The SCAN table may be assembled separately from the DTFPT macro.

TRANS=symbol

Specifies the label of an optional *translation table* for any type of paper tape file but an input file with shifted codes.

The translation table is coded in the form expected by the assembler for operand 2 of the BAL *translate* (TR) instruction. It may be assembled separately from the DTFPT macro call. The TRANS keyword should not be specified in company with the FTRANS or LTRANS keyword. If it is so misspecified, the following keywords are ignored (if they are specified), and a diagnostic message is listed in the DTFPT macro expansion: TRANS, FTRANS, LTRANS, and SCAN.

TYPEFLE=INPUT

May be specified to indicate that the paper tape file defined by this DTFPT macro is an input file, to be read. Whether he specifies it explicitly or by default, the user must always specify the EOFADDR keyword for an input file.

TYPEFLE=OUTPUT

Must be specified to define an output paper tape file, that is to be punched.

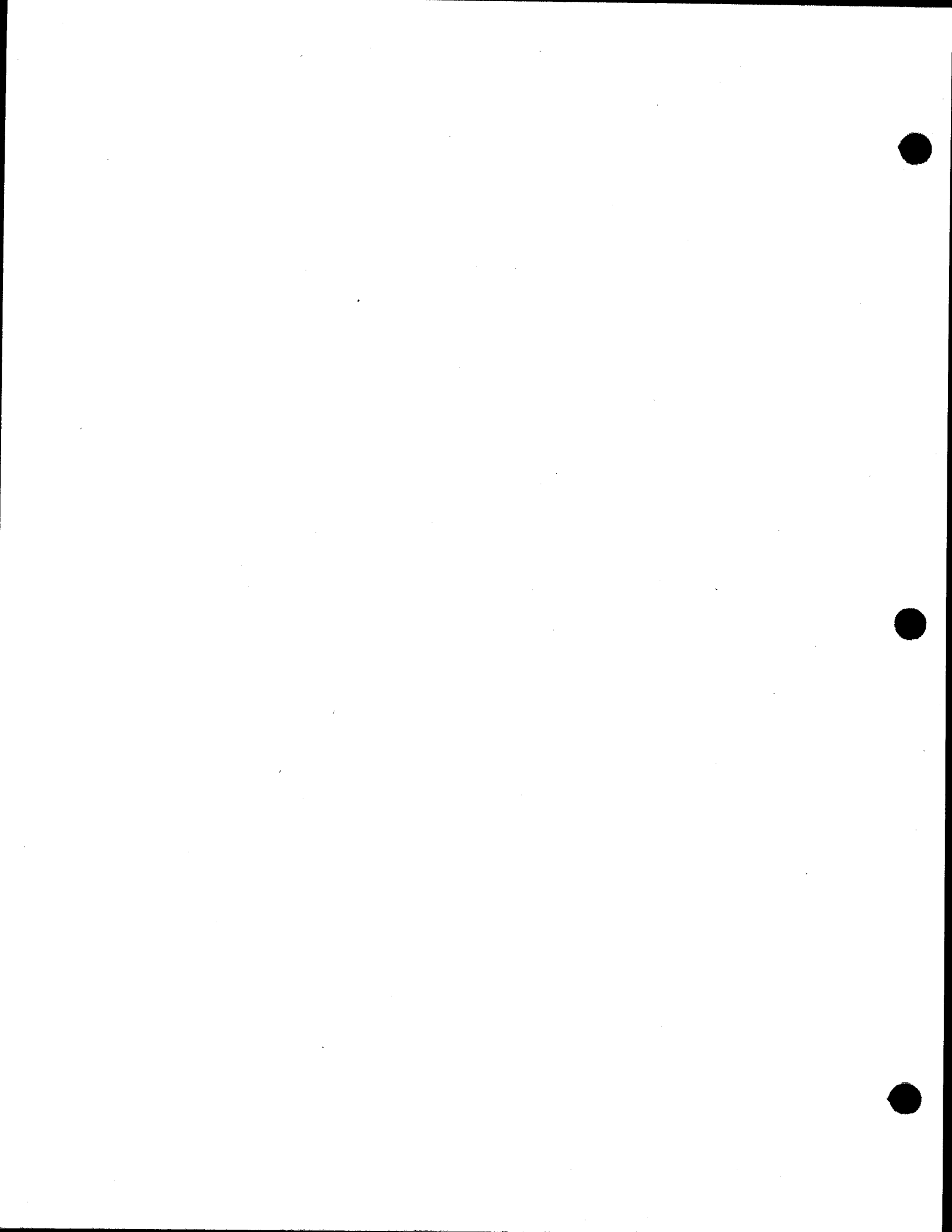
There is no combined type of paper tape file in OS/3: because an input and an output tape take different paths through the *0920 paper tape subsystem*, the same piece of tape cannot be read and punched in the same pass. To both punch and read the same piece of tape in the same program, the user must define it with two DTFPT declarative macros, assign two different *filenames*, and allocate each file with a separate DVC-LFD device assignment set.

WORKA=YES

Specifies *double-buffering* via work areas, the labels of which the user must specify with the multiple-parameter forms of the GET and PUT macros. For an input file, data management moves data from the I/O area to the work area before making it available to the user. For an output file, data management moves the user's data from the specified work area to the I/O area before initializing an I/O operation to the punch.

Ignored if specified with the IOREG keyword.

10. Indexed Random Access Method



DTFIR

Function:

The DTFIR declarative macro defines indexed or nonindexed IRAM files processed in sequential, direct, sequential-by-key access modes. The DTFIR macro establishes a 304-byte file table. Table A—19 provides a summary of DTFIR keyword parameters which are described in detail following the DTFIR declarative macro format.

Format:

LABEL	△ OPERATION △	OPERAND
filename	DTFIR	<pre> [ACCESS= { EXC EXCR SRD SRDO }] [ADD=YES] ,BFSZ=n [,EOFA=symbol] [,ERRO=symbol] [,INDA=symbol] [,INDS=n] [,INDX=YES] ,IOA1=symbol [,IOA2=symbol] [,IORG=(r)] [,KARG=symbol] [,KLEN=n] [,KLOC=n] [,MODE= { RAND SEQ }] [,OPTN=YES] ,RCSZ=n [,SKAD=symbol] [,SQCK=YES] [,TYPE= { INPUT OUTPUT }] [,UPDT=YES] [,VERFY=YES] [,VMNT=ONE] [,WORK=YES] </pre>

DTFIR

Parameters:

ACCESS=EXC

Specifies the exclusive read/update/add use of the file. No other jobs can access the file while it is being used.

NOTE:

This specification is equivalent to the default value for the LOCK parameter; that is, LOCK=NO is omitted.

ACCESS=EXCR

Specifies read/update/add use of the file and also allows other jobs to read from the file while it is being used.

ACCESS=SRD

Specifies that only the read function is allowed for the file and allows other jobs read/update/add use of the file.

ACCESS=SRDO

Specifies that only the read function is allowed for the file and also allows other jobs to read from the file. Writing to the file is not allowed from either the job associated with this DTF or from other jobs.

The ACCESS parameter should not be specified on the same DTF with a LOCK parameter; however, if both the ACCESS and LOCK parameters are specified on the same DTF, the ACCESS parameter specification overrides the LOCK parameter specification.

ADD=YES

Indicates that records can be added to an input file during record retrieval. These additions can be made only to indexed files processed by key. If the ADD parameter is specified, the DTFIR declarative macro must also contain the INDX=YES keyword parameter.

BFSZ=n

Indicates the size of the data buffer in the IRAM file. The BFSZ parameter is always required; n represents the number of bytes in the data buffer. The size must be at least 256 bytes as well as a multiple of 256.

EOFA=symbol

Specifies the symbolic address (name) of the required user routine that handles the end-of-data condition for IRAM input files. The parameter is required if MODE=SEQ is included in the DTFIR declarative macro; however, it is optional for randomly processed input files.

DTFIR**ERRO=symbol**

Specifies the symbolic address (name) of the user error-handling routine. When data management transfers control to the error routine, filenameC contains information on the reasons for the error. If ERRO parameter is omitted, control returns to the user inline.

INDA=symbol

Specifies the symbolic address of this index block processing area in main storage. The specified area must be half-word aligned and its length must be specified in the INDS parameter. The location of the index area in main storage must immediately precede the primary I/O buffer area (IOA1). This parameter is required for all keyed or indexed IRAM file processing.

INDS=n

Indicates the number of bytes used in main storage for the index area named in the INDA parameter. The index area length must be at least 256 bytes and in addition, a multiple of 256 bytes. Both INDS and INDA parameters are required specifications for IRAM indexed files.

INDX=YES

Indicates that IRAM file processing is to be performed by key. This parameter is required for input and output indexed IRAM files. In addition, the parameters INDA, INDS, KARG, KLEN, and KLOC must be specified if the INDX=YES is used.

IOA1=symbol

Specifies the symbolic address of the I/O processing area. IOA1 must be half-word aligned and greater than or equal to 256 bytes, a multiple of 256, and consistent with the BFSZ specification. IOA1 must immediately follow the index buffer (INDA), if specified, and must immediately precede the secondary I/O buffer (IOA2), if specified.

IOA2=symbol

Specifies the symbolic address of a secondary I/O area. Similar to IOA1, the IOA2 must be half-word aligned, the same size as the required IOA1, and immediately follow the primary I/O buffer. IOA2 may not be specified if the UPDT=YES parameter is specified.

IORG=(r)

Indicates the register number used to point to the current I/O area. Registers 2 through 12 are available. Either the IOREG or WORK parameters may be specified, but not both. If both parameters are specified, the WORK parameter specification is used.

KARG=symbol

Provides the symbolic address of the field in the user's program where keys are placed. The length of KARG must be equal to the specification in the KLEN parameter plus 3. The KARG parameter is required for all keyed operations.

KLEN=n

Specifies the number of bytes in a key for an IRAM indexed file. All keys in an IRAM file must be the same length; the minimum length is 3 bytes and the maximum, 80 bytes. This parameter is required for all keyed operations.

KLOC=n

Indicates the number of bytes preceding the key of an IRAM record. The key location must be the same within all records of the file. If the key begins in the first byte of the record, KLOC=0 should be specified. If the KLOC parameter is omitted, IRAM assumes a value of zero; i.e., the keys begin in the first byte of each record. This parameter is required for all keyed operations.

DTFIR

↓

LOCK=NO

Is equivalent to specifying ACCESS=SRDO. It should not be used in the same DTF as the ACCESS keyword parameter. If it is, the ACCESS keyword parameter will override it.

Omitting both LOCK=NO and the ACCESS keyword parameter is equivalent to specifying ACCESS=EXC.

↑

MODE=RAND

RAND specifies random (direct) retrieval operations for an indexed file and random retrieval or random load operations for a nonindexed file.

MODE=SEQ

SEQ specifies sequential retrieval operations for an indexed IRAM file and sequential retrieval or sequential load operation for nonindexed IRAM files. Sequential mode is assumed if no MODE parameter is specified.

OPTN=YES

Specifies that the sequential input or output file defined by the DTFIR macro is an optional file. When this parameter is specified for an input file not allocated to a device by the DVC job control statement, data management transfers control to the user's EOFA routine on the first issue of input operation. When the OPTN parameter is specified for an output file, data management transfers control to the user's program inline and with no error.

RCSZ=n

Specifies the length of each record in bytes and is always required.

SKAD=symbol

Specifies the symbolic address of an area in your program into which you load the relative disc address for use in processing nonindexed files by relative record number. The form of a record address is a 4-byte value and the first record is relative record 1.

SQCK=YES

Specifies that data management should verify ascending key sequence on file creation for indexed files.

TYPE=INPUT

INPUT specifies a read-only DTFIR file. If omitted, data management assumes the TYPE=INPUT specification. Users may not issue an output function to this file unless:

- they specify either the UPDT=YES or ADD=YES parameters; or
- they close the file, reset the file processing direction, and reopen the file.

TYPE=OUTPUT

OUTPUT specifies a write-only file. Users may not issue an input function to this file unless they close, reset, and reopen the file.

UPDT=YES

Indicates that nonindexed or indexed input files (TYPE=INPUT) are to be updated. If this parameter is omitted, users cannot update a file.

DTFIR**VERFY=YES**

Indicates that data management should check parity of output records after they have been written to disc.

If it detects bad parity, data management sets the parity check flag (byte 2, bit 2) in filenameC and transfers control to the user's error routine or to his program inline if he has no error routine. Specifying this parameter results in an increase in execution times for output operations.

VMNT=ONE

Specifies that only one volume be processed online at any time. This parameter cannot be used if the MODE=RAND parameter is specified unless the INDX=YES parameter also is specified.

WORK=YES

Indicates the use of a work area. This parameter may not be specified in the same DTFIR with the IOREG parameter. When you issue input, update, or file addition operations, you specify the address of the work area. The WORK parameter is required for indexed files when TYPE=OUTPUT or when TYPE=INPUT and ADD=YES.



11. Multiple Indexed Random Access Method



DTFMI

Function:

The DTFMI declarative macro defines indexed or nonindexed MIRAM files. It establishes a 388-byte file table. Table A—20 provides a summary of DTFMI keyword parameters, which are described in detail following the DTFMI declarative macro format.

Format:

LABEL	Δ OPERATION Δ	OPERAND
filename	DTFMI	<pre> [ACCESS = { EXC EXCR SRD SRDO }] ,BFSZ=n [,EOFA=symbol] [,ERRO=symbol] [,INDA=symbol] [,INDS=n] ,IOA1=symbol [,IOA2=symbol] [,IORG=(r)] [,KARG=symbol] [,KEYn=(s,[I],[{NDUP}],[{DUP}],[{NCHG}],[{CHG}]))] [,LOCK=NO] [,MODE= { SEQ RAN RANH }] [,OPTN=YES] [,PROC= { KEY UNK }] [,RCB=NO] [,RCFM= { FIX VAR }] ,RCSZ=n [,RETR= { INF MOD REP }] ,SKAD=symbol [,VMNT=ONE] [,VERFY=YES] [,WORK=YES] </pre>

DTFMI

Parameters:

ACCESS=EXC

Specifies the exclusive read/update/add use of the file, no other jobs can access the file while it is being used.

NOTE:

This specification is equivalent to the default value for the LOCK parameter; that is, LOCK=NO is omitted.

ACCESS=EXCR

Specifies read/update/add use of the file and also allows other jobs to read from the file while it is being used.

ACCESS=SRD

Specifies that only the read function is allowed for the file and allows other jobs read/update/add use of the file.

ACCESS=SRDO

Specifies that only the read function is allowed for the file and also allows other jobs to read from the file. Writing to the file is not allowed from either the job associated with this DTF or from other jobs.

The ACCESS parameter should not be specified on the same DTF with a LOCK parameter; however, if both the ACCESS and LOCK parameters are specified on the same DTF, the ACCESS parameter specification overrides the LOCK parameter specification.

BFSZ=n

Is always required. n represents the number of bytes in the data buffer. The size must be at least 256 bytes as well as a multiple of 256 bytes.

EOFA=symbol

Specifies the symbolic address of your end of file handling routine. This parameter is required if MODE=SEQ is specified.

ERRO=symbol

Specifies the symbolic address of your error handling routine to which data management transfers control when an error is detected. When control is transferred to this routine, filenameC contains information on the reasons for the error. If this parameter is omitted, control is returned to your program inline.

DTFMI**INDA=symbol**

Specifies the symbolic address of the main storage area in which index blocks are processed. 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 must be specified: INDS, KARG, and KEYn. If INDA is specified and any of the related parameters are omitted, it is assumed that indexed operations were not intended to be used and none will be permitted.

INDS=n

Specifies the number of bytes to be used in main storage for the index area named in the INDA parameter. The length must be at least 256 bytes or a multiple of 256.

IOA1=symbol

Specifies the symbolic address of the primary I/O (data) buffer area. This area must be half-word aligned, greater than or equal to 256 bytes, a multiple of 256 bytes, and it must be consistent with the BFSZ parameter. It must immediately follow the index area (INDA), if specified, and must immediately precede the secondary I/O (data) buffer, if specified.

IOA2=symbol

Specifies the symbolic address of a secondary I/O (data) buffer area. As with IOA1, this buffer must be half-word aligned, the same size as IOA1, and must immediately follow the primary I/O (data) buffer (IOA1). A secondary data buffer is permitted only when performing keyed or nonkeyed sequential output or nonkeyed sequential input operations.

IORG=(n)

Specifies the number of the general register used to point to the current I/O (data) buffer when you do not reference records in a work area. Registers 2 through 12 may be used. Either IORG or WORK must be specified. If both are specified, the WORK parameter is used.

KARG=symbol

Specifies the symbolic address of the field in your program where keys are placed to effect the retrieval of records. The length of this field must be equal to the largest key in your program plus three bytes. This parameter is required for all keyed operations.

$$\text{KEY}_n = \left(s, [l], \left[\left\{ \begin{array}{c} \text{NDUP} \\ \text{DUP} \end{array} \right\} \right], \left[\left\{ \begin{array}{c} \text{NCHG} \\ \text{CHG} \end{array} \right\} \right] \right)$$

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. s specifies the size of the key and may range from 1 to 80 bytes. l specifies the number of bytes preceding the key. If l is omitted, 0 is assumed for fixed records and 4 for variable-length records. DUP specifies that duplicate keys are allowed. NDUP specifies that they are not allowed and is the default case. CHG specifies that the key can change during update; NCHG specifies that it cannot and is the default case.

KEYn parameters are required unless you intend to accept the KEYn parameters that were specified when the file was created. If so, no KEYn specification should be present. For IRAM characteristic files, $n > 1$, $s < 3$, DUP, and CHG are not permitted.

DTFMI

↓

LOCK=NO

Is equivalent to specifying ACCESS=SRDO. It should not be used in the same DTF as the ACCESS keyword parameter. If it is, the ACCESS keyword parameter will override it.

Omitting both LOCK=NO and the ACCESS keyword parameter is equivalent to specifying ACCESS=EXC.

↑

MODE=SEQ

Specifies sequential file processing. This is the default case.

MODE=RAN

Specifies random file processing.

MODE=RANH

Specifies random file processing (hold sequential position).

OPTN=YES

Specifies that the sequentially processed file is an optional file. When this parameter is specified for a file not allocated to a device by a DVC job control statement, control is transferred to your EOFAR routine when the first input operation is issued. Control is transferred to your program inline when the first output operation is issued.

PROC=KEY

Specifies keyed operations. This is the default case.

PROC=UNK

Specifies nonkeyed operations.

RCB=NO

Applies only to files that are being created. It is required for IRAM characteristic files and it specifies that each record is not to contain a record control byte (RCB). If specified, delete functions will not be permitted when the file is subsequently processed. The default case is that each record will contain an RCB. When a file creation program is completed and the file is closed, the format label is marked to indicate whether the RCB is present. Once the file is created, this format label indication cannot be changed; that is, if the file is subsequently processed and an attempt is made to use the RCB parameter, the format label indication will override it.

↓

RCFM=FIX

Specifies fixed-length records. This is the default case.

RCFM=VAR

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 effective record size that the user supplies on output and data management supplies on input. The record size includes the 4-byte RDW. Variable-length records are contained within a fixed-size slot that is equal to the RCSZ specification.

↑

RCSZ=n

Specifies the length of each record in bytes. For variable-length records, this parameter should specify the maximum size plus the 4-byte RDW required for variable-length records.

DTFMI**RETR=INF**

Specifies that records are to be retrieved for information purposes only. This is the default case.

RETR=MOD

Specifies that records are to be retrieved for modification.

RETR=REP

Specifies that records are to be retrieved for replacement.

SKAD=symbol

Specifies the symbolic address in your program where you place the relative disc address for use in processing files by relative record number. The form of the relative disc address is a 4-byte value. The first record is relative record 1.

VRFY=YES

Specifies that data management is to check the parity of output records after they have been written to disc. If it detects bad parity, data management sets the parity check flag (byte 2, bit 2) in filenameC and transfers control to your error routine or to your program inline if you have no error routine. Specifying this parameter results in an increase in execution times for output operations.

VMNT=ONE

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

WORK=YES

Specifies that input and output records will be processed in a work area rather than a data buffer area. The IORG parameter should not be specified when the work parameter is specified. If both are specified, the WORK parameter is used. When you issue input, output, update, or delete operations, you specify the address of the work area. The WORK parameter is required for all output, keyed, update, and delete operations.



**Appendix A. Tables Cited in Sections
2 Through 11**



Table A-1. Use of the IOREG Keyword Parameter for Sequential Processing of Blocked or Unblocked Card, Tape, Printer, or Disc Files

Number of I/O Areas		Number of Work Areas		Record Format		IOREG Specification Required	OS/3 Data Management Action: Input Files
1	2	0	>0	Blocked	Unblocked		
x		x		x		YES	Uses IOREG to point to address of current record within the block contained in IOAREA1.
x		x			x	NO	Delivers each unblocked record directly to user in IOAREA1.
x			x	x		NO	Deblocks in IOAREA1 and delivers each deblocked record to work area specified in positional parameter 2 of GET macro.
x			x		x	NO	Delivers unblocked record from IOAREA1 to work area specified in GET macro.
	x	x		x		YES	Delivers record to IOAREA specified by IOREG. Alternate areas are available for overlap processing.
	x	x			x	YES	Delivers record to IOAREA specified by IOREG. Alternate areas are available for overlap processing.
	x		x	x		YES	Deblocks in one IOAREA and delivers deblocked records to work area specified by GET macro. Other areas are available for overlap processing.
	x		x		x	YES	Delivers unblocked record to work area specified by GET macro, from IOAREA specified by IOREG. Alternates are available for overlap processing.
x		x		x		YES	PUT macro delivers logical record to data management in IOAREA1. Data management loads the starting address in the output area of the current record into IOREG index register, and automatically blocks fixed-length records. User blocks variable-length records constructed in I/O area; refer to VARBLD keyword parameter.
x		x			x	NO	PUT macro delivers logical record to data management in IOAREA1. Data management writes out each record singly. User references records directly, relative to IOAREA1.
x			x	x		NO	PUT macro delivers logical record to data management in work area. Data management automatically blocks fixed- and variable-length records and writes block out from work area. User specifies work area address with operand of each PUT macro issued.
x			x		x	NO	PUT macro delivers logical record to data management in work area; data management writes record out from work area. User specifies work area as operand of each PUT macro issued. For printer files, see notes 1 and 2.
	x	x		x		YES	Loads IOREG index register with address of next available I/O area. PUT macro delivers logical record to data management in current I/O area and automatically blocks fixed-length records. User must block variable-length records; refer to VARBLD keyword parameter. Data management calculates block size and loads it into BDW before writing block out.
	x	x			x	YES	Loads IOREG index register with address of next available I/O area. PUT macro delivers logical record to data management in current I/O area, which writes it out from there. Alternate I/O area available for overlap processing. For printer files, see notes 1 and 2.
	x		x	x		YES	Loads IOREG index register with address of next available I/O area. PUT macro delivers logical records from work area to current I/O area, from where data management writes block out to device, automatically blocking records. Address of work area is specified with operand of each PUT macro issued. For variable-length records, data management tests whether current record will fit into I/O area; if it will, it is added to current block. If not, data management first writes out current block and then starts new block with current record.
	x		x		x	YES	Loads IOREG index register with address of next available I/O area. PUT macro delivers logical records from work area to current I/O area; user specifies address of work area with operand of each PUT macro. Data management writes out records singly from current I/O area. For printer files, see notes 1 and 2.

NOTES:

- For DTFPR files, no additional processing speed is obtained with two I/O areas and a work area. Most efficient processing is obtained with either of the following combinations of DTFPR keywords:
 IOAREA1, IOAREA2, and IOREG
 or
 IOAREA1 and WORKA.
- For DTFPR files, the I/O area must provide space for everything included as part of the block length. The user must allocate I/O areas containing an even number of bytes, excluding the control character, if any. Before printing a record containing an odd number of characters to be printed, data management inserts a nonprinting character in the I/O area after the last byte of user-supplied data.
- This table is not applicable to DTFIS files. When IOROUT= ADD, IOROUT=ADDRTR, or IOROUT=LOAD is specified, a work area is always required. When IOROUT=RETRVE is specified, the user may specify either a work area or the IOREG keyword.

Table A-2. Sizes of DTFCD File Tables

Type of File	Keyword Specifications				MODE= BINARY	File Table Size in Bytes
	ITBL	OTBL	SAVAREA	CRDERR		
INPUT	No	NA	No	NA	NA	104
INPUT	No	NA	Yes	NA	NA	112
INPUT	Yes	NA	No	NA	NA	116
INPUT	Yes	NA	Yes	NA	NA	116
OUTPUT	NA	No	No	No	NA	104
OUTPUT	NA	No	Yes	No	NA	112
OUTPUT	NA			Yes	No	200
OUTPUT	NA			Yes	Yes	280
COMBINED	No	No	No	NA	NA	104
	No	No	Yes	NA	NA	112
	Yes	Yes	No	NA	NA	120
	Yes	Yes	Yes	NA	NA	120

Table A-3. Summary of DTFCD Keyword Parameters (Part 1 of 2)

Keyword	Specification	Files			Remarks
		Input	Output	Cmbnd	
ASCII	YES	X	X	X	Specifies processing in ASCII; if used, MODE=STD must be specified
AUE*	YES	X			Skips cards containing validity check errors.
BLKSIZE**	n	X	X	X	The maximum block size in bytes
CONTROL*	YES		X	X	Specifies that CNTRL macro is to be issued to control stacker selection on 0604 card punch.
CRDERR*	RETRY		X		For card punch error recovery.
EOFADDR	symbol	R		R	End-of-file routine for input and combined files
ERROR	symbol	X	X	X	Address of the user's unrecoverable error routine
IOAREA1	symbol	R	R	R	Address of input/output area; input area for a combined file
IOAREA2	symbol	X	X	X	Address of alternate input/output area; output area for a combined file
IOREG	(r)	X	X		General register (2-13) that contains the address of the current record when processing in two I/O areas. Omit WORKA=YES. Must not be used for a combined file

Table A-3. Summary of DTFCD Keyword Parameters (Part 2 of 2)

Keyword	Specification	Files			Remarks
		Input	Output	Cmbnd	
ITBL	symbol	X		X	Address of input translate table; required when MODE=TRANS is specified
MODE*	BINARY*	Y	Y	Y	Specifies cards are to be read or punched in column binary
	CC	Y	Y	Y	Specifies cards are to be read or punched in compressed code
	STD	Y	Y	Y	Specifies cards are to be read and punched in EBCDIC
	TRANS	Y	Y	Y	For cards to be read or punched in EBCDIC and translated by the table specified in the ITBL or OTBL entry, respectively
OPTION	YES	X	X	X	Specifies an optional file
ORLP	YES			X	Specifies that a read/punch unit file is to be processed in an overlap mode
OTBL	symbol		X	X	Address of output translate table; required when MODE=TRANS is specified
OUBLKSZ	n			R	Specifies the length of IOAREA2 for a combined file
RECFORM*	FIXUNB	R	Y	R	For fixed-length records
	UNDEF*		Y		For undefined records
	VARUNB		Y		For variable-length records
RECSIZE**†	(r)		X		For undefined records; general register (2-13) contains record size
	n	X	X		Specifies record size in bytes and is used with BLKSIZE for multisector I/O on diskette.
SAVAREA	YES	X	X	X	Specifies 72-byte register save area
STUB*	51	X			Stub card read for 51-column cards
	66	X			Stub card read for 66-column cards
TYPEFLE	INPUT	R			For input files
	OUTPUT		R		For output files
	CMBND			R	For combined read/punch file
WORKA	YES	X	X	X	Records are to be processed in work area. Omit IOREG

*Not applicable to 8413 diskette

**Parameter may be changed at run time via DD job control statement.

†Parameter may be changed at run time via DD job control statement only on 8413 diskette operations.

LEGEND:

- R Required
- X Optional
- Y One option required
- Value assumed if keyword not specified

Table A-4. Device-Independent Control Character Codes (Part 1 of 2)

Function	DI Code (Hex.)	Printer			
		0773 and 0778	0770	0768	0776
No-op	00				
Print and space n lines. Note 8 n =		Note 6			
0	10				
1	01				
2	02				
3	03				
4	04				Note 1
5	05				Note 1
6	06				Note 1
7	07				Note 1
8	08				Note 1
9	09				Note 1
10	0A				Note 1
11	0B				Note 1
12	0C				Note 1
13	0D				Note 1
14	0E				Note 1
15	0F			Note 1	
Print and skip to code n. Note 7 n =					
1 (OV)	11		Code 12 (OV)	Chan 9 (OV) Note 5 Note 5	Code 12 (OV)
2	12				
3	13				
4	14				
5	15				
6	16				
7 (HP)	17		Note 4	Chan 15 Note 3	Code 7 (HP) Note 4
8	18	Code 2			
9	19	Code 1 (OV)	Note 2 (OV)		Code 12 (OV)
10	1A	Code 3			
11	1B	Code 4			
12	1C	Code 1 (OV)	Note 2 (OV)	Chan 9 (OV)	Code 12 (OV)
13	1D	Code 5			
14	1E	Code 7 (HP)	Code 7 (HP)	Chan 15 Note 3	Code 7 (HP)
15	1F	Code 7 (HP)	Code 7 (HP)	Chan 15 Note 3	Code 7 (HP)
Space n lines. Note 8 n =		Note 6			
1	51				
2	52				
3	53				
4	54				Note 1
5	55				Note 1
6	56				Note 1
7	57				Note 1
8	58				Note 1
9	59				Note 1
10	5A				Note 1
11	5B				Note 1
12	5C				Note 1
13	5D				Note 1
14	5E				Note 1
15	5F			Note 1	

Table A-4. Device-Independent Control Character Codes (Part 2 of 2)

Function	DI Code (Hex.)	Printer			
		0773 and 0778	0770	0768	0776
Skip to code n. Note 7					
n = 1 (OV)	21		Code 12 (OV)	Chan 9 (OV)	Code 12 (OV)
2	22			Note 5	
3	23			Note 5	
4	24				
5	25				
6	26				
7 (HP)	27		Note 4	Chan 15 Note 3	Code 7 (HP) Note 4
8	28	Code 2			
9	29	Code 1 (OV)	Note 2		Code 12 (OV)
10	2A	Code 3			
11	2B	Code 4			
12	2C	Code 1 (OV)	Note 2	Chan 9 (OV)	Code 12 (OV)
13	2D	Code 5			
14	2E	Code 7 (HP)	Code 7 (HP)	Chan 15 Note 3	Code 7 (HP)
15	2F	Code 7 (HP)	Code 7 (HP)	Chan 15 Note 3	Code 7 (HP)

LEGEND:

OV Overflow code

HP Home paper code

NOTES:

- Line spacing of 4-15 lines on the 0768 printer is accomplished by issuing multiple I/O commands. The commands are issued by data management.
- Code 12 is the primary forms overflow code on the 0770 printer. Code 9 can also be detected as forms overflow code, using the PRTOV macro. If the PRTOV macro is not used, however, code 12 should be used as overflow code, and code 9 should not be placed in the VFB.
- On the 0768 printer, the user must employ both a vertical format buffer and a paper tape loop. Using DI codes 27, 2E, or 2F as control characters has no direct effect on line spacing; this is controlled by what is punched on the paper tape loop. The actual selection of 6 or 8 lines per inch spacing occurs when the operator sets up the form on the printer to register at home paper position and pushes the HP button twice. If channel 14 is used as home paper code on the paper tape loop, this causes printing at 6 lines per inch; using channel 15 results in 8 lines per inch spacing. These two codes should not be intermixed. When a DI code for skip to code 7 is issued, a skip is issued to channel 15 on the 0768 printer - this causes an advance to either channel 14 or 15, whichever is punched on the paper tape loop.
- Code 7 must be used as the home paper code on the 0770 and 0776 printers.
- For the 0768 printer, the system provides codes 2 and 3.
- Line spacing is switch-controlled on the 0773 and 0778 printers. The user issues instructions to the operator via the job control VFB statement, using its FORMNAME and DENSITY parameters, and the operator sets the line rate when the vertical format buffer is loaded. For the 0770 and 0776 printers, line spacing is software-controlled via the DENSITY parameter of the VFB statement.
- Code n specifies channel code CD1 through CD15 for skipping.
- Code n specifies number of lines to be spaced.

Table A-5. Device Skip Code Table

Function	Printer Code Substitution			
	0773 and 0778	0770	0768	0776
Skip to code n, m.				
n, m = 1 (OV)		Code 12 (OV)	Chan 9 (OV)	Code 12 (OV)
2			Note 7	
3			Note 7	
4				
5				
6				
7 (HP)		Note 2	Chan 15 Note 4	Code 7 (HP) Note 2
8	Code 2			
9	Code 1 (OV)	Note 5 (OV)		Code 12 (OV)
10	Code 3			
11	Code 4			
12	Code 1 (OV)	Note 5 (OV)	Chan 9 (OV)	Code 12 (OV)
13	Code 5			
14	Code 7 (HP)	Code 7 (HP)	Chan 15 Note 4	Code 7 (HP)
15	Code 7 (HP)	Code 7 (HP)	Chan 15 Note 4	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 the user has specified.
2. Code 7 must be used as the home paper code on the 0770 and 0776 printers.
3. A skip to code 7 control causes a skip to the home paper code on all printers. On the 0768 printer, the skip is to channel 14 or 15. (Skip to channel 15 is issued to the printer.) On the 0768 printer, the home paper code used on the tape loop sets the number of lines printed per inch. Channel 14 results in 6 lines per inch line spacing and channel 15 results in 8 lines per inch spacing.
4. A skip to channel 15 issued to the 0768 printer causes an advance to the home paper position, regardless of whether channel 14 or 15 is punched in the paper-tape forms control loop.
5. Code 12 is the primary forms overflow control code for the 0770 printer. Code 9 can also be detected as forms overflow code, using the PRTOV macro. If the PRTOV macro is not used, however, code 12 should be used as overflow code, and code 9 should not be placed in the VFB.
6. Data management accomplishes spacing greater than three lines on the 0768 printer by issuing a number of I/O commands. If the user issues several control commands in succession, specifying delayed spacing or skipping, only the last delayed spacing or skipping option is executed.
7. For the 0768 printer, the system provides codes 2 and 3.

Table A-6. Sizes of DTFPR File Table

SAVAREA Keyword Specified	File Table Size in Bytes
No	104
Yes	112

Table A-7. Summary of DTFPR Keyword Parameters

Keyword	Specifications	OUTPUT File	Remarks
BLKSIZE*	n	X	The maximum block size in bytes
CONTROL	YES	X	Specified if CNTRL macro instruction is issued to control line spacing or skipping
CTLCHR	DI	Y	Device-independent control characters
ERROR	symbolic label	X	Address of user's unrecoverable error routine
IOAREA1	symbolic label	R	Address of output area
IOAREA2	symbolic label	X	Address of alternate output area
IOREG	(r)=general register	X	General register (2 through 12) that contains the address of the current record each time a PUT is issued. Register 13 may be used when SAVAREA keyword parameter is used.
OPTION	YES	X	Specifies an optional file
PRAD	n	X	Number of lines (1 to 15) to be spaced
PRINTOV	SKIP	X	Automatic skip to home paper position on printer
	symbolic label	X	Address of user's overflow routine
	YES	X	Specifies execution of PRTOV imperative macros in user's program
RECFORM*	FIXUNB	Y	For fixed-length records
	UNDEF	Y	For undefined records
	VARUNB	Y	For variable-length, unblocked records
RECSIZE	(r)=general register	X	General register (2 through 13) that contains the length of each record when RECFORM=UNDEF is specified
SAVAREA	symbolic label	X	Specifies 72-byte register save area
UCS	OFF	Y	Character mismatches will be ignored.
	ON	Y	Operator is to be notified of character mismatches.
WORKA	YES	X	Process records in work area.

*Parameter may be changed at run time via DD job control statement.

LEGEND:

R Required
X OptionalY One option required
■ Value assumed if keyword not specified

Table A-8. Overflow and Home Paper Control Character Codes

Code	Printer			
	0773 and 0778	0770	0768	0776
Overflow	Code 1	Code 9 Code 12*	Code 9	Code 12
Home paper	Code 7	Code 7	Code 14 Code 15	Code 7

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

Table A-9. Summary of Imperative Macros Used with Magnetic Tape SAM

Macro Call	Function
OPEN	Performs validation checks and initiates magnetic tape file processing
CLOSE	Terminates file processing
PUT	Delivers logical output records to tape SAM
GET	Makes next logical record in input tape file available to user in I/O buffer or work area
SETF	Changes file processing mode for in/out file
TRUNC	Writes short blocks of output records to tape; required for variable blocked records
RELSE	Skips to next block of input records
FEOV	Forces end-of-volume procedures
LBRET	Control returns from user label processing routine
CNTRL	Controls tape unit functions

Table A-10. Summary of DTFMT Keyword Parameters (Part 1 of 3)

Keyword Parameter	Completion	Files			Remarks
		Input	Output	In/Out	
ASCII	YES	X	X	X	Specifies ASCII processing
BKNO	YES	X	X	X	Specifies that block numbers are accepted (without processing) in input or output files.
BLKSIZE*	n	X	X	X	Specifies the number of bytes in each block. If omitted, data management assumes BLKSIZE=256.
BUFOFF	n	X	X	X	For ASCII files; specifies length of block prefix in bytes
CKPTREC	YES	X	—	X	Specifies bypassing of checkpoint blocks on input files. If omitted, checkpoint records are read.
CLRW*	NORWD	X	X	X	Specifies that reel is not to be rewound at close
	RWD	X	X	X	Specifies that reel is to be rewound without interlock at close If omitted, tape is rewound with interlock at close. If specified, REWIND keyword is not specified.
EOFADDR	symbol	R	X	X	Required for input files; specifies address of user's routine for processing end-of-data
ERROPT	IGNORE	X	X	X	Specifies that the user will process input or output block containing parity error as if no error has occurred
	SKIP	X	X	X	Specifies that an input block on which a parity occurs is to be bypassed and not made available to the user. An output block containing a parity error is ignored as if written correctly. If omitted, parity errors transfer control to the user's error routine or to him inline.
ERROR	symbol	X	X	X	Specifies the address of the user's error handling routine. If omitted, errors return inline. Refer to Appendix B.
FILABL*		X	X	X	Specifies an unlabeled file. Refer to Appendix L.
	NSTD	X	X	X	Specifies a nonstandard labeled file; invalid for an ASCII file. Refer to Appendixes F and L.
	STD	X	X	X	Specifies a standard labeled file. Refer to Appendixes F and L.
IOAREA1	symbol	R	R	R	Specifies the address of I/O area for this file, half-word aligned. Required for all files
IOAREA2	symbol	X	X	X	Specifies address of optional secondary I/O area. If specified, either the WORKA or the IOREG keyword must also be specified.

Table A-10. Summary of DTFMT Keyword Parameters (Part 2 of 3)

Keyword Parameter	Completion	Files			Remarks
		Input	Output	In/Out	
IOREG	(r)	X	X	X	Specifies number of general register to be used as an index register. Registers 2 through 12 may be used, and register 13 if SAVAREA is specified. Required when two I/O areas are used without a work area, when records are blocked, or for read-backward processing of variable or undefined records.
LABADDR	symbol	X	X	X	Specifies address of user's routine for processing nonstandard labels or optional standard UHL or UTL. Omitted to bypass user labels on an input file
LENCHK	YES	X	—	—	Specifies checking or insertion of block length in variable records of ASCII files. Requires specification of ASCII=YES and BUFOFF=4
OPRW*	NORWD	X	X	X	Specifies that reels of this file are not to be rewound at open before labels are processed. REWIND is not specified. Assumed if READ=BACK is specified
OPTION	YES	X	X	X	Specifies optional input or output file
READ	BACK	X	—	X	Specifies read-backward input processing; invalid for multivolume file
	FORWARD	X	—	X	Specifies read-forward input processing
RECFORM*	FIXBLK	X	X	X	Specifies fixed, blocked records
	FIXUNB	X	X	X	Specifies fixed, unblocked records
	UNDEF	X	X	X	Specifies undefined record format
	VARBLK	X	X	X	Specifies variable, blocked records
	VARUNB	X	X	X	Specifies variable, unblocked records
RECSIZE*	n	X	X	X	Required for fixed-length, blocked records; specifies the record length in bytes
	(r)	X	X	X	Required for output processing with undefined records; optional for input files. Specifies block length register
REWIND*	NORWD	X	X	X	Specifies no rewind at open, and positioning between tape marks at close
	UNLOAD	X	X	X	Specifies rewind to load point at open, and rewind with interlock at close. Assumed if REWIND, OPRW, and CLRW are not specified
SAVAREA	symbol	X	X	X	Specifies a 72-byte general register save area, full-word aligned
TPMARK*	NO	—	X	X	Specifies no tape mark to be written between labels and data on unlabeled or nonstandard-labeled output files

Table A-10. Summary of DTFMT Keyword Parameters (Part 3 of 3)

Keyword Parameter	Completion	Files			Remarks
		Input	Output	In/Out	
TYPEFLE	INOUT	—	—	R	Specifies file used for input or output processing
	INPUT	X	—	—	Specifies input processing
	OUTPUT	—	R	—	Specifies output processing
VARBLD	(r)	—	X	X	Required for variable-length, blocked records built in I/O area; specifies register for residual buffer space
WORKA	YES	X	X	X	Specifies record processing in work area. Not specified if IOREG is specified

*Parameter may be changed at run time via DD job control statement.

LEGEND:

- R Required
 X Optional
 — Valid
 ■ Data management assumes this value has been specified if the keyword is omitted.

Table A-11. Effects of Job Control VOL and LBL Statements on Data Management OPEN Transient, Standard-Labeled Tape Files (Part 1 of 2)

VOL Statement, Specification of Positional Parameter 1	LBL Statement, Specification of Positional Parameter 2	Resulting Action By Open Transient		Notes
		VSN Field of VOL1 Label	File Serial Number Field of HDR1 Label	
Input Files				1
Blank (statement omitted)	Blank, or statement omitted	No check made	No check made	2
SCRTCH	Blank, or statement omitted	No check made	No check made	2
SCRTCH	VCHECK	No check made	Checks whether FSN in HDR1 label is identical to VSN in VOL1 label	2
Unique VSN	Unique FSN	Checks whether VSN in VOL1 label is identical to VSN specified in VOL statement	Checks whether FSN in HDR1 label is identical to VSN in the VOL1 label of the first volume of the file	—
Unique VSN	VCHECK	Checks whether VSN in VOL1 label is identical to VSN specified in VOL statement	Checks whether FSN in HDR1 label is identical to VSN in the VOL1 label of the first volume of the file	—

Table A-11. Effects of Job Control VOL and LBL Statements on Data Management OPEN Transient, Standard-Labeled Tape Files (Part 2 of 2)

VOL Statement, Specification of Positional Parameter 1	LBL Statement, Specification of Positional Parameter 2	Resulting Action By Open Transient		Notes
		VSN Field of VOL1 Label	File Serial Number Field of HDR1 Label	
Input Files (cont)				1
Unique VSN	Blank, or statement omitted	Checks whether VSN in VOL1 label is identical to VSN specified in VOL statement	No check made	—
Output Files				1
SCRTCH	Blank, or statement omitted	No check made	Creates FSN in HDR1 label, identical to VSN in VOL1 label	2
SCRTCH	VCHECK	No check made	Creates FSN in HDR1 label, identical to VSN in VOL1 label	2
Unique VSN	Unique FSN	Creates VSN in VOL1 label identical to the VSN specified in the VOL statement, or checks whether the VSNs are identical	Creates FSN in HDR1 label identical to VSN in the VOL1 label of the first volume of the file	3
Unique VSN	VCHECK	Creates VSN in VOL1 label identical to the VSN specified in the VOL statement, or checks whether the VSNs are identical	Creates FSN in HDR1 label identical to VSN in the VOL1 label of the first volume of the file	3
Unique VSN	Blank, or statement omitted	Creates VSN in VOL1 label identical to the VSN specified in the VOL statement, or checks whether the VSNs are identical	Creates FSN in HDR1 label identical to VSN in the VOL1 label of the first volume of the file	3

NOTES:

1. Any combination of specifications of the VOL and LBL statements other than those shown for input or output files is invalid and results in the issue of error message DM54. The user must correct his invalid job control specifications and rerun his program.
2. These VOL and LBL statement combinations are valid for single-volume files only.
3. When the user specifies (PREP) following the VSN in the VOL statement for an output file, he must specify any of these three combinations in the VOL and LBL statements.

Table A-12. Summary of DTFSD Keyword Parameters (Part 1 of 2)

Keyword	Specification	Files			Remarks
		Input	Output	In/Out	
ACCESS*	EXC	X	X	X	This DTF: read/update/add use Other jobs: no access
	EXCR	X	X	X	This DTF: read/update/add use Other jobs: read use
	SRD	X			This DTF: read use Other jobs: read/update/add use
	SRDO	X			This DTF: read use Other jobs: read use
BLKSIZE	n=maximum block size	R	R	R	The maximum block size, in bytes
EOFADDR	symbolic label	R		R	Identifies EOF routine
ERROPT	IGNORE	X	X	X	Ignore parity error
	SKIP	X	X	X	Bypass parity error
ERROR	symbolic label	X	X	X	Address of user's unrecoverable error routine
IOAREA1	symbolic label	R	R	R	Address of I/O area
IOAREA2	symbolic label	X	X	X	Address of alternate I/O area
IOREG	(r)=general register	X	X	X	Required if records are processed in the I/O area and records are blocked
LABADDR	symbolic label of user's label routine		X	X	Required if user header or trailer labels are to be created
		X			Required if user header or trailer labels are to be retrieved
LACE	n=lace factor	X	X	X	Specifies factor for record interface
LOCK	NO	X	X	X	At OPEN time, file lock is not to be set. This permits read-only access.
OPTION	YES	X	X	X	Specifies file not always to be processed
RECFORM	FIXBLK	Y	Y	Y	For fixed-length, blocked records
	FIXUNB	X	X	X	For fixed-length, unblocked records; assumed
	VARBLK	Y	Y	Y	For variable-length, blocked records
	VARUNB	Y	Y	Y	For variable-length, unblocked records
RECSIZE	n=number of bytes in record	X	X	X	For fixed-length, blocked records
SAVAREA	symbolic label	X	X	X	Specifies address of save area for contents of general registers
TRLBL	YES	X	X	X	Read or write user trailer labels when CLOSE issued to file. (Specify LABADDR also.)

*Parameter may be changed on DD job control statement.

Table A-12. Summary of DTFSD Keyword Parameters (Part 2 of 2)

Keyword	Specification	Files			Remarks
		Input	Output	In/Out	
TYPEFLE	INOUT			R	For I/O files
	INPUT	X			For input files; assumed
	OUTPUT		R		For output files
UPDATE	YES	X		X	Required if records are to be written back to the same location from which they were read
VARBLD	(r)=general register		X	X	Required for variable-length, blocked records built in output area; register contains number of bytes left in output area.
VERIFY	YES	X	X	X	Check parity after records have been written.
WORKA	YES	X	X	X	Process records in work area.

LEGEND:

- R Required
X Optional
Y One option required
▒ Assumed parameter, if none specified

Table A-13. Summary of DTFDA Keyword Parameters (Part 1 of 2)

Keyword	Specification	Files		Remarks
		Read	Write	
ACCESS*	EXC	X	X	This DTF: read/update/add use Other jobs: no access
	EXCR	X	X	This DTF: read/update/add use Other jobs: read use
	SRD	X		This DTF: read use Other jobs: read/update/add use
	SRDO	X		This DTF: read use Other jobs: read use
AFTER	YES		X	A capacity record on each track is assumed.
BLKSIZE	n=maximum block size	R	R	Length of IOAREA1, in bytes
ERROR	symbolic label	X	X	Address of user error routine
IDLOC	symbolic label	X	X	Address of field containing the record ID
IOAREA1	symbolic label	R	R	Name of I/O area defined by user
KEYARG	symbolic label	X	X	Address of field for key used for key search
KEYLEN	n=key length	X	X	Length of the key in bytes
LABADDR	symbolic label	X	X	Address of user label handling routine

*Parameter may be changed on DD job control statement.

Table A-13. Summary of DTFDA Keyword Parameters (Part 2 of 2)

Keyword	Specification	Files		Remarks
		Read	Write	
LACE	n=lace factor	X	X	Specifies factor for record interlace
LOCK	NO	X	X	Requests that a write-only <i>file lock</i> not be set on a lockable file when opened.
READID	YES	X		Record referenced by ID
READKEY	YES	X		Record referenced by KEY
RECFORM	FIXUNB	Y	Y	For fixed-length records
	VARUNB	Y	Y	Variable-length records
RELATIVE	R	X	X	Relative addressing – record
	T	X	X	Relative addressing – track
SAVAREA	symbolic label	X	X	Specifies the address of a save area for contents of general registers
SEEKADR	symbolic label	R	R	Address of track reference field
SRCHM	YES	X	X	Search multiple tracks. (If specified, file must be allocated on a cylinder basis.)
TRLBL	YES	X	X	User standard trailer labels are to be read or written when CLOSE issued to file. Specify LABADDR also.
TYPEFLE	INPUT	Y	Y	Check standard labels
	OUTPUT	Y	Y	Write standard labels
VERIFY	YES		X	Records are to be check-read
WRITEID	YES		X	Output record is located by means of its relative disc address (ID).
WRITEKEY	YES		X	Output record is located by key.

LEGEND:

- R Required
- X Optional
- Y One option required
- ▒ Assumed parameter, if none specified

Table A-14. Summary of DTFNI and DPCA Keyword Parameters (Part 1 of 2)

Keyword	Specification	Used for DPCA	Files			Remarks
			Input	Output	In/Out	
ACCESS*	EXC	No	X	X	X	This DTF: read/update/add use Other jobs: no access
	EXCR	No	X	X	X	This DTF: read/update/add use Other jobs: read use
	SRD	No	X			This DTF: read use Other jobs: read/update/add use
	SRDO	No	X			This DTF: read use Other jobs: read use
AFTER	YES	No		X	X	A WRITE, AFTER macro will be issued.
BLKSIZE	n=maximum block size	R	R	R	R	Length of IOAREA1, in bytes
EOFADDR	symbolic label	X	X	X	X	Address to which control is passed when end of sequentially processed file is reached
ERROPT	IGNORE	No	Y	Y	Y	Ignore parity error
	SKIP	No	Y	Y	Y	Bypass parity error
ERROR	symbolic label	No	X	X	X	Address of user error routine
IDLOC	symbolic label	No	X	X	X	Address of field containing the record ID
IOAREA1	symbolic label	R	R	R	R	Name of I/O area defined by user; always half-word aligned
IOAREA2	symbolic label	X	X	X	X	Name of alternate I/O area
IOREG	(r) general register	X	X	X	X	Required if records are processed in the I/O area and records are blocked
KEYARG	symbolic label	X	X	X	X	Address of field for key used for key search
KEYLEN	n=key length	X	X	X	X	Length of the key in bytes
LABADDR	symbolic label	No	X	X	X	Address of user label-handling routine
LACE	n=lace factor	X	X	X	X	Specifies the factor for record interlace
OPTION	YES	No	X	X	X	Specifies a file not always to be processed
LOCK	NO	No	X	X	X	Requests that a write-only <i>file lock</i> not be set on a lockable file when opened.
	YES	No	X	X	X	Requests an exclusive lock to be set on a lockable file at open time.
PCAn	symbolic label	No	X	X	X	Specifies the address of partitions (1 to 7) used to subdivide a file
READID	YES	No	X		X	A READ, ID macro will be issued.
READKEY	YES	No	X		X	A READ, KEY macro will be issued.
RECFORM	FIXBLK	Y	Y	Y	Y	Fixed-length, blocked records
	FIXUNB	Y	Y	Y	Y	Fixed-length records, unblocked
	VARBLK	Y	Y	Y	Y	Variable-length, blocked records
	VARUNB	Y	Y	Y	Y	Variable-length records, unblocked

*Parameter may be changed on DD job control statement.

Table A-14. Summary of DTFNI and DPCA Keyword Parameters (Part 2 of 2)

Keyword	Specification	Used for DPCA	Files			Remarks
			Input	Output	In/Out	
RECSIZE	n=number of bytes in record	X	X	X	X	Specifies record size
RELATIVE	R	No	X	X	X	Relative addressing – record
	T	No	X	X	X	Relative addressing – track
SAVAREA	symbolic label	No	X	X	X	Specifies save area for contents of general registers
SEEKADR	symbolic label	No	R	R	R	Address of track reference field
SIZE	n=percent	X	X	X	X	Specifies percentage of total file allocation to be initially assigned to partition
SRCHM	YES	No	X	X	X	Search multiple tracks. (If specified, file must be allocated on a cylinder basis.)
SUBFILE	YES	X	X	X	X	Support division of file partitions into subfiles.
TRLBL	YES	No	X	X	X	Read or write user trailer labels when CLOSE issued to file. (Specify LABADDR also).
TYPEFLE	INOUT	No			R	This file may be used as either an input file or an output file.
	INPUT	No	X			Read and check standard labels for file. If file is processed sequentially, the PUT macro may not be issued for this file unless UPDATE=YES has also been specified in the DTF.
	OUTPUT	No		R		Write standard labels for this file. If file is processed sequentially, the GET macro instruction may not be issued.
UOS	n=percent	X	X	X	X	Specifies percentage of secondary allocation to be used for extending partition.
UPDATE	YES	No	X	X	X	Write records back into same location from which they were read.
VARBLD	(r)=general register	X	X	X	X	Required for variable-length, blocked records that are built in output area; register contains number of bytes left in output area.
VERIFY	YES	No		X		Records are to be check-read.
WORKA	YES	Yes	X	X	X	Process records in work area.
WRITEID	YES	No		X	X	A WRITE, ID macro will be issued.
WRITEKEY	YES	No		X	X	A WRITE, KEY macro will be issued.

LEGEND:

- R Required
X Optional
Y One option required
Assumed parameter, if none specified

Table A-15. Sizes of DTFIS File Table

Specification of IOROUT Keyword Parameter	File Table Size in Bytes
=LOAD	332
=RETRVE	372
=ADD	396
=ADDRTR	396

Table A-16. Summary of DTFIS Keyword Parameters (Part 1 of 2)

Keyword	Specification	File Function				Remarks
		L	A	S	T	
ACCESS*	EXC	S	S	S	S	This DTF: read/update/add use Other jobs: no access
	EXCR			S	S	This DTF: read/update/add use Other jobs: read use
	SRD			S	S	This DTF: read use Other jobs: read/update/add use
	SRDO			S	S	This DTF: read use Other jobs: read use
BLKSIZE	n	R	R	R	R	Specifies data block size
ERROR	symbolic label	O	O	O	O	Address of subroutine to handle errors and exceptions
INDAREA	symbolic label	R	O	O	O	Address of main storage area to contain index
INDEXED	NO	O	O	O	O	Specifies that file is not to be indexed
INDSIZE	n (in bytes)	R	O	O	O	Size of index area in main storage; minimum size is 256 bytes.
IOAREA1	symbolic label	R	R	R	R	Address of I/O area in main storage
IOAREA2	symbolic label	O	O	O	O	Address of a second I/O area in main storage to speed processing
IOREG	(r)=general register			O	O	Contains address of the I/O area in main storage
IOROUT	ADD		R			Insert new records into file.
	ADDRTR		R	S	S	Insert new records and retrieve records randomly or sequentially.
	LOAD	R				Create new file or extend existing file.
	RETRVE			S	S	Retrieve and/or update randomly or sequentially.
KEYARG	symbolic label			O	R	Address of field containing key of desired record
KEYLEN	n	O	O	O	O	Key length in bytes for file. When specified for ASAM files, a sequence check of keys is made.
KEYLOC	n	O	O	O	O	Location, in bytes, of the key within a record. If omitted, KEYLOC=0 for fixed-length records; KEYLOC=2 for variable-length records.

*Parameter may be changed on DD job control statement.

Table A-16. Summary of DTFIS Keyword Parameters (Part 2 of 2)

Keyword	Specification	File Function				Remarks
		L	A	S	T	
LOCK	NO	O	O	O	O	Requests that a write-only <i>file lock</i> not be set on a lockable file when opened.
PCYLOFL	nn (00 to 80)	O				Percentage of cylinder (blocks) available for overflow
RECFORM	FIXBLK	S	S	S	S	Specifies fixed-blocked records
	VARIABLE	S	S	S	S	Specifies variable-blocked records
RECSIZE	n	O	O	O	O	Size, in bytes, of fixed records to be processed
SAVAREA	symbolic label	O	O	O	O	Address of general register save area
TYPEFLE	RANDOM				O	Specifies random file processing function
	RANSEQ			O	O	Specifies random/sequential file processing function
	SEQNTL			O		Specifies sequential file processing function
UPDATE	NO		O	O	O	Prevents inadvertent update
VERIFY	YES	O	O	O	O	Specifies a parity check of data records is to be made after being written to output disc
WORK1	symbolic label	R	R		O	Address of work area for records being loaded, reloaded, extended, or inserted
WORKS	NO			O		No record work area available for sequential retrieval

LEGEND:

- | | | | |
|---|----------------------------|---|--|
| L | File creation or extension | O | Optional |
| A | Record insertion | R | Required |
| S | Sequential retrieval | S | Select one |
| T | Random retrieval | █ | Value assumed if keyword is not specified. |



Table A-17. Summary of DTFPT Keyword Parameters (Part 1 of 2)

Keyword	Completion	Remarks	Use With File Type		EXTRN	Use With Processing Mode		Use With Shifted Codes	Use Without Shifted Codes	Do Not Specify With
			INPUT	OUTPUT		BINARY	STD			
BLKSIZE	n	Required for all files. Specifies maximum length, in bytes, of largest logical record; n is a decimal number in the range 1-4095.	R	R	-	R	R	-	-	-
EOFADDR	symbol	Specifies label of user's end-of-tape processing routine; required for all input files.	R	No	Yes	-	-	-	-	-
EORCHAR	expression	Required for output files with undefined records; specifies end-of-record stop character (delimiter).	No	R	No	No	R	-	-	RECFORM=FIXUNB, MODE=BINARY
ERROR	symbol	Specifies label of user's error routine. If not specified, errors return inline.	O	O	Yes	O	O	-	-	-
FSCAN	symbol	Required, with LSCAN keyword, to specify label of user's figure scan table for punching files with shifted codes.	No	O	Yes	No	O	R	No	MODE=BINARY
FTRANS	symbol	Required, with LTRANS and SCAN, to specify label of user's figure translation table for processing a shifted input file.	O	No	Yes	No	O	R	No	TRANS, MODE=BINARY
IOAREA1	symbol	Required for all files; specifies label of primary I/O buffer.	R	R	Yes	R	R	-	-	-
IOAREA2	symbol	Specifies label of optional secondary I/O buffer. Requires specification of IOREG if work area processing is not specified (WORKA keyword).	O	O	Yes	O	O	-	-	-
IOREG	(r)	Specifies, in mandatory parentheses, the number of the general register to be used as I/O index register. Required when IOAREA2 keyword is specified, but work area processing is not (WORKA keyword).	O	O	-	O	O	-	-	WORKA
LSCAN	symbol	Required, with FSCAN keyword, to specify label of user's letter scan table to punch files with shifted codes.	No	O	Yes	No	O	R	No	MODE=BINARY
LTRANS	symbol	Required, with FTRANS and SCAN keywords, to specify the label of the user's letter translation table for an input file with shifted codes.	O	No	Yes	No	O	R	No	TRANS, MODE=BINARY
MODE	BINARY	Required to specify binary processing mode.	O	O	-	R	No	No	-	RECFORM=UNDEF
	STD	Specifies character (nonbinary) processing mode.	O	O	-	No	O	Yes	Yes	-
OPTION	YES	Specifies optional file processing.	O	O	-	O	O	-	-	-
OVBLKSZ	n	Specifies use and length of oversized I/O buffers for processing fixed, unblocked records containing shifted codes; n is a decimal number ranging from 2 through 4095 and must be at least one byte larger than the BLKSIZE specification.	O	O	-	No	O	O	No	RECFORM=UNDEF, MODE=BINARY
RECFORM	FIXUNB	Specifies fixed, unblocked record format.	O	O	-	R	O	Yes	Yes	-
	UNDEF	Specifies undefined record format (varying length). Records require delimiter (end-of-record stop). Output files require RECSIZE register.	O	O	-	No	O	Yes	Yes	MODE=BINARY
RECSIZE	(r)	Specifies, in mandatory parentheses, the number of the general register to be used to refer to length of undefined records. Required for output files; optional for input.	O	R	-	No	R	Yes	Yes	RECFORM=FIXUNB, MODE=BINARY
SAVAREA	symbol	Specifies label of 72-byte storage area, fullword-aligned, in which data management saves contents of user's general registers during execution of imperative macros. If not specified, data management expects save area address in register 13.	O	O	Yes	O	O	-	-	-

Table A-17. Summary of DTFPT Keyword Parameters (Part 2 of 2)

Keyword	Completion	Remarks	Use With File Type		EXTRN	Use With Processing Mode		Use With Shifted Codes	Use Without Shifted Codes	Do Not Specify With
			INPUT	OUTPUT		BINARY	STD			
SCAN	symbol	Specifies label of user's input file shift code scan table. Required for input files with shifted codes (MODE=STD); FTRANS and LTRANS keywords must also be specified. Optional for software character deletion in binary mode. Optional, with TRANS keyword, for character deletion in either mode.	O	No	Yes	O	O	R	Yes	-
TRANS	symbol	Specifies label of user's translation table, for any file but a shifted input file.	O	O	Yes	O	O	Yes	Yes	FTRANS, LTRANS
TYPEFLE	INPUT	Specifies an input file - read only.	R	No	-	O	O	Yes	Yes	-
	OUTPUT	Required to specify an output file - punch only.	No	R	-	O	O	Yes	Yes	-
WORKA	YES	Specifies double buffering via work areas, which user must specify with each GET or PUT macro. Ignored if IOREG keyword is specified.	O	O	-	O	O	Yes	Yes	IOREG

LEGEND:

- R Required to be specified, explicitly or by default.
- O Specification is optional.
- Default value assumed by data management if keyword is not specified.
- Not pertinent.

NOTE:

A "Yes" entry in the column headed "EXTRN" indicates that the assembler generates an EXTRN pseudo-op for the symbolic label specified. This means that the user's coding that defines the subroutine, table, or main storage area in point may be assembled separately from the DTFPT macro call. The user will need an ENTRY for each EXTRN.

→ *Table A—18. Relative Disc Address (ID) Returned to IDLOC Field by WAITF Macro after a READ or WRITE Macro Is Issued to a Direct Access File*

Imperative Macros	DTF Keyword Parameters	ID Returned by WAITF Macro	Form* of ID Returned if:	
			RELATIVE=R	RELATIVE=T
READ,KEY	READKEY=YES	ID of the block retrieved	rrrr	tttr
WRITE,KEY	WRITEKEY=YES	ID of the block previously retrieved		
READ,ID	READID=YES	ID of the <i>next</i> block in the file	rrrr	tttr
WRITE,ID	WRITEID=YES			
WRITE,AFTER	AFTER=YES			
WRITE,AFTER,EOF	AFTER=YES	none	—	—
WRITE,RZERO	AFTER=YES	none	—	—

*Discontinuous binary, where:

rrrr is the 4-byte relative block number.

ttt is the relative track number.

r is the absolute block number on the relative track.

Table A-19. Summary of DTFIR Keyword Parameters

Keyword	Specification	Keyed Operations		Nonkeyed Operations		Restrictions	Remarks
		Input	Output	Input	Output		
ACCESS*	EXC	O	O	O	O		This DTF: read/update/add use Other jobs: no access
	EXCR	O	O	O	O		This DTF: read/update/add use Other jobs: read use
	SRD	O	O	O	O		This DTF: read use Other jobs: read/update/add use
	SRDO	O	O	O	O		This DTF: read use Other jobs: read use
ADD	YES	O	N	X	X	Used only with keyed operations	Indicates new records are to be added to a file
BFSZ*	n	R	R	R	R	Always required	Supplies data buffer size
EOFA	symbol	R	N	R	N	Required if MODE=SEQ	Address of end of file routine
ERRO	symbol	O	O	O	O		Address of error handling routine
INDA	symbol	R	R	X	X	Used only with keyed operations	Address of main storage area to contain index
INDS**	n	R	R	X	X	Used only with keyed operations	Indicates size of index area
INDX	YES	R	R	X	X	Used only with keyed operations	Indicates keyed operations
IOA1	symbol	R	R	R	R	Always required	Address of primary buffer
IOA2	symbol	O	O	O	O	Not permitted when UPDT= YES	Address of secondary buffer
IORG	(r)=general register	O	X	O	O	Not permitted when WORK= YES	Indicates I/O buffer index register
KARG	symbol	R	R	X	X	Used only with keyed operations	Address of field containing key of desired record
KLEN**	n	R	R	X	X	Used only with keyed operations	Indicates key length
KLOC**	n	R	R	X	X	Used only with keyed operations	Indicates the byte number location of the key within a record
MODE	SEQ	S	S	S	S		Sequential file processing (default)
	RAND	S	S	S	S		Random file processing
OPTN	YES	O	O	O	O		Optional file for input
RCSZ*	n	R	R	R	R	Always required	Indicates record size
SKAD	symbol	X	X	R	R	Required if MODE=RAND	Address of seek address field
SOCK	YES	N	O	X	X	Used only with keyed operations	Indicates that sequence of keys for ordered load should be verified
TYPE	INPUT	R	X	R	X		Indicates input file type (default)
	OUTPUT	X	R	X	R		Indicates output file type
UPDT	YES	O	N	O	N	Input only	Indicates update capability
VERFY	YES	O	O	O	O	Used for TYPE=INPUT and permitted only when ADD= YES or UPDT= YES.	Read/check of output records to be performed
VMNT	ONE	O	O	O	O	Not permitted when MODE= RAND unless INDX= YES	Defines file to be processed with only one volume inline at any time
WORK	YES	O	R	O	O	Also required for keyed operations when TYPE=INPUT and ADD= YES. Not permitted in conjunction with IORG.	Indicates that the record processing is in a work area.

* Parameter may be changed on DD job control statement.
**Parameter may be changed on DD job control statement for index mode only.

LEGEND:

- N Valid only, if SETF macro was used to change TYPE
- O Optional
- R Required
- S Select one
- X Invalid specification

Table A-20. Summary of DTFMI Keyword Parameters (Part 1 of 2)

Keyword	Specification	Keyed Operations	Nonkeyed Operations	Restrictions	Remarks
ACCESS*	EXC	S	S		This DTF: read/update/add use Other jobs: no access
	EXCR	S	S		This DTF: read/update/add use Other jobs: read use
	SRD	S	S		This DTF: read use Other jobs: read/update/add use
	SRDO	S	S		This DTF: read use Other jobs: read use
BFSZ*	n	R	R	Always required	Supplies data buffer size
EOFA	symbol	O	O		Address of end of file routine
ERRO	symbol	O	O		Address of error handling routine
INDA	symbol	R	X	Used only with keyed operations	Address of index buffer
INDS**	n	R	X	Used only with keyed operations	Indicates size of index buffer
IOA1	symbol	R	R	Always required	Address of primary data buffer
IOA2	symbol	O	O	Only allowed with sequential output or unkeyed sequential input	Address of secondary data buffer
IORG	(r)=general register	O	O	Not permitted when WORK=YES	Indicates I/O buffer index register
KARG	symbol	R	X	Used only with keyed operations	Address of field containing key of desired record
KEYn**	n	R	X	Used only with keyed operations	Indicates key length key location, and whether duplicate keys or changes to keys are allowed
LOCK	NO	O	O		Indicates file lock
MODE	SEQ	S	S		Sequential file processing (default)
	RAN	S	S		Random file processing
	RANH	S	S		Random file processing (hold sequential position)
OPTN	YES	O	O		Optional file
PROC	KEY	S	X		Keyed (index and data) operation (default)
	UNK	X	R		Nonkeyed (data) operation
RCB	NO	O	O	Required if delete is to be allowed	Record control byte

Table A—20. Summary of DTFMI Keyword Parameters (Part 2 of 2)

Keyword	Specification	Keyed Operations	Nonkeyed Operations	Restrictions	Remarks
RCFM	FIX	S	S		Fixed length records (default).
	VAR	S	S		Variable length records
RCSZ*	n	R	R	Always required	Indicates record size
RETR	INF	S	S	Update not allowed	Retrieval for information (default)
	MOD	S	S		Retrieval for modification
	REP	S	S		Retrieval for replacement
SKAD	symbol	R	R	Always required	Address of seek address field
VERFY	YES	O	O		Check parity of output records after they have been written
VMNT	ONE	O	O	Nonkeyed random or random output operations not allowed	Defines file to be processed with only one volume online at a time
WORK	YES	O	O	Required for all output, keyed update, and delete functions	Indicates that record processing is in a work area

*Parameter may be changed on DD job control statement.

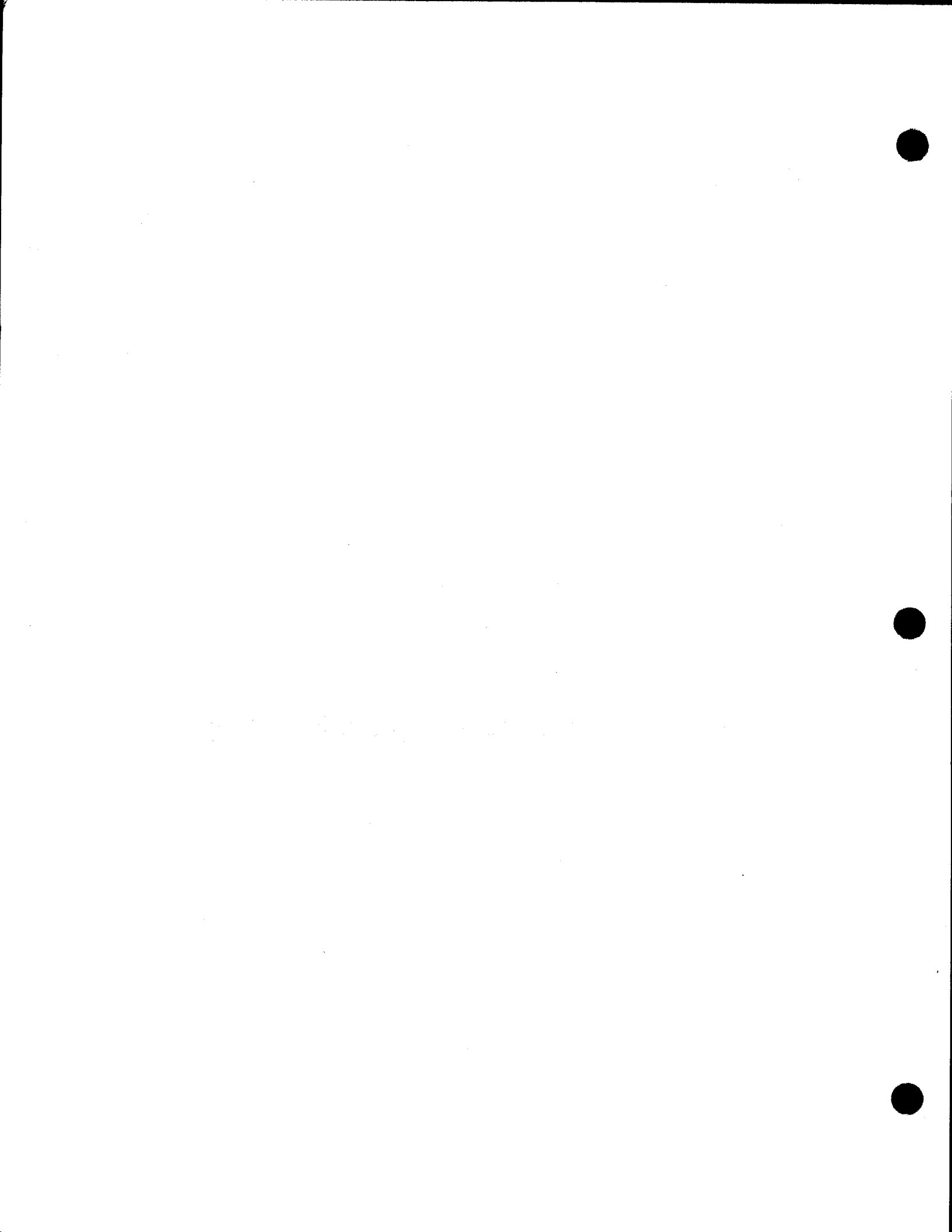
**Parameter may be changed on DD job control statement for indexed file only.

LEGEND:

O Optional
R Required
S Select one
X Not used



Appendix B. Error and Exception Handling



GENERAL

All programs using the services of SPERRY UNIVAC Operating System/3 (OS/3) Data Management execute imperative macro instructions to obtain specific processing. OS/3 data management performs part of the desired data manipulation itself, but frequently calls on other OS/3 system programs (such as the physical IOCS or disc space management) to perform other parts of the task. Most of the time, the desired service is performed exactly as requested, and control is returned to the user inline with no need to issue messages to the system console or to the log. Sometimes, however, errors or exceptions to desired performance occur; these may be detected by data management or the other system programs at various points in processing.

OS/3 data management is responsible for noting all errors and exceptions reported to it by the other system programs, as well as for testing, within itself, for other types of errors or exceptions. When any such condition is detected, OS/3 data management will always:

- make appropriate entries in certain fields of the DTF file table, which the user may address in order to learn of exceptions and errors and take the proper course of action when control returns to him;
- log and display messages that call for operator intervention or are helpful in after-the-fact tracing of the user program action;
- branch to an error/exception routine in the user's program.

RETURN OF CONTROL

The design policy of OS/3 data management is never to terminate a user program. This means that data management will always return control to the user after detecting an error or exception. If the user provides the address of an error/exception routine in his DTF macro instruction, data management returns control to this address for all conditions of error or exception. If he does not provide this address, data management returns control to the user inline, at the next sequential instruction after the macro call. Retries by PIOCS have already been performed to correct the error condition at this point of processing. ←

ERROR FLAGGING PROCEDURES

All OS/3 data management programs set bits in a special field of the DTF file table to serve as error flags, providing the user with particular information on the error. Disc and tape programs set the bits in this field and then call the logging transient; card and printer modules go directly to the logging transient. When an error is detected during the execution of a data management transient routine, the logging transient is called after the setting of the error flag bits is completed or bypassed.

filenameC

The error flag field of the DTF file table is designated *filenameC* and is addressed by concatenating the EBCDIC character "C" to the 7-byte logical *file name*. *filenameC* is one byte long in the DTFCD, DTFPT, and DTFPR file tables and four bytes long in the DTFs for disc and tape files: it is full-word aligned in all DTFs. Each of the bits in *filenameC* has its own significance as a status or error flag, according to the access method in use. The user has the responsibility of providing the coding to test these bits and to take action appropriate to the condition reported. If he has provided an error/exception handling routine, data management returns control to this in all events other than successful, unexceptional performance of the requested function. In the absence of such a routine, control invariably returns to the user inline. The user can avoid the need to check *filenameC* at the inline return points by including the necessary coding in his error routine. Table B—1 lists the significance of the bits set to 1 by data management in the DTF file tables for certain error and exception conditions encountered in processing all but paper tape files. Table B—2 summarizes the meanings of these bits in *filenameC* of the DTFPT macro.

Table B-1. Significance of Bits in filenameC (Part 1 of 4)

BYTE 0					
Bit	DTFMI DTFIR DTFIS	DTFSD, DTFDA, DTFNI	DTFMT	DTFPR	DTFCD
0	Last block on track accessed	→	Reserved	Line truncated (data too large)	Record size invalid (too large or too small)
1	Invalid ID	→	Reserved	Invalid control character	Reserved
2	Invalid DTF	Invalid PCA/DTF	Invalid DTF	Character mismatch	Validity check (unique unit error)
3	Hardware error				→
4	Error found in OPEN				→
5	Error found in CLOSE				→
6	Invalid macro sequence				→
7	Reserved	(DTFSD: reserved) WAITF required	Reserved	Record size invalid (too small)	Reserved

Table B-1. Significance of Bits in filenameC (Part 2 of 4)

BYTE 1			
Bit	DTFMI DTFIR DTFIS	DTFSD, DTFDA, DTFNI	DTFMT
0	I/O completed	→	→
1	Unrecoverable error	→	→
2	Unique unit error	→	→
3	Record not found	→	Reserved
4	Unit exception	→	→
5	Wrong length found	→	Reserved
6	End of track	→	Reserved
7	End of cylinder	→	Reserved

Table B-1. Significance of Bits in filenameC (Part 3 of 4)

BYTE 2			
Bit	DTFMI DTFIR DTFIS	DTFSD, DTFDA, DTFNI	DTFMT
0	Command rejection		
1	Intervention required		
2	Output parity check		
3	Equipment check		
4	Data check		
5	Overrun		
6	STOP state		Word count = zero
7	Device check		Data converter check (7-track only)

Table B-1. Significance of Bits in filenameC (Part 4 of 4)

BYTE 3			
Bit	DTFMI DTFIR DTFIS	DTFSD, DTFDA, DTFNI	DTFMT
0	Invalid record size		
1	Logical end of file		
2	File space exhausted (DTFIS) Logical end of volume (DTFIR and DTFMI)	Logical end of volume	
3	Processing inhibited	Invalid subfile	Wrong length error
4	Invalid index	Reserved	Reserved
5	Sequence error (DTFIS and DTFIR only)	Reserved	Reserved
6	Duplicate record	Reserved	Reserved
7	ADD rejected (DTFIS only)	Reserved	Reserved

Table B-2. Significance of Bits in filenameC, DTFPT Declarative Macro

Bit	Hexadecimal if Set Alone	Binary if Set Alone	Significance for DTFPT File	Other Bits Set (Note 1)	Data Management Error Messages Issued to Log
0	80 (Note 2)	1000 0000	DTF error	4	DM61: INVALID DTF FIELD, PARAMETER, OR PARAMETER COMBINATION
1	40	0100 0000	Wrong length error	—	DM25: WRONG LENGTH ERROR DETECTED
2	20	0010 0000	Unique (parity) error	—	None
3	10	0001 0000	Unrecoverable error	— (Note 4)	DM23: UNRECOVERABLE I/O ERROR DETECTED
4	08	0000 1000	Error detected in OPEN	—	None
5	04	0000 0100	Error detected in CLOSE	—	None
6	02	0000 0010	Invalid imperative macro	—	DM13: ATTEMPTED ACCESS TO AN UNOPENED FILE, or: DM14: INVALID IMPERATIVE MACRO/MACRO SEQUENCE
7	01	0000 0001	Invalid record size	4 (Note 3) —	DM17: INVALID BLOCKSIZE SPECIFIED, or: DM18: RECORD SIZE INVALID

NOTES:

1. The "Other Bits Set" column shows only those bits invariably set by paper tape data management. Others may also be set: for example, to indicate which errors are detected during OPEN or CLOSE processing.
2. Bit 0 is never set alone; bit 4 is always set with it. The hexadecimal value of the *filenameC* byte is then 88.
3. When bit 4 is set with bit 7, the hexadecimal value of the *filenameC* byte is 09.
4. When an unrecoverable error is detected during OPEN processing, bit 4 is also set with bit 3, and *filenameC* contains hexadecimal 18. When detected during CLOSE processing, bit 5 is set with bit 3, and *filenameC* contains hexadecimal 14.

Summary of Data Management Error Processing for Paper Tape Files

■ DTF error

Bit 0 is set (with bit 4) when the user has:

1. omitted the BLKSIZE keyword or specified it improperly;
2. omitted the EOFADDR keyword for an input file;
3. omitted the IOAREA1 keyword; or
4. specified an invalid address in the DTF.

- Wrong length error

- Bit 1 is set for an input file with undefined records when data management has filled the I/O buffer with a number of bytes that equals the BLKSIZE specification but has not detected the *end-of-record stop* character.
- Bit 1 is also set for an input file with fixed, unblocked records when the last record on the tape is not full-sized—that is, the number of bytes of data yielded by the final record, stripped of *shift characters* and *deletes*, does not equal the BLKSIZE specification.

- Unique (parity) error

Bit 2 is set only for an input file processed in character mode (MODE=STD); the file must have been created with a parity track punched on tape, and the *0920 paper tape subsystem* must have its *program connector* board wired to check the parity track for odd or even parity on input. This bit is set to indicate detection of a parity error in one or more characters on tape. Furthermore, each character on which a parity error is detected has its most significant bit set to binary 1.

- Unrecoverable error

Bit 3 is set when an unrecoverable hardware or software error is detected and the operator has replied "U" to the message usually issued by PIOCS to indicate that the *0920 paper tape subsystem* is in the stop state. Under certain conditions, the *error detected in OPEN* or *error detected in CLOSE* flag (bit 4 or 5) may also be set.

- Error detected in OPEN

- Bit 4 is set when any errors are detected during the processing performed by the OPEN transients. The file is not opened and may not be processed. It is not necessary to issue the CLOSE macro to the file. Other bits may be set to indicate which error was detected; for example:
 1. Bit 0 is set when the DTF is in error; data management issues error message DM61.
 2. Bit 3 is set when the error detected in OPEN is an unrecoverable error. Data management issues error message DM17.
 3. Bit 7 is set when the BLKSIZE or OVBLKSZ specification is not a positive decimal integer in the range 1—4095, or the OVBLKSZ specification does not exceed block size. Data management issues error message DM17.
- Bit 4 is set for an *optional file* when the user has omitted to specify the OPTION keyword in the DTFPT declarative macro and one of the following occurs:
 1. The user has specified the OPT positional parameter in the job control *DVC statement* for the file, which is not available at execution time; or
 2. The user has not provided a job control *device assignment set* for the file.

Data management issues error message DM21, INVALID OR MISSING DEVICE ASSIGNMENT.

- Error detected in CLOSE

Bit 5 is set when errors are detected during the processing performed by the CLOSE transients. Other bits may also be set to indicate which error was detected. The CLOSE processing goes to completion, and the user may reopen the file.

- Invalid imperative macro

- Bit 6 is set when the user issues an inappropriate imperative macro to process his file. Data management issues error message DM14.
- Bit 6 is also set if the user issues any imperative macro except OPEN to an unopened file (including one not opened because an error was detected during OPEN processing) or when the user has omitted to specify the OPTION keyword for an optional file that is not available or assigned at execution time, and data management encounters an imperative macro issued to the file in his program. Data management issues error message DM13.

- Invalid record size

Bit 7 is set only when the user is processing an output paper tape file containing undefined records and places a number in the mandatory RECSIZE register that is negative, zero, or larger than his BLKSIZE specification (minus one byte). The upper limit is 4094 bytes. Data management issues error message DM18.

Recovery from Parity Errors in Input Paper Tape Files

When a parity error is detected in reading an input paper tape file, the physical IOCS issues a standard message to the operator. He is able to move the paper tape back to the beginning of the error record and to retry the read. If the retry is not successful, data management takes the error processing actions set forth in the preceding paragraph; the user's recovery options depend on whether his data contains shifted codes.

- Data not shifted

The user may:

- stop processing records and close his file;
- continue processing the record by branching to the normal return point (at the address contained in register 14); or
- store register 14 and issue another GET macro to skip the record containing bad parity and read in the next. If one next record is free of parity errors, the user may restore register 14 and resume processing.

- Data contains shift codes

Data management has not performed shift processing on the error record; it has neither removed the shift codes, removed software delete characters, nor translated intervening data. The error record remains in the I/O area. The most significant bit of each byte containing a character with bad parity has been set to binary 1. The user may:

- stop processing records and close the file;
- skip the record containing bad parity and read in the next; or
- deal with the erroneous record in his error routine. This option involves:
 1. Locating the record, either by means of the IOREG register or (if he has specified two I/O areas but no IOREG register) via *filenameD*. He must not change the contents of *filenameD*.

2. Locating each character of the record that has a parity error and resetting its most significant bit to binary 0.
3. Removing software delete characters and compressing the record.
4. Removing shift codes and translating the intervening data, further compressing the record, until it is left-justified in the I/O buffer.
5. If he has specified work area *double-buffering*, moving the record to the work area.

The user applies his SCAN table as operand 2 of the BAL *translate and text* (TRT) instruction, and his FTRANS and LTRANS tables as operand 2 of the BAL *translate* (TR) instruction, his data serving as operand 1.

SYSTEM ERROR MESSAGES

In OS/3, system error messages are contained in a general file of canned messages and are listed or displayed under control of the OS/3 supervisor. They are documented in the system messages programmer/operator reference, UP-8076 (current version).

Data Management Error Messages

When OS/3 data management detects a loggable error, it acts through a logging transient routine to provide the supervisor with the proper code for the specific message to be logged; the transient picks up this code from an internal error code field in the DTF file table, which is addressable by the user program at *filenameE* in the file tables.

The internal error code is, in hexadecimal, the numeric component of the designation of the data management error message issued to the system log; e.g., 10_{16} is the internal error code corresponding to error message DM10.



Appendix C. Peripherals



Table C-1. SPERRY UNIVAC Card Reader Subsystem Characteristics (Part 1 of 2)

Characteristic	Description
0717 Card Reader Subsystem	
Card orientation (80-, 66-, and 51-column cards)	Face in, with column 1 leading, and row 9 down
Card rate	500 cpm (maximum)
Read technique	Dual redundant, solar cell technique using photo transistors. Column 0 amplifier checking
Read modes	Image mode – 160 six-bit characters per card Translate mode – 80 characters per card Available code – 8-bit EBCDIC
Read station sensing	Column by column
Hopper capacity	2400 cards
Stacker capacity	2000 cards
0716 Card Reader Subsystem	
Card orientation (80-, 66-, and 51-column cards)	Face in, with column 1 leading and row 9 down (Model available to read 96-column cards)
Card rate	600 or 1000 cpm
Read technique	Dual redundant, solar cell technique using photo transistors. Column 0 amplifier checking
Read modes	Image mode – 160 six-bit characters per card Translate mode – 80 characters per card Three available codes: <ul style="list-style-type: none"> ■ 8-bit ASCII ■ 8-bit EBCDIC (required) ■ Compressed code
Read station sensing	Column by column
Hopper capacity	2400 80-column cards 2000 96-column cards
Stacker capacity Normal (stacker 2) Reject (stacker 1)	2000 cards (stacker 2) 2000 cards (stacker 1)
0719 Card Reader Subsystem	
Card orientation (80-, 66-, and 51-column cards)	Face down, column 1 to left and row 9 facing away
Card rate	300 cpm
Read technique	Two columns of photosensitive sensors and light-emitting diodes Dual redundant Column amplifier checking

Table C-1. SPERRY UNIVAC Card Reader Subsystem Characteristics (Part 2 of 2)

Characteristic	Description
0719 Card Reader Subsystem (cont)	
Read modes	Image mode: 160 six-bit characters per card Translate mode: 80 characters per card
Read station sensing	Column by column
Hopper capacity	1000 cards
Stacker capacity Normal Reject	1000 cards 1000 cards

Table C-2. SPERRY UNIVAC Card Punch Subsystem Characteristics (Part 1 of 2)

Characteristic	Description
0603 Card Punch Subsystem	
Media	80-column cards
Punch mode	2-column serial
Check mode	Punch motion check
Feed mode	On demand
Punch rate	75 cpm (full card) 160 cpm (28 columns only)
Input capacity	700 cards
Output capacity	700 cards (primary stacker) 100 cards (reject stacker)
Reading	Optional
Read rate	160 cpm
Punch translation Image mode Translate mode Available code: EBCDIC	160 six-bit characters per card 80 characters per card
0604 Card Punch Subsystem	
Media	80-column cards
Punch mode	Row
Check mode	Read of punched data
Feed mode	On demand
Punch rate	250 cpm
Input hopper capacity	1000 cards

Table C-2. SPERRY UNIVAC Card Punch Subsystem Characteristics (Part 2 of 2)

Characteristic	Description
0604 Card Punch Subsystem (cont)	
Output stacker capacity	1000 cards (normal and select stacker)
Reading	Optional
Read rate	250 cpm
Punch translation Image mode Compressed mode	160 six-bit characters per card 80 characters per card

Table C-3. SPERRY UNIVAC Printer Subsystem Characteristics (Part 1 of 4)

Characteristic	Description		
0773 Printer Subsystem			
Print speed	110 to 680 lpm, depending on character contingencies:		
	Available character sets	Number of bands per set	Nominal print rate (lpm)
	48-character business	5	500
	63-character print	4	400
	48/16-character print	4	400/670
	85-character print	3 (plus 1 character)	310
	128-character special	2	217
	96/(16-16)-character ASCII 256-character special	2 1	217/500 114
Line advance timing	8.75 ms for spacing first line; for skipping each subsequent line: 3.33 ms at 6 lpi 2.50 ms at 8 lpi	8.75 ms for spacing first line; for skipping each subsequent line: 2.22 ms at 6 lpi 1.67 ms at 8 lpi	8.75 ms for spacing first line; for skipping each subsequent line: 1.67 ms at 6 lpi 1.25 ms at 8 lpi
Number of print positions	120 print positions (columns) by standard printer; 132 or 144 columns by feature		
Form advance control	Vertical format buffer		
Line advance rate	Single space only, 22 inches per second		
Form dimensions	3 to 18.75 inches wide 1 to 24 inches long		
Character set	Standard 48-character set. Any number of characters, up to 256, with options		
Horizontal spacing	10 characters per inch		
Vertical spacing	6 or 8 lines per inch, operator-selectable		

Table C-3. SPERRY UNIVAC Printer Subsystem Characteristics (Part 2 of 4)

Characteristic	Description		
0770 Printer Subsystem			
Print speed	0770-00	0770-02	0770-04
	112 to 1435 lpm, depending on character contingencies	213 to 2320 lpm, depending on character contingencies	337 to 3000 lpm, depending on character contingencies
	112 lpm - 384 contiguous characters	213 lpm - 384 contiguous characters	337 lpm - 384 contiguous characters
	800 lpm - 48 contiguous characters	1400 lpm - 48 contiguous characters	2000 lpm - 48 contiguous characters
	1435 lpm - 24 contiguous characters	2320 lpm - 24 contiguous characters	3000 lpm - 24 contiguous characters
Line advance timing	Advance and print	Time in ms	
		6 lpi	8 lpi
	1 line 2 lines 3 lines n+1 line	120.0 127.6 135.2 120 + 7.6n	118.0 123.7 129.4 118 + 5.7n
Number of print positions	Full print width of 132 print positions placed anywhere on a 16.5-inch form. With 22-inch form, only central 13.2-inch portion can be used (160 print positions with feature).		
Form advance control	Vertical format buffer		
Line advance rate	50 ips	75 ips	100 ips
Form dimensions	Continuous forms with standard edge sprocket-holes from 4 to 22 inches in width. Carbons may be attached or unattached with multicopy forms to a maximum of six parts. Recommended pack thickness not to exceed 0.0155 inch for high quality print.		
Character set	Standard 48-character set. Any number of characters up to 384 with options		
Horizontal spacing	10 characters per inch		
Vertical spacing	6 or 8 lines per inch, as determined by program		

Table C-3. SPERRY UNIVAC Printer Subsystem Characteristics (Part 3 of 4)

Characteristic	Description				
0768 Printer Subsystem					
Print speed	0768-00	0768-02		0768-99	
	900 through 1100 lpm	840 through 2000 lpm		1200 through 1600 lpm	
Line advance timing	11.5 + 5.1 (n-1) ms - 6 lines per inch 11.5 + 5.7 (n-1) ms - 8 lines per inch where: n = number of lines advanced				
Number of print positions	132 character print positions including spaces				
Form advance control	Vertical format buffer and loop control; up to 132 lines per command				
Line advance rate	25 ips				
Form dimensions	4 to 22 inches wide 1 to 22 inches long				
Character set	0768-00 63 characters 0768-02 94 characters 0768-99 132 characters				
Horizontal spacing	10 characters per inch				
Vertical spacing	6 to 8 lines per inch				
0776 Printer Subsystem					
Print Speed	Available character sets	Character sets per band	Nominal print rate (lpm)		
			0776-00, 01	0776-02	0776-03
	384	1	115	150	145
	192	2	225	290	280
	128	3	325	420	400
	96	4	420	540	520
	64	6	600	750	730
	48	8	760	940	900
	32	12	1030	1250*	1220
	24	16	1090*	1250*	1250*
*For duty cycle reasons, maximum speed in lpm is limited by a minimum time between consecutive line feeds: 55 ms for the 0776-00, 01 and 48 ms for the 0776-02, 03.					
Line advance timing	Advance and print		Time in ms		
			6 lpi	8 lpi	
	1 line	16	14.2		
	2 lines	23.6	19.9		
3 lines	31.2	25.6			
4 lines	38.8	31.3			
5 lines	46.4	37			
n+1 lines	16+7.6n	14.2+5.7n			
Number of print positions	136 character print positions including spaces				

Table C—3. SPERRY UNIVAC Printer Subsystem Characteristics (Part 4 of 4)

Characteristic	Description		
0776 Printer Subsystem (cont)			
Form advance control	Vertical format buffer		
Line advance rate	22 inches per second		
Form dimensions	4 to 18.75 inches wide 1 to 24 inches long		
Character set	Standard 48-character set. Any number of characters up to 384 with options		
Horizontal spacing	10 characters per inch		
Vertical spacing	6 or 8 lines per inch, as determined by program		
0778 Printer Subsystem			
Print speed	240 to 560 lpm, depending on character contingencies, at 6 lines per inch (lpi) (2.36 lines per cm), and single line spacing:		
	Available character set	Nominal print rate (lpm)	
	48-character business	300/500	
	64-character print	415	
	48/16 character print*	415/560	
	128 character print	240	
	335 character print*	335	
Line advance timing (in milliseconds)	Number of lines	6 lpi (2.36 lpcm)	8 lpi (3.15 lpcm)
	single	35 ms	35 ms
	double	53 ms	51 ms
	triple	61 ms	57 ms
Number of print positions	120 print positions (columns) per line; 136 columns by feature.		
Number of characters	Standard 48- or 64-character set, with five sets on a 240-character band; or up to 256 characters with expanded character set control feature; 48 character set band repeats five times resulting in 240 characters; 64 character set band repeats four times resulting in 256 characters.		
Forms advance control	Vertical format buffer		
Line advance rate	Single space only, 22 inches (55.88 cm) per second (slew rate).		
Ribbon feed	Bidirectional, self-reversing; ribbon removal without rewinding		
Ribbon type	Fabric or plastic film		
Codes	EBCDIC, ASCII, or any 8-bit code		
Form dimensions	Continuous single-part and multipart (up to six parts or up to 0.018 inch (0.457 mm) thick) with standard edge sprocket holes. Printer can also accept continuously sprocketed, 1-part card stock forms of weights typically used for punch cards, postcards, or offset masters. Form widths from 4.0 inches (101.60 mm) to 18.75 inches (476.2 mm) and lengths up to 24 inches (609.6 mm) can be accommodated. Forms longer than 17 inches (431.8 mm) can be run with casework door open, but noise level increases with door open.		
Horizontal spacing	10 characters per inch (2.54 mm per print position)		
Vertical spacing	6 lpi (4.23 mm per line) or 8 lpi (3.17 mm per line), operator-selectable		

*The "16" array is commonly a numeric subset. Extra 16 arrays are included in the 96/16-16 arrangement to make up a total number of 256 characters in a band.

Table C-4. SPERRY UNIVAC Disc Subsystem Characteristics

Characteristic	Description									
	8411 Disc Subsystem	8413 Diskette Subsystem	8414 Disc Subsystem	8415 Disc Subsystem	8416 Disc Subsystem	8418 Disc Subsystem	8424 Disk Subsystem	8425 Disk Subsystem	8430 Disc Subsystem	8433 Disc Subsystem
Data capacity (8-bit bytes)	7.25 million	242,944 bytes (using tracks 1-73 for data)	29.17 million	33.1 million per track	28.95 million	28.9 million or 57.9 million	58.35 million	58.35 million	100 million	200 million
Number of disc units	1 to 8	2 to 4	2 to 8	1 to 2	2 to 8	2 to 8	1 to 4	2 to 8	1 to 8 (with optional feature up to 16)	1 to 8 (with optional feature up to 16)
Disc/diskette speed	2400 rpm	360 rpm	2400 rpm	2800 rpm	2800 rpm	2800 rpm	2400 rpm	2400 rpm	3600 rpm	3600 rpm
Rotation period (ms/rotation)	25	166.7	25	21.5	21.5	21.5	25	25	16.7	16.7
Data bit rate	1.25 MHz	.250 MHz	2.5 MHz	5.0 MHz	5.0 MHz	5.0 MHz	2.5 MHz	2.5 MHz	6.45 MHz	6.45 MHz
Bit density	1100 ppi	3268 ppi	2200 ppi	4040 fixed 4040 removable	4040 pulses per inch (ppi)	4040 ppi	2200 ppi	2200 ppi	4040 ppi	4040 ppi
Track density	100 tracks per inch (free format)	48 track per inch	200 tracks per inch	370 fixed 185 removable	192 tracks per inch	370 tracks per inch	400 tracks per inch	400 tracks per inch	192 tracks per inch	370 tracks per inch
Track capacity (byte)	3625	3,328 bytes per track	7294	10,240*	10,240*	10,240	7294	7294	13,030	13,030
Number of tracks	200 + 3 spare usable tracks per disc surface	77 total, 73 for data use per disc surface	200 + 3 spare usable tracks per disc surface	808+7 spare tracks 404+4 spare tracks	404 + 7 spare usable tracks per disc surface	404 or 808+7 spare usable tracks per disc surface	400 + 6 spare usable tracks per surface	400 + 6 spare usable tracks per surface	404 + 7 spare usable tracks per disc surface	808 + 7 spare tracks per disc surface
Number of surfaces per disk unit	10	1 surface	20	Data 3 Positioning 1 fixed Data 2 removable	Data 7 Positioning 1	Data 7 Positioning 1	20	20	19	19
Positioning time (seek time)	Minimum Average Maximum	— 83.33 ms —	25 ms 60 ms 130 ms	10 ms 33 ms 60 ms	10 ms 30 ms 60 ms	10 ms 27 ms 45 ms	10 ms 30 ms 55 ms	7.5 ms 29 ms 55 ms	7 ms 27 ms 50 ms	10 ms 30 ms 55 ms
Transfer rate	156 kilobytes per second	128 bytes in < 6ms	312 kilobytes per second	625 kilobytes per second	625 kilobytes per second	628 kilobytes per second	312 kilobytes per second	312 kilobytes per second	806 kilobytes per second	806 kilobytes per second

Table C-6. SPERRY UNIVAC 0920 Paper Tape Subsystem Characteristics

Characteristic	Description
Reader mounting	Mounted on a 7- by 9-inch panel having a pin spindle for handling reels containing up to 50 feet of tape (for tape reader without an optional spooler)
Tape read	Unidirectional (right to left)
Tape channel capacity	Capable of reading 11/16-inch, 7/8-inch, or 1-inch paper tape; 3-position tape guide available to adjust to tape width used
Read speed	300 characters per second at 10 characters per inch
Type of tape	All conventional perforated tapes with a light transmissivity of 40% or less
Stop and start capacities	Can stop on character or before next character; on start, unit reaches full speed within two characters
Tape spooler	Up to 5-inch reels can be used with the spooler to allow reeling of approximately 300 feet of paper tape
Tape leader	Approximately 3 feet of tape leader when spooler mechanism is used
Tape trailer	A 12-inch trailer is provided to prevent false broken tape indication
Punch mounting	Mounted within a 14- by 19-inch panel
Tape channel capacity	Handles paper tape width of 11/16 inch or 1 inch; five levels of tape characters with 11/16-inch paper tape being used; or 5, 6, 7, or 8 levels of tape characters with 1-inch paper tape in use. Tape guide adjusts to conform to paper tape width.
Punch speed	110 characters per second at 10 characters per inch
Type of tape	Oil base paper tape is provided. A compatible tape utilizing a paper-plastic-paper sandwich is also available.
Stop and start capabilities	Punching is performed one character at a time. Tape punch is capable of stopping and starting between characters.
Tape feeding	The tape punch handles a paper tape reel of 1000 feet with sensing signals to indicate low paper tape supply.



Appendix D. Disc File Labels



GENERAL

This appendix describes the system standard labels for disc files in the SPERRY UNIVAC Operating System/3 (OS/3), as well as the optional user standard labels which are supported by OS/3 data management for processing disc files described by the DTFSD, DTFNI, and DTFDA macros. Note that OS/3 ISAM does not support user labels for DTFIS files.

Because user files within a disc volume may be stored in various locations, a directory listing the addresses of the fragments of the files is required. This directory, called the volume table of contents (VTOC), and the files within a disc volume require various standard labels in predefined formats to describe the properties of the files and the volumes on which they reside.

The system standard disc labels include the volume label (VOL1 label) and seven types of format labels. These labels may, according to their use, be separated 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.

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 D—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 occupied by the VTOC and the device-dependent characteristics of the volume; it also supplies a link to the first format 0 label.

Each label in the VTOC which describes label records not in use is defined as a format 0 label and is linked to the next available format 0.

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 will be present only if split-cylinder allocation has taken place.

The first (or only) 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.

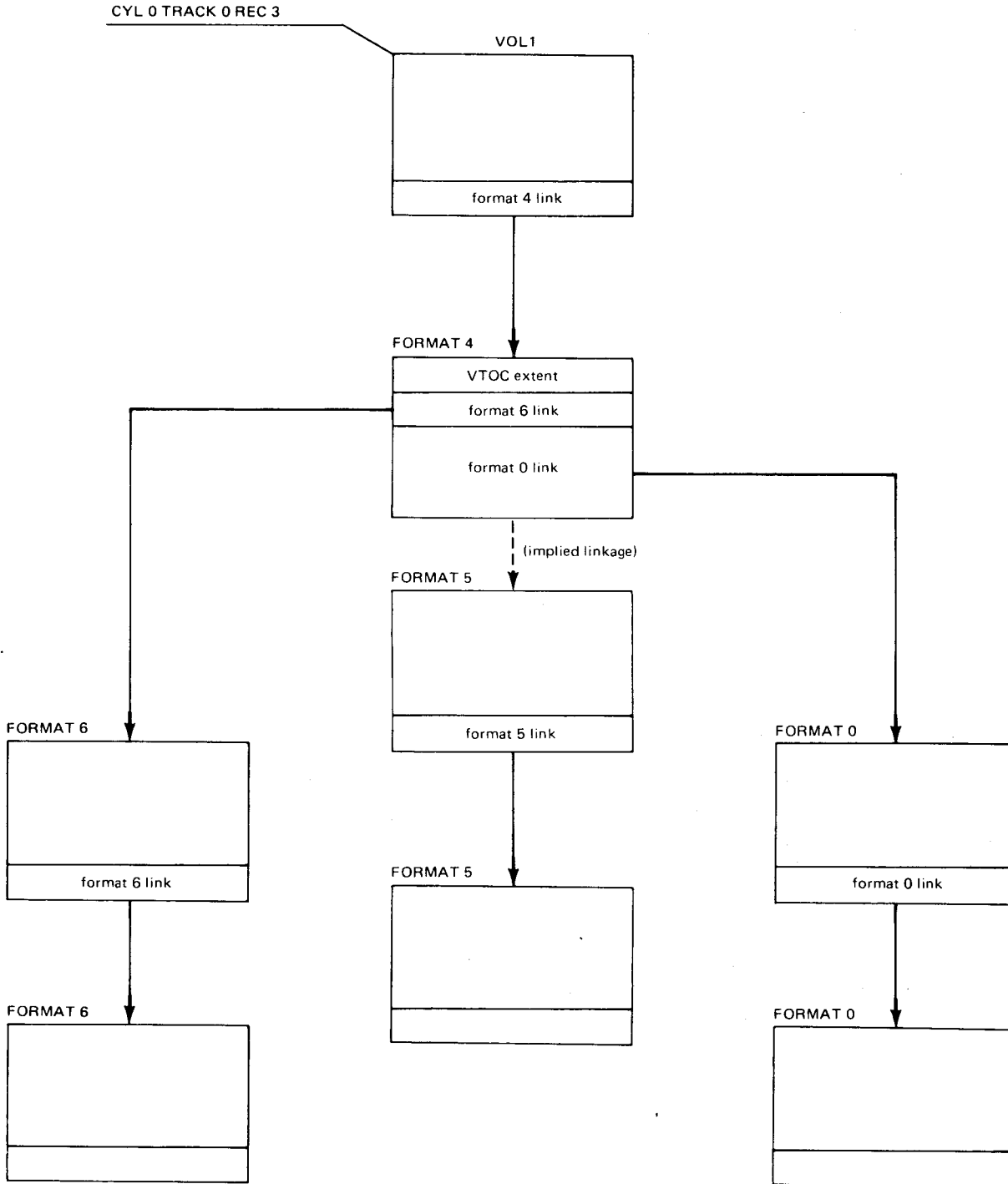


Figure D-1. VTOC Volume Information Label Group

VOL1 Label

As each disc 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 disc initialization routine at cylinder 0, head 0, record 3.

The VOL1 label is the standard volume label in the OS/3. 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 D-2; Table D-1 summarizes its contents.

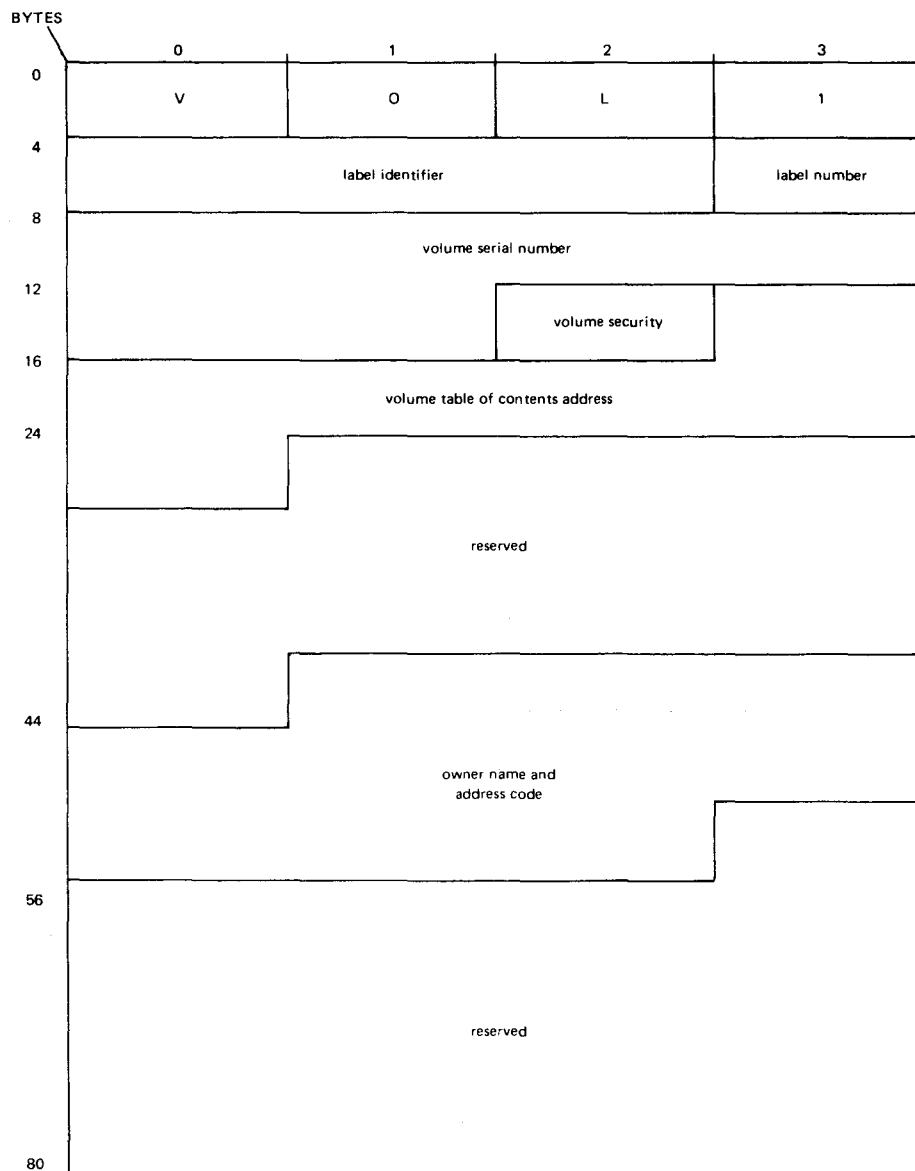


Figure D-2. VTOC VOL1 Label

Table D—1. Contents of VOL1 Label

Label	Initialized by	Bytes	Code	Description
DL\$VL	Disc 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 disc pack when it enters the system. The same code should appear visually on the disc pack for operator identification.
DL\$VSB	Disc 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 <i>cchhr</i> in bytes 15 through 19. Bytes 20 through 24 are 0.
		25-44		Reserved
DL\$ONR	Disc 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.

Disc Format 4 Label

The format 4 label describes the VTOC itself and is the first record of the VTOC. An indicator in the format 4 label states whether the format 5 label contains valid information. 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.

The format 4 label is written by the disc initialization routine at the disc 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 (i.e., a format 0 label) is placed in the format 4 label for use by OS/3 disc space management. An additional linkage that is created and maintained by disc space management specifies the first active format 6 label and is used only during split-cylinder allocation of data files. Figure D—3 shows the format 4 label; Table D—2 summarizes its contents.

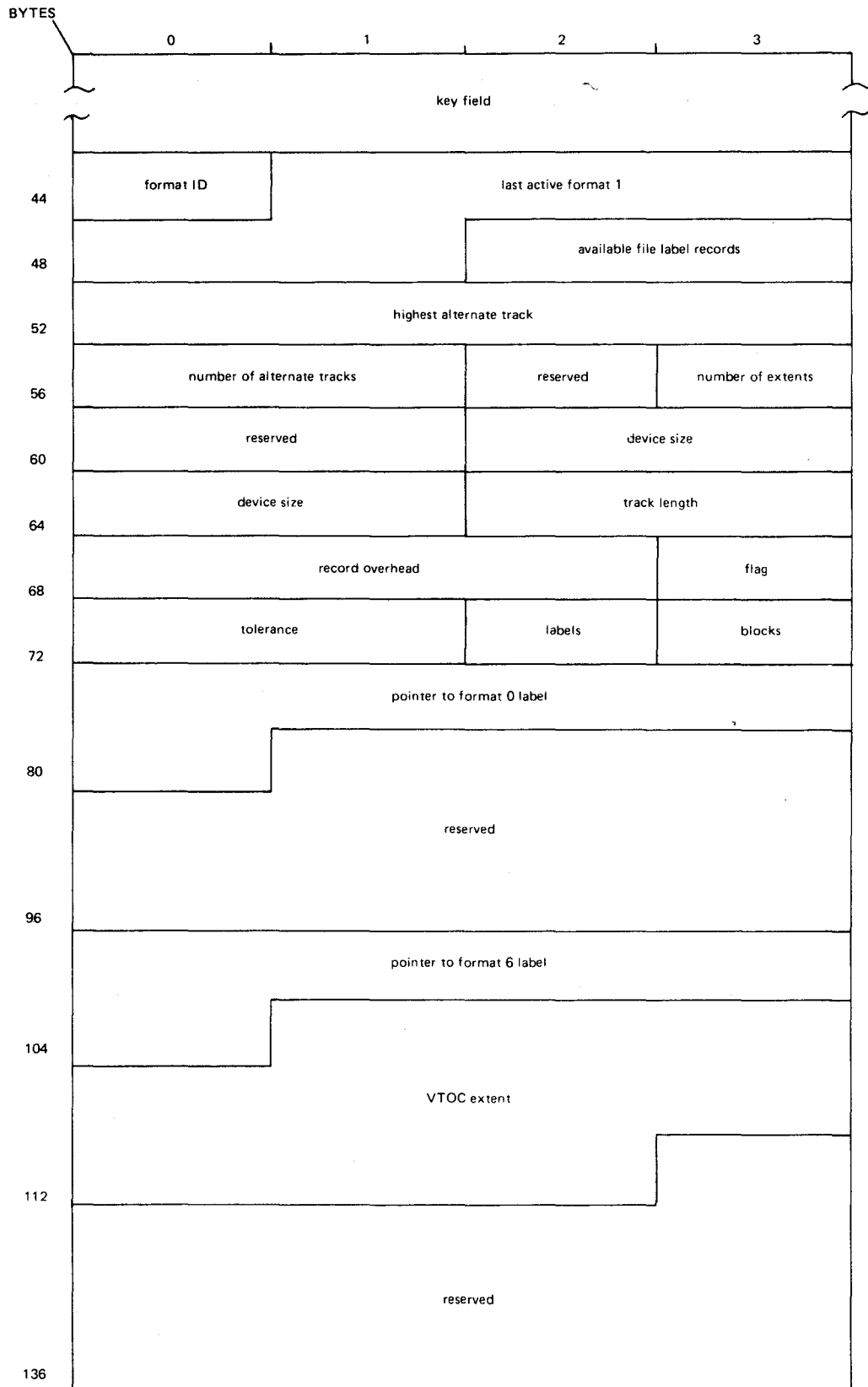


Figure D-3. Disc Format 4 Label

Table D-2. Contents of Disc Format 4 Label (Part 1 of 2)

Label	Initialized by	Bytes	Code	Description									
DL\$KY4	Disc prep	0-43	Hexadecimal	Key field - each byte of this field contains 04 ₁₆									
DL\$ID4	Disc prep	44	EBCDIC	Format ID - always 4 for format 4 label									
DL\$LF4	Space mgmt	45-49	Discontinuous binary	Last active format - the address, in the form <i>cchhr</i> , used for a search on filename									
DL\$AF4	Disc prep	50-51	Binary	Available file label records - number of unused records in the VTOC									
DL\$HA4	Disc prep	52-55	Discontinuous binary	Highest alternate track - address, in the form <i>cchh</i> , of alternate tracks set aside in case of bad tracks									
DL\$AT4	Disc prep	56-57	Binary	Number of alternate tracks									
DL\$VI4	Space mgmt	58		Reserved for VTOC indicators - <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Bit</th> <th>Contents</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>A format 5 label, if present, contains invalid information.</td> </tr> <tr> <td>1-7</td> <td>0</td> <td>Unused</td> </tr> </tbody> </table>	Bit	Contents	Meaning	0	1	A format 5 label, if present, contains invalid information.	1-7	0	Unused
Bit	Contents	Meaning											
0	1	A format 5 label, if present, contains invalid information.											
1-7	0	Unused											
DL\$XC4	Disc prep	59	Binary	Number of extents - contains 01 ₁₆ to indicate the one extent in the VTOC									
	Disc prep	60-61		Reserved									
DL\$DS4	Disc prep	62-65		Device size - indicates the number of cylinders and the number of heads per cylinder on the device, in the form <i>cchh</i>									
DL\$TL4	Disc prep	66-67		Track length - number of available bytes on a track exclusive of home address and record 0									

Table D-2. Contents of Disc Format 4 Label (Part 2 of 2)

Label	Initialized by	Bytes	Code	Description				
DL\$RO4	Disc prep	68-70		Record overhead — ILK describes overhead bytes on track, where I is for keyed record which is not the last on track, L is for keyed record which is the last on track, and K is a decrement applied to records which have no key.				
DL\$FG4	Disc prep	71	Binary	Flag — <table border="1"> <thead> <tr> <th>Bit</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0-7</td> <td>Reserved for device-dependent characteristics</td> </tr> </tbody> </table>	Bit	Meaning	0-7	Reserved for device-dependent characteristics
Bit	Meaning							
0-7	Reserved for device-dependent characteristics							
DL\$TO4	Disc prep	72-73		Tolerance — a device-dependent factor which is used to calculate effective record lengths for that device				
DL\$LT4	Disc prep	74		Labels per track — a device-dependent factor specifying the number of 140-byte labels possible in a VTOC track				
DL\$BK4	Disc 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\$FO4	Disc prep	76-80	Discontinuous binary	Format 0 address in the form <i>cchhr</i> — points to the first available format 0 record in the VTOC.				
		81-99		Reserved				
DL\$F64	Space mgmt	100-104		Format 6 address in the form <i>cchhr</i> — points to the first format 6 label created by space management				
DL\$VX4	Disc 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				

Disc 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 disc initialization routine and maintained by the disc 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 D—4 shows the format 5 label; Table D—3 summarizes its contents.

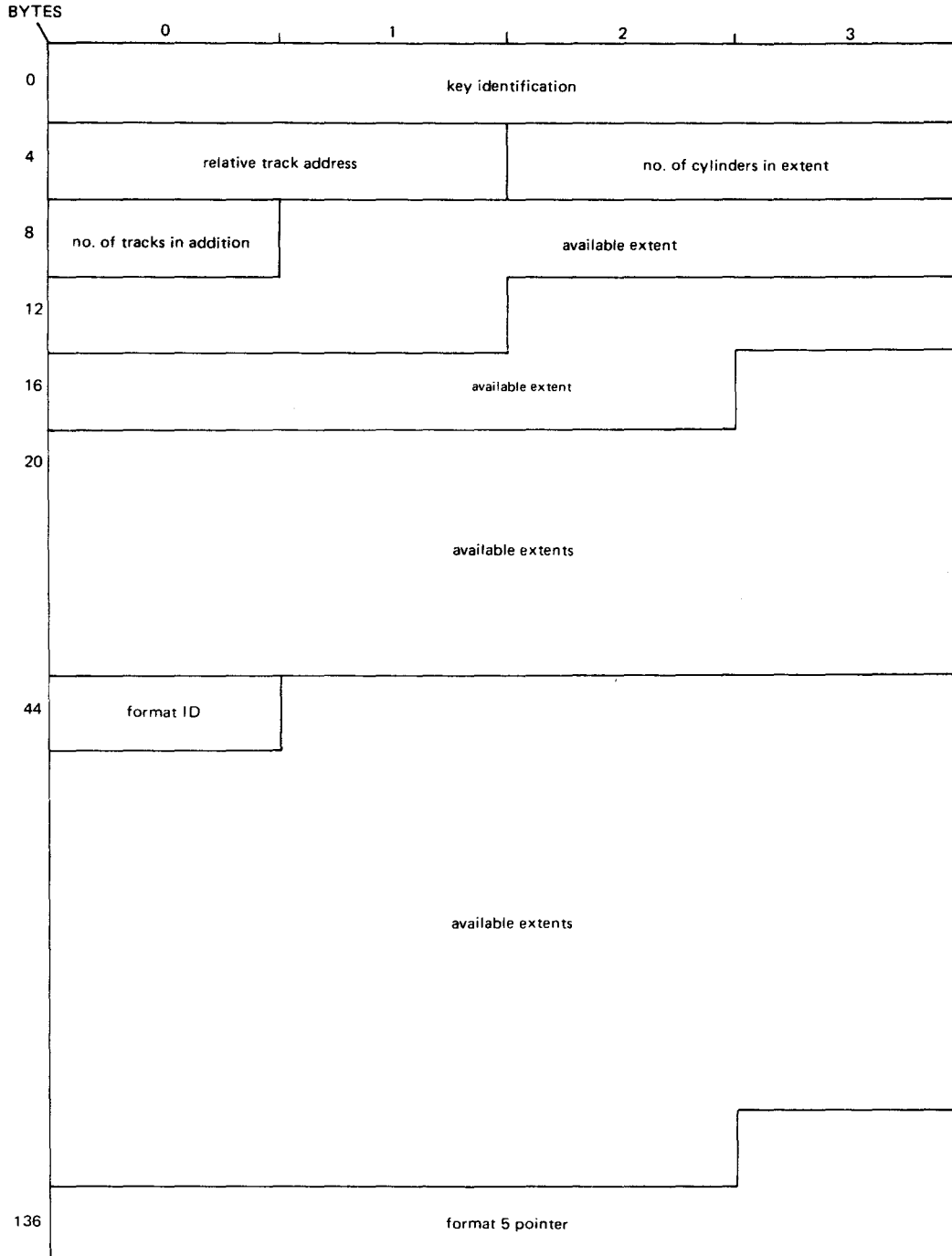


Figure D—4. Disc Format 5 Label

Table D-3. Contents of Disc Format 5 Label

Label	Initialized by	Bytes	Code	Description
DL\$ID5	Disc prep	0-3	Hexadecimal	Key identification — each byte of this field contains 05 ₁₆
DL\$XT5	Disc prep	4-5	Discontinuous binary	Relative track address — start of extent
DL\$XC5	Disc prep	6-7	Binary	Number of cylinders in extent
DL\$XE5	Disc prep	8	Binary	Number of tracks in extent in addition to the cylinders
	Space mgmt	9-13		Available extent — describes another extent in fields with the same format as bytes 4 through 8 above
	Space mgmt	14-43		Six more available extents
DL\$F15	Disc prep	44	EBCDIC	Format ID — always 5, for format 5 label
DL\$XS5	Space mgmt	45-134		Eighteen more available extents
DL\$CP5	Space mgmt	135-139	Discontinuous binary	Pointer — indicates the address of another format 5 label, in the form <i>cchr</i> . Binary 0 if no further label

Disc Format 6 Label

The format 6 label is used to control split-cylinder allocation. Each format 6 label controls a single split-cylinder set and contains a code which 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 1. 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. The format 1 disc address of each member of the set is stored following the head pool. A format 6 label is created for each split-cylinder set defined.

The format 6 label is created and maintained by the disc space management routines. Each label contains the disc address of the format 3 label that defines the extents allocated to that split member set. The disc address of the first format 6 label is maintained in the format 4 label. If more than 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 and format 6 label. Figure D-5 shows the format 6 label; Table D-4 summarizes its contents.

Table D-4. Contents of Disc Format 6 Label

Label	Initialized by	Bytes	Code	Description
DL\$ID6	Space mgmt	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	Disc address of the format 3 label shared by all member files (<i>ccrh</i>)
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 mgmt	12-29	Hexadecimal	Nine additional entries for low and high available head
		30-33	Hexadecimal	Disc address of format 1 label of primary member (<i>ccrh</i>)
		34-109	Hexadecimal	Format 1 label disc addresses of up to 19 additional members of the split-cylinder set in the same format as bytes 30-33
		110-133		Reserved
DL\$F16		134	Hexadecimal	Format identification F6 ₁₆
DL\$CP6		135-139	Discontinuous binary	Pointer to next format 6 label in the form <i>cchr</i>

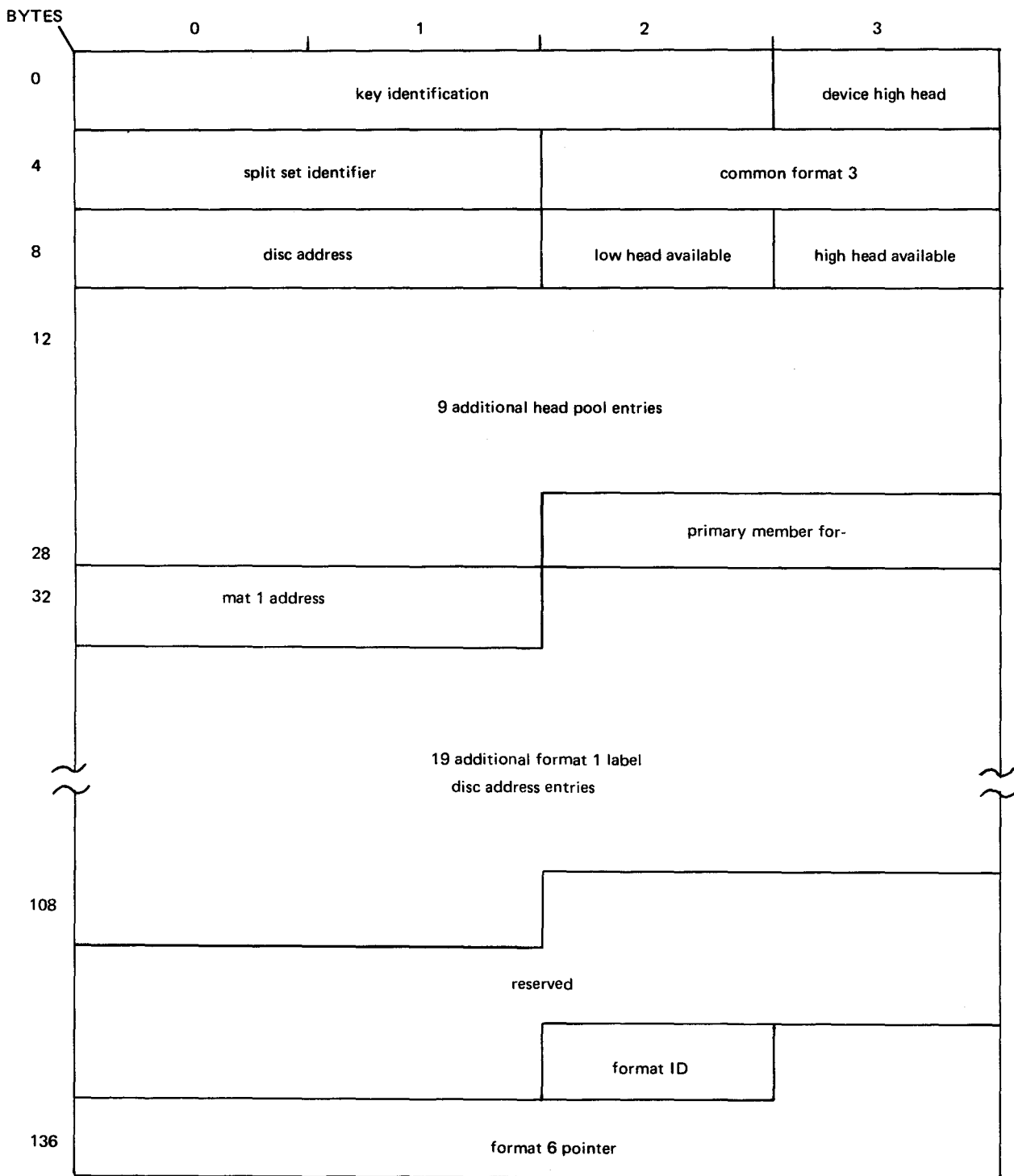


Figure D-5. Disc Format 6 Label

Disc 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 disc 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 0's. Figure D-6 shows the format 0 label; Table D-5 summarizes its contents.

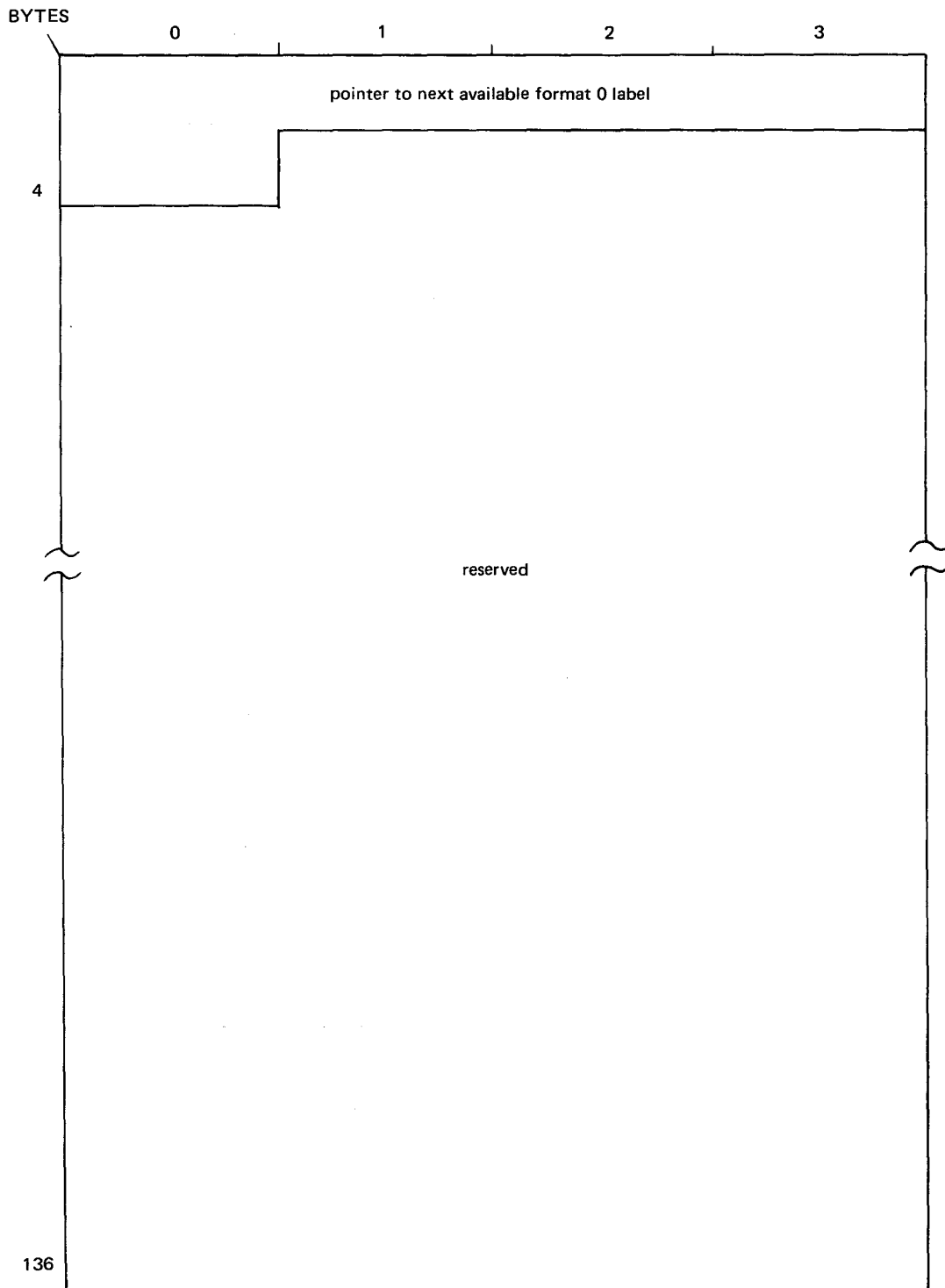


Figure D-6. Disc Format 0 Label

Table D-5. Contents of Disc Format 0 Label

Label	Initialized by	Bytes	Code	Description
	Disc prep	0-4	Discontinuous binary	Disc address in the form <i>cchr</i> of the next available format 0 label.
		5-139	Binary 0	Reserved

FILE INFORMATION GROUP

The file information group (Figure D-7) is composed of the format 1, format 2, and format 3 labels. The format 1 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 will exist for all files created by using split-cylinder allocation. For all other files, the format 3 label will exist only if the file occupies more than three separate extent areas.

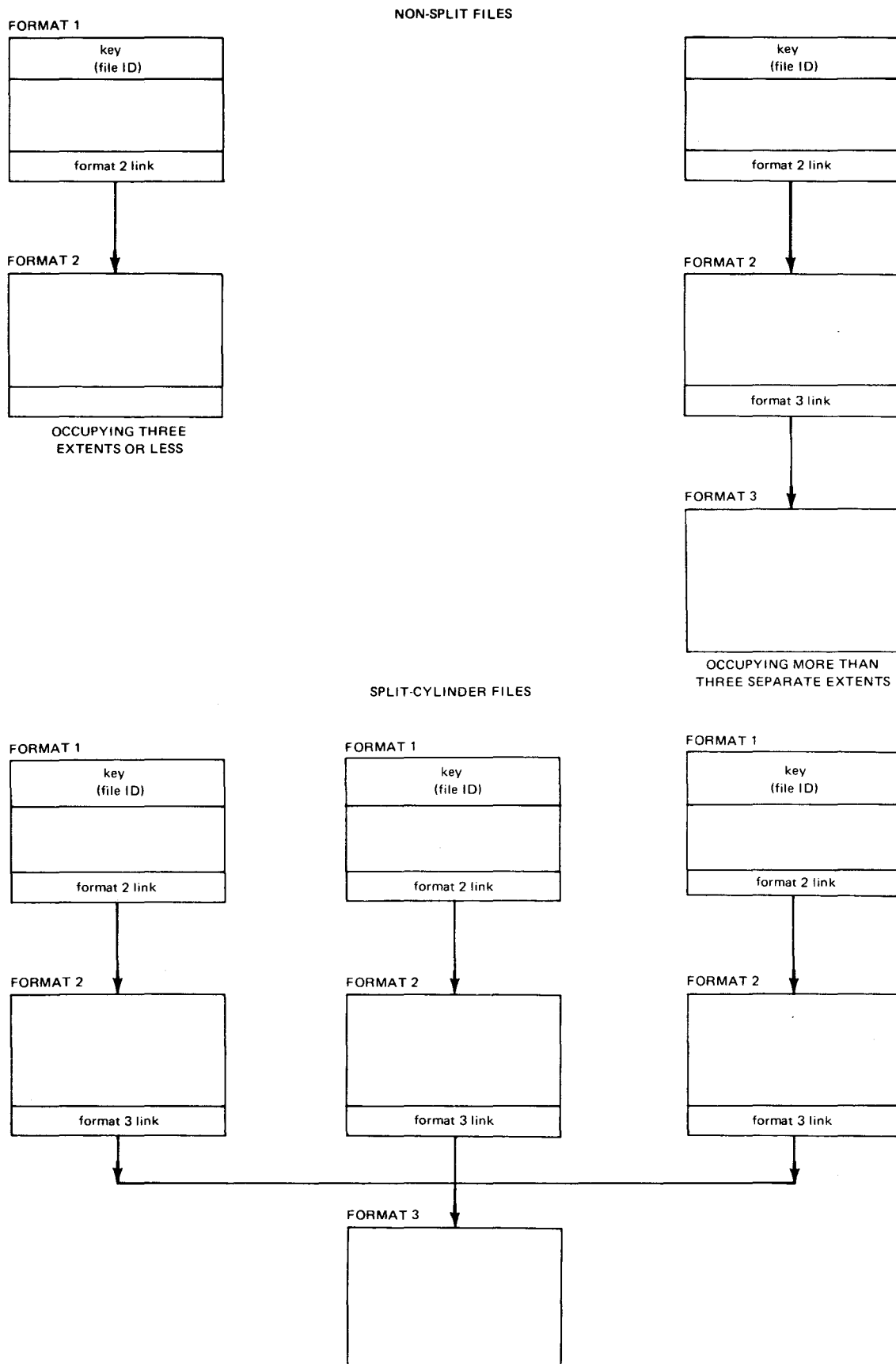


Figure D-7. File Information Group Label Chain

Disc 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 disc 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 D—8 shows the format 1 label; Table D—6 summarizes its contents.

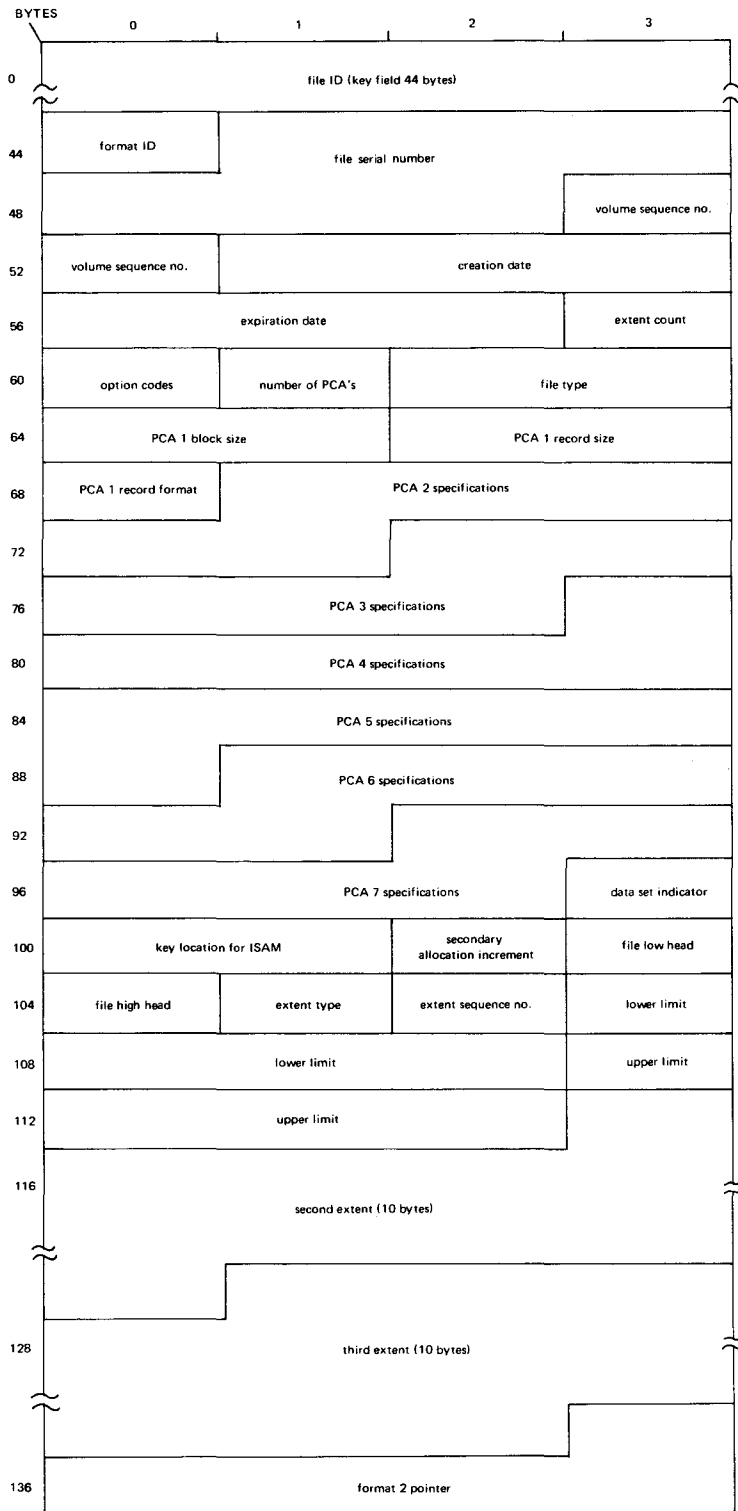


Figure D—8. Disc Format 1 Label

Table D-6. Contents of Disc Format 1 Label (Part 1 of 4)

Label	Initialized by	Bytes	Code	Description																					
DL\$KEY1	Space mgmt	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 mgmt	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 the 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 mgmt	53-55	Discontinuous binary	Creation date - format is <i>ydd</i> (year-day-day), where <i>y</i> is 0 to 99, and <i>dd</i> 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																					
DL\$OC1	Space mgmt	60	Binary	Option codes <table border="1"> <thead> <tr> <th>Bit</th> <th>Content</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>Preformatted by VTOC</td> </tr> <tr> <td>1</td> <td>1</td> <td>Allocation by cylinder</td> </tr> <tr> <td>2</td> <td>1</td> <td>New file</td> </tr> <tr> <td>3</td> <td>1</td> <td>Partitions cylinder aligned</td> </tr> <tr> <td>4</td> <td>1</td> <td>File contains user labels</td> </tr> <tr> <td>5-7</td> <td></td> <td>Unused</td> </tr> </tbody> </table>	Bit	Content	Meaning	0	1	Preformatted by VTOC	1	1	Allocation by cylinder	2	1	New file	3	1	Partitions cylinder aligned	4	1	File contains user labels	5-7		Unused
Bit	Content	Meaning																							
0	1	Preformatted by VTOC																							
1	1	Allocation by cylinder																							
2	1	New file																							
3	1	Partitions cylinder aligned																							
4	1	File contains user labels																							
5-7		Unused																							

Table D-6. Contents of Disc Format 1 Label (Part 2 of 4)

Label	Initialized by	Bytes	Code	Description
DL\$PC1	Data mgmt	61	Binary	PCA count — number of partitions which constitute the file
DL\$FT1	Data mgmt	62	Hexadecimal	File type Hexadecimal Code Meaning 20 Sequential (DTFSD) 40 Direct access (DTFDA) 60 Nonindexed (DTFNI) 80 Indexed sequential (DTFIS) 90 IRAM (DTFIR) or MIRAM (DTFMI) 02 SAT (DTFPF) 00 Undefined
		63*	Hexadecimal	File type Hexadecimal Code Meaning 00 IRAM file, nonindexed 11 IRAM file, indexed 80 MIRAM file, IRAM characteristics C0 MIRAM file, MIRAM characteristics
DL\$BL1	Data mgmt	64-65	Binary	Reserved for PCA1 block length — size of fixed-length blocks or maximum size of variable-length blocks
DL\$RL1	Data mgmt	66,67	Binary	Reserved for PCA1 record length — size of fixed-length records or maximum size of variable-length records
DL\$RF1	Data mgmt	68	Binary	Reserved for PCA1 record format
				Bit Content Meaning 0 Reserved 1 0 PCA has no subfiles. 1 1 PCA has subfiles. 2 0 Records have no keys. 1 1 Records have keys.

*Byte 63 is meaningless unless byte 62 = X'90'

Table D-6. Contents of Disc Format 1 Label (Part 3 of 4)

Label	Initialized by	Bytes	Code	Description																		
				(Record format, cont) <table border="1"> <thead> <tr> <th>Bit</th> <th>Content</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>3</td> <td>1</td> <td>Fixed-length blocked records</td> </tr> <tr> <td>4</td> <td>1</td> <td>Variable-length blocked records</td> </tr> <tr> <td>5</td> <td>1</td> <td>Fixed-length unblocked records</td> </tr> <tr> <td>6</td> <td>1</td> <td>Variable-length unblocked records</td> </tr> <tr> <td>7</td> <td>1</td> <td>Records are inter-laced.</td> </tr> </tbody> </table>	Bit	Content	Meaning	3	1	Fixed-length blocked records	4	1	Variable-length blocked records	5	1	Fixed-length unblocked records	6	1	Variable-length unblocked records	7	1	Records are inter-laced.
Bit	Content	Meaning																				
3	1	Fixed-length blocked records																				
4	1	Variable-length blocked records																				
5	1	Fixed-length unblocked records																				
6	1	Variable-length unblocked records																				
7	1	Records are inter-laced.																				
	Data mgmt	69-73	Discontinuous binary	Partition descriptor 2; block size, record size, and record format for partition 2																		
	Data mgmt	74-78	Discontinuous binary	Partition descriptor 3																		
	Data mgmt	79-83	Discontinuous binary	Partition descriptor 4																		
	Data mgmt	84-88	Discontinuous binary	Partition descriptor 5																		
	Data mgmt	89-93	Discontinuous binary	Partition descriptor 6																		
	Data mgmt	94-98	Discontinuous binary	Partition descriptor 7																		
DL\$DS1	Space mgmt	99	Binary	Data set indicators - reserved for future use																		
DL\$KL1	Data mgmt	100-101	Binary	Key location - high order position of key field within each data record of an indexed-sequential file																		

Table D-6. Contents of Disc Format 1 Label (Part 4 of 4)

Label	Initialized by	Bytes	Code	Description																		
DL\$SA1		102	Binary	Secondary allocation increment — the number of cylinders of disc storage to be requested for each dynamic extension 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 — <table border="0"> <thead> <tr> <th><u>Code</u></th> <th><u>Meaning</u></th> </tr> </thead> <tbody> <tr> <td>00</td> <td>No valid extent described</td> </tr> <tr> <td>20</td> <td>Sequential file (DTFSD)</td> </tr> <tr> <td>40</td> <td>Direct access file (DTFDA)</td> </tr> <tr> <td>60</td> <td>Nonindexed file (DTFNI)</td> </tr> <tr> <td>80</td> <td>Indexed sequential file (DTFIS)</td> </tr> <tr> <td>90</td> <td>IRAM (DTFIR) or MIRAM (DTFMI)</td> </tr> <tr> <td>02</td> <td>SAT (DTFPF)</td> </tr> <tr> <td>FF</td> <td>Job control</td> </tr> </tbody> </table>	<u>Code</u>	<u>Meaning</u>	00	No valid extent described	20	Sequential file (DTFSD)	40	Direct access file (DTFDA)	60	Nonindexed file (DTFNI)	80	Indexed sequential file (DTFIS)	90	IRAM (DTFIR) or MIRAM (DTFMI)	02	SAT (DTFPF)	FF	Job control
<u>Code</u>	<u>Meaning</u>																					
00	No valid extent described																					
20	Sequential file (DTFSD)																					
40	Direct access file (DTFDA)																					
60	Nonindexed file (DTFNI)																					
80	Indexed sequential file (DTFIS)																					
90	IRAM (DTFIR) or MIRAM (DTFMI)																					
02	SAT (DTFPF)																					
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 <i>cchh</i>																		
DL\$XU1		111-114	Discontinuous binary	Upper limit — the address specifying the end of the extent, in the form <i>cchh</i>																		
		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 mgmt	135-139	Discontinuous binary	Continuation pointer — the address of a format 2 label. The address is in the form <i>cchhr</i> .																		

Disc Format 2 Label

The format 2 label is used as an extension to the format 1 label to further describe the file.

For nonindexed files, bytes 1 through 43 are used to carry partition information in a maximum of seven 6-byte entries. For indexed files, bytes 13 through 43 are used to carry index control information. For IRAM and MIRAM files, bytes 13 through 43 are used to carry index control and file characteristic information. For library files, bytes 32 through 47 are used to carry information on the library text and directory; bytes 13 through 31 contain binary 0's.

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 will point to a format 3 label, if used. This pointer will be 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 0's. Figure D-9 shows the format 2 label; Table D-7 summarizes its contents. The format of the ISAM file information area is shown in Figure D-10, and the contents of this area are listed in Table D-8. The format of the IRAM/MIRAM file information area is shown in Figure D-11, and the contents of this area are listed in Table D-9. The format of the library file information area is shown in Figure D-12, and the contents of this area are listed in Table D-10.

BYTES	0	1	2	3
0	key ID	nonindexed LBC	key length or lace factor	reserved
4	EOD ID			nonindexed LBC
8	key length or lace factor	reserved	EOD ID	
12	EOD ID	(up to five additional partition descriptors)		
40				
44	reserved		blocks/track, PCA1	
48	PCA1ID 0 2 3	relative track addr-1* 15 16	tracks 31	
52	PCA2ID 0 2 3	relative track addr-2* 15 16	tracks 31	
⋈				
128	tracks per cylinder		file low head no.	
132	relative extent count		flags	
136	format 3 pointer			

*Thirteen bits can represent a maximum relative track address (RTA) of 8191_{10} ($1FFF_{16}$). To support the larger 8433 disc, the high-order bit of the tracks field (bit 16) of the logical extent is used to indicate that the RTA must be increased by a constant value of 8192_{10} . (See Table D-7.)

Figure D-9. Disc Format 2 Label, Nonindexed Files (DTFSD, DTFDA, DTFNI)

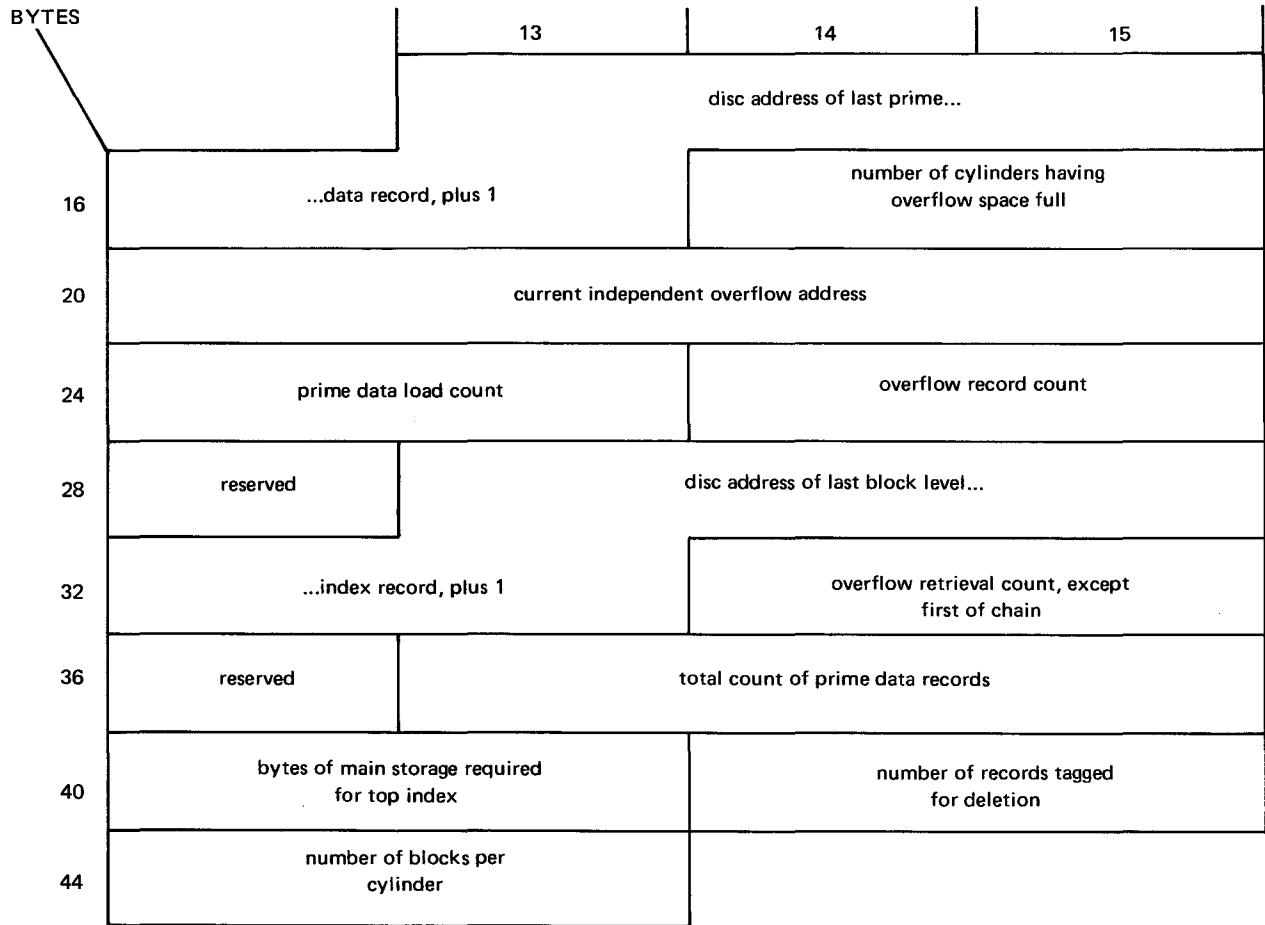
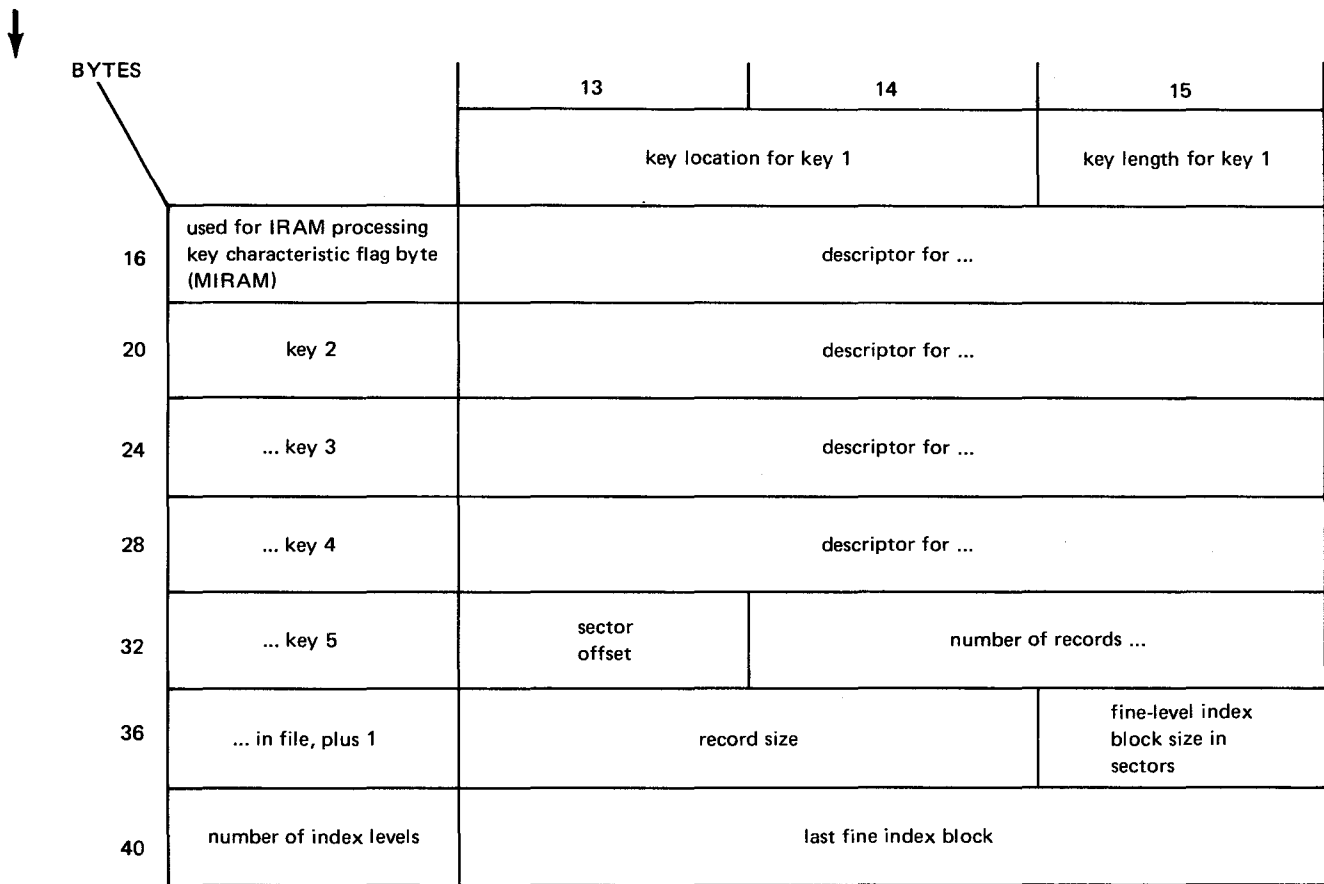


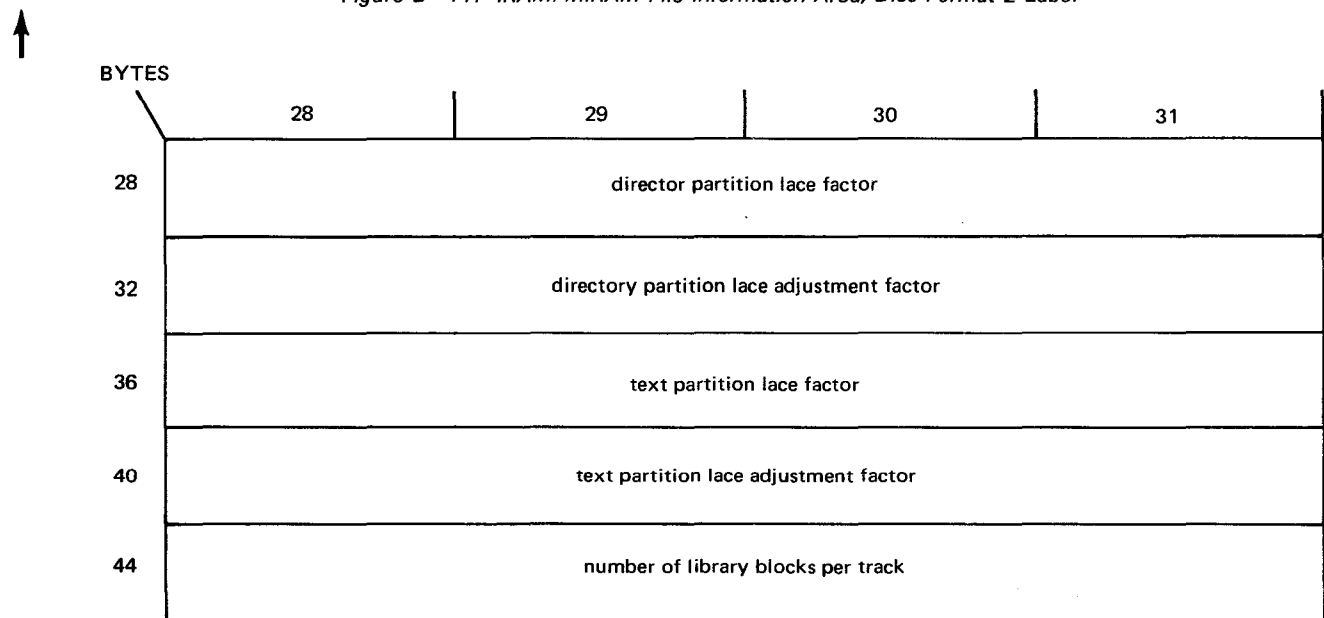
Figure D-10. ISAM (DTFIS) File Information Area, Disc Format 2 Label



NOTE:

Descriptions which pertain to IRAM files also apply to MIRAM files with IRAM characteristics.

Figure D-11. IRAM/MIRAM File Information Area, Disc Format 2 Label



NOTE:

In the format 2 label for library files, byte 3 and bytes 13-27 are reserved and contain binary 0.

Figure D-12. Library File Information Area, Disc Format 2 Label

Table D-7. Contents of Disc Format 2 Label (Part 1 of 2)

Label	Initialized by	Bytes	Code	Description
DL\$SID2	Space mgmt	0	Hexadecimal	Key identification 02 ₁₆
DL\$SPC2	Data mgmt	1		Nonindexed last block control – the number of logical records in the last block of the partition for fixed-length blocked files
DL\$SLF2		2		Key length or lace factor
DL\$SLA2		3		Reserved
DL\$SEP2		4–6		End of data ID – relative block address plus 1 of the last block written into the partition
		7–12		A 6-byte partition descriptor entry in the same form as bytes 1–6
	Data mgmt	13–43	Hexadecimal	For nonindexed files, up to five additional partition descriptors For ISAM files, see index information area (Table D-8 and Figure D-10) For IRAM and MIRAM files, see IRAM/MIRAM information area (Table D-9 and Figure D-11). For library files, see library information area (Table D-10 and Figure D-12)
		44–45		Unused (binary 0) 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 mgmt	48–127	Discontinuous binary	Logical extent table area – These entries are four bytes in length and specify PCA ID, starting relative track address, and number of tracks at that address.

Table D-7. Contents of Disc Format 2 Label (Part 2 of 2)

Label	Initialized by	Bytes	Code	Description																					
DL\$XAR2 (cont)				<p>From 1 to 20 four-byte logical extent entries may be placed in this 80-byte area. Each 4-byte entry has the following format:</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0-2</td> <td>The high order 3 bits of the logical extent identify the partition to which it is assigned. (This value may be from 1 to 7.)</td> </tr> <tr> <td>3-15</td> <td>The next 13 bits indicate the relative track address of the logical extent.</td> </tr> <tr> <td>16-31</td> <td>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 relative track address of the logical extent on the 8433 disc. The remaining 16 bits indicate the number of tracks contained in the extent.</td> </tr> </tbody> </table>	Bit	Meaning	0-2	The high order 3 bits of the logical extent identify the partition to which it is assigned. (This value may be from 1 to 7.)	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 relative track address of the logical extent on the 8433 disc. The remaining 16 bits indicate the number of tracks contained in the extent.													
Bit	Meaning																								
0-2	The high order 3 bits of the logical extent identify the partition to which it is assigned. (This value may be from 1 to 7.)																								
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 relative track address of the logical extent on the 8433 disc. The remaining 16 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\$XCT2		132-133		Number of relative extents contained in this label																					
DL\$SFL2		134		<p>Flags</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Content</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td rowspan="2">Reserved</td> <td></td> </tr> <tr> <td>1</td> </tr> <tr> <td>2</td> <td>1</td> <td>Library text is interlaced.</td> </tr> <tr> <td>3</td> <td>1</td> <td>Library directory is interlaced.</td> </tr> <tr> <td>4</td> <td rowspan="3">Reserved</td> <td></td> </tr> <tr> <td>5</td> </tr> <tr> <td>6</td> </tr> <tr> <td>7</td> <td>1</td> <td>9400 SAT compatible</td> </tr> </tbody> </table>	Bit	Content	Meaning	0	Reserved		1	2	1	Library text is interlaced.	3	1	Library directory is interlaced.	4	Reserved		5	6	7	1	9400 SAT compatible
Bit	Content	Meaning																							
0	Reserved																								
1																									
2	1	Library text is interlaced.																							
3	1	Library directory is interlaced.																							
4	Reserved																								
5																									
6																									
7	1	9400 SAT compatible																							
DL\$SCID2	Space mgmt	135-139	Discontinuous binary	The disc address, in the form <i>cchhr</i> , of the format 3 label (if required) associated with this file																					

Table D-8. Contents of Indexed File Information Area, Disc Format 2 Label

Label	Initialized By	Bytes	Code	Description
DL\$PID2	Data management	13-17	Discontinuous binary	Disc address of last prime data record (plus 1), in the form <i>rrrbb</i> , where <i>rrr</i> =relative block address and <i>bb</i> =displacement within the block.
DL\$NMA2		18-19	Hexadecimal	Count of cylinders having overflow space filled.
DL\$IOF2		20-23		Current address of independent overflow (<i>rrr</i>).
DL\$PDL2		24-25	Discontinuous binary	Prime data load count.
DL\$NMO2	Data management	26-27	Hexadecimal	Count of the number of overflow records in the file.
-		28		Reserved
DL\$BID2		29-33	Discontinuous binary	Disc address of last block level index record (plus 1), in the same form as bytes 13-17.
DL\$NMR2		34-35		Overflow chain retrieval count, not first of chain.
-		36		Reserved
DL\$NMP2		37-39		Total count of number of prime data records.
DL\$NMS2	Data management	40-41	Hexadecimal	Number of bytes required to hold top index in main storage
DL\$NMT2		42-43		Number of records user has tagged for deletion.
-		44-45		Number of blocks per cylinder.

Table D-9. Contents of IRAM/MIRAM File Information Area, Disc Format 2 Label (Part 1 of 2)

Label	Initialized by	Bytes	Code	Description			
DL\$XILOC	Data mgmt	13-14	Hexadecimal	Key location for key 1			
		15		Key length for key 1			
		16	Hexadecimal (IRAM)	Used for IRAM file index processing			
Binary (MIRAM)	For MIRAM files, byte 16 contains key 1 characteristics: <table border="0"> <tr> <td><u>This Bit On</u></td> <td><u>Means</u></td> </tr> <tr> <td>Bit 0</td> <td>Duplicates allowed for this key</td> </tr> <tr> <td>Bit 1</td> <td>Key change allowed for this key during update</td> </tr> </table>		<u>This Bit On</u>	<u>Means</u>	Bit 0	Duplicates allowed for this key	Bit 1
<u>This Bit On</u>	<u>Means</u>						
Bit 0	Duplicates allowed for this key						
Bit 1	Key change allowed for this key during update						

Table D-9. Contents of IRAM/MIRAM File Information Area, Disc Format 2 Label (Part 2 of 2)

Label	Initialized by	Bytes	Code	Description
DL\$XILOC (cont)		16 (cont)	Binary (MIRAM) (cont)	<u>This Bit On</u> <u>Means</u>
				Bit 2-4 Unused
				*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 binary	Descriptor for key 2 (binary zeros for IRAM)
		21-24		Descriptor for key 3 (binary zeros for IRAM)
		25-28		Descriptor for key 4 (binary zeros for IRAM)
		29-32		Descriptor for key 5 (binary zeros for IRAM)
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 key 2 through 5.

NOTE:

Descriptions pertaining to IRAM files also apply to MIRAM files with IRAM characteristics.

Table D-10. Contents of Library Information Area, Disc Format 2 Label

Label	Initialized By	Bytes	Code	Description
DL\$DIRL2	Data management	28-31		Hardware-adjusted lace factor for the directory partition
DL\$DIRF2	Data management	32-35		Rotational adjustment for directory lace factor
DL\$XTL2	Data management	36-39		Hardware-adjusted lace factor for the library text partition
DL\$XTF2	Data management	40-43		Rotational adjustment factor for the library's text
-	Data management	44-47		The number of library blocks per track

Disc 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 will exist only when more than three extents are required.

The format 3 label is initialized and maintained by the disc space management routines. The format 3 label, when required, will always be linked from a format 2 label. Figure D—13 shows the format 3 label; Table D—11 summarizes its contents.

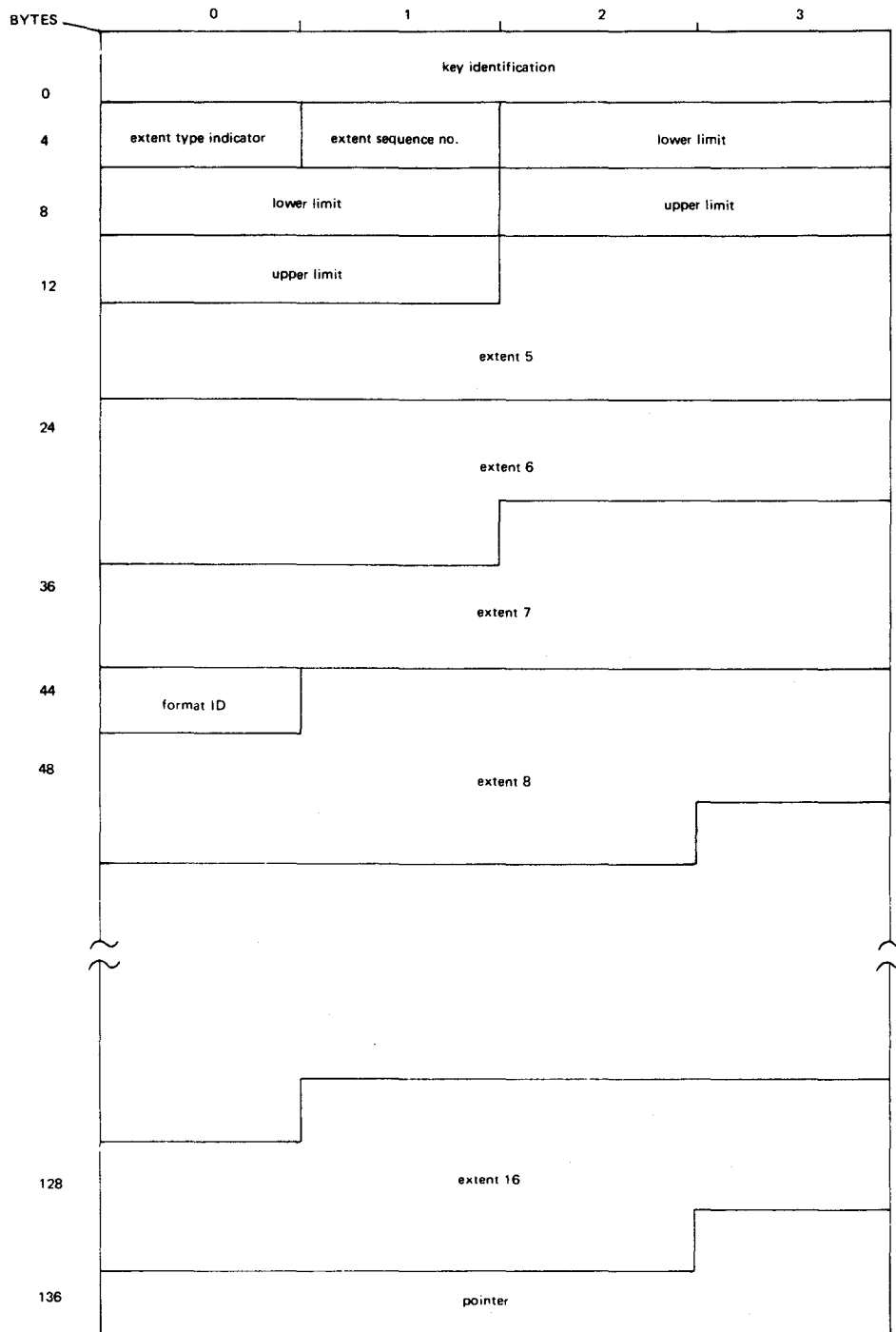


Figure D—13. Disc Format 3 Label

Table D-11. Contents of Disc Format 3 Label

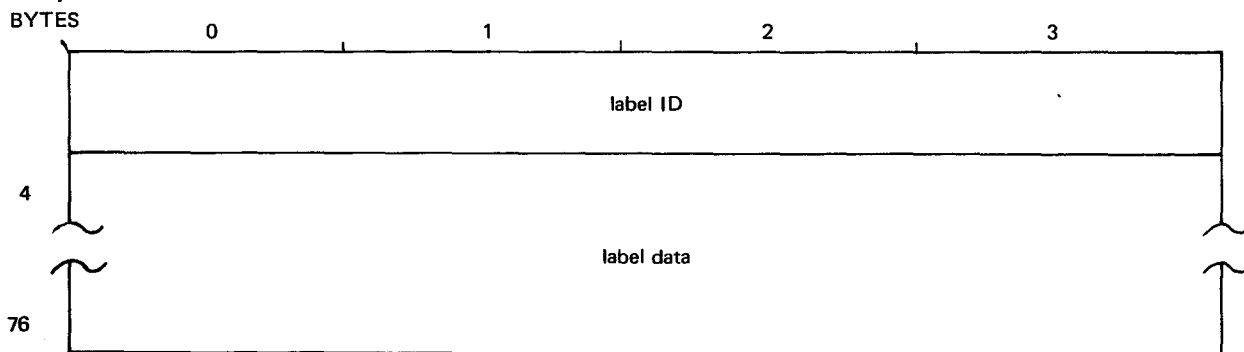
Label	Initialized by	Bytes	Code	Description						
DL\$ID3	Space mgmt	0-3	Hexadecimal	Key identification — each byte contains 03 ₁₆						
DL\$XT3		4	Hexadecimal	Extent type indicator — <table border="1"> <thead> <tr> <th>Code</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>No valid extent described</td> </tr> <tr> <td>01*</td> <td>Prime data area</td> </tr> </tbody> </table>	Code	Meaning	00	No valid extent described	01*	Prime data area
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 <i>cchh</i>						
DL\$XU3		10-13	Discontinuous binary	Upper limit — terminating track address of the extent, in the form <i>cchh</i>						
		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						
DL\$XS3		45-134		Extents 8 through 16						
DL\$CP3		135-139	Discontinuous binary	Pointer — address of next format 3 label, in the form <i>cchhr</i> . Binary 0 if no further label						

OPTIONAL USER STANDARD LABELS

Optional user standard labels are records made available to the user's label processing routine (LABADDR) at the opening or closing of a disc volume. OS/3 data management supports user standard labels for SAM and DAM disc files described by the DTFSD, DTFNI, and DTFDA declarative macro instructions; it does not support them for ISAM files, which are described by the DTFIS macro.

User Header Labels

If the user requires user header labels, these will be written on the first track of each volume of a DTFSD file, and on the first track of the first volume of a DTFDA or DTFNI file. The user may write a maximum of eight labels. Figure D-14 shows the format of the 80-byte user header label; its contents are explained in the table below the figure.

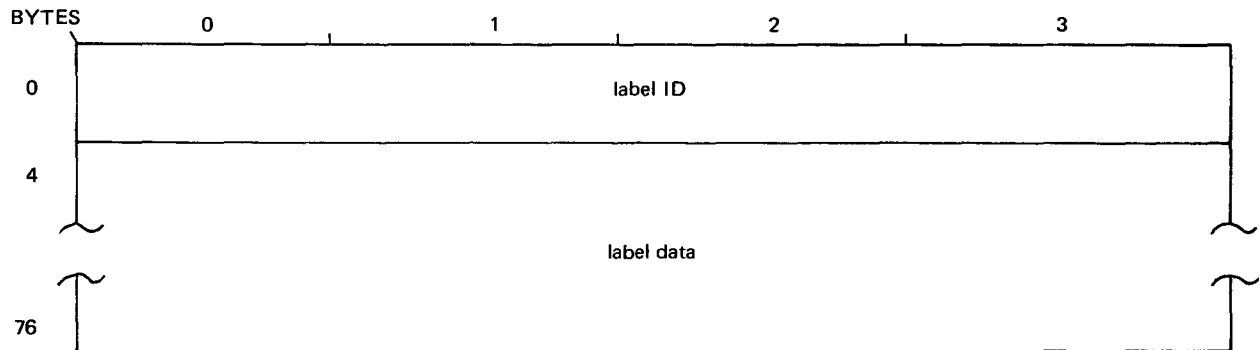


<u>Field</u>	<u>Bytes</u>	<u>Code</u>	<u>Description</u>
Label ID	0-3	EBCDIC	Contains 4-byte label identifier: UHL, followed by a label number which ranges from 1 through 8.
Label data	4-79	User option	Contains 76 bytes of user-specified header label data.

Figure D-14. Optional User Standard Header Label

User Trailer Labels

If the user needs user trailer labels, these will be written on the first track of each volume of a DTFSD file and on the first track of the first volume of DTFDA or DTFNI files, following the user header labels. The user may write a maximum of eight labels on DTFDA and DTFNI files, and eight labels per volume on DTFSD files. Figure D—15 shows the format of the 80-byte user trailer label; its contents are explained in the table below the figure.



<u>Field</u>	<u>Bytes</u>	<u>Code</u>	<u>Description</u>
Label ID	0-3	EBCDIC	Contains 4-byte label identifier: UTL, followed by a label number which ranges from 1 through 8.
Label data	4-79	User option	Contains 76 bytes of user-specified trailer label data

Figure D—15. Optional User Standard Trailer Label

DISKETTE FILE LABEL

Each user file on a diskette must have a valid label before data management can process it. The index track contains the VOL1 label in sector 7. The index track also contains information defining files recorded on tracks 01 through 73 of the diskette. This information is in sectors 08 through 26 of the index track. Each file (up to 19) is represented by one sector of information on the index track. These sectors (08—26) of the index track are called file labels. Figure D—16 illustrates the 8413 diskette file label format. Table D—12 explains the contents of each field in the diskette file label.

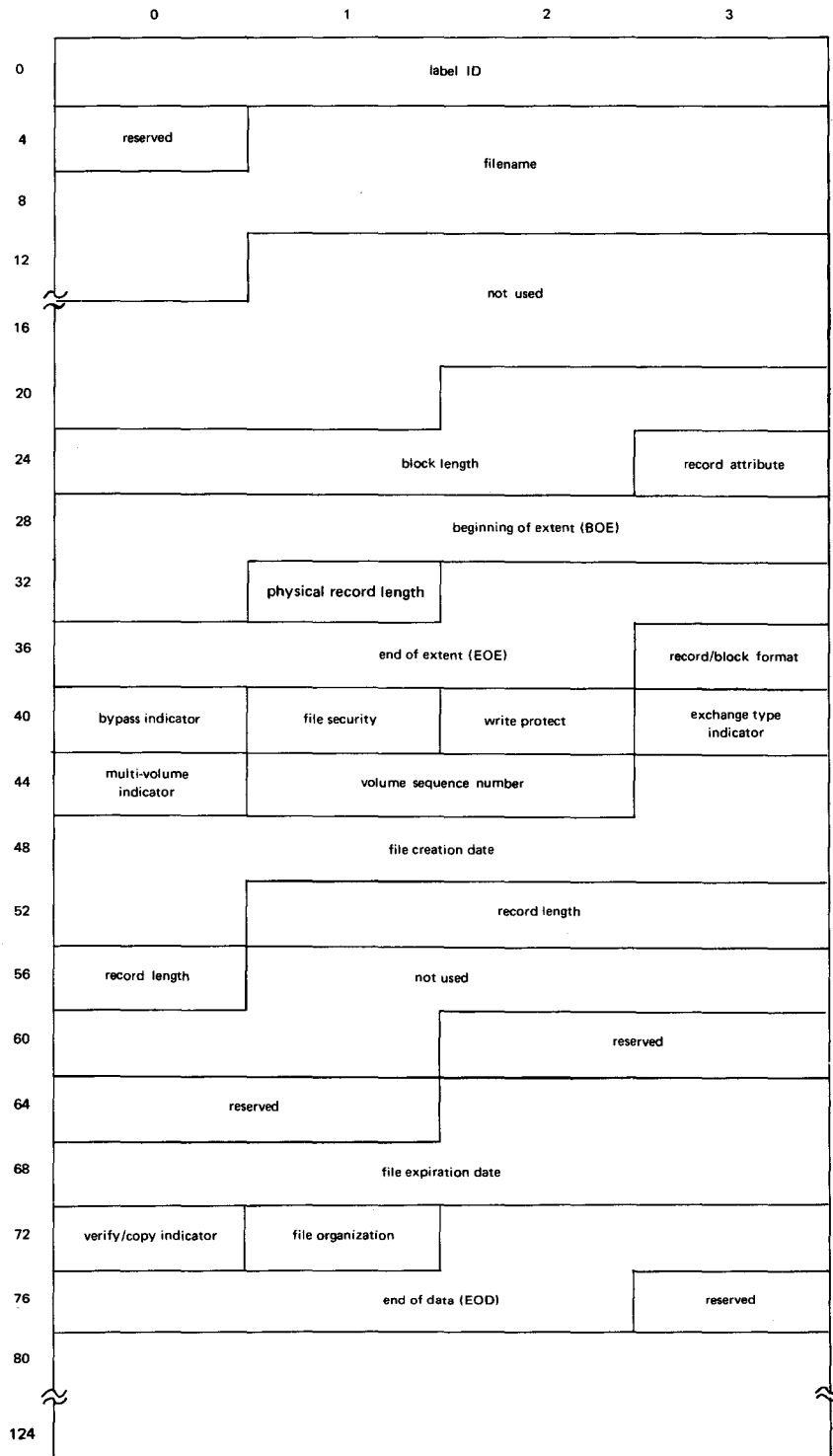


Figure D—16. Diskette File Label Layout

Table D-12. Diskette File Label Description (Part 1 of 2)

Field	Byte Position	Description								
Label ID	0-3	Identifies the application.								
Reserved	4	Reserved								
Filename	5-12	Names user file 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. Only the first 8 characters are used for data exchange.								
—	13-21	Not used.								
Block length	22-26	Indicates the record size as follows: <table border="1"> <thead> <tr> <th>Character</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>blank</td> <td>Record size will be obtained from the DTF.</td> </tr> <tr> <td>1-128</td> <td>Actual record size; for example, 80.</td> </tr> </tbody> </table>	Character	Meaning	blank	Record size will be obtained from the DTF.	1-128	Actual record size; for example, 80.		
Character	Meaning									
blank	Record size will be obtained from the DTF.									
1-128	Actual record size; for example, 80.									
Record attribute	27	Blank indicates unblocked 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 and is blank meaning 128 bytes per record. When exchange type, position 43, is blank, this field must also be blank.								
End of extent (EOE)	34-38	Indicates address of last sector reserved for this file and uses the same format as BOE.								
Record/block format	39	Contains either a blank or F indicating fixed-length records in fix blocks. When position 43 is blank, this field must be blank also.								
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.								
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.								
Multi-volume indicator	44	<table border="1"> <thead> <tr> <th>Character</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>blank</td> <td>file on 1 diskette</td> </tr> <tr> <td>C</td> <td>file continued on next diskette</td> </tr> <tr> <td>L</td> <td>last diskette on which file resides</td> </tr> </tbody> </table>	Character	Meaning	blank	file on 1 diskette	C	file continued on next diskette	L	last diskette on which file resides
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 multi-volume 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 indicate nonsignificance of creation date.								
Record length	53-56	Defines record length If blank, record length equals block length (position 22). (This field is ignored because blank in position 43 means the same.)								

Table D-12. Diskette File Label Description (Part 2 of 2)

Field	Byte Position	Description								
—	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.) blank = file date expired 999999 = file date never expires								
Verify/copy indicator	72	<table border="0"> <thead> <tr> <th><u>Character</u></th> <th><u>Meaning</u></th> </tr> </thead> <tbody> <tr> <td>blank</td> <td>file is created</td> </tr> <tr> <td>V</td> <td>file is verified</td> </tr> <tr> <td>C</td> <td>file data successfully transferred to another medium.</td> </tr> </tbody> </table>	<u>Character</u>	<u>Meaning</u>	blank	file is created	V	file is verified	C	file data successfully transferred to another medium.
<u>Character</u>	<u>Meaning</u>									
blank	file is created									
V	file is verified									
C	file data successfully transferred to another medium.									
File organization	73	If position 43 contains blank, this field must also contain blanks.								
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 sector beyond file extent, the entire extent was used.								
Reserved	79	Reserved								
—	80-127	Padded with binary zeros								

NOTE:

ISAM does not support 8413 diskette.



Appendix E. Tape File Labels



OS/3 SYSTEM STANDARD LABELS FOR MAGNETIC TAPE

This appendix describes the system standard labels for magnetic tape volumes in SPERRY UNIVAC Operating System/3 (OS/3). Appendix J describes OS/3 magnetic tape volume organization and file conventions. *Tape user label processing* and *tape volume label processing* in OS/3 are described in the glossary.

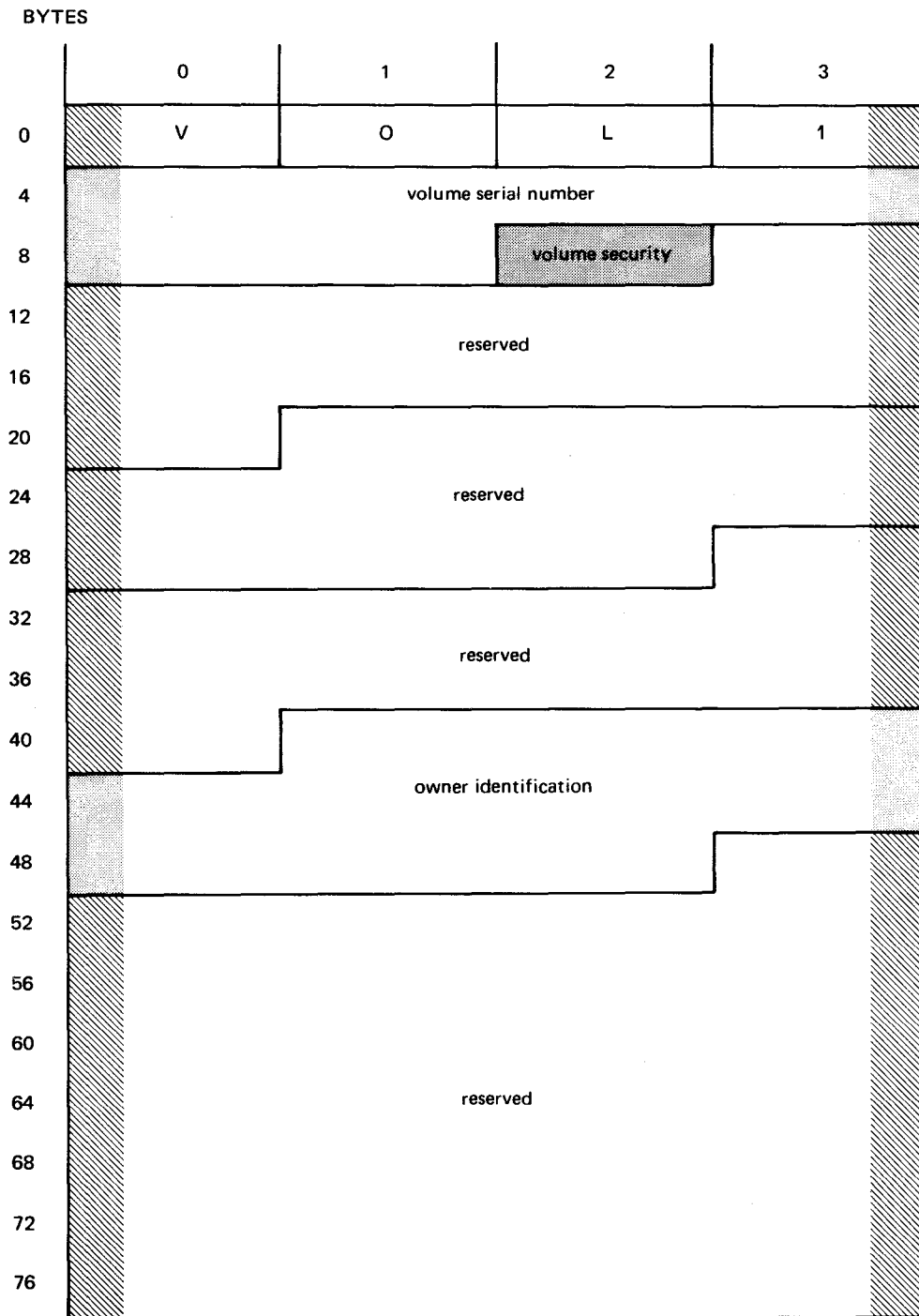
SYSTEM STANDARD TAPE LABEL GROUPS

All standard tape labels including optional user header labels (UHL) and user trailer labels (UTL) are in blocks of 80 bytes. There are five tape label groups, three required and two optional:

- Volume label group
- File header label group
- User header label group (optional)
- File trailer label group
- User trailer label group (optional)

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. (Refer to Table A—11 for these actions of the OPEN transient.) Figure E—1 shows the format of the VOL1 label; its fields are described in Table E—1.



LEGEND:



-  Generated by data management or reserved for system expansion
-  Written by data management from user-supplied data

Figure E-1. Tape Volume 1 (VOL1) Label Format for an EBCDIC Volume

Table E—1. Tape Volume 1 (VOL1) Label Format, Field Description for an EBCDIC Volume

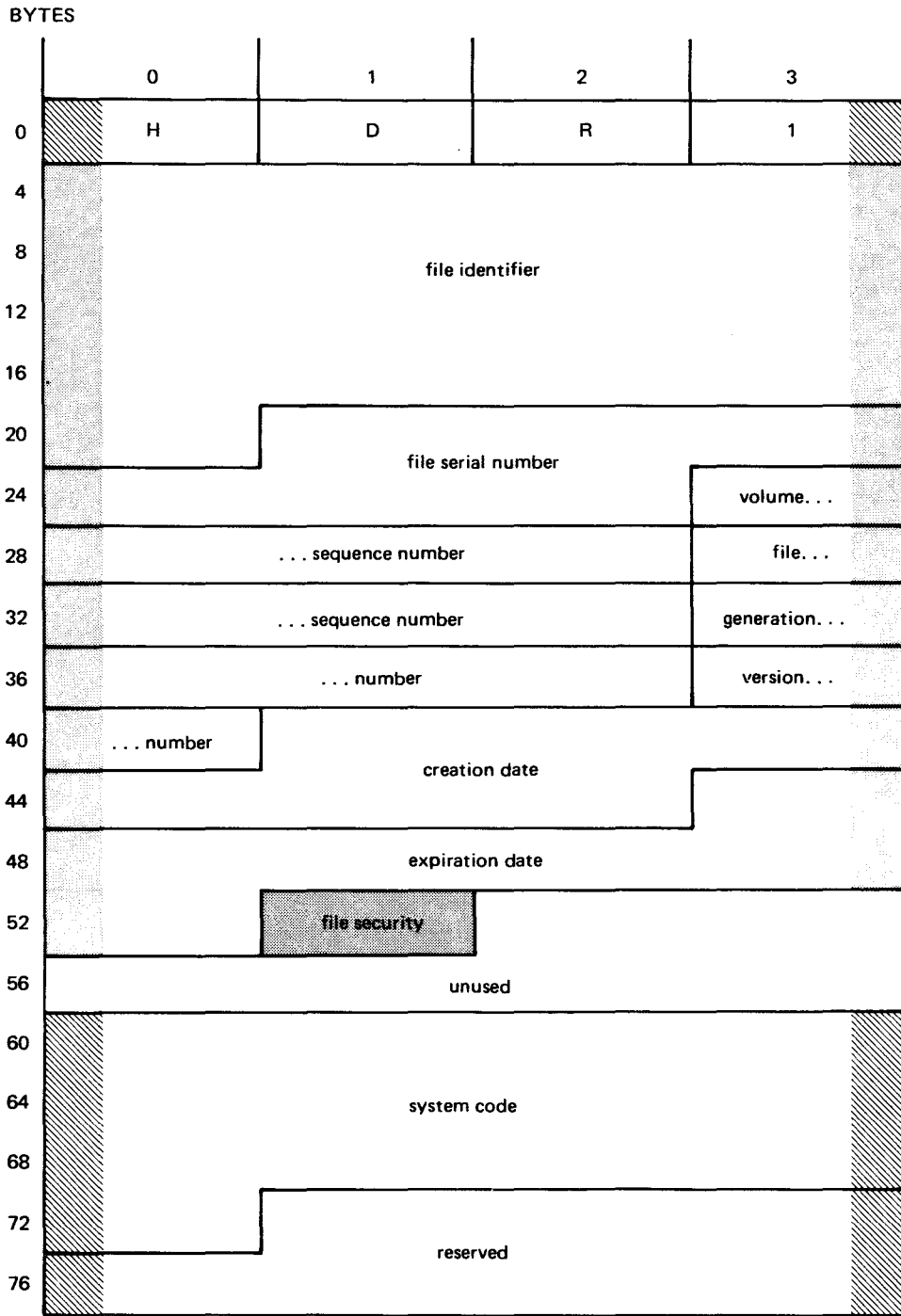
Field	Initialized by	Bytes	Code	Description
Label identifier	Tape prep	0–2	EBCDIC	Contains "VOL" to indicate that this is a volume label
Label number	Tape prep	3	EBCDIC	Always "1", for the initial volume label
Volume serial number	Tape prep	4–9	EBCDIC	Unique identification number assigned to this volume by the user's installation. Tape SAM expects one to six alphanumeric characters, the first of which is alphabetic.
Volume security	Data Mgt	10	EBCDIC	Reserved for future use by installations requiring security at the reel level. Currently blank (40 ₁₆)
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 identification	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)

File Header Label Group

The file header label group comprises two labels, HDR1 and HDR2, described in the following subparagraphs.

First File Header Label (HDR1)

The first file header (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 E—2. All fields in the HDR1 label may be specified in the job control stream; these are described in Table E—2. Refer also to Table A—11 for actions of the OPEN transient checking or creating the file serial number field of the HDR1 label.



LEGEND:



-  Generated by data management or reserved for system expansion
-  Written by data management from user-supplied data

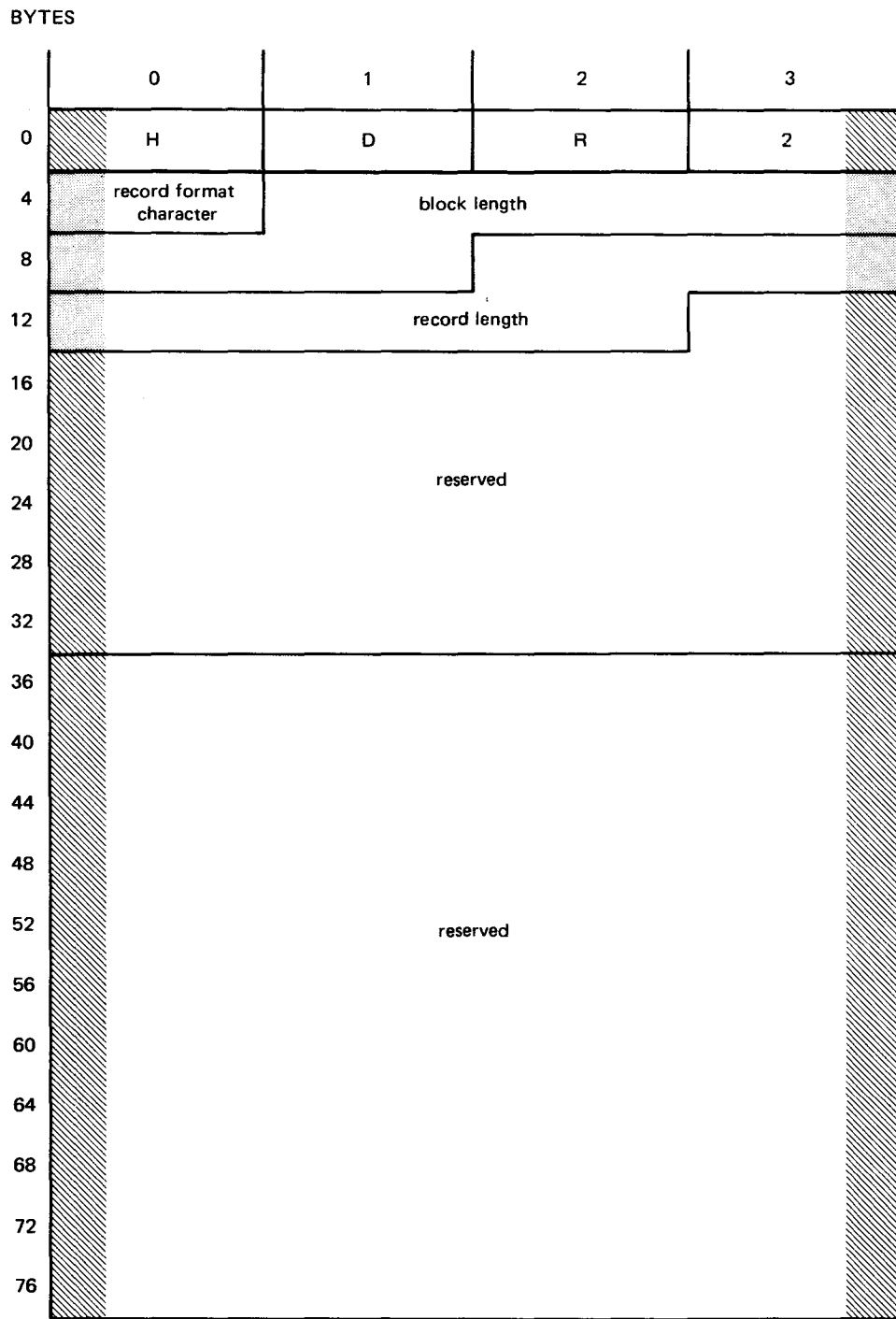
Figure E-2. First File Header Label (HDR1) Format for an EBCDIC Tape Volume

Table E-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 on which the file was created, expressed in the form <i>yyddd</i> 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	Contains EBCDIC 0's
System code	60-72	Reserved for system code, the unique identification of the operating system that produced the file
Reserved	73-79	Contains blanks (40_{16})

Second File Header Label (HDR2)

The second file header label (HDR2) acts as an extension of the HDR1 label and is required in standard-labeled files. Figure E-3 shows the format of the HDR2 label; Table E-3 describes its fields.



LEGEND:



-  Generated by data management or reserved for system expansion
-  Written by data management from user-supplied data

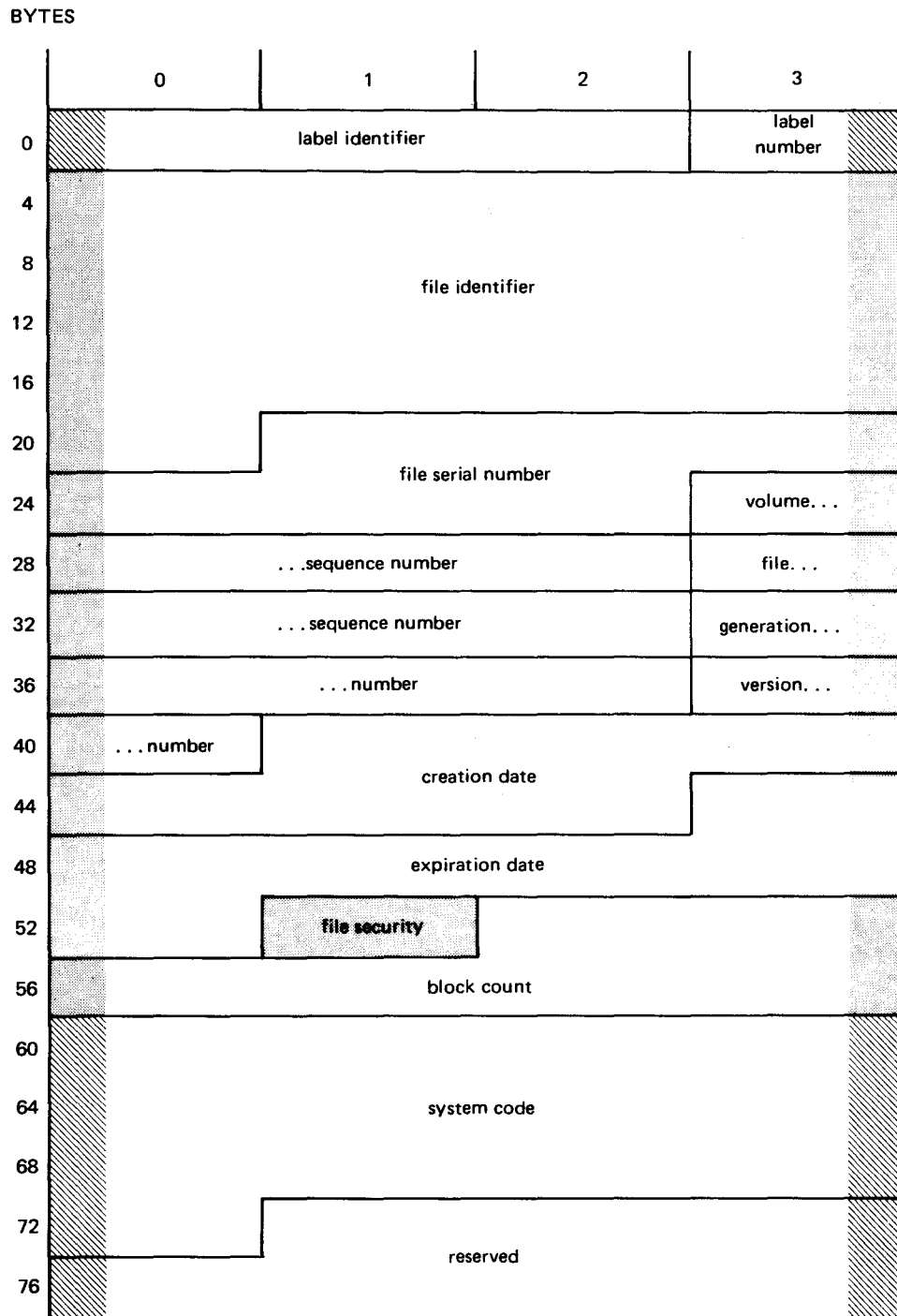
Figure E-3. Second File Header Label (HDR2), Format for an EBCDIC Tape Volume

Table E-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	<table border="1"> <thead> <tr> <th>Character</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>D</td> <td>Variable-length, with length fields specified in decimal</td> </tr> <tr> <td>F</td> <td>Fixed-length</td> </tr> <tr> <td>U</td> <td>Undefined</td> </tr> <tr> <td>V</td> <td>Variable-length, with length fields specified in binary</td> </tr> </tbody> </table>	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
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 0's.										
Reserved	15-35	For future system use										
Reserved	36-79	For future system use										

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. Figure E-4 illustrates the format of the EOF1 or EOV1 label; Table E-4 summarizes the contents of its fields. Similarly, Figure E-5 and Table E-5 describe the EOF2 or EOV2 label.



LEGEND:



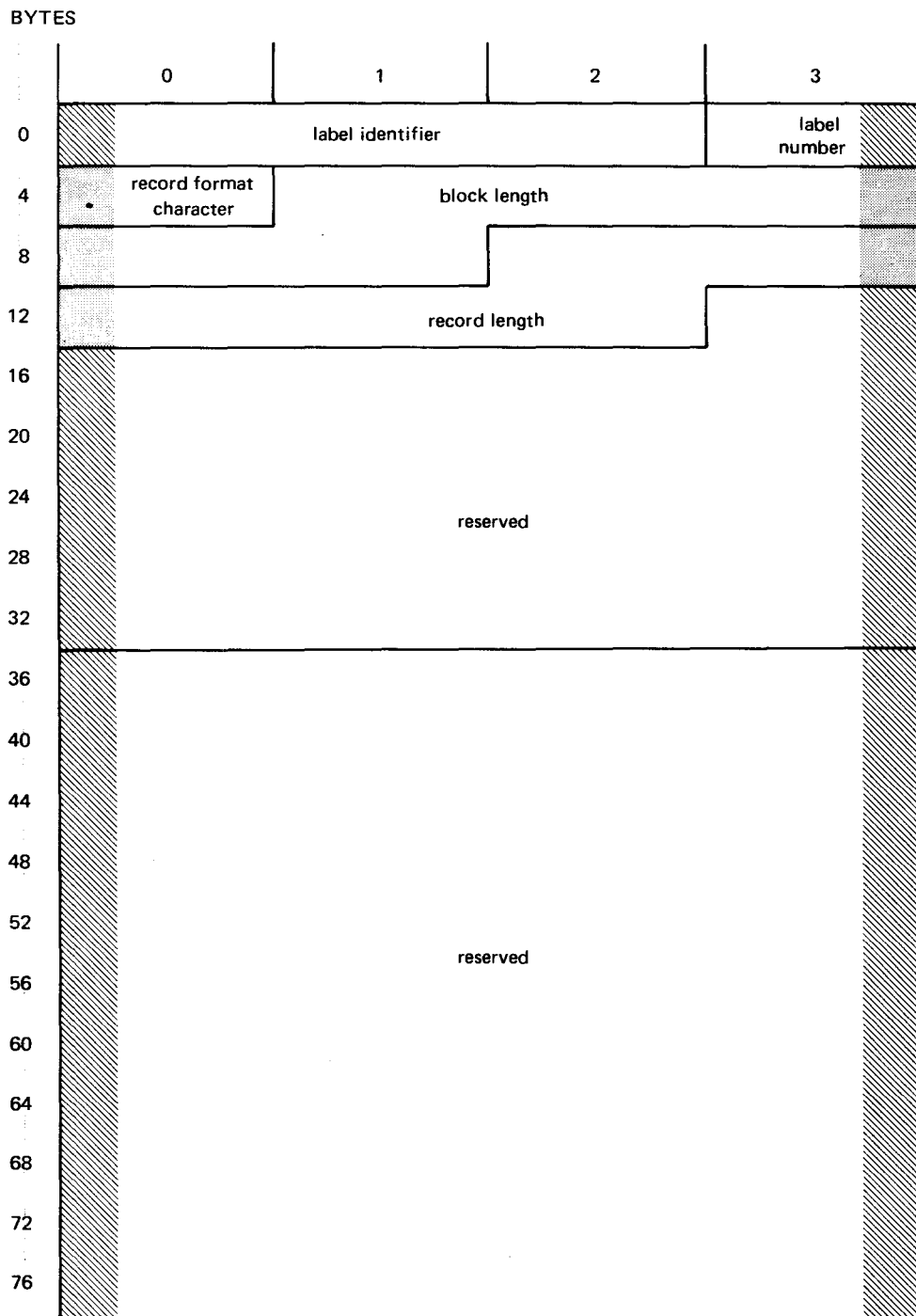
-  Generated by data management or reserved for system expansion
-  Written by data management from user-supplied data

Figure E-4. Tape File EOF1 and EOVI Label Formats

Table E-4. Tape File EOF1 and EOVI 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 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 on which the file was created, expressed in the form <i>yyddd</i> 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. Tape SAM 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	Contains blanks (40 ₁₆)



LEGEND:



-  Generated by data management or reserved for system expansion
-  Written by data management from user-supplied data

Figure E-5. Tape File EOF2 and EO2 Label Formats

Table E—5. Tape File EOF2 and EO2 Labels, Field Description

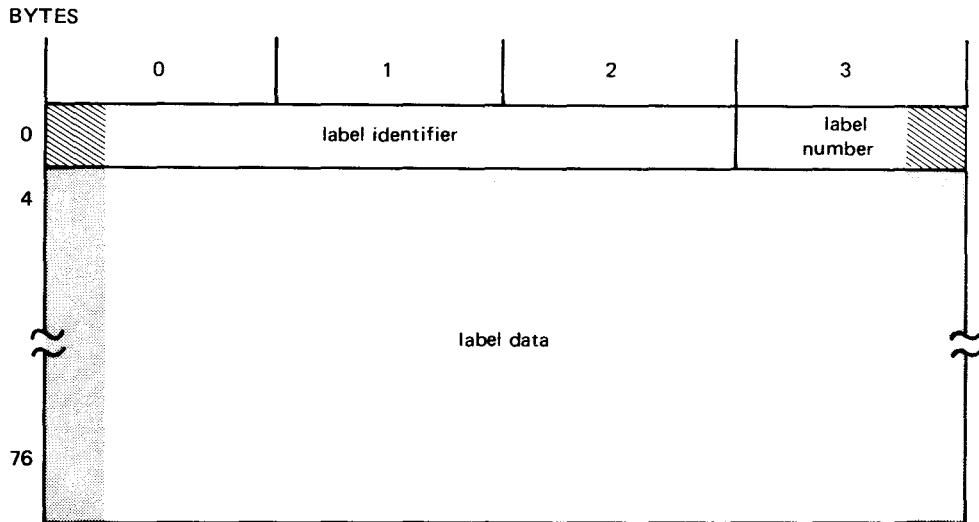
Field	Bytes	Description										
Label identifier	0–2	Indicates that this is a file trailer label. Contains "EOF" for an end-of-file label, or "EOV" for an end-of-volume label										
Label number	3	Always "2"										
Record format character	4	<table border="1"> <thead> <tr> <th>Character</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>D</td> <td>Variable-length, with length fields specified in decimal</td> </tr> <tr> <td>F</td> <td>Fixed-length</td> </tr> <tr> <td>U</td> <td>Undefined</td> </tr> <tr> <td>V</td> <td>Variable-length, with length fields specified in binary</td> </tr> </tbody> </table>	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
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 0's.										
Reserved	15–35	For future system use										
Reserved	36–79	For future system use										

Standard User Header and Trailer Labels

In an ASCII or a standard-labeled EBCDIC file, only the standard UHL and UTL may be used. Within the OS/3 tape SAM conventions, their use is entirely optional. The EBCDIC user may have one or as many as eight labels of each type, or none at all. There is no upper limit to the range of the label number in ASCII files. The user may have as many UHLs as he wants. The number of UTLs the user may have is not restricted by data management but by the requirement of recorded magnetic tape standards that the area available for recording information after the end-of-tape (EOT) marker be limited to 10 feet. If they are used, they must be in the OS/3 standard 80-byte format and content described in Figure E—6 and Table E—6. Their positions in a standard-labeled tape volume are prescribed in Appendix J.

Table E—6. Optional User Header and Trailer Labels, Field Description for ASCII and Standard-Labeled EBCDIC Tape Files

Field	Bytes	Description
Label identifier	0–2	Contains "UHL" for use header label; "UTL" for user trailer label
Label number	3	Ranges from 1 through 8
Label data	4–79	Contains 76 bytes of user-specified information



LEGEND:

- Written by user's LABADDR routine
- Written by data management for EBCDIC files; however, to write the label number in ASCII files is the user's responsibility.

Figure E-6. Optional User Header and Trailer Label Format, ASCII and Standard-Labeled EBCDIC Tape Files

ASCII STANDARD MAGNETIC TAPE LABELS

The figures and tables that follow describe the labels written (and expected to be read) by OS/3 magnetic tape SAM 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*.

OS/3 magnetic tape SAM writes and processes the following ASCII standard labels: VOL1, HDR1, HDR2, EOVI, EOVI, 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 for EBCDIC files), 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.

For ASCII record formats and reel organizations, refer to Appendix J.

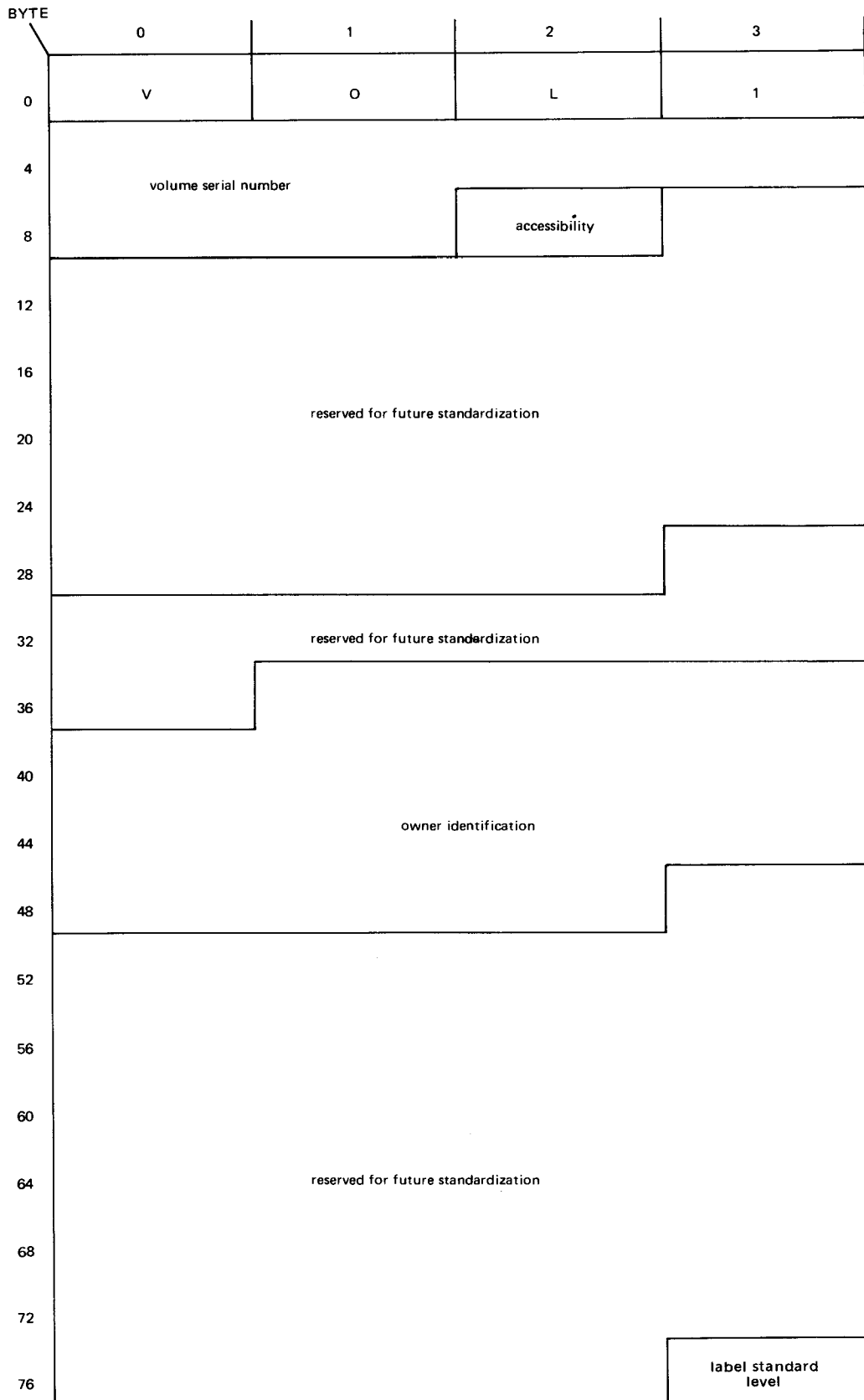


Figure E-7. Volume Header Label (VOL1) for an ASCII Magnetic Tape Volume

Table E-7. Volume Header Label (VOL1), Field Description for an ASCII 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 the volume. A "space" means unlimited access; any other character means special handling, in the manner agreed upon between the interchange parties. Data management always writes a "space" in the accessibility field in output files and places no restrictions on the content of this field in input files. 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 <i>American National Standard X3.27-1969</i> . "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 (American National Standard Code for Information Interchange).

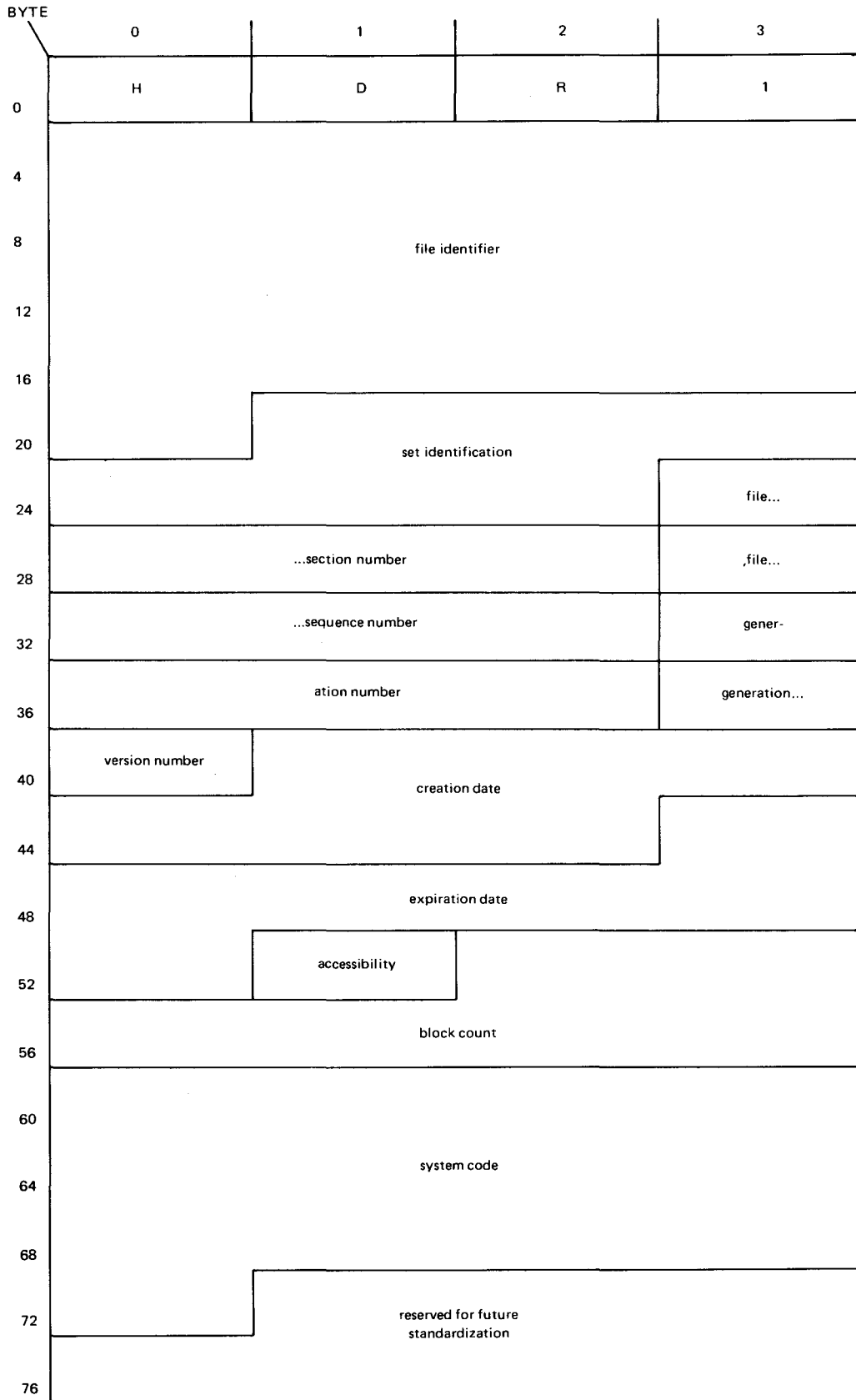


Figure E-8. First File Header Label (HDR1) for an ASCII Magnetic Tape Volume

Table E-8. First File Header Label (HDR1), Field Description for an ASCII 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 will contain 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 restriction on who may have access to the information in the file. A "space" means unlimited access; any other character means special handling, in a manner agreed upon between the interchange parties. Data management always writes a "space" in the accessibility field in output files and places no restriction on the content of this field. See Notes 1 and 2.
Block count	54-59	Must be "zeros"
System code (optional)	60-72	Thirteen "a" characters identifying the operating system that recorded this file. Output tapes written by OS/3 tape SAM contain "OS3".
(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 X3.4-1968*), 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".

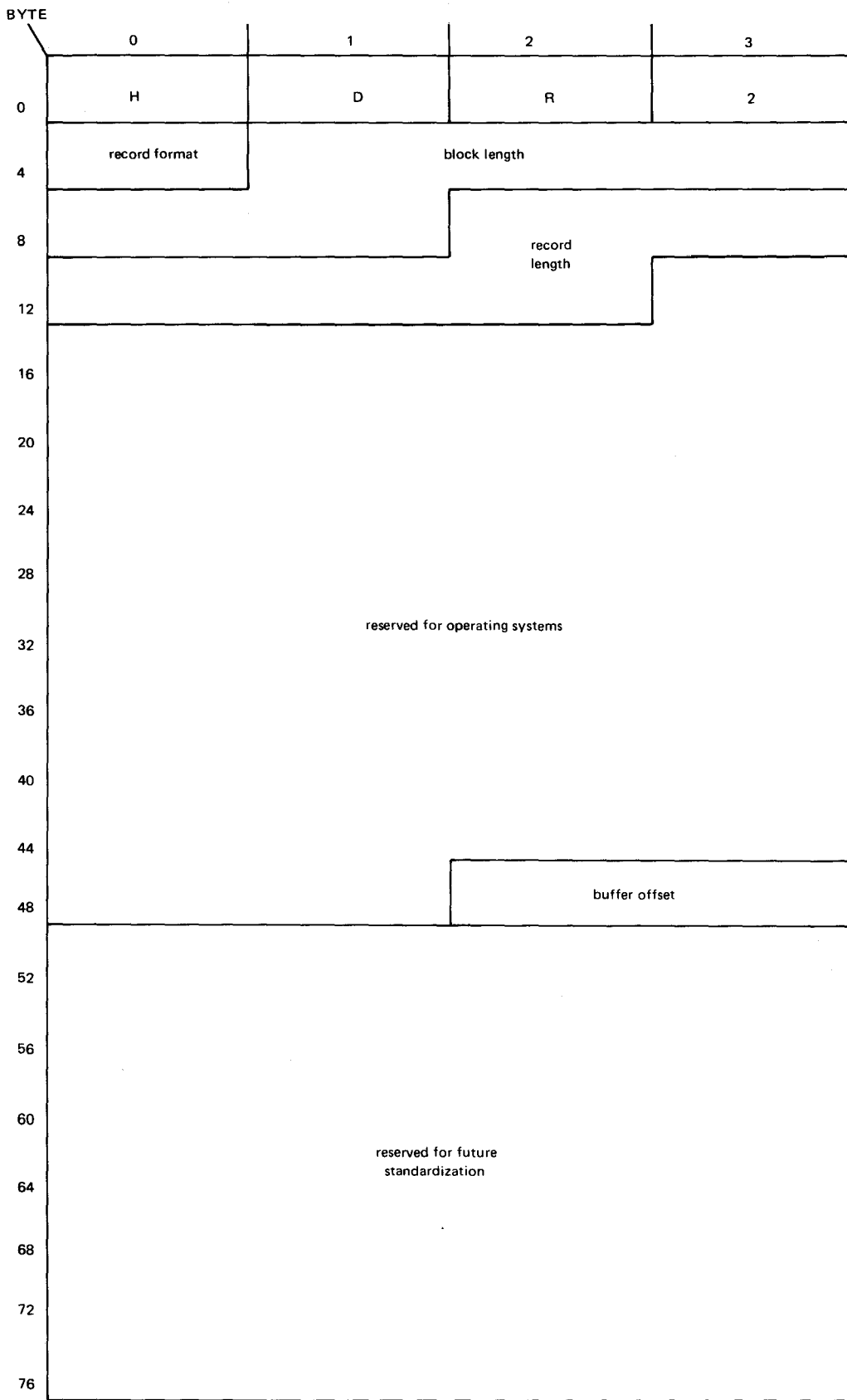


Figure E—9. Second File Header Label (HDR2) for an ASCII Magnetic Tape Volume

Table E-9. Second File Header Label (HDR2), Field Description for an ASCII Volume

Field	Bytes	Description										
Label identifier	0-2	Must be "HDR"										
Label number	3	Must be "2"										
Record format	4	<table border="1"> <thead> <tr> <th>Character</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>F</td> <td>Fixed length</td> </tr> <tr> <td>D</td> <td>Variable, with the number of characters in the record specified in decimal</td> </tr> <tr> <td>V</td> <td>Variable, with the number of characters specified in binary. See Note 1.</td> </tr> <tr> <td>U</td> <td>Undefined</td> </tr> </tbody> </table>	Character	Meaning	F	Fixed length	D	Variable, with the number of characters in the record specified in decimal	V	Variable, with the number of characters specified in binary. See Note 1.	U	Undefined
Character	Meaning											
F	Fixed length											
D	Variable, with the number of characters in the record specified in decimal											
V	Variable, with the number of characters specified in binary. See Note 1.											
U	Undefined											
Block length	5-9	Five "n" characters specifying the maximum number of characters per block. See Note 2.										
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.										
(reserved)	15-49	Reserved for OS/3 use; currently "spaces".										
Buffer offset (optional)	50-51	Two "n" characters specifying the length in characters of any additional field inserted before data block - for example, block length. This length is included in the block length. See Notes 2 and 4.										
(reserved)	52-79	Reserved for future standardization; must be "spaces". See Note 3.										

NOTES:

- OS/3 magnetic tape SAM does not support the ASCII "V-format" record.
- An "n" character is any ASCII numeric digit, from 0 through 9.
- A "space" is the normally nonprinting graphic character occupying position 2/0 in ASCII.
- "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".

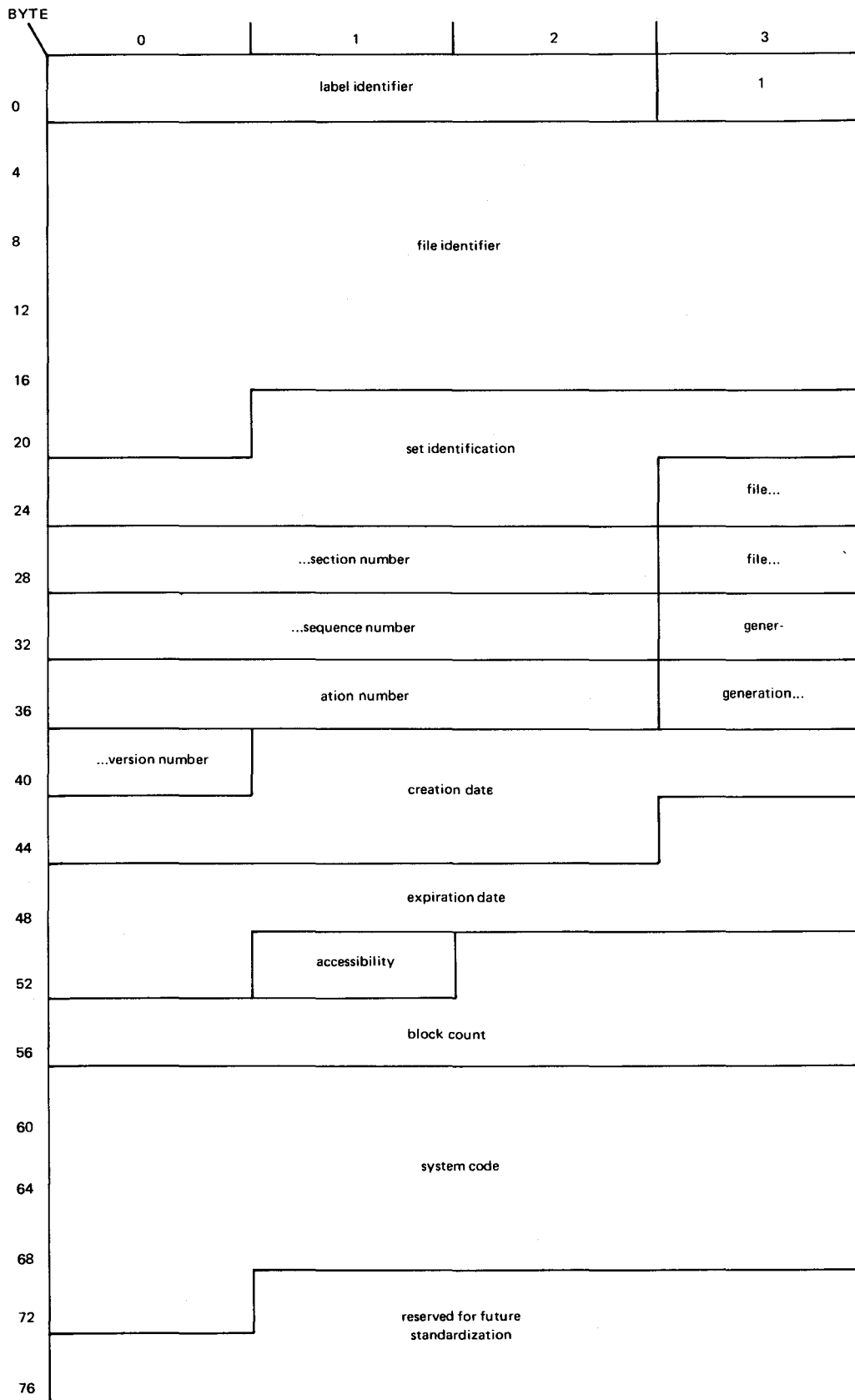


Figure E-10. First End-of-File or End-of-Volume Label (EOF1/EOV1) for an ASCII Magnetic Tape Volume

Table E-10. First End-of-File or End-of-Volume Label (EOF1/EOV1), Field description for an ASCII 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.
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 will contain 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 restriction on who may have access to the information in the file. A "space" means unlimited access; any other character means special handling, in a manner agreed upon between the interchange parties. Data management always writes a "space" in the accessibility field of output files and places no restriction on the content of this field in input files. See Notes 1 and 2.
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. Output tapes written by OS/3 tape SAM contain "OS3". See Note 1.
(reserved)	73-79	Reserved for future standardization; must contain "spaces". See Note 3.

NOTES:

1. An "a" character is any of the characters occupying the center four columns of ASCII (depicted in *American National Standard X3.4-1968*), 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".

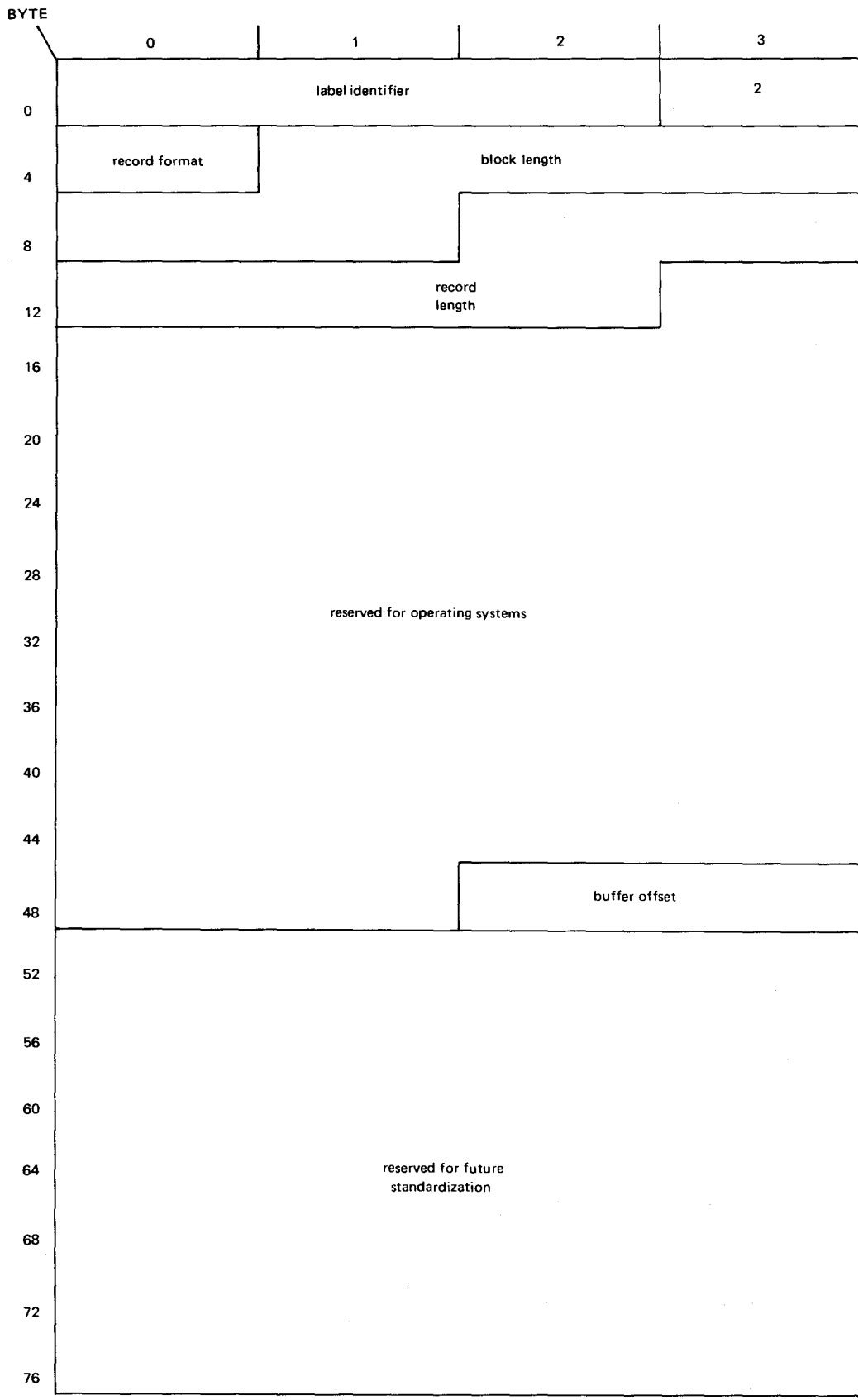


Figure E-11. Second End-of-File or End-of-Volume Label (EOF2/EOV2) for an ASCII Magnetic Tape Volume

Table E-11. Second End-of-File or End-of-Volume Label (EOF2/EOV2), Field Description for an ASCII 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	<table border="1"> <thead> <tr> <th>Character</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>F</td> <td>Fixed-length</td> </tr> <tr> <td>D</td> <td>Variable, with the number of characters in the record specified in decimal</td> </tr> <tr> <td>V</td> <td>Variable, with the number of characters specified in binary. See Note 1.</td> </tr> <tr> <td>U</td> <td>Undefined</td> </tr> </tbody> </table>	Character	Meaning	F	Fixed-length	D	Variable, with the number of characters in the record specified in decimal	V	Variable, with the number of characters specified in binary. See Note 1.	U	Undefined
Character	Meaning											
F	Fixed-length											
D	Variable, with the number of characters in the record specified in decimal											
V	Variable, with the number of characters specified in binary. See Note 1.											
U	Undefined											
Block length	5-9	Five "n" characters specifying the maximum number of characters per block. See Note 2.										
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 Notes 1 and 2.										
(reserved)	15-49	Reserved for OS/3 use; currently "spaces". See Note 3.										
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 2 and 4.										
(reserved)	52-79	Reserved for future standardization; must be "spaces". See Note 3.										

NOTES:

- OS/3 magnetic tape SAM does not support the ASCII "V-format" record.
- An "n" character is any ASCII numeric digit, from 0 through 9.
- A "space" is the normally nonprinting graphic character occupying position 2/0 in ASCII.
- "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".

Appendix F. Nonstandard DTF Keyword Parameters



DTFCD KEYWORDS ACCEPTED, BUT NOT IMPLEMENTED

SPERRY UNIVAC 9200/9300 Systems

The following 9200/9300 keywords are accepted by the OS/3 DTFCD macro, but are not implemented:

CHNL	INCR	TYPC
DEVA	SENT	

SPERRY UNIVAC OS/4 System

The only OS/4 keyword which is not implemented in the OS/3 DTFCD macro is the CTLCHR keyword.

SPERRY UNIVAC OS/7 System

The following OS/7 DTFCD keywords are accepted, but not implemented in the OS/3 DTFCD macro:

CTLCHR	RDONLY
OPENXIT	SEPASMB

IBM 360/20 System

The following 360/20 keywords are accepted but not implemented in the OS/3 DTFCD macro:

BINARY	OVERLAP	RFXIT
CRDPRA	PFORMAT0—PFORMAT9	SEQNCE
CRDPRL1—CRDPRL6	PFXIT	SEQXIT
DEVICE	RFORMAT0—RFORMAT9	

Other IBM S/360 Systems

The following S/360 keywords are accepted by the OS/3 DTFCD macro but are not implemented:

CTLCHR	MODNAME
DEVADDR	RDONLY
DEVICE	SEPASMB

ALTERNATE DTFCD KEYWORD SPELLINGS

The following are alternate keyword spellings accepted by the OS/3 DTFCD macro:

<u>OS/3, OS/7, 360/20 Spelling</u>	<u>9200/9300 Spelling</u>	<u>9400 Spelling</u>
BLKSIZE=n	—	BKSZ=n
CONTROL=YES	CNTL=YES	CNTRL=YES
CRDERR=RETRY	PUNR=YES	PUNR=YES
EOFADDR=symbol	EOFA=symbol	EOFA=symbol
IOAREA1=symbol	INAR=symbol (input and combined files) OUAR=symbol (output files)	IOA1=symbol
IOAREA2=symbol	OUAR=symbol (combined files)	IOA2=symbol
IOREG=(r)	—	IORG=(r)
OUBLKSZ=n	—	OBSZ
RECFORM	—	RCFM
RECSIZE=(r)	—	RCSZ=(r)
TYPEFLE = $\left\{ \begin{array}{l} \text{INPUT} \\ \text{OUTPUT} \\ \text{CMBND} \end{array} \right\}$	TYPEF = $\left\{ \begin{array}{l} \text{INPUT} \\ \text{OUTPUT} \\ \text{COMBND} \end{array} \right\}$	TYPE = $\left\{ \begin{array}{l} \text{INPUT} \\ \text{OUTPUT} \\ \text{COMBND} \end{array} \right\}$
WORKA=YES	—	WORK=YES

The following 360/20 keywords, when specified for a combined file, are handled in the following manner:

INBLKSZ=n	is treated as	BLKSIZE=n
INAREA=symbol	is treated as	IOAREA1=symbol
OUAREA=symbol	is treated as	IOAREA2=symbol

DTFPR KEYWORDS ACCEPTED, BUT NOT IMPLEMENTED**SPERRY UNIVAC 9200/9300 Systems**

The following 9200/9300 keywords are accepted by the OS/3 DTFPR macro, but are not implemented:

CH1—CH7	FONT	RPNO
CHNL	GENO	SWAP
CRDT	HDR	VOLN
DEVA	OTBL	XPDT
FLID	PBUF	

SPERRY UNIVAC OS/4 System

The following OS/4 keywords are accepted by the OS/3 DTFPR macro, but are not implemented:

ASCII	CODE
AUE	CTLCHR=YES

Note that a CTLCHR=DI keyword parameter is implemented in OS/3.

SPERRY UNIVAC OS/7 System

The following OS/7 DTFPR keywords are accepted, but are not implemented in the OS/3 DTFPR macro:

ASCII	CTLCHR=NOTRANS	RONLY
CODE	CTLCHR=ASA	SEPASMB
CTLCHR=YES	OPENXIT	VFB

It should be noted that a CTLCHR=DI keyword parameter is implemented. The OS/3 device-independent control characters are a compatible subset of the OS/7 device-independent control characters. Refer to Table A-4.

IBM 360/20 System

The only 360/20 DTFPR keyword for printer files which is accepted but not implemented in the OS/3 DTFPR macro is the DEVICE keyword.

Other IBM S/360 Systems

The following S/360 DTFPR keywords are accepted by, but are not implemented in, the OS/3 DTFPR macro:

DEVADDR	RONLY
DEVICE	SEPASMB
MODNAME	STLIST

ALTERNATE DTFPR KEYWORD SPELLINGS

The following alternate keyword spellings are accepted by the OS/3 DTFPR macro:

<u>OS/3, OS/7, 360/20 Spelling</u>	<u>9200/9300 Spelling</u>	<u>9400 Spelling</u>
BLKSIZE=n	BKSZ=n	BKSZ=n
CONTROL=YES	CNTL=YES	CNTRL=YES
IOAREA1=symbol	—	IOA1=symbol
IOAREA2=symbol	—	IOA2=symbol
IOREG=(r)	—	IORG=(r)
PRINTOV=SKIP	PROV=YES	PRTOV=YES
PRINTOV=symbol	PROV=symbol	PRTOV=symbol
RECFORM	—	RCFM
RECSIZE=(r)	—	RCSZ=(r)
WORKA=YES	—	WORK=YES

NONSTANDARD DTFMT KEYWORD PARAMETERS

The following nonstandard forms are accepted for the DTFMT declarative macro:

<u>OS/3 Standard Form</u>	<u>Accepted and Implemented</u>	<u>Accepted But Not Implemented</u>
BLKSIZE	BKSZ	CRDT
CKPTREC	CKPT	GENO
EOFADDR	EOFA	DEVA
ERROR	ERRO	VOLN
IOAREA1	IOA1	FLBL
IOAREA2	IOA2	FLID
IORG	IORG	XPDT
LABADDR	LBAD	
RECFORM	RCFM	
RECSIZE	RCSZ	
TYPEFLE	TYPEF,TYPE	
TPMARK	TPMK	
VARBLD	VBLD	
WORKA	WORK	

NONSTANDARD DTFDA, DTFSD, DTFNI, AND DPCA KEYWORD PARAMETERS

The following nonstandard forms are accepted for all DTFs in the OS/3 nonindexed disc file processor system:

<u>OS/3 Standard Form</u>	<u>Acceptable Nonstandard Spellings</u>
AFTER	AFTR
BLKSIZE	BKSZ
EOFADDR	EOFA
ERROPT	ERRO
IOAREA1	IOA1
IOAREA2	IOA2
IOREG	IORG
KEYARG	KARG
KEYLEN	KLEN
LABADDR	LBAD
RECFORM	RCFM
RECSIZE	RCSZ
READID	RDID
READKEY	RDKY
RELATIVE	REL
SEEKADR	SKAD
SRCHM	SRCM
TYPEFLE	TYPF
UPDATE	UPDT
VARBLD	VBLD
VERIFY	VERFY
WRITEID	WRID
WRITEKEY	WRKY
WORKA	WORK

NONSTANDARD DTFIS KEYWORD PARAMETERS

The following nonstandard forms are accepted for the DTFIS declarative macro:

<u>OS/3 Standard Form</u>	<u>Acceptable Nonstandard Spellings</u>
BLKSIZE	BKSZ
ERROR	ERRO
INDAREA	INDA
INDSIZE	INDS
IOAREA1	IOA1, IOAREAL, IOAREAR, IOAREAS
IOAREA2	IOA2
IOREG	IORG
IOROUT	IORT
KEYARG	KARG
KEYLEN	KLEN
KEYLOC	KLOC
PCYLOFL	PCYL, CYLOFL, CYL
RECFORM	RCFM
RECSIZE	RCSZ
SAVAREA	SAVE
TYPEFLE	TYPF
UPDATE	UPDT
VERIFY	VERFY
WORK1	WRK1, WORKL, WORKR

Note that there are no acceptable variations for the INDEXED or the WORKS keywords and that none of the completion values may vary from the OS/3 standards given in Section 8.

NONSTANDARD DTFIR KEYWORD PARAMETERS

The following variant spellings for the DTFIR keyword parameters are accepted:

<u>DTFIR Spelling</u>	<u>OS/3 Standard Form</u>
BFSZ	BLKSIZE/BKSZ
EOFA	EOFADDR
ERRO	ERROR
INDA	INDAREA
INDS	INDSIZE
INDX	INDEXED
IOA1	IOAREA1
IOA2	IOAREA2
IORG	IOREG
KARG	KEYARG
KLEN	KEYLEN
KLOC	KEYLOC
OPTN	OPTION
RCSZ	RECSIZE
SKAD	SEEKADDR
TYPE	TYPEFLE/TYPF
UPDT	UPDATE
VERFY	VERIFY
WORKA	WORKA

NONSTANDARD DTFMI KEYWORD PARAMETERS

The following variant spellings for the DTFMI keyword parameters are accepted:

<u>DTFMI Spelling</u>	<u>OS/3 Standard Form</u>
BFSZ	BLKSIZE/BKSZ
EOFA	EOFADDR
ERRO	ERROR
INDA	INDAREA
INDS	INDSIZE
IOA1	IOAREA1
IOA2	IOAREA2
KARG	KEYARG
OPTN	OPTION
RCFM	RECFORM
RCSZ	RECSIZE
SKAD	SEEKADDR
VERFY	VERIFY
WORK	WORK

DTFPT KEYWORD COMPATIBILITY WITH OS/4

All OS/4 DTFPT keyword parameters are accepted in OS/3; all but the OS/4 ERRO keyword are implemented.

DTFPT KEYWORD COMPATIBILITY WITH 9200/9300

All 9200/9300 Series DTFPT keywords are accepted in OS/3; all are implemented but the following:

ATTN
CHNL
DEVA
FIGS
LTRS

OS/3 does not implement the 9200/9300 letter/shifting capability for input files. To run a 9200/9300 program in OS/3 that uses this capability, the user must remove the FIGS and LTRS keywords from the DTF and substitute the FTRANS, LTRANS, and SCAN keywords, providing the necessary *scan* and *translation tables* in his program. OS/3 does not have a combined paper tape file processing capability: to punch a paper tape and then read it for error checking in OS/3, the user must code two DTFPT declarative macros (one for output, one for input, using different file names) and specify a separate job control *device assignment set* for each file. Error checking must take place in a different pass from punching.

DTFPT KEYWORD COMPATIBILITY WITH IBM S/360 DOS

OS/3 accepts all IBM S/360 DOS keyword parameters and implements all but the following:

DELCHAR

DEVADDR

DEVICE

ERROPT

MODNAME

SEPASM

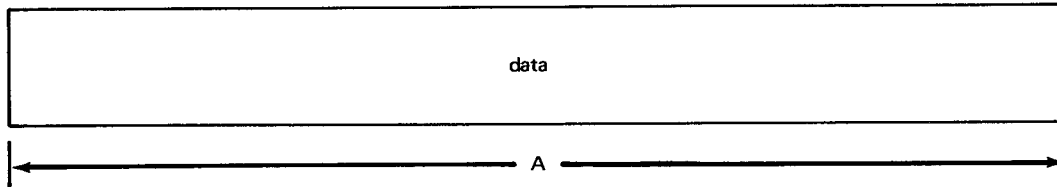
WLRERR



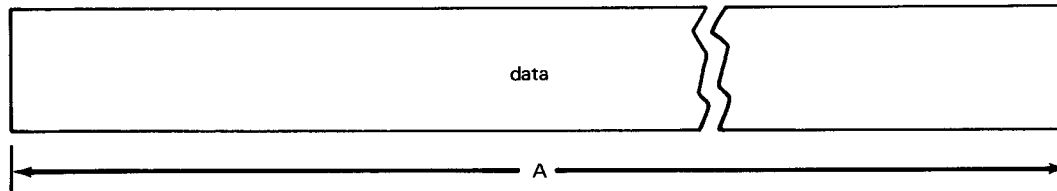
Appendix G. File and Record Formats



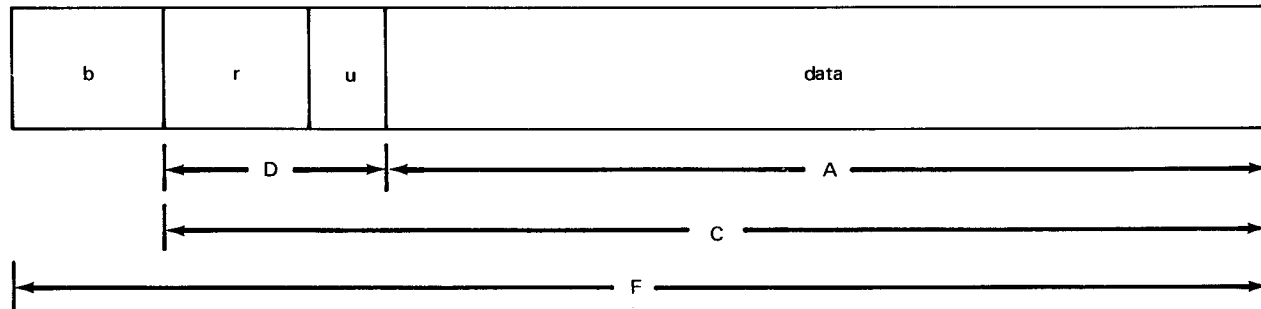
FIXED-LENGTH



UNDEFINED



VARIABLE-LENGTH



LEGEND:

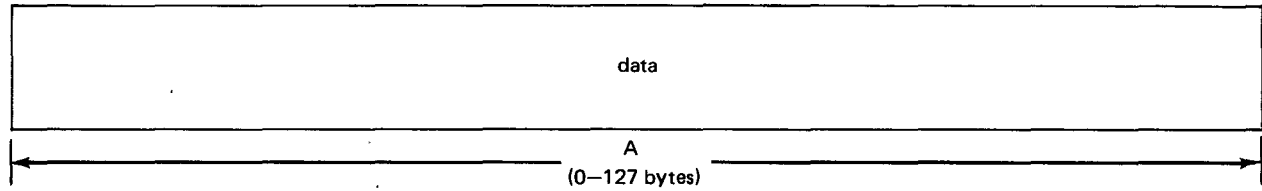
- b Block size field, four bytes, reserved
- r Record length field, two bytes, binary
- u Reserved (two bytes); may be any two characters chosen by the user but does not include his record data
- A Data record length
- C Variable record length
- D Record size field
- F I/O area layout

NOTES:

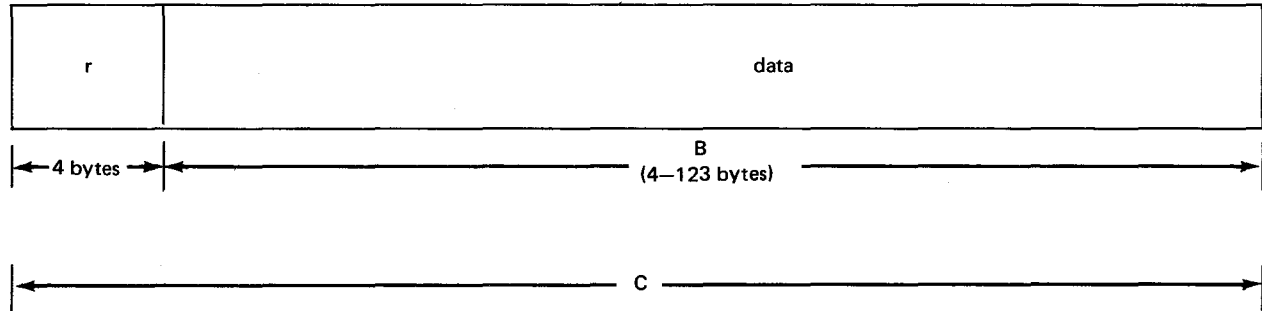
1. An I/O area must be so aligned that the first character to be punched falls on a half-word boundary.
2. Record length, as a binary number, must be placed by the user in the first two bytes of the record-length field (r) before punching a variable-length, unblocked record.
3. An even number of bytes should be allocated for data in I/O areas, even through an odd number of columns are to be punched. The I/O areas for a file with an odd block size should provide at least one byte more than the block size specification.

Figure G-1. Card Punch (Output File) Record Formats

Fixed-length unblocked



Variable-length unblocked



LEGEND:

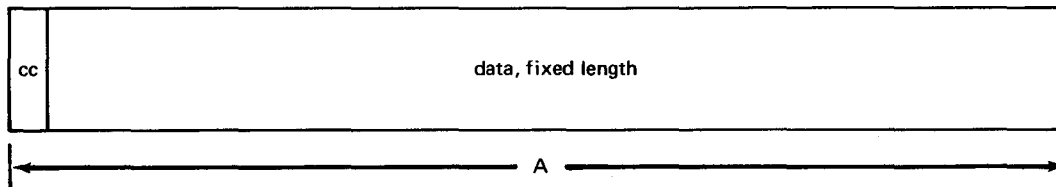
- r Record length field, four bytes
- A Data record length
- B Variable record length
- C I/O area layout

NOTES:

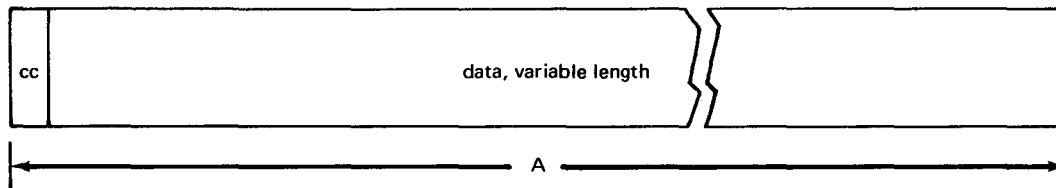
1. For input files, data management passes the record length field and data portion of the record to the user.
2. A record corresponds to one sector and may be up to 128 bytes long in fixed unblocked format and up to 124 bytes long in variable unblocked format.
3. BLKSIZE=n specification on DTFC macro and file label record length determines the size of record processed during OPEN processing. The maximum block size of multi-sector I/O is 1024 bytes.
4. Unused sector space is hardware padded.
5. Under multi-sector I/O, user IOAREA contains multiple records in either fixed unblocked or variable unblocked formats.

Figure G—2. Diskette (Output File) Record Formats

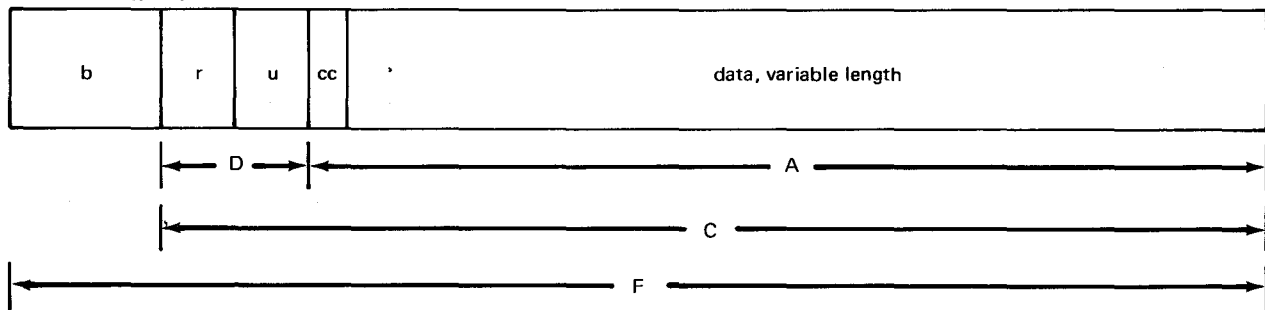
FIXED-LENGTH



UNDEFINED



VARIABLE-LENGTH



LEGEND:

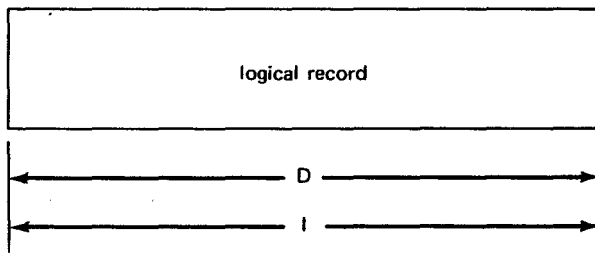
- b Block size field, four bytes
- cc Control character, one byte, optional
- r Record length field, two bytes, binary
- u Reserved (two bytes); can be any two characters you choose
- D Record size field
- A Data record length
- C Variable record length

NOTES:

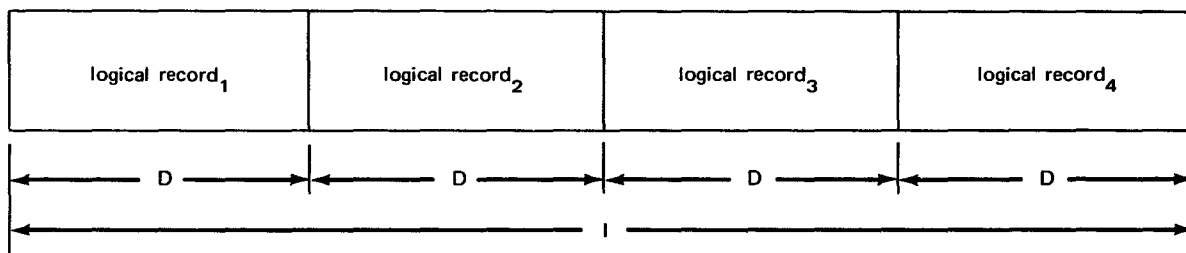
1. User must align an I/O area so that the first character to be printed falls on a half-word boundary.
2. User must place record length, as a binary number, in the first two bytes of the record length field (r) before printing a variable-length, unblocked record. The record length includes the 4-byte record length field and the control character, if any.
3. User should allocate an even number of bytes for data in I/O areas, even though an odd number of columns are to be printed. To print an odd number of columns, the user allocates data areas one byte larger than the number of columns to be printed.

Figure G-3. Printer Record Formats

FIXED-LENGTH, UNBLOCKED RECORD



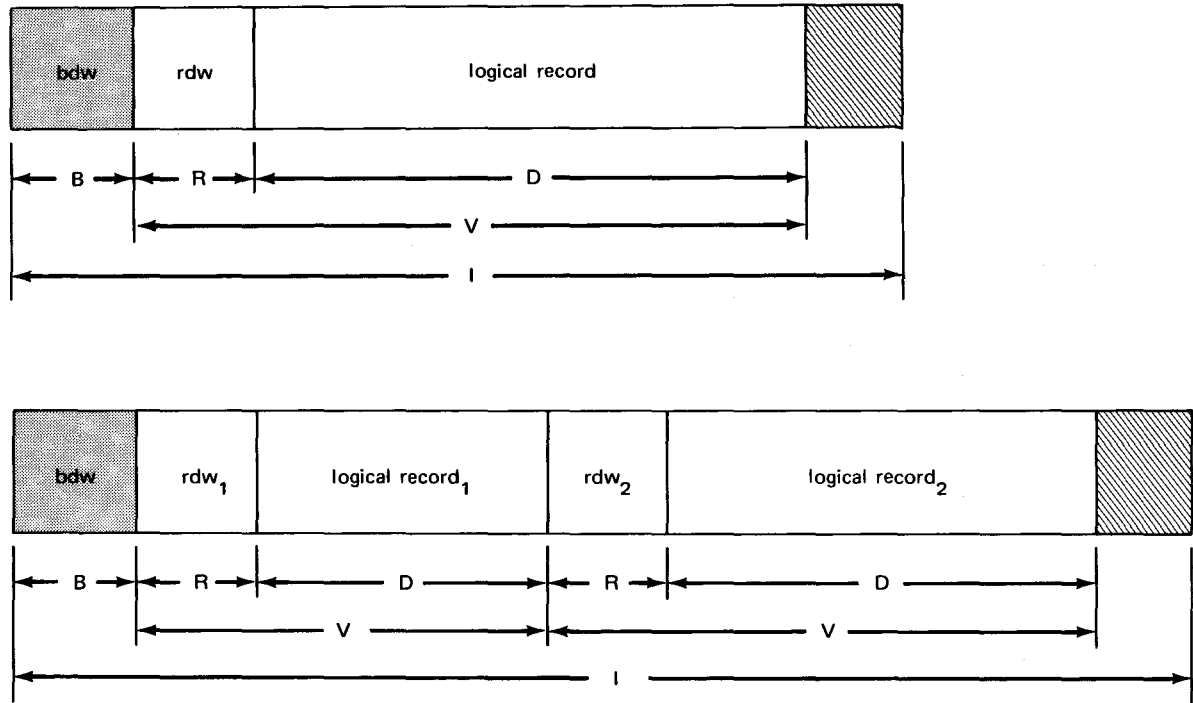
FIXED-LENGTH, BLOCKED RECORDS



LEGEND:

- D Length of data in each logical record; user specifies this length only when his records are blocked, using the RECSIZE keyword parameter in the DTF macro.
- I Minimum length of I/O area; user always specifies this with the BLKSIZE keyword parameter of the DTF macro instruction. If the file is to reside on an 8416 disc the I/O buffer supplied for the specified BLKSIZE must be a multiple of 256 bytes, and therefore the length of the I/O area may exceed the nearest multiple of fixed-record length.

Figure G—4. Fixed-Length Physical Record Formats, Nonindexed Disc Files without Keys



LEGEND:

- B Block descriptor word; four bytes. User must reserve space for this in the I/O area; it is also part of the block on disc. Data management calculates the block length in bytes and writes this in the first two bytes of the BDW; the last two bytes are reserved.
- R Record descriptor word; always the first four bytes of each variable-length record. User determines the length of each logical record in bytes, including this 4-byte RDW, and places it in the first two bytes of the RDW. The last two bytes are reserved.
- D Variable length of the data portion of logical record.
- V Length of the variable record; includes four bytes for the RDW. User inserts this number (measured in bytes) into the first two bytes of the RDW, in binary form.
- I Length of the logical block, both on disc and in the I/O area. The logical block size length, specified via the BLKSIZE keyword parameter, must accommodate the largest variable-length block.



Unused space



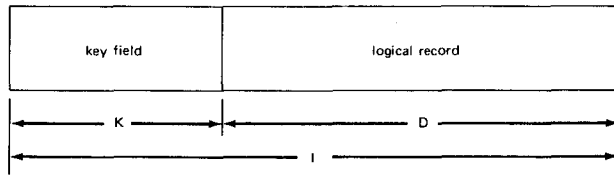
Supplied by data management

NOTE:

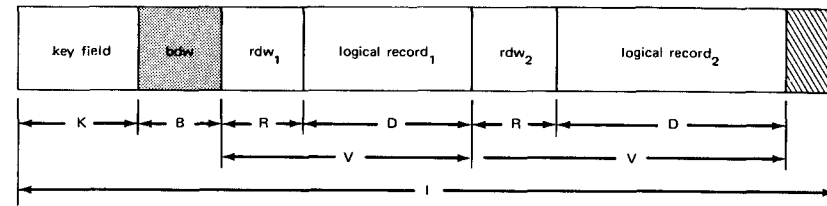
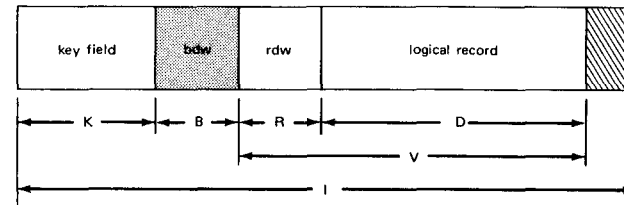
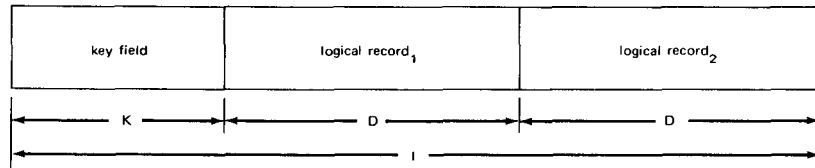
If the file is to reside on an 8416 disc, the length of the I/O buffer supplied must be a multiple of 256 bytes, even though BLKSIZE is less.

Figure G-5. Variable-Length Physical Record Formats, Nonindexed Disc Files without Keys

FIXED-LENGTH, UNBLOCKED RECORD



FIXED-LENGTH, BLOCKED RECORDS



LEGEND:

- K Key field, supplied by user. Minimum 3 bytes; maximum 255. All keys in same partition or file must have same length.
- B Block descriptor word; always four bytes. User reserves space for this at the head of the block, after the key field; data management calculates and inserts block length into first two bytes. Last two bytes are reserved.
- R Record descriptor word; always the first four bytes of a variable-length record. User determines the length of each logical record in bytes, including this 4-byte RDW, and places it in the first two bytes of the RDW. The last two bytes are reserved.
- D Length of the data portion of the logical record; varies for variable-length records. Always the same for each fixed-length record throughout file.

- V Length of a variable record; includes four bytes for the RDW. User inserts this number (measured in bytes) into the first two bytes of the RDW, in binary form.
- I Length of the physical block, both on disc and in the I/O area. The logical block length, specified via the BLKSIZE keyword parameter, must accommodate the largest physical block in a file of variable-length records.

- Unused space, if any
- Supplied by data management

NOTE:

If the file is to reside on an 8416 disc, the length of the I/O buffer supplied must be a multiple of 256 bytes, even though BLKSIZE is less.

Figure G—6. Keyed Fixed- and Variable-Length Physical Record Formats, Nonindexed Disc Files

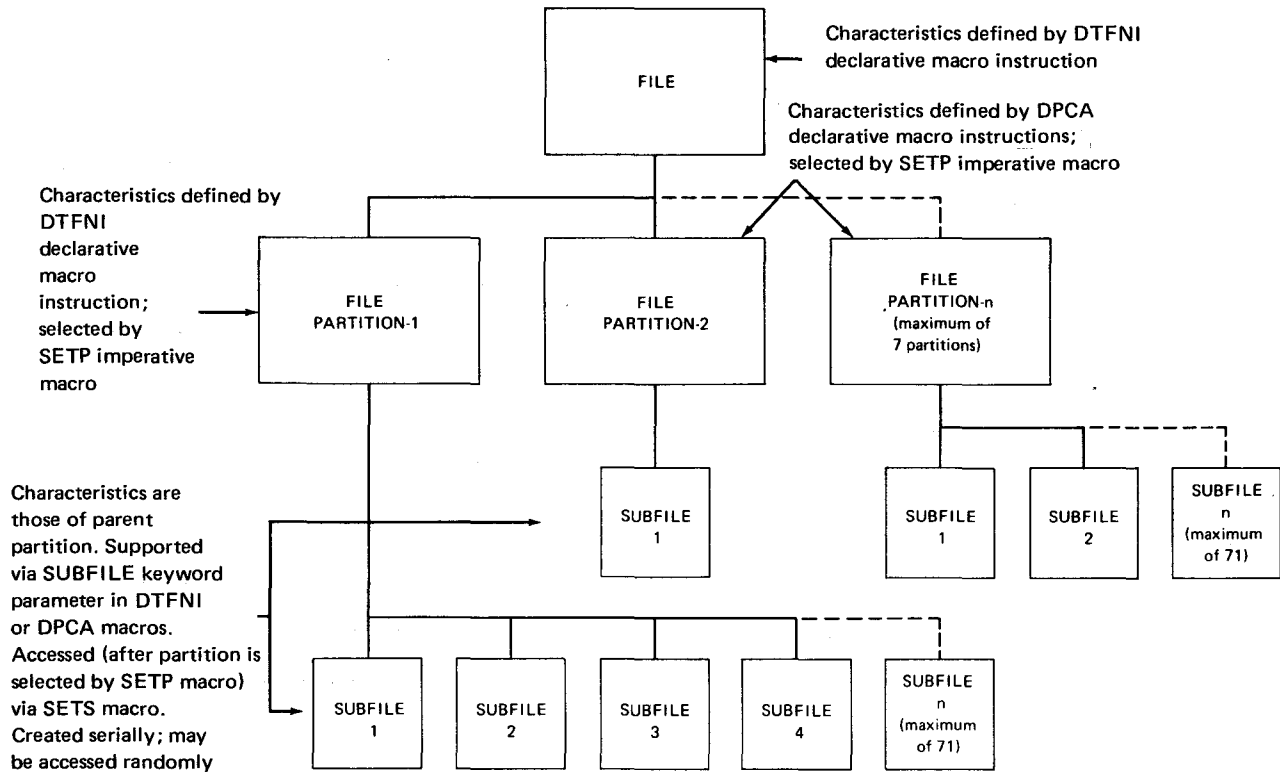
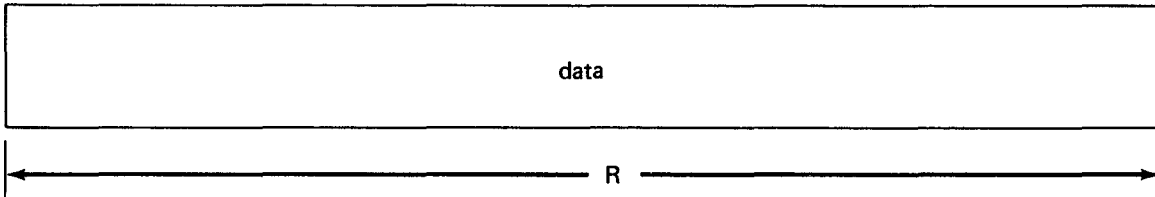
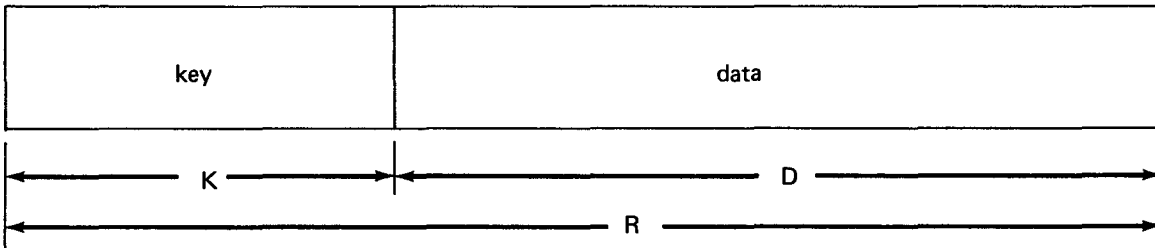


Figure G-7. Organization of a DTFNI Disc File into Partitions and Subfiles

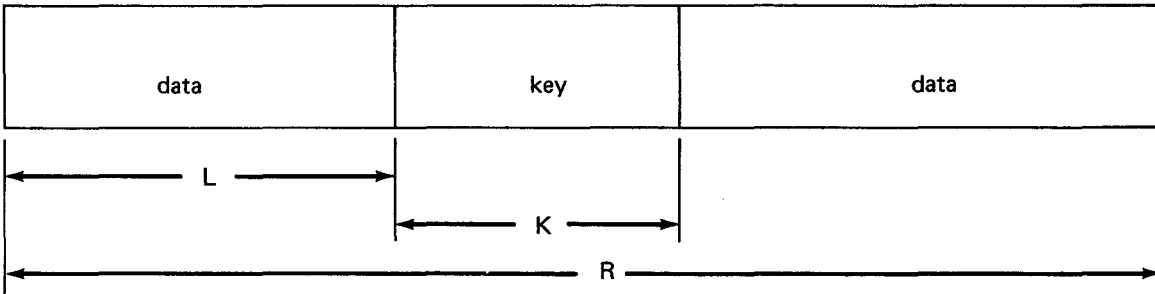
WITHOUT KEY



KEY AT HEAD OF RECORD



KEY INTERNAL TO RECORD

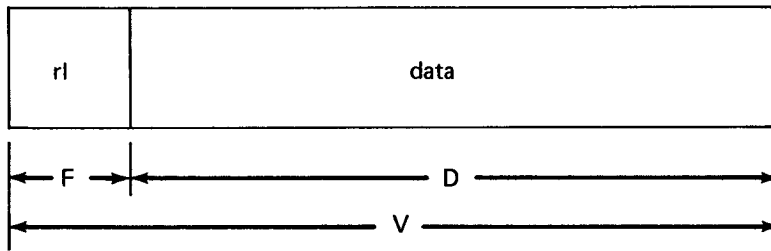


LEGEND:

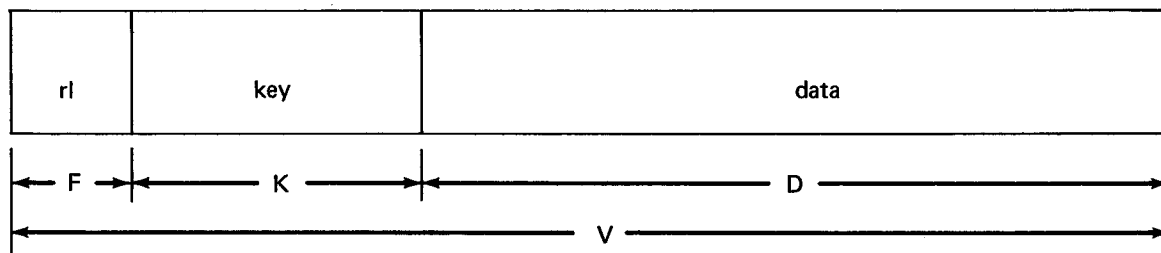
- K Record key. All keys in a keyed file must have the same length; each record in a keyed file must have one (and only one) unique key; and the starting location of the key must be the same in each record. User specifies the length of the key with the KEYLEN keyword parameter. Minimum length is 3 bytes; maximum is 253.
- L Key location. The starting location of the key must be the same in each record. User may specify the number of bytes of data preceding the key with the KEYLOC keyword parameter. If keyword is omitted, ISAM assumes the key starts in the first byte of a fixed record.
- D Data portion of the logical record
- R Length of logical fixed-length record (key plus data). User specifies this length, measured in bytes, with the RECSIZE keyword parameter; it must never exceed the value of data block size, less seven bytes.

Figure G-8. Fixed-Length ISAM and ASAM Records, with and without Keys

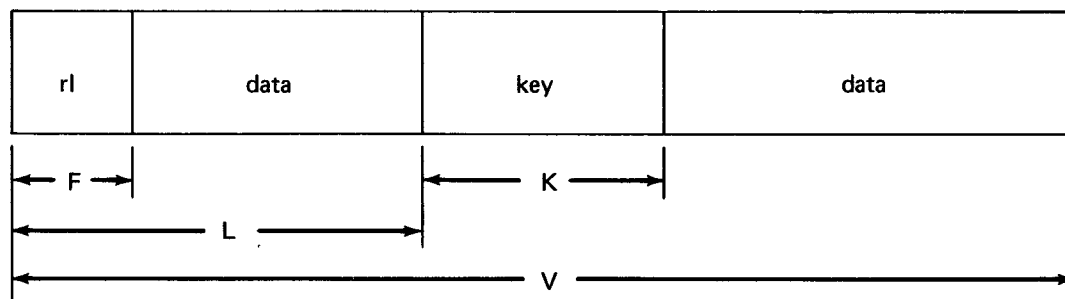
WITHOUT KEY



KEY AT HEAD OF RECORD



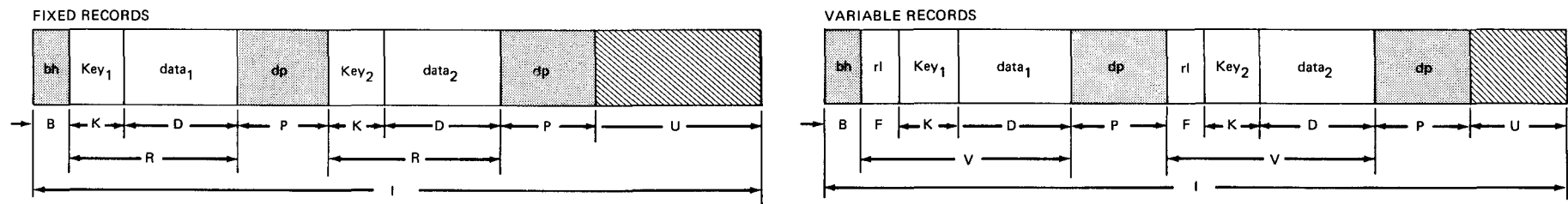
KEY INTERNAL TO RECORD



LEGEND:

- F 2-byte record length field. User inserts the length of each variable record into this field in binary; the length includes this 2-byte field and is equivalent to V in this figure.
- D Data portion of the logical record
- V Length of a variable record. Includes key plus data, plus two bytes of the record length field. User does not specify this length with a keyword parameter but places it in the leading two bytes of each variable record (in the field represented by F in this figure).
- K Record key. All keys in a keyed file must have the same length; each record in a keyed file must have one (and only one) unique key; and the starting location of the key must be the same in each record. Minimum key length is 3 bytes; the maximum is 253. User specifies the length of the key with the KEYLEN keyword parameter.
- L Key location. The starting location of the key must be the same in each record. User may specify the number of bytes that precede the key with the KEYLOC keyword parameter. If user omits the keyword, ISAM assumes the key begins in the third byte of a variable record.

Figure G-9. Variable-Length ISAM and ASAM Records, with and without Keys



LEGEND:

- B** Block header, written by data management in the buffer. This is always two bytes long and contains (in binary) the number of bytes in the data block which hold usable information. In the example shown, this figure would equal (I) minus (U); it includes the total space occupied by the records themselves and the 5-byte data pointers, one of which follows each of the records. Because data management appends the block header in the buffer, and it is placed in the buffer with the block when it is retrieved, user must allow for this 2-byte header in calculating the data block size, which he specifies with the BLKSIZE keyword parameter. However, it is not moved to the record work area (the address of which is specified with the WORK1 keyword parameter) where data management presents user records, one by one.
- K** Record key. This may be internal to the record instead of being located (as shown) at the head of the record. All keys in a file must have the same length; each record in a keyed file must have one and only one unique key; and the starting location of the key must be the same in each record of the file. User specifies the starting location of the key with the KEYLOC keyword parameter, and its length with the KEYLEN parameter; minimum key length is 3 bytes and maximum is 253. (When user presents a key that he wants ISAM to match by search, he loads it in an area of his program specified by the KEYARG keyword parameter.)
- D** Data portion of a logical record
- F** 2-byte record length field of a variable record
- P** Record pointer, a 5-byte divider which follows every record that data management writes into the data block. The record pointers are written by data management, but user must allow space for them in calculating I/O area length, which is not specified with the BLKSIZE keyword parameter. The BLKSIZE keyword specifies the size of data blocks. I/O area size, specified with BAL define storage (DS) instructions must equal or be greater than data block size. (See note.) The record pointer contains, in binary, the block number and byte position in that block of the next sequential logical record in the file, in the form *rrbb* where: *rrr* is the block number, relative to the data partition; and *bb* is the displacement, measured in bytes, of the record into that block. (A record preceded by 125 bytes will have a displacement value of 125.) The programmer will use this 5-byte "address" to retrieve records directly, rather than by key. (This address is always returned to user by data management after each record is loaded or added to the file; it is user's responsibility to access it and store it for later use, if he plans to use it. Data management places this 5-byte value in the field of the DTF called *filenameH*.)

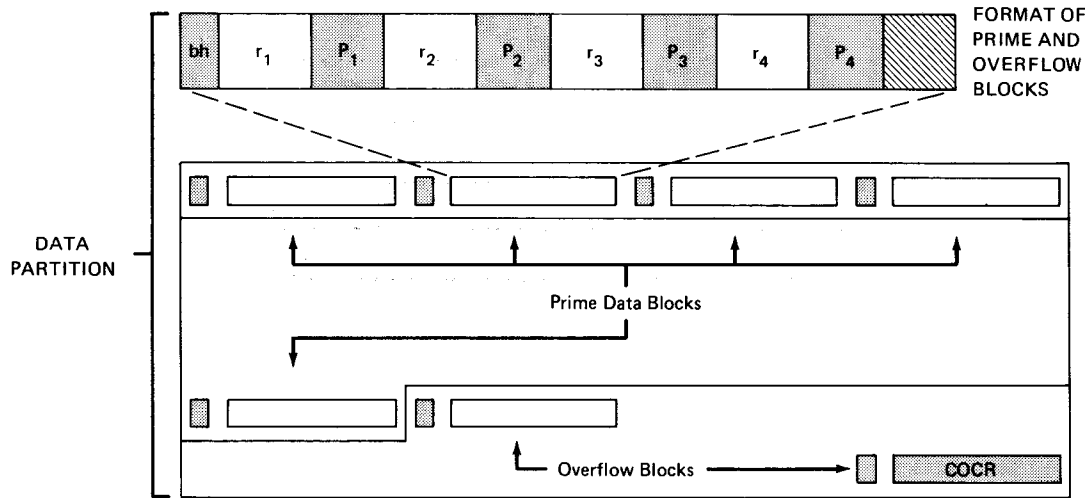
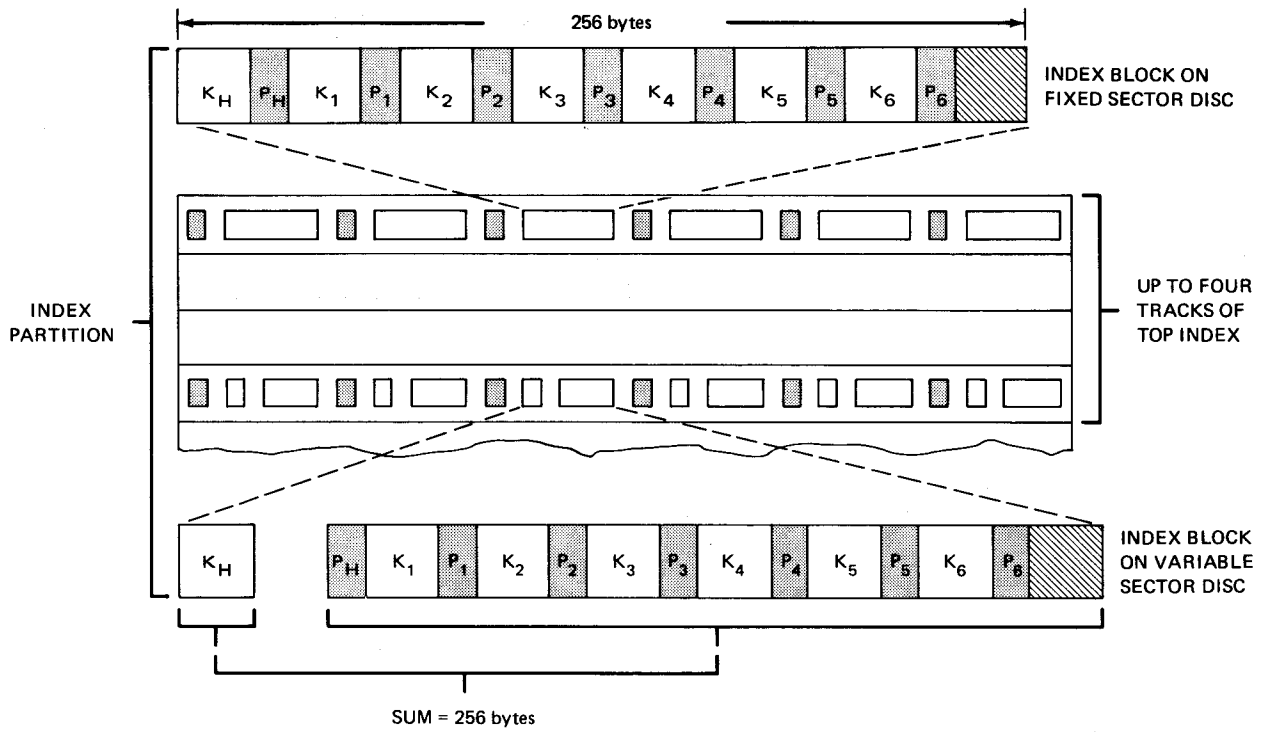
- U** Any unused space at the end of a data block. Usually unavoidable in data blocks containing variable-length records, this dead space may also occur in files of fixed records, if the sum of the logical record size (R) plus five bytes for the record pointer (P) does not divide evenly into (I), the data block size.
- R** Length of logical fixed-length record (key plus data). User specifies this, measured in bytes, with the RECSIZE keyword parameter. This must never exceed the value of the data block size, less seven bytes.
- V** Length of logical variable-length record (2-byte record-length field, plus key, plus data). User does not specify this length with a keyword parameter, but places it in the 2-byte record-length field at the head of each logical record. Like the length of a fixed record, it may never exceed the value of data block size, less seven bytes.
- I** Length of logical block which user specifies with the BLKSIZE keyword parameter. It includes the 2-byte header length, plus the record length of each logical record in the block, plus the 5-byte record pointer that must follow each record. It also includes such unused space as user has been unable to avoid.

Unused space
 Written by data management

NOTE:

If the file is to reside on an 8416 disc, the length of the I/O buffer supplied must be a multiple of 256 bytes, even though BLKSIZE is less.

Figure G—10. Layout of ISAM Data Blocks (Prime or Overflow) on Disc, Each Containing Two Logical Records



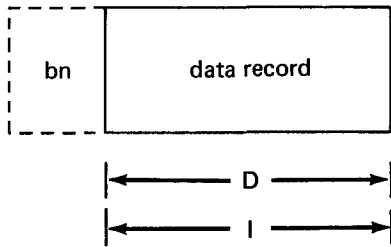
LEGEND:

- K Record key
- P Pointer
- bh Block header
- r Logical record
- Unused space at block end
- COCR Cylinder overflow control record; written by data management
- System-supplied pointers, headers, and count fields on disc

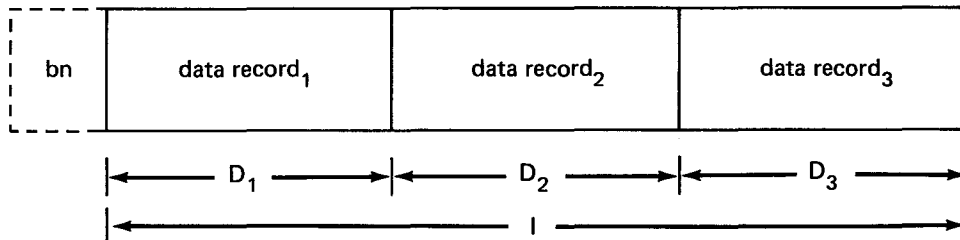
Figure G-11. OS/3 ISAM File Structure

EBCDIC

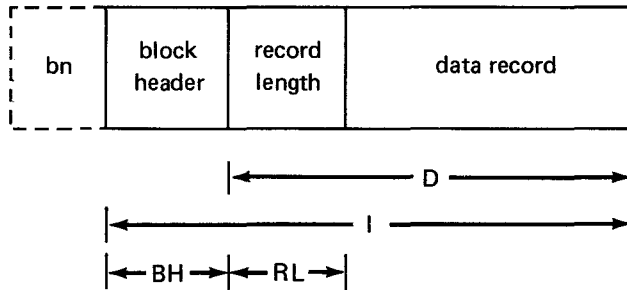
FIXED-LENGTH, UNBLOCKED RECORD



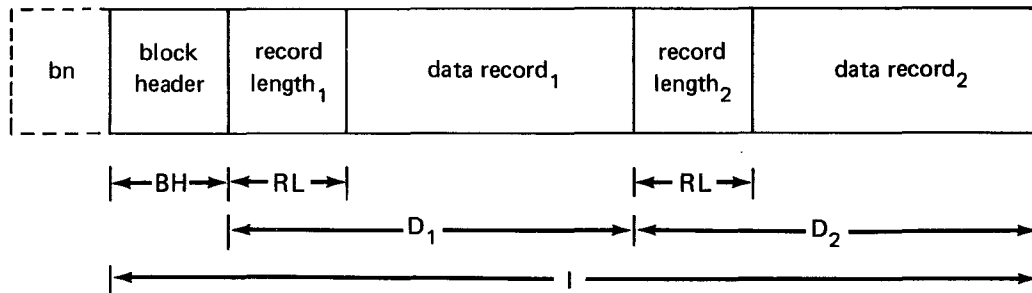
FIXED-LENGTH, BLOCKED RECORDS



VARIABLE-LENGTH, UNBLOCKED RECORD



VARIABLE-LENGTH, BLOCKED RECORDS



UNDEFINED RECORD FORMAT

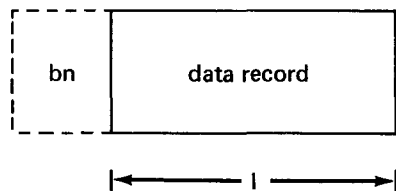
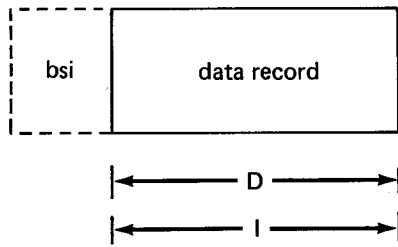


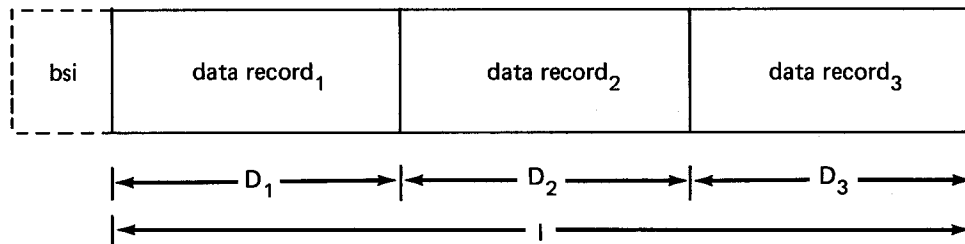
Figure G-12. Record and Block Formats for Magnetic Tape Files, ASCII and EBCDIC (Part 1 of 3)

ASCII

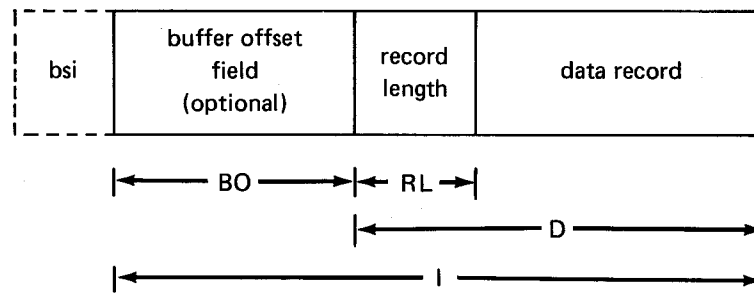
FIXED-LENGTH, UNBLOCKED RECORD (FORMAT F)



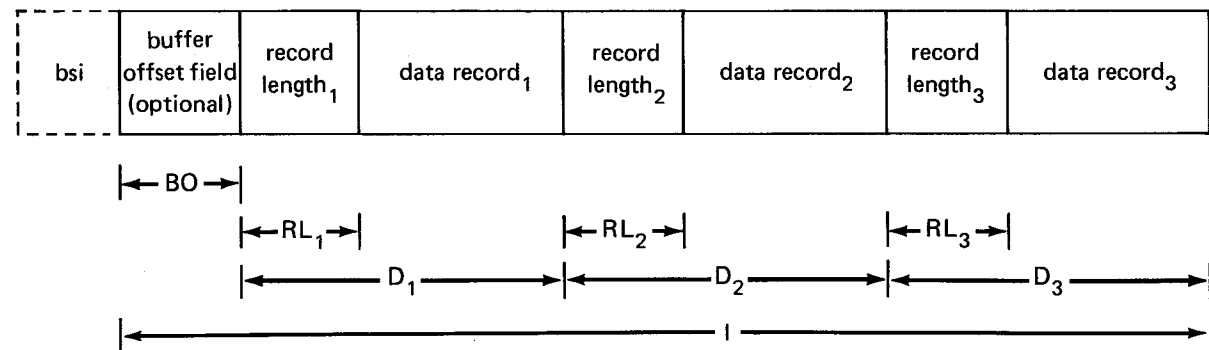
FIXED-LENGTH, BLOCKED RECORDS (FORMAT F)



VARIABLE-LENGTH, UNBLOCKED RECORD (FORMAT D)



VARIABLE-LENGTH, BLOCKED RECORDS (FORMAT D)



UNDEFINED RECORD FORMAT (FORMAT D)

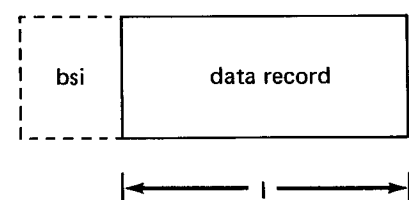


Figure G-12. Record and Block Formats for Magnetic Tape Files, ASCII and EBCDIC (Part 2 of 3)

LEGEND:

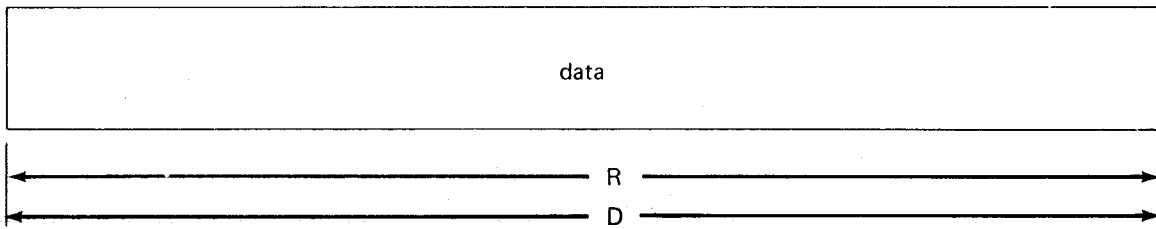
- D = Record length, measured in bytes. This measure is entered in the most significant two bytes of the 4-byte record length field; the two least significant bytes are reserved. For ASCII variable records, however, although the user enters this measure in the field in the same manner as in EBCDIC, data management converts this into a 4-byte ASCII numeric and writes it in the 4-byte record length field.
- I = Block length, measured in bytes. Minimum block length is 18 bytes. This measure is entered in the most significant two bytes of the 4-byte block header of EBCDIC variable-length records (blocked or unblocked); the two least significant bytes are reserved. When the buffer offset field of ASCII variable records is a 4-byte field tape SAM assumes that, in an input file, it contains the length of the block; during output, tape SAM creates the 4-byte field.
- RL = Record length field of variable length records, a 4-byte field in ASCII and EBCDIC records. Its own length is included in the measure inserted here. In EBCDIC records, record length is read and written in binary; in ASCII records, it is recorded on tape in the ASCII code, although the user presents it to data management in binary and processes it in binary (see Note 1).
- BH = Block header, a 4-byte field at the head of the block format in which all EBCDIC variable-length records, blocked or unblocked, appear on magnetic tape. Most significant two bytes contain the length of the block, which includes the length of the header itself; the two least significant bytes are reserved.
- BO = Buffer offset field, an optional block prefix that may be placed at the head of each block of ASCII variable records. Its content is recorded in ASCII; its length ranges from 0 to 99 bytes. When its length is four bytes, tape SAM assumes that this field contains the length of the block (which includes the length of this field itself). the index register specified by the IOREG keyword parameter points here, to the first byte of the record length field of variable-length records.
- bsi = Optional 1-byte block sequence indicator in ASCII numeric code. May not be created for output files. Data management accepts the block sequence indicator in ASCII input files, but does not process it.
- bn = Optional 3-byte block number in EBCDIC numeric code. May not be created for output files. Data management accepts the block number in EBCDIC input files, but does not process it.

NOTES:

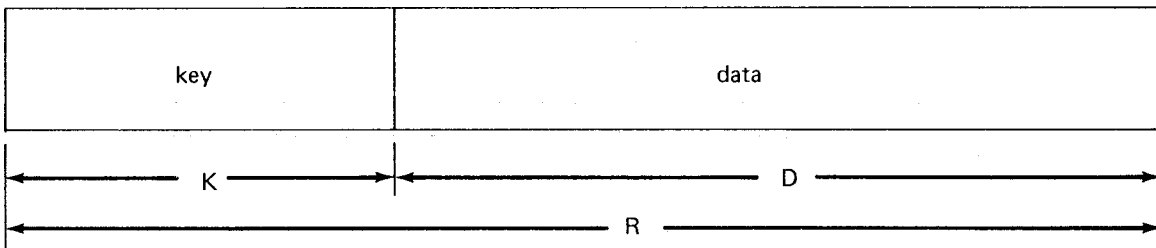
1. Although the *American National Standard X3.27-1969* also provides for a variable ASCII record with its record length specified in binary (the so-called "V-format" record), OS/3 magnetic tape SAM does not support this format.
2. Spanned records (those extending beyond one block) are not supported by OS/3 for either EBCDIC or ASCII magnetic tape files.
3. A single, unblocked variable-length EBCDIC record always exists on magnetic tape in the block format shown here, with a block header.

Figure G—12. Record and Block Formats for Magnetic Tape Files, ASCII and EBCDIC (Part 3 of 3)

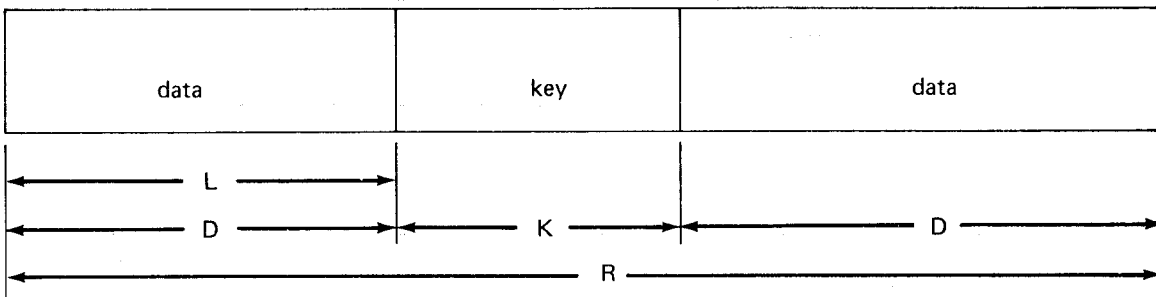
WITHOUT KEY



KEY AT HEAD OF RECORD



KEY INTERNAL TO RECORD

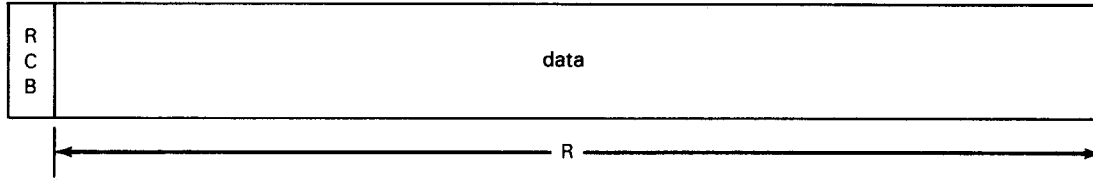


LEGEND:

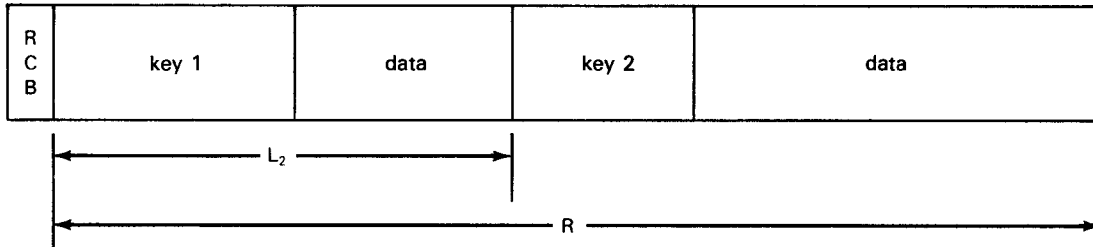
- K = Record key. All keys in a keyed file must have the same length; each record in a keyed file must have one unique key; and the starting location of the key must be the same in each record. You specify the length of the key, which may range from a minimum length of 3 bytes to a maximum of 80. (The maximum key length for RPG II records is 29 bytes.) No byte of any key may contain the hexadecimal value 'FF'.
- L = Key location. The starting location of the key must be the same in each record. You may specify the number of bytes of data preceding the key. If you default, IRAM assumes the key starts in the first byte of the record.
- D = Data portion of your logical record
- R = Length of logical record (key plus data). You specify this length, measured in bytes. All records in an IRAM file must have the same length.

Figure G-13. IRAM Data Record Formats

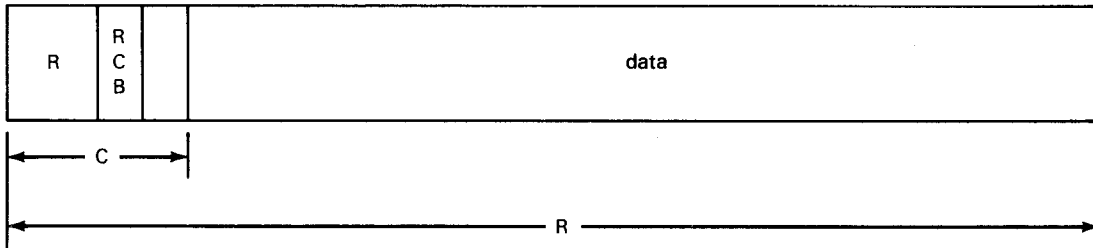
FIXED-LENGTH WITHOUT KEYS



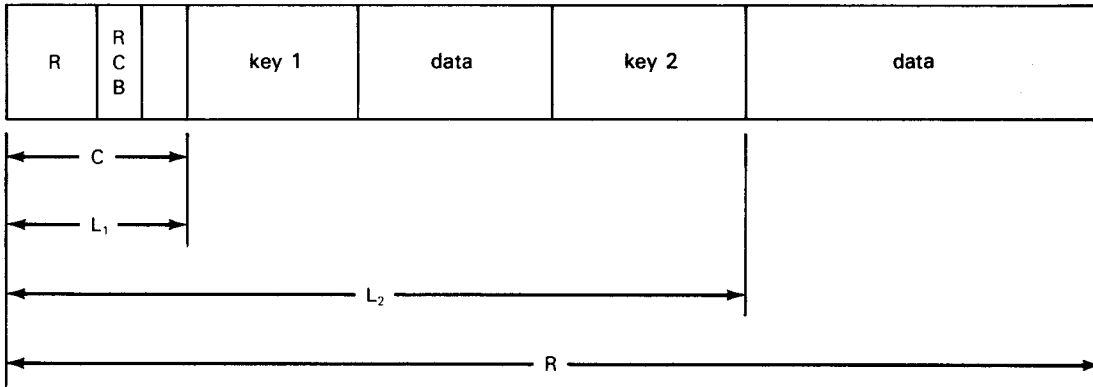
FIXED-LENGTH WITH KEYS



VARIABLE-LENGTH WITHOUT KEYS



VARIABLE-LENGTH WITH KEYS



LEGEND:

- RCB = Record control byte. Used to indicate that a record has been logically deleted from the file. For MIRAM fixed-length records, this byte is automatically placed at the beginning of each record. For variable-length records, the third byte of the control area (C) is used as the RCB (if RCB is requested).
- R = Length of the logical record (control area plus keys plus data). You specify this length as the number of bytes. For variable-length records, this value, expressed in binary, must be placed in the first two bytes of the control area (C).
- C = 4-byte control area for variable-length records. The first two bytes contain the logical record length (R) expressed in binary; the third byte is used as the RCB (if requested); the fourth byte is not used.
- L_n = The starting location of record key n ($n = 1$ through 5) of a MIRAM characteristic file data record when the key does not start in the first byte of the record (L_n) represents the number of bytes (control area plus data) that precede key n . The starting location of key n must be the same in each record. Key n must have the same length in each record (a minimum of 1 byte and a maximum of 80), and no byte may contain the hexadecimal value 'FF'.

Figure G-14. MIRAM Data Record Formats

Appendix H. Lace Factor Calculation



TWO-STEP CALCULATION

The first step is to calculate the sector time, which is the number of milliseconds each block of user data requires to pass under the disc head:

$$\text{sector time} = \frac{\text{user block size}}{256 \text{ bytes}} \times 0.535 \text{ ms}$$

The second step is to divide the calculated sector time into the time frame (a program-dependent figure approximating the number of milliseconds required to process records between accesses), add 1, and round to the next higher integer:

$$\text{lace factor} = \frac{\text{time frame}}{\text{sector time}} + 1 ; \text{round high}$$

Note that, when the time frame exceeds 21.4 ms, it should be divided by 21.4 and the remainder should be used as the time frame in the foregoing calculation.

Determining Time Frame

The programmer may estimate the average time frame his program requires to process records between accesses by using the GETIME and SETIME imperative macros of the OS/3 supervisor; or, arbitrarily specifying successively larger lace factors and noting the running time for his program with each factor, he may pinpoint the optimum factor for the program. The optimum is the factor giving the sharpest decrease in running time, followed by a quick increase when the next higher factor is used.

Programming Considerations

Once a file has been created with record interlace, the resulting physical arrangement of blocks is a permanent characteristic of the file. All programs referencing the file thereafter must specify the same lace factor used to create it. Erroneous specification will cause data management to reset the LACE specification to the value used when the file was created. Because the time frame for each program accessing a file is program dependent, the determination of the lace factor to use in creating a file for access by a number of programs is a matter of compromise.

Files defined by the DTFSD, DTFDA, and DTFNI declarative macros, as well as file partitions defined by the DPCA declarative macro, may be created with record interlace. However, most increase in throughput is realized from record interlace operations with sequentially processed files. If both the LACE and KEYLEN keywords are specified for a file, the lace factor specification is ignored.

Device-Independence

The preceding formulas apply to all disc subsystems supported by OS/3. Data management automatically adjusts the specified lace factor to the characteristics of the device on which the file resides.



Appendix I. Code Correspondences



GENERAL

This appendix presents a cross-reference table and figures useful for visualizing the correspondences among the following codes commonly used in data processing and in the SPERRY UNIVAC Operating System/3 (OS/3):

- Hollerith punched card code
- EBCDIC (Extended Binary Coded Decimal Interchange Code)
- ASCII (American National Standard Code for Information Interchange)
- Binary bit-pattern (bit-configuration) representation for an 8-bit code
- Hexadecimal representation
- Compressed code for punched cards
- Binary (image) mode for punched cards

EBCDIC/ASCII/HOLLERITH CORRESPONDENCE

Table I—1 is a cross-reference table depicting the correspondences among the Hollerith punched card code, ASCII, and EBCDIC. The table is arranged in the sorting (or collating) sequence of the binary bit-patterns which have been assigned to the codes, with 0000 0000 being the lowest value in the sequence and 1111 1111 the highest.

Note that the column headed *Decimal* uses decimal numbers to represent the positions of the codes and bit patterns in this sequence, but counts the position of the lowest value as position 0 rather than position 1. Thus, the position of the highest value bit-pattern 1111 1111 is represented in the decimal column by 255, whereas it is actually the 256th in the sequence. This scheme corresponds to the common convention for numbering bytes, in which the first byte of a group is byte 0, and is convenient when the user is constructing a 256-byte translation table.

Hollerith Punched Card Code

The standard Hollerith punched card code specifies 256 hole-patterns in 12-row punched cards. Hole-patterns are assigned to the 128 characters of ASCII and to 128 additional characters for use in 8-bit coded systems. These include the EBCDIC set. Note that no sorting sequence is implied by the Hollerith code itself.

EBCDIC

EBCDIC is an extension of Hollerith coding practices. It comprises 256 characters, each of which is represented by an 8-bit pattern. Table I—1 shows the EBCDIC graphic characters only; the EBCDIC control characters are not indicated.

ASCII

ASCII comprises 128 coded characters, each represented by an 8-bit pattern, and includes both control characters and graphic characters. Only the latter are shown in Table I—1; the shading in the graphic character column indicates where the 128-character ASCII code leaves off.

Table I-1. Cross-Reference Table: EBCDIC/ASCII/Hollerith (Part 1 of 5)

EBCDIC					ASCII	
Decimal	Hexa-decimal	Binary	EBCDIC Graphic Character	Hollerith Punched Card Code	ASCII Graphic Character	Hollerith Punched Card Code
0	00	0000 0000		12-0-9-8-1		12-0-9-8-1
1	01	0000 0001		12-9-1		12-9-1
2	02	0000 0010		12-9-2		12-9-2
3	03	0000 0011		12-9-3		12-9-3
4	04	0000 0100		12-9-4		9-7
5	05	0000 0101		12-9-5		0-9-8-5
6	06	0000 0110		12-9-6		0-9-8-6
7	07	0000 0111		12-9-7		0-9-8-7
8	08	0000 1000		12-9-8		11-9-6
9	09	0000 1001		12-9-8-1		12-9-5
10	0A	0000 1010		12-9-8-2		0-9-5
11	0B	0000 1011		12-9-8-3		12-9-8-3
12	0C	0000 1100		12-9-8-4		12-9-8-4
13	0D	0000 1101		12-9-8-5		12-9-8-5
14	0E	0000 1110		12-9-8-6		12-9-8-6
15	0F	0000 1111		12-9-8-7		12-9-8-7
16	10	0001 0000		12-11-9-8-1		12-11-9-8-1
17	11	0001 0001		11-9-1		11-9-1
18	12	0001 0010		11-9-2		11-9-2
19	13	0001 0011		11-9-3		11-9-3
20	14	0001 0100		11-9-4		9-8-4
21	15	0001 0101		11-9-5		9-8-5
22	16	0001 0110		11-9-6		9-2
23	17	0001 0111		11-9-7		0-9-6
24	18	0001 1000		11-9-8		11-9-8
25	19	0001 1001		11-9-8-1		11-9-8-1
26	1A	0001 1010		11-9-8-2		9-8-7
27	1B	0001 1011		11-9-8-3		0-9-7
28	1C	0001 1100		11-9-8-4		11-9-8-4
29	1D	0001 1101		11-9-8-5		11-9-8-5
30	1E	0001 1110		11-9-8-6		11-9-8-6
31	1F	0001 1111		11-9-8-7		11-9-8-7
32	20	0010 0000		11-0-9-8-1	SP	No punches
33	21	0010 0001		0-9-1	!	12-8-7
34	22	0010 0010		0-9-2	"	8-7
35	23	0010 0011		0-9-3	#	8-3
36	24	0010 0100		0-9-4	\$	11-8-3
37	25	0010 0101		0-9-5	%	0-8-4
38	26	0010 0110		0-9-6	&	12
39	27	0010 0111		0-9-7	'	8-5
40	28	0010 1000		0-9-8	(12-8-5
41	29	0010 1001		0-9-8-1)	11-8-5
42	2A	0010 1010		0-9-8-2	*	11-8-4
43	2B	0010 1011		0-9-8-3	+	12-8-6
44	2C	0010 1100		0-9-8-4	,	0-8-3
45	2D	0010 1101		0-9-8-5	-	11
46	2E	0010 1110		0-9-8-6	.	12-8-3
47	2F	0010 1111		0-9-8-7	/	0-1
48	30	0011 0000		12-11-0-9-8-1	0	0
49	31	0011 0001		9-1	1	1
50	32	0011 0010		9-2	2	2
51	33	0011 0011		9-3	3	3
52	34	0011 0100		9-4	4	4
53	35	0011 0101		9-5	5	5
54	36	0011 0110		9-6	6	6

Table I-1. Cross-Reference Table: EBCDIC/ASCII/Hollerith (Part 2 of 5)

EBCDIC					ASCII	
Decimal	Hexadecimal	Binary	EBCDIC Graphic Character	Hollerith Punched Card Code	ASCII Graphic Character	Hollerith Punched Card Code
55	37	0011 0111		9-7	7	7
56	38	0011 1000		9-8	8	8
57	39	0011 1001		9-8-1	9	9
58	3A	0011 1010		9-8-2	:	8-2
59	3B	0011 1011		9-8-3	;	11-8-6
60	3C	0011 1100		9-8-4	<	12-8-4
61	3D	0011 1101		9-8-5	=	8-6
62	3E	0011 1110		9-8-6	>	0-8-6
63	3F	0011 1111		9-8-7	?	0-8-7
64	40	0100 0000	SP	No punches	@	8-4
65	41	0100 0001		12-0-9-1	A	12-1
66	42	0100 0010		12-0-9-2	B	12-2
67	43	0100 0011		12-0-9-3	C	12-3
68	44	0100 0100		12-0-9-4	D	12-4
69	45	0100 0101		12-0-9-5	E	12-5
70	46	0100 0110		12-0-9-6	F	12-6
71	47	0100 0111		12-0-9-7	G	12-7
72	48	0100 1000		12-0-9-8	H	12-8
73	49	0100 1001		12-8-1	I	12-9
74	4A	0100 1010	[12-8-2	J	11-1
75	4B	0100 1011	.	12-8-3	K	11-2
76	4C	0100 1100	<	12-8-4	L	11-3
77	4D	0100 1101	(12-8-5	M	11-4
78	4E	0100 1110	+	12-8-6	N	11-5
79	4F	0100 1111	!	12-8-7	O	11-6
80	50	0101 0000	&	12	P	11-7
81	51	0101 0001		12-11-9-1	Q	11-8
82	52	0101 0010		12-11-9-2	R	11-9
83	53	0101 0011		12-11-9-3	S	0-2
84	54	0101 0100		12-11-9-4	T	0-3
85	55	0101 0101		12-11-9-5	U	0-4
86	56	0101 0110		12-11-9-6	V	0-5
87	57	0101 0111		12-11-9-7	W	0-6
88	58	0101 1000		12-11-9-8	X	0-7
89	59	0101 1001		11-8-1	Y	0-8
90	5A	0101 1010	}	11-8-2	Z	0-9
91	5B	0101 1011	\$	11-8-3	[12-8-2
92	5C	0101 1100	*	11-8-4	\	0-8-2
93	5D	0101 1101)	11-8-5]	11-8-2
94	5E	0101 1110	:	11-8-6	^	11-8-7
95	5F	0101 1111	^	11-8-7	-	0-8-5
96	60	0110 0000	~	11	`	8-1
97	61	0110 0001	/	0-1	a	12-0-1
98	62	0110 0010		11-0-9-2	b	12-0-2
99	63	0110 0011		11-0-9-3	c	12-0-3
100	64	0110 0100		11-0-9-4	d	12-0-4
101	65	0110 0101		11-0-9-5	e	12-0-5
102	66	0110 0110		11-0-9-6	f	12-0-6
103	67	0110 0111		11-0-9-7	g	12-0-7
104	68	0110 1000		11-0-9-8	h	12-0-8
105	69	0110 1001		0-8-1	i	12-0-9
106	6A	0110 1010	!	12-11	j	12-11-1
107	6B	0110 1011	,	0-8-3	k	12-11-2
108	6C	0110 1100	%	0-8-4	l	12-11-3
109	6D	0110 1101	—	0-8-5	m	12-11-4

Table I-1. Cross-Reference Table: EBCDIC/ASCII/Hollerith (Part 3 of 5)

EBCDIC					ASCII	
Decimal	Hexa-deci-mal	Binary	EBCDIC Graphic Character	Hollerith Punched Card Code	ASCII Graphic Character	Hollerith Punched Card Code
110	6E	0110 1110	>	0-8-6	n	12-11-5
111	6F	0110 1111	?	0-8-7	o	12-11-6
112	70	0111 0000		12-11-0	p	12-11-7
113	71	0111 0001		12-11-0-9-1	q	12-11-8
114	72	0111 0010		12-11-0-9-2	r	12-11-9
115	73	0111 0011		12-11-0-9-3	s	11-0-2
116	74	0111 0100		12-11-0-9-4	t	11-0-3
117	75	0111 0101		12-11-0-9-5	u	11-0-4
118	76	0111 0110		12-11-0-9-6	v	11-0-5
119	77	0111 0111		12-11-0-9-7	w	11-0-6
120	78	0111 1000	'	12-11-0-9-8	x	11-0-7
121	79	0111 1001		8-1	y	11-0-8
122	7A	0111 1010	:	8-2	z	11-0-9
123	7B	0111 1011	#	8-3	{	12-0
124	7C	0111 1100	@	8-4		12-11
125	7D	0111 1101	,	8-5	~	11-0
126	7E	0111 1110	=	8-6		11-0-1
127	7F	0111 1111	"	8-7		12-9-7
128	80	1000 0000		12-0-8-1		11-0-9-8-1
129	81	1000 0001	a	12-0-1		0-9-1
130	82	1000 0010	b	12-0-2		0-9-2
131	83	1000 0011	c	12-0-3		0-9-3
132	84	1000 0100	d	12-0-4		0-9-4
133	85	1000 0101	e	12-0-5		11-9-5
134	86	1000 0110	f	12-0-6		12-9-6
135	87	1000 0111	g	12-0-7		11-9-7
136	88	1000 1000	h	12-0-8		0-9-8
137	89	1000 1001	i	12-0-9		0-9-8-1
138	8A	1000 1010		12-0-8-2		0-9-8-2
139	8B	1000 1011		12-0-8-3		0-9-8-3
140	8C	1000 1100		12-0-8-4		0-9-8-4
141	8D	1000 1101		12-0-8-5		12-9-8-1
142	8E	1000 1110		12-0-8-6		12-9-8-2
143	8F	1000 1111		12-0-8-7		11-9-8-3
144	90	1001 0000		12-11-8-1		12-11-0-9-8-1
145	91	1001 0001	j	12-11-1		9-1
146	92	1001 0010	k	12-11-2		11-9-8-2
147	93	1001 0011	l	12-11-3		9-3
148	94	1001 0100	m	12-11-4		9-4
149	95	1001 0101	n	12-11-5		9-5
150	96	1001 0110	o	12-11-6		9-6
151	97	1001 0111	p	12-11-7		12-9-8
152	98	1001 1000	q	12-11-8		9-8
153	99	1001 1001	r	12-11-9		9-8-1
154	9A	1001 1010		12-11-8-2		9-8-2
155	9B	1001 1011		12-11-8-3		9-8-3
156	9C	1001 1100		12-11-8-4		12-9-4
157	9D	1001 1101		12-11-8-5		11-9-4
158	9E	1001 1110		12-11-8-6		9-8-6
159	9F	1001 1111		12-11-8-7		11-0-9-1

Table I-1. Cross-Reference Table: EBCDIC/ASCII/Hollerith (Part 4 of 5)

EBCDIC					ASCII	
Decimal	Hexa-deci-mal	Binary	EBCDIC Graphic Character	Hollerith Punched Card Code	ASCII Graphic Character	Hollerith Punched Card Code
160	A0	1010 0000		11-0-8-1		12-0-9-1
161	A1	1010 0001	~	11-0-1		12-0-9-2
162	A2	1010 0010	s	11-0-2		12-0-9-3
163	A3	1010 0011	t	11-0-3		12-0-9-4
164	A4	1010 0100	u	11-0-4		12-0-9-5
165	A5	1010 0101	v	11-0-5		12-0-9-6
166	A6	1010 0110	w	11-0-6		12-0-9-7
167	A7	1010 0111	x	11-0-7		12-0-9-8
168	A8	1010 1000	y	11-0-8		12-8-1
169	A9	1010 1001	z	11-0-9		12-11-9-1
170	AA	1010 1010		11-0-8-2		12-11-9-2
171	AB	1010 1011		11-0-8-3		12-11-9-3
172	AC	1010 1100		11-0-8-4		12-11-9-4
173	AD	1010 1101		11-0-8-5		12-11-9-5
174	AE	1010 1110		11-0-8-6		12-11-9-6
175	AF	1010 1111		11-0-8-7		12-11-9-7
176	B0	1011 0000		12-11-0-8-1		12-11-9-8
177	B1	1011 0001		12-11-0-1		11-8-1
178	B2	1011 0010		12-11-0-2		11-0-9-2
179	B3	1011 0011		12-11-0-3		11-0-9-3
180	B4	1011 0100		12-11-0-4		11-0-9-4
181	B5	1011 0101		12-11-0-5		11-0-9-5
182	B6	1011 0110		12-11-0-6		11-0-9-6
183	B7	1011 0111		12-11-0-7		11-0-9-7
184	B8	1011 1000		12-11-0-8		11-0-9-8
185	B9	1011 1001		12-11-0-9		0-8-1
186	BA	1011 1010		12-11-0-8-2		12-11-0
187	BB	1011 1011		12-11-0-8-3		12-11-0-9-1
188	BC	1011 1100		12-11-0-8-4		12-11-0-9-2
189	BD	1011 1101		12-11-0-8-5		12-11-0-9-3
190	BE	1011 1110		12-11-0-8-6		12-11-0-9-4
191	BF	1011 1111		12-11-0-8-7		12-11-0-9-5
192	C0	1100 0000	{	12-0		12-11-0-9-6
193	C1	1100 0001	A	12-1		12-11-0-9-7
194	C2	1100 0010	B	12-2		12-11-0-9-8
195	C3	1100 0011	C	12-3		12-0-8-1
196	C4	1100 0100	D	12-4		12-0-8-2
197	C5	1100 0101	E	12-5		12-0-8-3
198	C6	1100 0110	F	12-6		12-0-8-4
199	C7	1100 0111	G	12-7		12-0-8-5
200	C8	1100 1000	H	12-8		12-0-8-6
201	C9	1100 1001	I	12-9		12-0-8-7
202	CA	1100 1010		12-0-9-8-2		12-11-8-1
203	CB	1100 1011		12-0-9-8-3		12-11-8-2
204	CC	1100 1100		12-0-9-8-4		12-11-8-3
205	CD	1100 1101		12-0-9-8-5		12-11-8-4
206	CE	1100 1110		12-0-9-8-6		12-11-8-5
207	CF	1100 1111		12-0-9-8-7		12-11-8-6
208	D0	1101 0000	}	11-0		12-11-8-7
209	D1	1101 0001	J	11-1		11-0-8-1

Table I-1. Cross-Reference Table: EBCDIC/ASCII/Hollerith (Part 5 of 5)

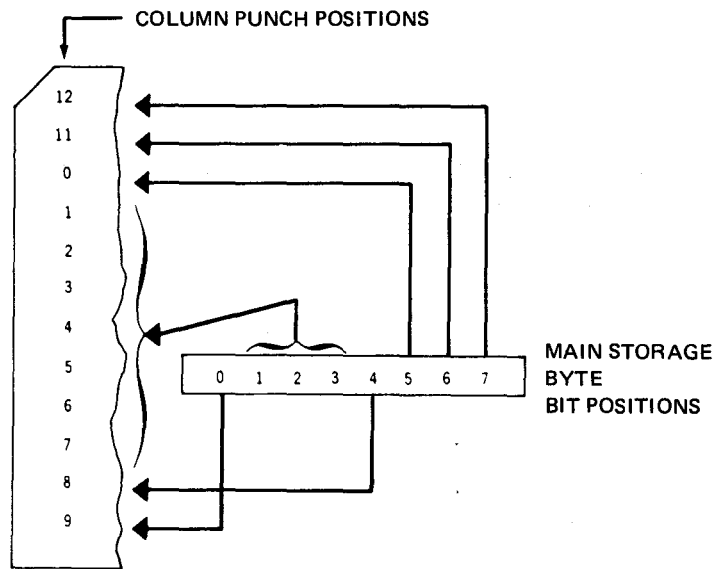
EBCDIC					ASCII	
Decimal	Hexa-decimal	Binary	EBCDIC Graphic Character	Hollerith Punched Card Code	ASCII Graphic Character	Hollerith Punched Card Code
210	D2	1101 0010	K	11-2		11-0-8-2
211	D3	1101 0011	L	11-3		11-0-8-3
212	D4	1101 0100	M	11-4		11-0-8-4
213	D5	1101 0101	N	11-5		11-0-8-5
214	D6	1101 0110	O	11-6		11-0-8-6
215	D7	1101 0111	P	11-7		11-0-8-7
216	D8	1101 1000	Q	11-8		12-11-0-8-1
217	D9	1101 1001	R	11-9		12-11-0-1
218	DA	1101 1010		12-11-9-8-2		12-11-0-2
219	DB	1101 1011		12-11-9-8-3		12-11-0-3
220	DC	1101 1100		12-11-9-8-4		12-11-0-4
221	DD	1101 1101		12-11-9-8-5		12-11-0-5
222	DE	1101 1110		12-11-9-8-6		12-11-0-6
223	DF	1101 1111		12-11-9-8-7		12-11-0-7
224	E0	1110 0000	\	0-8-2		12-11-0-8
225	E1	1110 0001		11-0-9-1		12-11-0-9
226	E2	1110 0010	S	0-2		12-11-0-8-2
227	E3	1110 0011	T	0-3		12-11-0-8-3
228	E4	1110 0100	U	0-4		12-11-0-8-4
229	E5	1110 0101	V	0-5		12-11-0-8-5
230	E6	1110 0110	W	0-6		12-11-0-8-6
231	E7	1110 0111	X	0-7		12-11-0-8-7
232	E8	1110 1000	Y	0-8		12-0-9-8-2
233	E9	1110 1001	Z	0-9		12-0-9-8-3
234	EA	1110 1010		11-0-9-8-2		12-0-9-8-4
235	EB	1110 1011		11-0-9-8-3		12-0-9-8-5
236	EC	1110 1100		11-0-9-8-4		12-0-9-8-6
237	ED	1110 1101		11-0-9-8-5		12-0-9-8-7
238	EE	1110 1110		11-0-9-8-6		12-11-9-8-2
239	EF	1110 1111		11-0-9-8-7		12-11-9-8-3
240	F0	1111 0000	0	0		12-11-9-8-4
241	F1	1111 0001	1	1		12-11-9-8-5
242	F2	1111 0010	2	2		12-11-9-8-6
243	F3	1111 0011	3	3		12-11-9-8-7
244	F4	1111 0100	4	4		11-0-9-8-2
245	F5	1111 0101	5	5		11-0-9-8-3
246	F6	1111 0110	6	6		11-0-9-8-4
247	F7	1111 0111	7	7		11-0-9-8-5
248	F8	1111 1000	8	8		11-0-9-8-6
249	F9	1111 1001	9	9		11-0-9-8-7
250	FA	1111 1010		12-11-0-9-8-2		12-11-0-9-8-2
251	FB	1111 1011		12-11-0-9-8-3		12-11-0-9-8-3
252	FC	1111 1100		12-11-0-9-8-4		12-11-0-9-8-4
253	FD	1111 1101		12-11-0-9-8-5		12-11-0-9-8-5
254	FE	1111 1110		12-11-0-9-8-6		12-11-0-9-8-6
255	FF	1111 1111		12-11-0-9-8-7		12-11-0-9-8-7

OTHER CARD CODES

Two other punched card coding systems can be handled with OS/3 data management and all card reader and card punch subsystems in the SPERRY UNIVAC 90/30 System: the compressed code and the column binary (or image) code.

Compressed Card Code

Figure I—1 indicates the construction of the compressed card code; each card column is represented by an 8-bit pattern in one byte of main storage



NOTE:

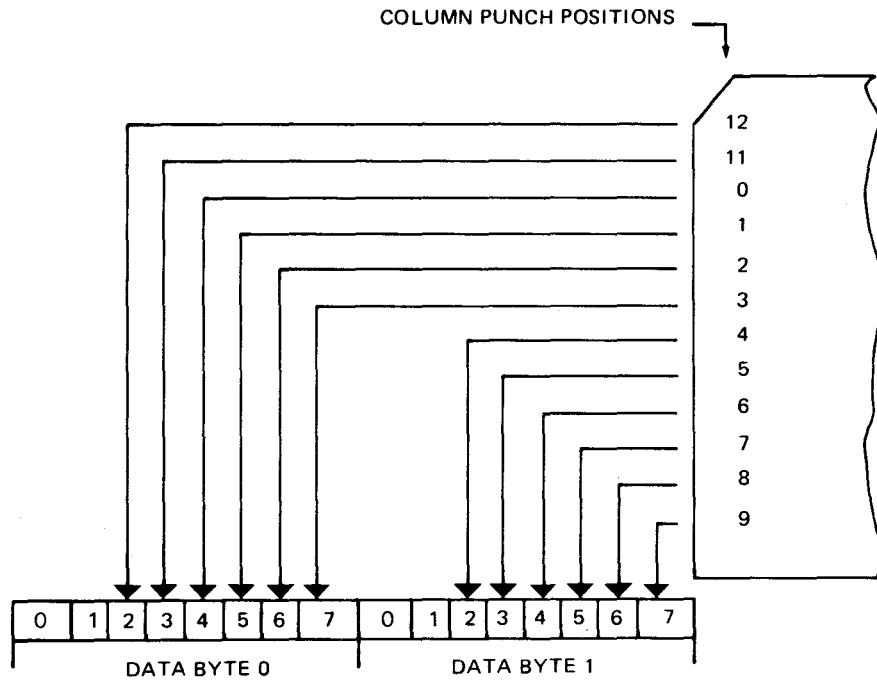
Punch positions 1 through 7 are indicated in bits 1 through 3, according to the following table:

PUNCH ROWS 1 THROUGH 7	BITS 123
NONE	000
1	011
2	101
3	001
4	010
5	100
6	111
7	110

Figure I—1. Compressed Card Code

Column Binary (Image) Code

Figure I—2 indicates the construction of this code. Note that each card column requires two bytes of main storage; an I/O area of 160 bytes is required for an 80-column card.



NOTE:

Bits 0 and 1 are cleared to 0's on an image read.

Figure I—2. Column Binary (Image) Card Code

DATA CONVERSION

In OS/3 data management, there are five ways in which data, held in main storage in 8-bit bytes, may be converted into hole-patterns in punched cards, and vice versa:

- Standard mode (EBCDIC)
- Standard mode (ASCII)
- Compressed code mode
- Binary (image) mode
- Translate mode for reading or punching.

In EBCDIC standard mode (MODE=STD), data in main storage in EBCDIC is punched into cards in the Hollerith punched card code. Cards are read in Hollerith, and the data is stored in EBCDIC.

In ASCII standard mode (MODE=STD and ASCII=YES), data management translates data stored in 8-bit ASCII in main storage to EBCDIC, and then the cards are punched in Hollerith. The reverse process is used when cards are read, unless a hardware ASCII translate feature is available on the card reader, when data management omits the EBCDIC-to-ASCII translation.

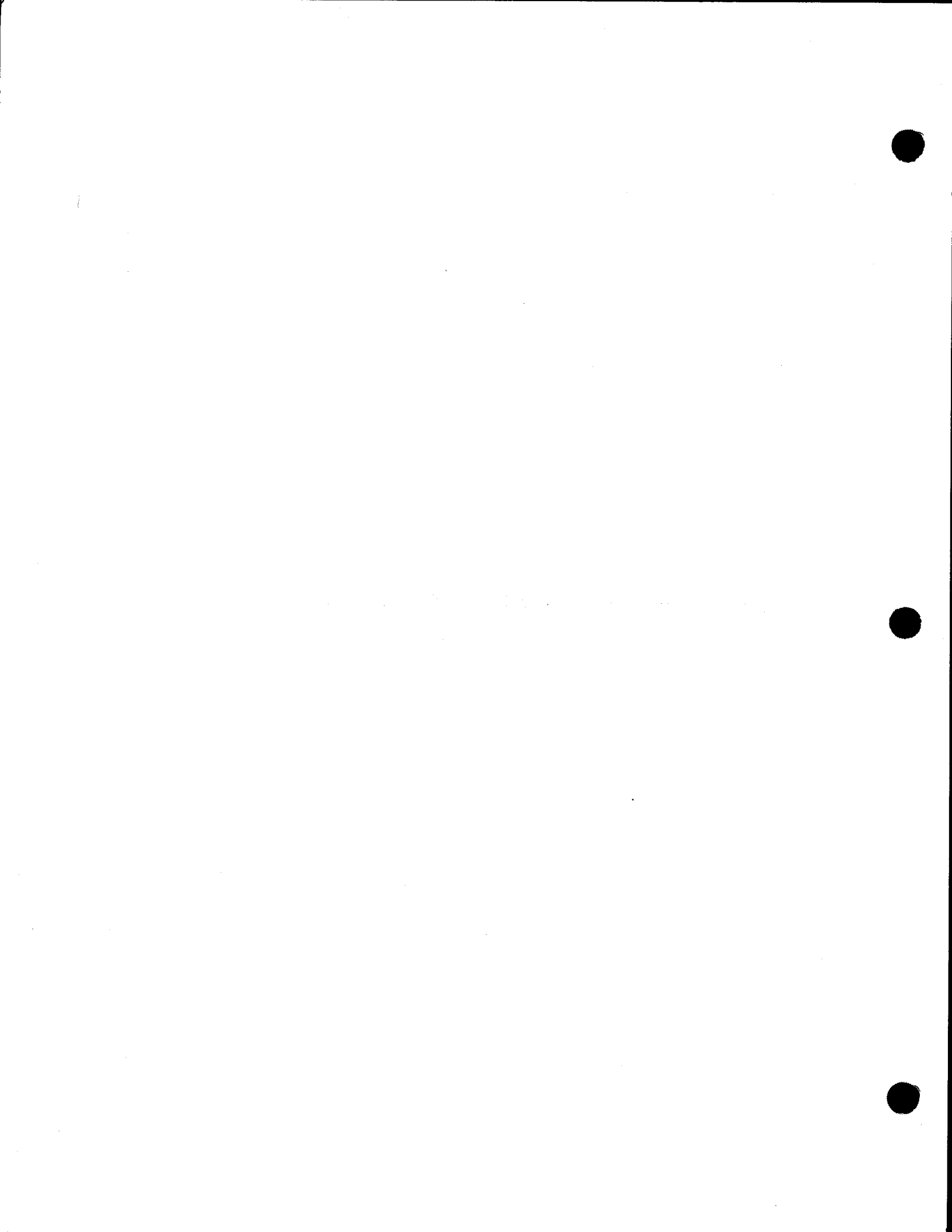
In the compressed code mode (MODE=CC), an 8-bit data byte is converted by data management into a single-column hole-pattern (Figure I-1).

In the binary or image mode (MODE=BINARY), there is a one-to-one correspondence between 12 data bits in main storage (data is stored in the least significant six bits of two 8-bit bytes) and the 12 possible row punches in a card column (Figure I-2).

In the translate mode (MODE=TRANS), the user makes his own assignment of 8-bit patterns to the 256 hole-patterns listed in Table I-1, in the order these are shown in the table.



**Appendix J. Magnetic Tape Formats
and File Conventions**



TAPE VOLUME AND FILE ORGANIZATION

Using the ASCII, FILABL, and TPMARK keyword parameters in his DTFMT declarative macro instruction, the BAL programmer indicates to data management whether his file is an ASCII or EBCDIC file; whether it contains standard, nonstandard, or no labels; and whether, in a nonstandard or unlabeled file, data management is to write a tape mark preceding the blocks of data. These specifications reflect the four types of reel or volume organization used with OS/3 tape SAM:

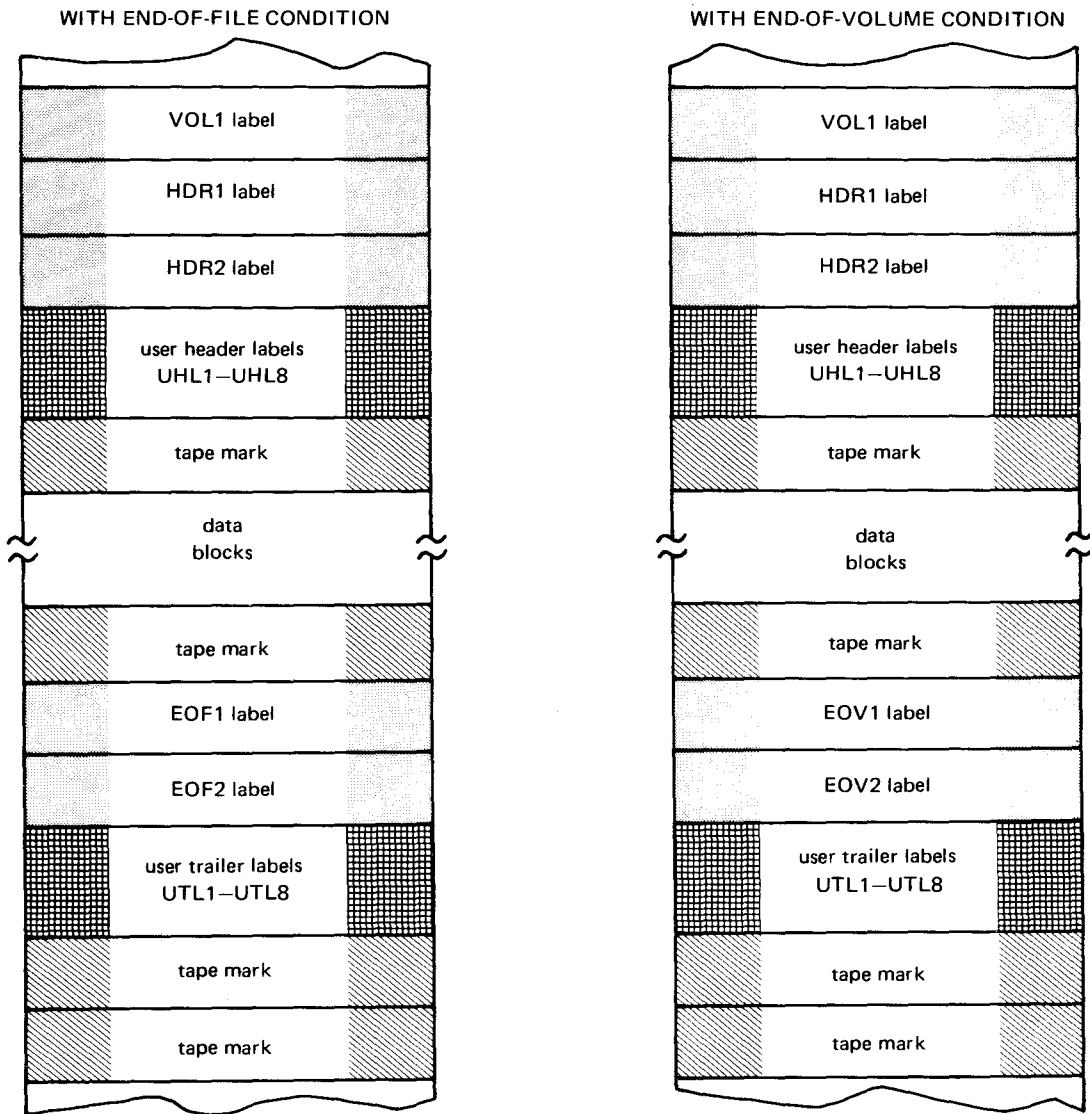
- EBCDIC:
 - Standard labeled
 - Nonstandard labeled
 - Unlabeled
- ASCII
 - Standard labeled

The following paragraphs and figures describe the organization of files and reels with respect to these four conventions. Refer to Appendix E for descriptions of the standard system labels for tape files and the standard user labels, which are optional.

EBCDIC Standard Volume Organization

A standard volume has system standard labels and required tape marks; it may also, at the user's option, contain standard user header and trailer labels (UHL and UTL). All standard tape labels are written in blocks of 80 bytes. A standard volume may not contain nonstandard labels. Data management assumes that the labels appear on tape in the order shown in Figures J-1, J-2, and J-3, which illustrate the reel organization for standard-labeled EBCDIC volumes with an *end-of-file* (EOF) and an *end-of-volume* (EOV) condition.

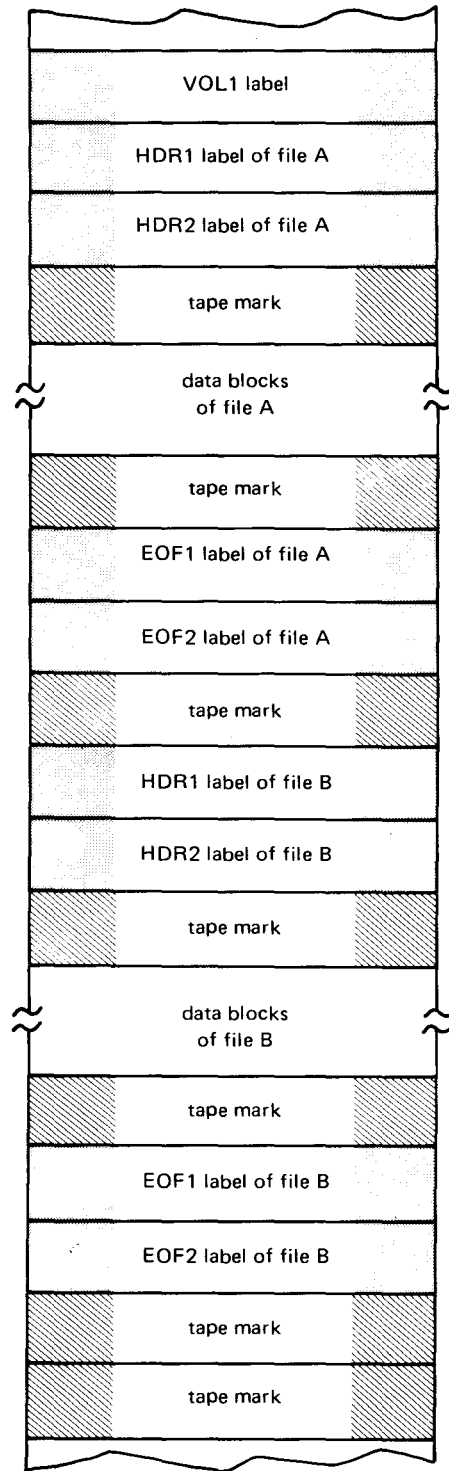
A standard-labeled EBCDIC volume processed by OS/3 data management ends in either an end-of-file or an end-of-volume label group, followed by two tape marks. The second tape mark indicates that no valid information follows. No provision is made for creating additional volume, header, or EOF/EOV labels on output files; if they exist on input files, data management bypasses them.



LEGEND:

- Content supplied by user
- Required and generated by data management
- Generated by data management; user supplies content for certain fields.
- Generated by user's LABADDR routine, at his option; content is at user's option except for content of 4-byte label ID fields. User is limited to eight UHL and eight UTL. Refer to Appendix D.

Figure J-1. Reel Organization for EBCDIC Standard-Labeled Tape Volumes, Containing a Single File



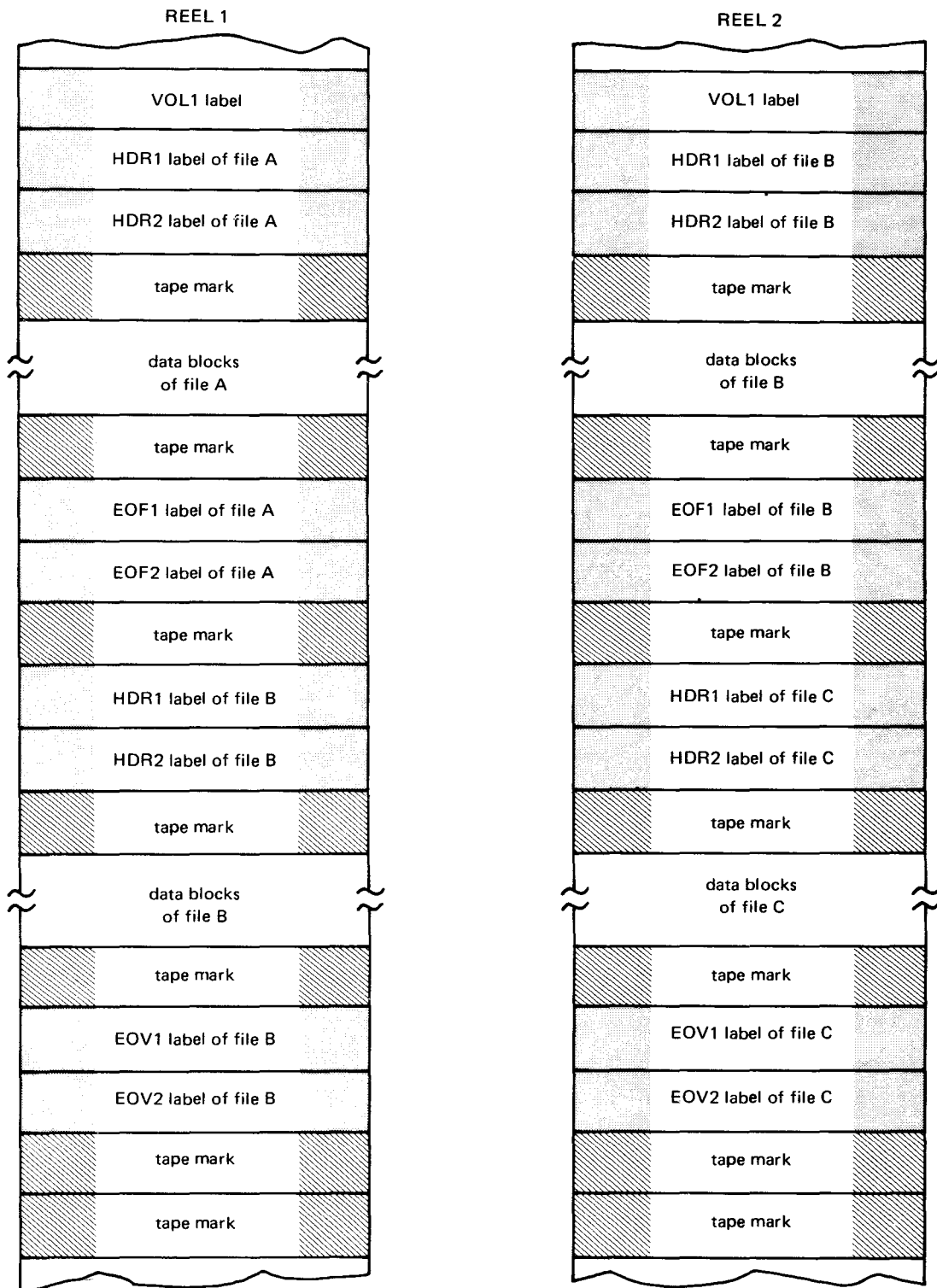
LEGEND:

- Content supplied by user
- Required and generated by data management
- Generated by data management; user supplies content for certain fields.

NOTE:

Assume that file B completes on this volume.

Figure J-2. Reel Organization for EBCDIC Standard-Labeled Tape Volume:
Multifile Volume with End-of-File Condition



LEGEND:

- Content supplied by user
- Required and generated by data management
- Generated by data management; user supplies content for certain fields.

NOTE:

Assume that file C is not completed on reel 2, but carries over (like file B) onto another volume. If file C were completed on reel 2, its EOV1 and EOV2 labels shown here would be replaced with EOF1 and EOF2 labels.

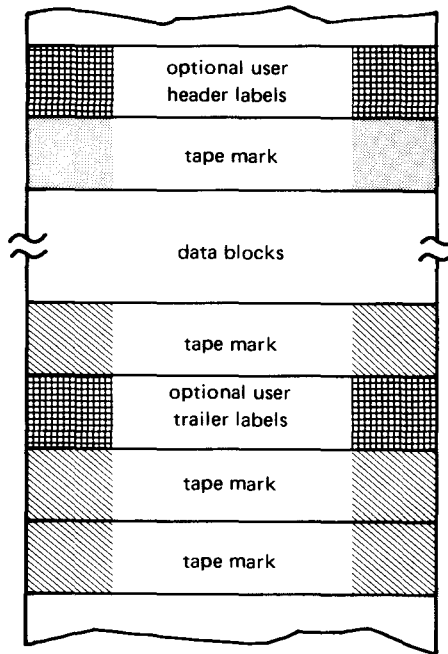
Figure J-3. Reel Organization for EBCDIC Standard-Labeled Tape Volumes: Multifile Volumes with End-of-Volume Condition

EBCDIC Nonstandard Volume Organization

A nonstandard volume is any volume containing only nonstandard labels; certain tape marks are required. Nonstandard user header and trailer labels (UHL and UTL) are optional; these may be of any format, length, or number because they are handled by the user's label routine. (Refer to the LABADDR and TPMARK keyword parameters of the DTFMT declarative macro). Figures J-4 and J-5 illustrate the reel organization for EBCDIC nonstandard volumes; a specification of a nonstandard volume for an ASCII file is invalid. (Refer to the ASCII and FILABL keyword parameters.)

The address of the user's label-handling routine to process nonstandard labels is usually specified, in which event the tape mark following the UHLs may be omitted; it is required only if label checking is to be omitted or a read-backward operation is specified. (Refer to the READ keyword parameter.) If nonstandard labels appear on an input file but are not to be checked when the file is read, the user omits specifying the address of his label-handling routine, but the tape mark must be present.

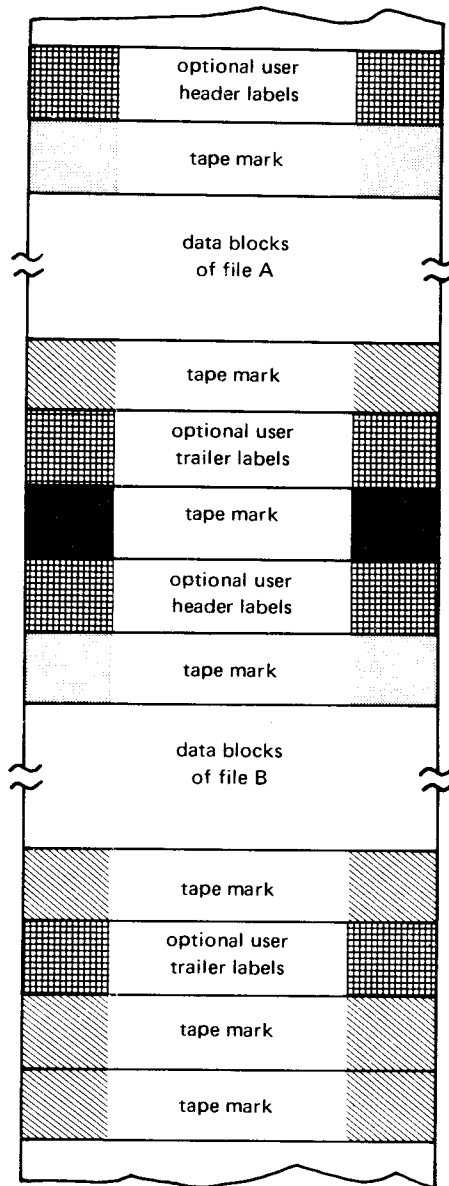
The tape mark following the data blocks is required and is written by data management, which also writes two required tape marks after the UTLs, if they are present. If the optional UTLs are not present, data management writes only one additional tape mark after the one following the data blocks. This second tape mark is always present when this file is the only file or the last file on the reel; it is overwritten by the next file to be written on a multifile volume.



LEGEND:

- Content supplied by user
- Required and written by data management; only two tape marks follow data blocks if UTLs are not present.
- Generated by data management unless user specifies TPMARK=NO; required only if label checking is omitted or user specifies READ=BACK
- Presence, content, format, and number entirely at user's option

Figure J-4. Reel Organization for EBCDIC Nonstandard Volume Containing a Single File



LEGEND:






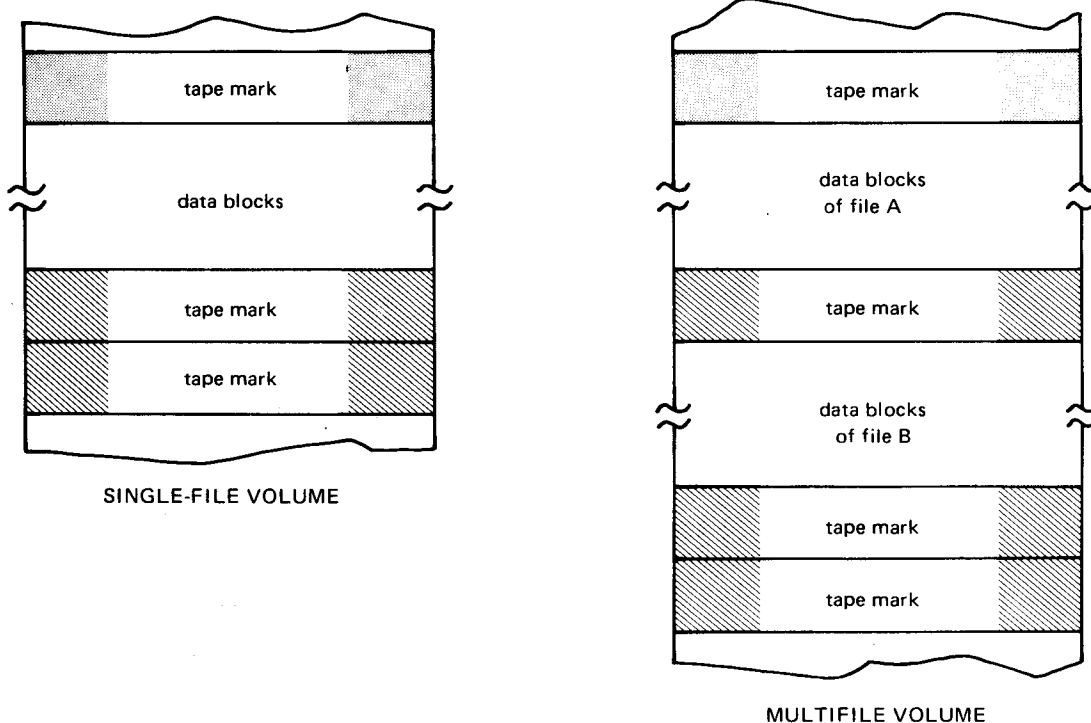
-  Content supplied by user
-  Required and generated by data management; only two tape marks follow data blocks of last file on volume if UTLs are not present
-  Generated by data management unless user specifies TP MARK=NO; required only if label checking is omitted or user specifies READ=BACK
-  Presence, content, format, and number entirely at user's option
-  Always present; written by data management

Figure J—5. Reel Organization for EBCDIC Nonstandard Multifile Volume

EBCDIC Unlabeled Volume Organization

OS/3 tape data management can also process unlabeled tape volumes. The user specifies `FILABL=NO`, or omits this keyword parameter, to indicate an unlabeled volume or file. A tape mark is expected or written by data management preceding the data blocks unless the user has specified `TPMARK=NO` in the `DTFMT` declarative macro.

Figure J—6 illustrates the reel organization for unlabeled EBCDIC volumes. The tape mark following the data blocks is required on both single-file and multifile volumes and is supplied by data management on output operations. A second tape mark is always written by data management following the last or only file on each volume and is overwritten by the next file to be written on a multifile volume.



LEGEND:

- Content supplied by user
- Required and written by data management; two tape marks follow data blocks of last file on volume.
- Generated by data management unless user specifies `TPMARK=NO`; required only when user specifies `READ=BACK`

Figure J—6. Reel Organization for Unlabeled EBCDIC Volumes

ASCII Standard Volume Organizations

The *American National Standard Magnetic Tape Labels for Information Interchange, X3.27—1969*, provides for the following *file sets* (collections of one or more related files recorded on one or more volumes):

- Single file, single volume
- Single file, multivolume
- Multifile, single volume
- Multifile, multivolume.

The standard ASCII reel organizations, or label configurations, used in OS/3 are depicted in Figures J—7 through J—10. Note that the standard does not provide for unlabeled ASCII volumes, and also that the DTF specification FILABL=NSTD is not valid for an ASCII file.

Refer to Appendix F for the format and content of the ASCII standard labels.

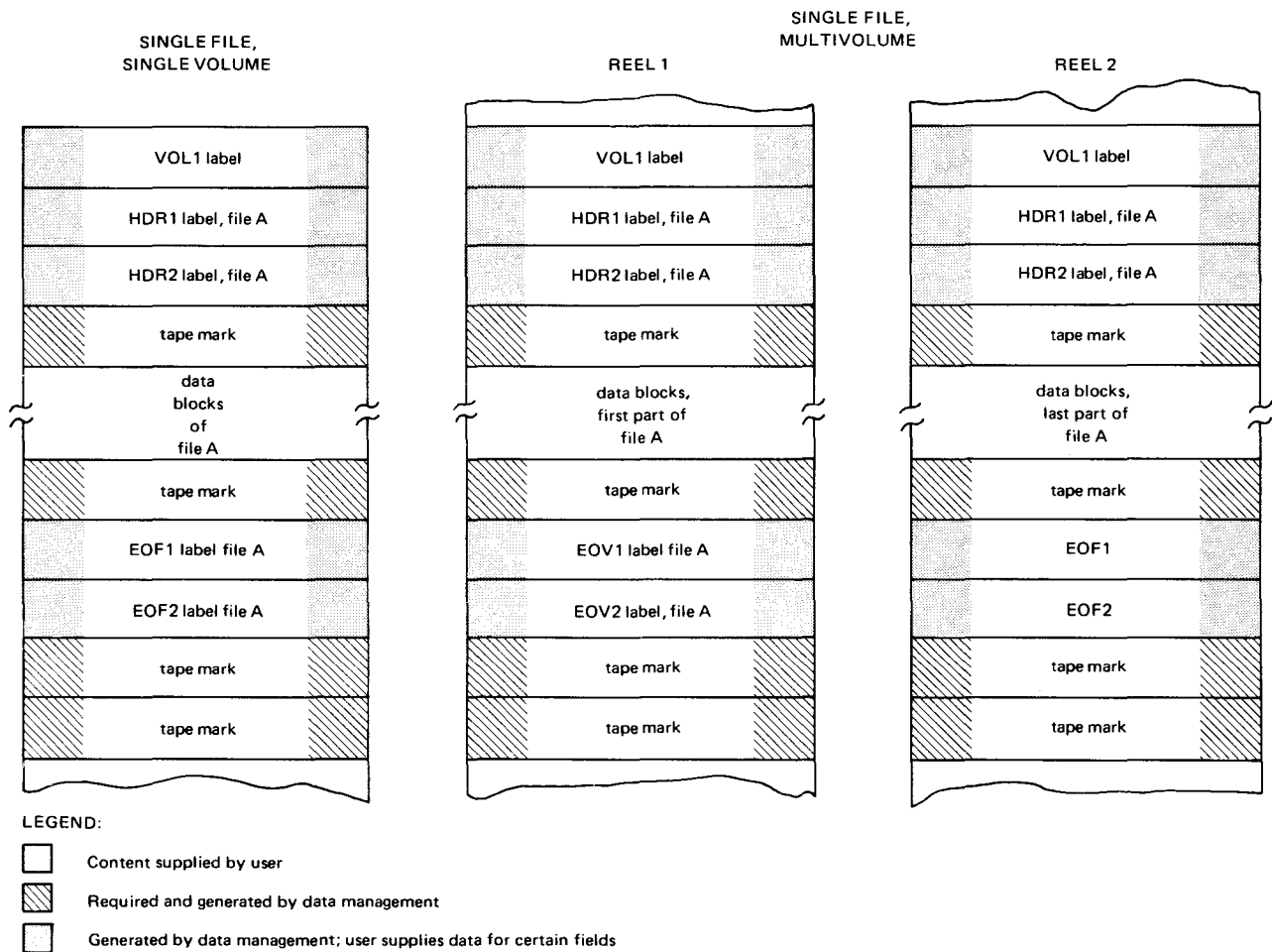
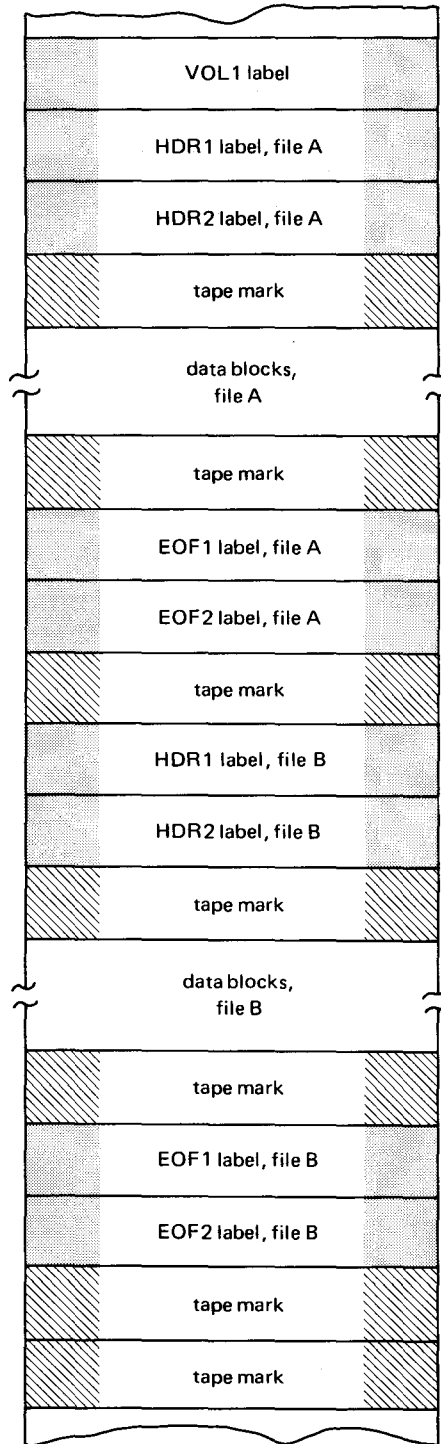


Figure J—7. Label Configuration, ASCII Single-File, Single-Volume and Multivolume Sets

MULTIFILE, SINGLE VOLUME



LEGEND:




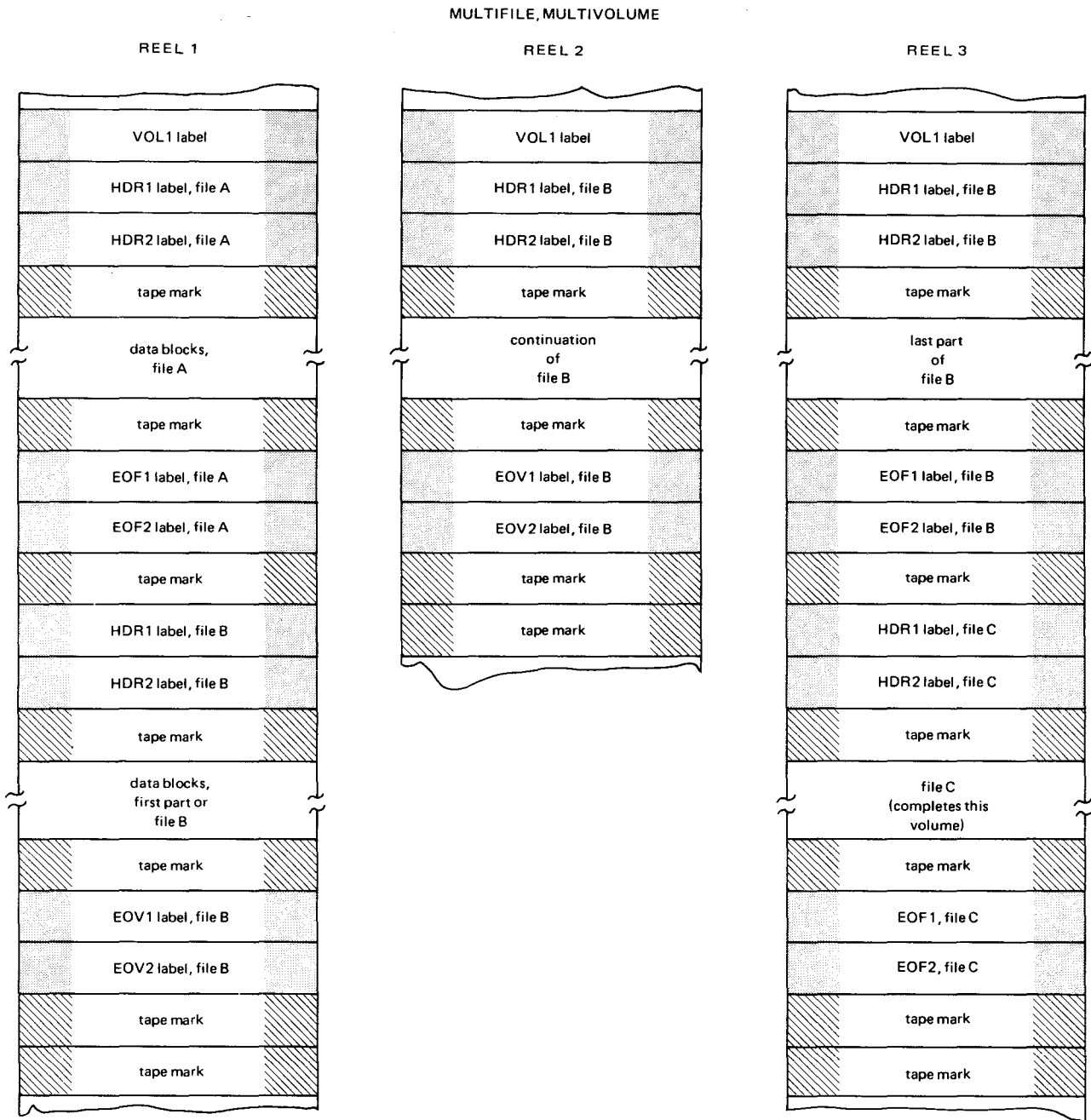
-  Content supplied by user
-  Required and generated by data management
-  Generated by data management; user supplies data for certain fields

Figure J-8. Label Configuration, ASCII Multifile, Single-Volume Set



LEGEND:

- Content supplied by user
- Required and generated by data management
- Generated by data management; user supplies data for certain fields

Figure J-9. Label Configuration, ASCII Multifile, Multivolume Set

END-OF-FILE AND END-OF-VOLUME COINCIDENCE

American National Standard X3.27-1969 provides that, whenever a volume ends within a file, the last block of the file in that volume is followed by an end-of-volume label (EOV1); it also allows the second end-of-volume (EOV2) label, which is standard in OS/3. A single tape mark precedes, and two tape marks follow these; further, no *file set* may be terminated by end-of-volume labels.

Whenever end-of-volume and end-of-file coincide, however, the standard provides that the labeling configuration shall follow one of the two options shown in Figure J—10. In general, Option 1 occurs when the end-of-tape warning mark is reached while OS/3 tape SAM is writing the last block of a file. Usually, data management does not yet know that this is the last, and records the EOV1 and EOV2 labels at this time. On the other hand, Option 2 occurs when the end-of-tape warning mark is reached after the EOF1 and EOF2 label group have been started.

MULTIFILE, MULTIVOLUME

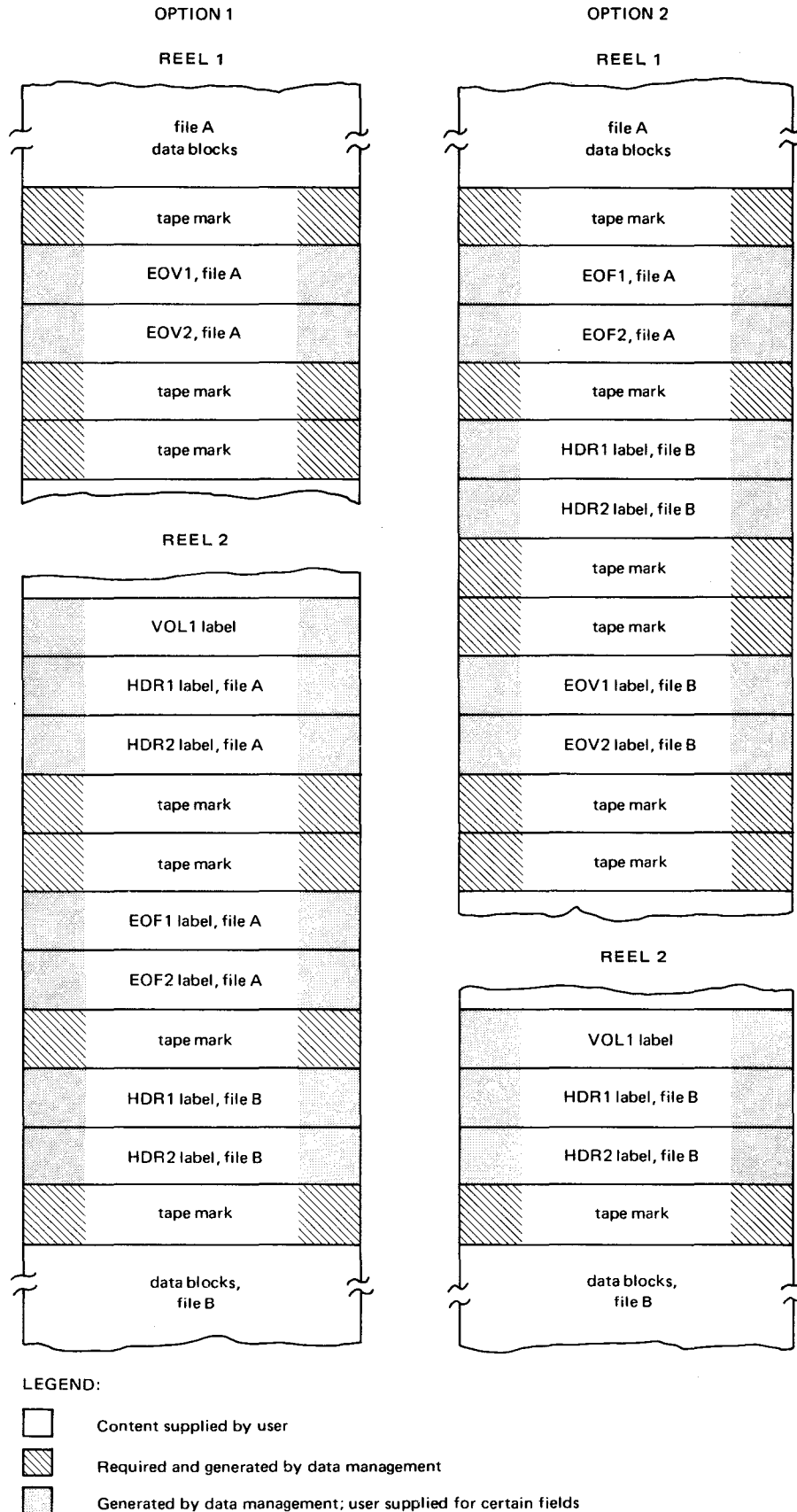
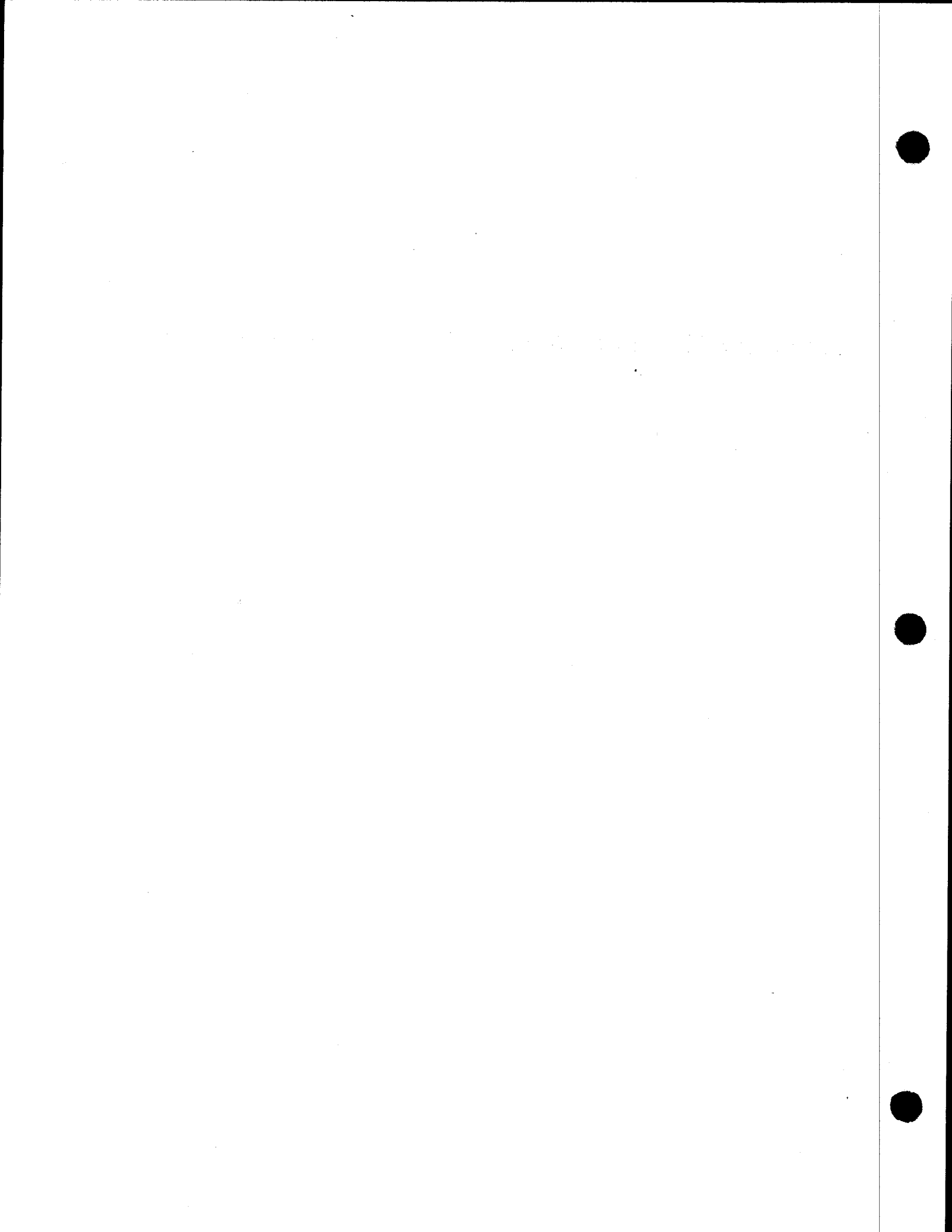


Figure J-10. Label Configuration Options, ASCII Multifile, Multivolume Set, When End-of-Volume and End-of-File Coincide

Appendix K. Load Code and Vertical Format Buffers



LOAD CODE AND VERTICAL FORMAT BUFFERS

This appendix explains, from the point of view of the basic assembly language (BAL) user of data management, how to specify codes for the *load code buffer* (LCB) and the *vertical format buffer* (VFB) of the printer subsystems supported by SPERRY UNIVAC Operating System/3 (OS/3). For these purposes, the BAL programmer uses the OS/3 job control *LCB* and *VFB statements*; for full details, refer to the job control programmer reference, UP-8217 (current version).

The printer subsystems in question are:

- SPERRY UNIVAC 0768 Printer Subsystem
- SPERRY UNIVAC 0770 Printer Subsystem
- SPERRY UNIVAC 0773 Printer Subsystem
- SPERRY UNIVAC 0776 Printer Subsystem
- SPERRY UNIVAC 0778 Printer Subsystem

Refer to Table C—3 for operational characteristics of these printers.

LOAD CODE BUFFER INTERCHANGEABILITY

There is no interchangeability of printer load code buffers across devices; an LCB job control statement specified for a particular printer and print band or drum cannot be used for any other.

LCB STATEMENT SPECIFICATION

The user specifies the codes to be assigned to each graphic symbol on the print band or drum, using the X (hexadecimal) or the C (character) positional parameters of the *LCB statement*. He must specify a character code or a hexadecimal specification for each symbol on the band or drum, and he may intermix X and C specifications. Each X or C specification must be complete on a single card. As many specifications as are necessary to specify an entire band or a single repeated font may be made.

The *space* or nonprinting *code* should be specified through the SPACE keyword parameter, and not included in the sequence of codes specified through the positional parameters.

If the number of characters is specified with the NUMBCHAR keyword, it should include only the number of codes specified for graphic symbols and should not include the *space code*.

If the CARTNAME keyword is specified, the operator will receive a message to mount the specified band, and program execution is suspended until the operator replies to the message. If the CARTNAME keyword is not specified, no operator message is issued.

LCB Specification for the 0773 and 0778 Printers

The user may specify 48, 63, 64, or 256 characters for the 0773 and 0778 printers; however, any band having more than 64 characters requires the specification of 256 characters. A 128-character cartridge requires the specification of 256 characters. A 128-character cartridge requires that the 128 characters be specified twice on the LCB statement.

Dualing applies to 48-character bands only; the user specifies dualing with the DUAL keyword of the LCB statement. Four *dualing characters* may be specified for the 0773 and 0778 printers; these correspond to the 39th, 40th, 44th, and 47th characters on the band.

The CARTID specification is optional for the 0773 and 0778 printers.

LCB Specification for the 0770 and 0776 Printers

The user may specify from 24 to 384 characters for the load code buffer of the 0770 or 0776 printer. For repeating fonts ranging from 24 to 192 symbols, he needs only to specify the characters for a single font: for example, the user would specify only 128 characters through the LCB statement for a repeating font of 128 characters.

Dualing for the 0770 and 0776 printers involves specifying up to four pairs of codes with the DUAL parameter. Each pair consists of one code that has been specified for the load code buffer, followed by one code that has not. Assuming, for example, that a band contains the question mark symbol(?), but not the vertical bar (|), the user could substitute ? in his printout for | by specifying DUAL=C'?'|. Every time his program outputs the EBCDIC code for a vertical bar to be printed, a question mark appears on the printed listing.

For 0770 or 0776 printers, the user must specify the CARTID parameter, and the code he specifies must be the correct one for the cartridge he intends to use.

The user may also specify a *mismatch character* for the 0770 or 0776 printers: that is, he may specify what character, other than blank (space), is to be printed whenever a *character mismatch* occurs.

LCB Specification for the 0768 Printer

The user need only specify the string of codes for the load code buffer and the MISM, SPACE, and TYPE parameters. He may also specify the optional NUMBCHAR parameter, but the other parameters of the LCB statement do not apply to the 0768 printer.

VERTICAL FORMAT BUFFER INTERCHANGEABILITY

Table K—1 summarizes the conditions under which a properly specified VFB statement for one printer may be used with other devices. There is no difference in the appearance of the printed results if the same VFB statement is used from machine to machine under these conditions.

VFB STATEMENT SPECIFICATION

Specifying a VFB job control statement involves visualizing the form with numbered lines. All 11-inch form to be printed at a density of 8 lines per inch has 88 lines. At 6 lines per inch, an 11-inch form has 66 lines. The first printable line on a form is line 1. The last line on an 11-inch form, printed at 8 lines per inch, is line 88.

Because lines may be printed (and the form advanced) beyond the overflow position, users must provide enough space between the overflow code position and the bottom of the form for any lines (and form advances) that must fit on the page. Users must allow at least four lines between the overflow code position and the bottom of the form. This is particularly important for VFBs that are used to print dumps, librarian runs, assemblies, etc.

Specifying Home Paper Position

The HP code specified on the VFB job control statement gives the line number location of the *home paper* position: the specification HP=5 places the home paper position on the fifth line of the form.

Table K-1. VFB Statement Specification and Interchangeability

Specification of TYPE Keyword	Statement may be used with Printer types (Note 2)	Other Keywords that the User May Specify (Note 1):																	
		HP	OVF	OVF2	CD1	CD2	CD3	CD4	CD5	CD6	CD7	CD8	CD9	CD10	CD11	CD12	CD13	CD14	CD15
=0773 (or keyword omitted)	0768 } TYPE 0770 } keyword 0776 } omitted 0773 0778	X	X			X	X	X	X	X									
=0768	0768 } TYPE 0770 } keyword 0776 } omitted	X	X			X	X	X	X	X		X		X	X		X	X	X
= 0770 Note 3	0770 Note 4	X	X	X		X	X	X	X	X		X		X	X		X		
=0776	0768 } TYPE 0770 } keyword omitted 0776	X	X			X	X	X	X	X		X		X	X		X		

LEGEND:

- X Keyword may be specified.
- █ Keyword may not be specified.

NOTES:

1. This table is concerned with only the keywords shown; the user may always specify the LENGTH, DENSITY, FORMNAME, and USE keywords.
2. The TYPE keyword should be specified only if a particular printer type must be used. A VFB statement designed for a particular printer (using the permitted keywords shown for that printer in Table M-1) may be used with other printers only if the TYPE keyword is omitted.
3. The user should specify TYPE=0770 only if he specifies a secondary overflow code (OVF2) or if he specifies multiple home paper positions.
4. If the user does not specify the OVF2 keyword, he may use the VFB statement (TYPE keyword omitted) with the 0768, 0770, and 0776 printers.
5. The secondary overflow code (OVF2) should be specified only by a data management user who issues the PRTOV imperative macro. Refer to the PRINTOV keyword parameter of the DTFPR and PRIO declarative macros.

Specifying Forms Overflow Position

The BAL programmer uses the OVF keyword of the VFB job control statement to specify the *forms overflow* position to printer SAM. He should not specify the OVF keyword if he does not intend to use it.

When an overflow code is placed in the buffer, a space form operation (advance paper *n* lines) which would move the form to or beyond the overflow position causes forms overflow to be detected. On detecting forms overflow, printer SAM takes action according to the user's specifications of the PRINTOV keyword parameter in the DTFPR declarative macro.

No indication of overflow is returned to the user, except that printer SAM transfers control to his overflow routine if he has so specified. In this routine, he may take such actions as skipping to the top of the next page, printing page numbers, printing subtotals, and so forth.

You must specify the overflow code position so that enough space is left between the overflow position and the bottom of the form to print and space all of the lines that are to appear on the page. Printer SAM may print a line on or below the overflow code position and perform spacing before branching to your overflow routine.

If a PUT appears in the overflow routine, the effect will be to perform spacing and/or print a line between the overflow position and the bottom of the form.

If the user does not specify the PRINTOV keyword in the DTF, data management takes no overflow action; in this event, the BAL programmer who is not counting lines in his program runs some risk of tearing the form by printing on or too near the perforations.

Table K—1 summarizes the combinations of device-independent control character codes permitted when using each type of printer with or without the TYPE parameter specification on the VFB job control statement. Because they describe the allowable use of device-independent control character codes. Tables K—1 and A—4 should be used conjunctively. Table A—4 interprets the control character codes associated with each of four printer functions (print and space, print and skip, spacing, and skipping). Note that the 0770 printer has two overflow codes (9 and 12) which the data management PRTOV imperative macro can detect selectively; the user should specify a secondary overflow code (hexadecimal code 9, specified through the OVF2 keyword) only with the 0770 printer and only if he is using the PRTOV macro, or (if he is not using data management) its PIOCS equivalent.

Specifying Special Forms

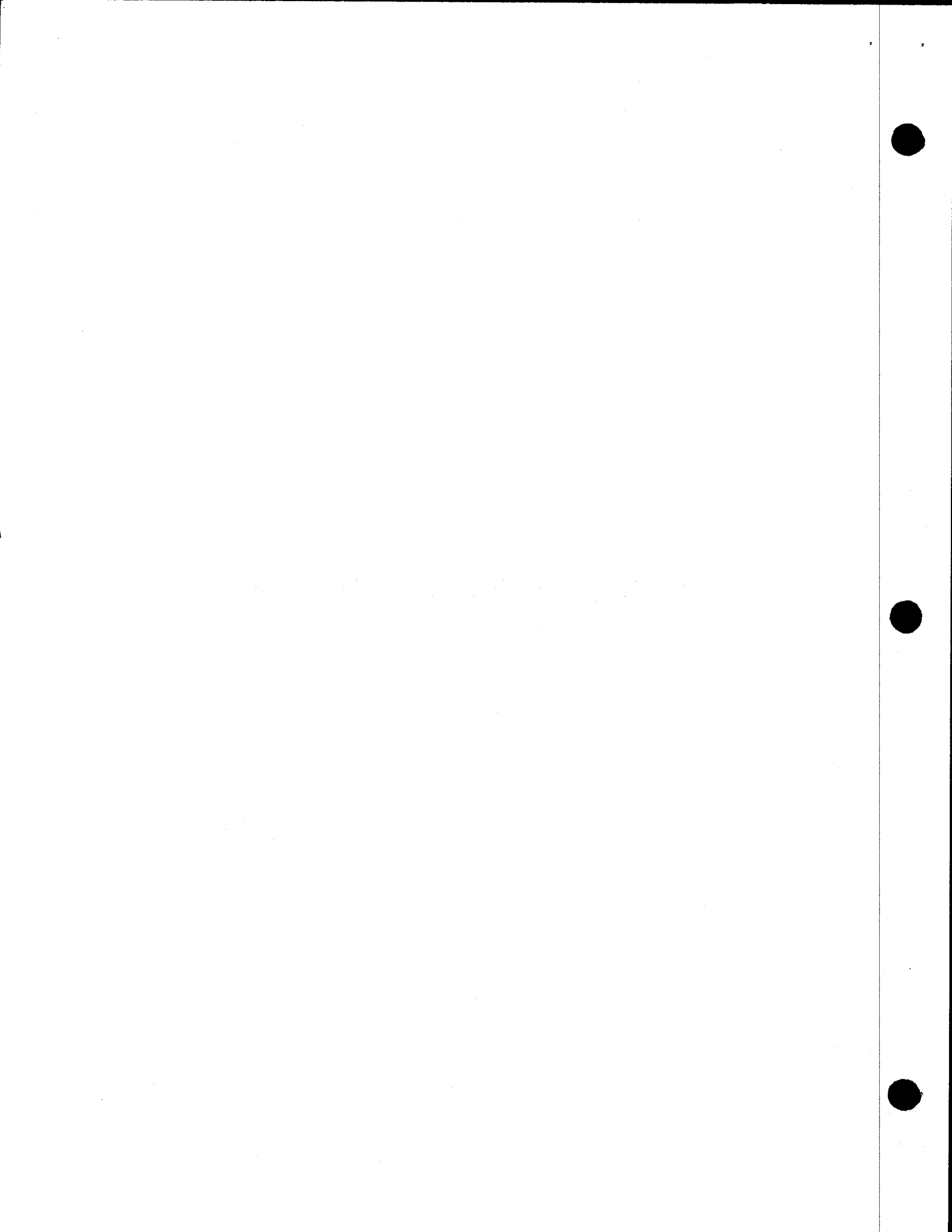
If the user specifies the FORMNAME keyword in the VFB job control statement, the operator is issued a message to mount the specified form, and program execution is halted until the operator replies.

Paper Tape Loop, 0768 Printer

For the *0768 printer*, the user must provide *both* a paper tape loop and a VFB job control statement. The paper tape loop should be punched to agree exactly with the VFB statement, with the following exceptions:

1. A 7 should not be punched on the tape. Home paper should be punched either as 15 (for 8 lines per inch spacing) or as 14 (for 6 lines per inch). Only one home paper code may be punched on the tape. The home paper code punched on the tape determines the line spacing rate.
2. Channel 1, 2, 3, or 12 should not be punched on the tape.

L. USER FILE LOCK FEATURE



USING THE FILE LOCK FEATURE

The OS/3 file lock feature allows you to control the sharability of a file while you are using it. Sharability control only applies to lockable files. To use the file lock feature, proceed as follows:

1. At system generation, you specify the FILELOCK parameter to indicate which files are lockable.
2. In your file definition (within your BAL program) and in the device assignment set for the file (regardless of the program type), you specify your read/write requirements and indicate whether other jobs or tasks within a job can share the file.

The following paragraphs describe how to indicate which files can be locked and how to set various degrees of sharability.

INDICATING WHICH FILES ARE LOCKABLE

You indicate which files are lockable by using the FILELOCK parameter in the SUPGEN section of the parameter processor at system generation time.

If you choose FILELOCK=NO or omit the parameter, only the system files prefixed with \$Y\$ are lockable. No user files can be locked.

If you choose FILELOCK=YES, all system files (prefixed with \$Y\$) and all files prefixed with \$LOK01-\$LOK99 are lockable.

If you choose FILELOCK=SHARE, all files are lockable.

SETTING FILE LOCKS FOR DATA FILES IN BAL PROGRAMS

After you specify which files are lockable, you specify the degree of sharability for each of these files.

If you choose FILELOCK=YES at system generation time, you can lock any file whose file name you prefixed with \$LOK01-\$LOK99 in the // LBL job control statement in the device assignment set. If you do nothing more, any prefixed file will be exclusively locked when it is opened during the execution of your program. You have exclusive use of the file. You can read, update, and add to the file. No other user can open the file until you close it.

If you choose FILELOCK=SHARE at system generation time, you can lock all of your files. If you do nothing more, each file will be exclusively locked when it is opened during the execution of your program. You have exclusive use of the file. You can read, update, and add to the file. No other user can access the file until you close it.

In both cases (FILELOCK=YES or FILELOCK=SHARE specified at system generation), you can override this lock by specifying one of the options of the ACCESS parameter or by specifying the LOCK=NO parameter in the DTF macroinstruction for a file.

You can also override this lock at program execution time two ways. The first way is to include a // DD job control statement that specifies one of the ACCESS parameter options in your device assignment set. The second is to prefix the filename with an asterisk (*) in the // LFD job control statement in the device assignment set. This will cause a read-only lock to be applied to the file; that is, you can only read from the file and all other users can only read from the file. refer to the job control user guide, UP-8065 (current version) for further information on the // DD and // LFD job control statements.

NOTE:

To set file locks on SAT files, see the supervisor user guide, UP-8075 (current version).

SETTING FILE LOCKS FOR DATA FILES IN NON-BAL PROGRAMS

After you specify which files are lockable, you specify the degree of sharability for each of these files.

If you choose FILELOCK=YES at system generation time, you can lock any file whose file name is prefixed with \$LOK01-\$LOK99 in the // LBL job control statement in the device assignment set. If you do nothing more, any prefixed file will be exclusively locked when it is opened in your program. You have exclusive use of the file. you can read, update, or add to the file. No other user can access the file until you close it.

In both cases (FILELOCK=YES or FILELOCK=SHARE specified at system generation) you can override this lock at program execution time. There are two ways to do this. The first way is to include a // DD job control statement that specifies one of the ACCESS parameter options in the device assignment set. The second is to prefix the file name with an asterisk (*) in the // LFD job control statement in the device assignment set. This will cause a read-only lock to be applied to the file; that is, you can only read from the file and all other users can only read from the file. Refer to the job control user guide, UP-8065 (current version) for further information on the // DD and // LFD job control control statements.

FILE LOCK FEATURE SUMMARY

Table L—1 summarizes the data management file lock feature. Remember, before using this feature, you indicated which files are lockable at system generation time through the FILELOCK keyword parameter. Once again, the available options are:

- FILELOCK=NO indicates that only the system files prefixed with \$Y\$ are lockable. No user files can be locked.
- FILELOCK=YES indicates that all system files (prefixed with \$Y\$) and all files prefixed with \$LOK01-\$LOK99 are lockable.
- FILELOCK=SHARE indicates that all files are lockable.

Table L—1. File Lock Summary

LOCK Keyword	Action	ACCESS Keyword	Action
LOCK=NO ① not specified	This DTF: read use/ update use/add use Other jobs: no access	ACCESS=EXC ①	This DTF: read use/ update use/add use Other jobs: no access
LOCK=NO ②	This DTF: read use Other jobs: read use	ACCESS=EXCR	This DTF: read use/ update use/add use Other jobs: read use
		ACCESS=SRDO ②	This DTF: read use Other jobs: read use
		ACCESS=SRD	This DTF: read use Other jobs: read use/update use/ add use

NOTES:

- ① LOCK=NO not specified and ACCESS=EXC are functionally equivalent.
- ② LOCK=NO and ACCESS=SRDO are functionally equivalent.

Glossary

A

absolute disc address

Not used in OS/3 disc file data management; see *relative disc address*.

accessibility field

In *ASCII* magnetic tape labelling practice, a field that occurs in the *VOL1*, *HDR1*, *EOF1*, and *EOV1* labels; its content indicates any restrictions on who may have access to the information in the volume.

During forward-read input of an ASCII file in OS/3, an ASCII space character (2/0) encountered in this field of both the *VOL1* and *HDR1* labels allows processing to continue. Any other character in either field causes data management to issue error message DM57 (ACCESS ATTEMPTED TO UNEXPIRED FILE RIC) to the operator's console. If the operator responds "I", processing may continue; his response "C" cancels the job. He may mount the correct volume and reply "R".

In OS/3, there is no option to write a nonspace character in the accessibility field of any system label in ASCII output files; data management always creates this field as a "space".

activity

An inherent characteristic of a file; the percentage of records processed on a run, or the frequency of reference to it. Activity affects the choice of file organization; if activity is low, a direct organization is preferable because any record may be located quickly without reference to any other. Those records processed most frequently should be those most quickly locatable. The amount of activity also affects file organization; an active file should be so organized as to minimize time involved in locating records.

add

In OS/3 ASAM and ISAM, to add records to an existing file is to write them (using the ADD macro, or its equivalent: WRITE,NEWKEY) to the *overflow* area of the file; this function does not affect the location of unused *prime data* space.

Adding records requires specification of IOROUT=ADD or IOROUT=ADDRTR in the DTFIS declarative macro, and prior specification of the PCYOFL keyword parameter to allocate cylinder overflow area, during the *load* function creating the file. Specification of all other DTFIS keywords must be the same as when the file was loaded.

The user presents new records randomly in the work area. Keys of added records may be higher than the key of the final prime data record, which is contained in the data block marked as the *logical end-of-file*. Data management moves them singly to the I/O area, from which it writes data blocks to disc. Each record is *chained* from another record so as to give it a logical position in the series.

If the user adds enough new records to an ASAM or ISAM file, some cylinder overflow space will become filled. When this occurs, data management resorts to using space on the earliest cylinder having available overflow; it adds records at the earliest possible point of the following sequence:

1. On the cylinder reached by prime search.
2. On the cylinder identified as current independent overflow.
3. On a cylinder newly set as being the current *independent overflow* cylinder.

addressable DTF fields

See *filenameA*; *filenameB*; etc.

alignment

In OS/3 data management, certain user-specified fields in main storage require alignment on specific boundaries:

<u>Type of Field</u>	<u>Alignment</u>	<u>Related DTF Keywords</u>
I/O areas (buffers)	Half-word	IOAREA1, IOAREA2
General register save areas	Full-word	SAVAREA
Search argument	Half-word	KEYARG
ISAM index area	Half-word	INDAREA

In OS/3 card SAM, the I/O area must be so located that the first data byte is half-word aligned. Work areas and *translation tables* require no special alignment. For the alignment of user-addressable fields in the DTF file tables, see *filenameA*, *filenameB*, etc.

alternate sequential access method (ASAM)

An access method, unique to OS/3 ISAM, in which files are created and processed without an *index*. Records are directly accessed by relative address rather than by key, but may be keyed or unkeyed. Sequential processing is the same as in ISAM. Records may be added in any number and, logically, to any point of the file. ASAM files are defined with the DTFIS macro, by specifying INDEXED=NO and omitting the INDAREA and INDSIZE keywords. ASAM uses all of the ISAM imperative macro repertoire but READ,KEY; SETL,KEY; and SETL,GKEY.

American National Standard Code for Information Interchange (ASCII)

A standard code, using a code character set consisting of 7-bit coded characters (eight bits, including parity check) and used for information interchange among data processing systems and associated equipment. The ASCII set comprises control characters and graphic characters (Appendix J); the ASCII code itself is documented in *American National Standard Code for Information Interchange, X3.4—1968*. Related standards of interest to the OS/3 data management user are *American National Standard Magnetic Tape Labels for Information Interchange, X3.27—1969*, and *American National Standard Perforated Tape Code for Information Interchange, X3.30—1971*. Synonymous with *USASCII*. See also *translation table* (2) and (3).

ASAM

Alternate sequential access method.

ASCII

American National Standard Code for Information Interchange.

B**BDW**

Block descriptor word.

beginning of file (BOF)

In OS/3 ISAM, the logical start of a file.

binary mode

In paper tape data management, a mode of processing and of operating the *0920 paper tape subsystem* in which all eight bits in a byte in main storage correspond to the eight data tracks (or *levels*) punched on paper tape. Only 8-level tape may be processed. The *program connector* board of the paper tape subsystem is bypassed in binary mode. Data may be translated, but may not contain *shifted characters*.

Refer to the MODE keyword parameter of the DTFPT declarative macro. Contrast with *character mode*.

block

A set of contiguous records handled or recorded as a unit by OS/3 data management; the portion of a file transferred into or out of main storage by a single access. May contain all of one or more *logical records*. Synonymous with *physical block* or *physical record*, terms used largely in other systems.

block descriptor word (BDW)

In variable-length record format, the first four bytes of each logical block, the first two bytes of which contain the length of the logical block.

block index

In OS/3 ISAM, that part of the index structure containing pointers to *prime data* blocks. Not used in OS/3 ASAM. See also *top index*; *intermediate index*.

block numbering

In OS/3 magnetic tape SAM, an option that the user should specify with the BKNO keyword parameter of the DTFMT declarative macro when his input file contains block numbers (or block sequence indicators).

If the user specifies BKNO=YES in his DTF, tape SAM will accept but not process block numbers on EBCDIC tape files (or block sequence indicators on ASCII files). The option may be used with 9-track or 7-track tape; the user must reserve a special 4-byte storage area immediately *preceding* each of the I/O buffers, and these must be full-word aligned. The following table summarizes data management responses:

BKNO Keyword Specified in DTFMT Declarative Macro	Block Numbers Physically Present on Tape	OS/3 Data Management Response
no	no	none
no	yes	For standard-labelled files (FILABL=STD): issues error message DM55 (STD SYSTEM/USER LABEL NOT FOUND).
		For nonstandard or unlabelled files (FILABL=NSTD, FILABL=NO): the last three bytes of data in an EBCDIC file (or the last byte of data in an ASCII file) will be lost from each block read.

BKNO Keyword Specified in DTFMT Declarative Macro	Block Numbers Physically Present on Tape	OS/3 Data Management Response
no	yes	If records or blocks are variable-length, data management issues error message DM18 (RECORD SIZE INVALID) because of apparently incorrect record length. See Note 1.
yes	no	For standard-labelled files (FILABL=STD): the BKNO specification is ignored, and correct file processing continues.
		For nonstandard or unlabelled files (FILABL=NSTD, FILABL=NO): the last 3 bytes of data in an EBCDIC file (or the last byte of data in an ASCII file) will be lost from each block read. If records or blocks are variable-length, data management issues error message DM18 (RECORD SIZE INVALID) because of apparently improper record length. See Note 1.
yes	yes	Correct file processing continues (see Notes 2 and 3).

NOTES:

1. If the user's records are fixed-length in a nonstandard or unlabelled input file, he should provide for checking on the presence of block numbers because data management cannot determine the need for error message DM18.
2. In an ASCII input file, all labels, including optional user header or trailer labels, should be exactly 81 bytes long to ensure correct processing. Those in a standard-labeled EBCDIC file must be 83 bytes long.
3. OS/3 tape SAM does not support the optional padding feature for ASCII input files. Label and data blocks that contain padding may not be processed correctly.

The 4-byte storage area reserved preceding each of the I/O buffers is not included in the BLKSIZE specification, nor in the space reserved for the IOAREA1 or IOAREA2 buffer with the BAL *define storage* (DS) instructions.

block pointer

In OS/3 ISAM, a 3-byte pointer, located in the *block index* entries and containing in binary the data-partition-relative block number of the block to be read. See also *relative disc address*.

block prefix

See *buffer offset*.

block sequence indicator

See block numbering.

BOF

Beginning of file.

braces

In OS/3 data management documentation, braces comprise a pair of symbols {} used to enclose two or more mandatory entries in a format delineation to indicate that one must be chosen and coded by the programmer. Braces themselves are never coded.

brackets

In OS/3 data management documentation, brackets comprise a pair of symbols [] used to enclose optional entries in a format delineation to indicate that they may be omitted or coded, depending on program requirements. Certain entries enclosed in brackets are mandatory under specified conditions. *Braces* within brackets signify that one of the enclosed entries must be chosen if the optional parameter is chosen. Brackets themselves are never coded.

buffer offset

In *ASCII* magnetic tape labeling practices, the term applied to any information required at the front of each data block on tape, not specified by the standard but needed for a user application. This information may include such items as block length, block address of the last record in the block, initial padding for word machines, dates, times of transmission, etc. The length of the buffer offset field, if used, is specified in field 7 of the second *file header label* (HDR2) and is included in the block length specified in field 4. Refer to BUFOFF keyword parameter of the DTFMT declarative macro, and to *American National Standard Magnetic Tape Labels for Information Interchange, X3.27—1969*.

C**capacity tables**

Refer to Appendix E.

cardpunch records

1. Variable-length

Before issuing PUT macro, user must determine size of record and insert this, in binary, in the first two bytes of the 4-byte record-length field. This field follows a reserved 4-byte field at the head of each variable record.

2. Undefined

Before issuing PUT macro, user must determine record size and place it, in binary, in the general register specified by the RECSIZE keyword parameter. Refer to Figure G—1.

CCB

Command control block.

CCW

Command control word; channel command word

chaining

In OS/3 ISAM and ASAM, the technique of inserting a 5-byte *record pointer* after each data record on disc to identify the location of the next record, in logical sequence, in the file. In ISAM and ASAM, new records may be chained from *prime data* records or from records in the *overflow* area. See also *add; load; relative disc address*.

channel command word (CCW)

A control block that specifies the operation, data address, byte count, and other control information used by the selector channel. CCW is also the call for a macro of the OS/3 supervisor that generates a channel control word to provide a hardware parameter interface to the selector channels for use by PIOCS. See also *command control word*.

character deletion

See *delete character; scan table*.

character mismatch

The *unique unit error* in OS/3 printer SAM. Occurs when the printer attempts to print a bit configuration which is not in the printer's *load code buffer*. Refer to the UCS keyword parameter of the DTFPR macro instruction.

In OS/3 job control, the MISM keyword of the *LCB statement* may be used to force reporting of this error condition on the system log, and the *mismatch character* may be supplied by the MISMCHAR keyword. Refer to the OS/3 job control programmer reference, UP-8217 (current version), and to Appendix K.

character mode

In paper tape data management, the standard or nonbinary mode of operating the *0920 paper tape subsystem*. In this hardware mode, five to seven bits of a byte in main storage correspond to the *data levels* or tracks in paper tape; five to seven levels can therefore be read or punched, with an optional odd or even parity track. Data may be translated and may contain *shifted characters*. By wiring the program connector board, the user causes the hardware to recognize an *end-of-record stop* character and one hardware *delete* character. Refer to the MODE keyword parameter of the DTFPT declarative macro. Contrast with *binary mode*.

character shifting

See *shift character*.

check cycle

In disc operations, an immediate reading back of data written on one revolution to ensure that it has registered properly. Effectively doubles the nominal write time because a second revolution is needed to check each item written.

checkpoint blocks

In order to bypass checkpoint blocks interspersed with data in an input EBCDIC magnetic tape file, the user specifies the CKPTREC keyword parameter of the DTFMT declarative macro. If this keyword is not specified and the file contains checkpoint records, these are processed by the user's program. In order for tape SAM to recognize the checkpoint set, the user's BLKSIZE specification must equal or exceed the length of a header or trailer label of a checkpoint set. Checkpoint records are not supported for ASCII files, input or output.

In OS/3 tape files, the first and last blocks of a checkpoint dump begin with the following:

```
//△CHKPT△//nnttCsss
```

where:

nn

Is the number, in binary, of image records plus control blocks, less 1, not including the header or trailer labels.

tt

Is the total number, in EBCDIC, of checkpoint records following the header label, including the trailer label (*tt* is 00 in a trailer label).

C

Is a constant, coded in EBCDIC as shown.

sss

Is the serial number of the checkpoint, in EBCDIC.

COCR

Cylinder overflow control record.

code correspondences

Refer to Appendix I.

column binary

A punched card code so constructed that each card column requires two bytes of main storage. Refer to Figure I—2. Synonymous with *image code*. Contrast with *compressed card code*.

combined card file

In OS/3 card SAM, a file defined by specifying TYPEFLE=COMBND in the DTFCD declarative macro. Cards may be read and punched from a combined file in the same pass; if they are to be processed in the *overlap mode*, the ORLP keyword parameter must also be specified.

A combined card file may contain only fixed-length, unblocked records; no format specification other than RECFORM=FIXUNB may be used.

See also *read/punch*.

command control block (CCB)

A command block, located anywhere in main storage, which contains constants and control words, or addresses of control words, used by the *physical IOCS* to execute a command.

command control word (CCW)

A *channel command word*, located anywhere in main storage, which specifies the I/O operation to be performed, the start of the buffer, and the number of bytes to be transferred.

compressed card code

A punched card code so constructed that each card column is represented by an 8-bit pattern in one byte of main storage. Refer to Figure I—1. Contrast with *column binary*.

control characters

See *first character control*.

current I/O area

When user is processing records in more than one I/O area, he specifies the address of the second area with the IOAREA2 keyword parameter in his DTF declarative macro and uses an *index register*, specified with the IOREG keyword, to point to the address of the current record. Refer to Table A—1.

cylinder capacity

Refer to Appendix E.

cylinder overflow control record (COCR)

In OS/3 ASAM and ISAM files for which *overflow* is specified, the 5-byte record on each cylinder that is created by data management and used to record the location of the remaining available overflow space on the cylinder. It is contained in the last data block in the cylinder. See also *independent overflow*; *relative disc address*.

D**DAM**

Direct access method.

data conversion

Refer to Appendix J.

data level

With reference to paper tape, indicates the number of data holes that may be punched per character on tape: in a 7-level tape, for example, there are seven hole positions available for each character. Such a tape is also sometimes called a "7-track" or a "7-channel" tape; the number of distinct characters that may be punched in it is 2^7 or 128.

deallocation

See *scratch*.

declarative macro instruction

A macro instruction that allows specification of and causes generation of machine instructions to define a table such as a DTF.

Declarative macros are used in OS/3 data management to define files and to specify logical IOCS processors. The format of the *define the file* (DTF) declarative macro is:

LABEL	△ OPERATION △	OPERAND
filename	XXXXXX	keyword-1=x,...,keyword-n=z

The symbolic name of the file must appear in the label field, may have a maximum of seven characters, and must begin with an alphabetic character. The operation field contains the mnemonic macro call. In the operand field, the keyword parameters may be coded in any order, but must be separated by commas. A keyword parameter consists of an alphanumeric character string immediately followed by an equal sign and one completion, or specification. General registers are specified in the keyword parameters of DTF macros by enclosing the register number in parentheses.

The format of the declarative macro used to configure logical IOCS processors is:

LABEL	△ OPERATION △	OPERAND
ignored	XXXXXX	keyword-1=x,...,keyword-n=z

Data management ignores any coding in the label field. The operation and operand fields are coded as for the DTF declarative macro.

default specification

In OS/3 documentation, a shaded background is used in instruction format delineations, macro parameter descriptions, and tables summarizing parameters to indicate the default values assumed by the system when an optional parameter is omitted by the user.

define the file (DTF)

Any of a series of tables generated by a *declarative macro instruction* which defines a file and the data management technique which will be used in processing the file.

delete character

In paper tape data management, a special character punched by the user to obliterate unwanted data from an already punched paper tape record; sometimes called a "rub-out" character. Its usual configuration is a punch in each *data level* available on the tape.

In paper tape data management, a character deleted from an input paper tape record by hardware or software action before the user is given access to the character's representation in main storage.

When the *0920 paper tape subsystem* is operated in character mode (MODE=STD), its *program connector board* may be wired so that the hardware automatically recognizes one delete character; this character, often called a "hardware delete" is not transferred to main storage from input records.

In addition, through the use of a *scan table*, the user may specify one or more "software deletes", which data management removed from an input record before it makes the "compressed" result available to the user in his data area. For delete characters, the user may specify any of the codes that may be punched in the number of *data levels* in his tape.

Because the program connector board is bypassed when processing in binary mode, it is not possible to use it to specify a hardware delete character. Using a scan table is the user's only way to delete characters from binary mode input records.

In character mode, delete characters may be used to constitute an *interrecord gap* or the *paper tape leader or trailer*; only the *null character* may be used for these purposes in binary mode.

Refer to the FSCAN, LSCAN, and SCAN keyword parameters of the DTFPT declarative macro.

deletion

See *delete character*; *record deletion*; *scratch*.

density

Same as *line spacing* (1).

device assignment set

In OS/3 job control, the set of job control statements by means of which the user specifies the relationship between his files and the peripheral devices allocated to his job. Every file referenced by the user's program must be represented in the job control stream by a device assignment set. The minimum set comprises at least a DVC statement followed by an *LFD statement*. A full device assignment set for disc files also requires a *VOL, EXT,* and *LBL* statement; tape and unit record files do not use the *EXT* statement. Printer files may require the *LCB* and *VFB statements*.

direct access

Access to a record on disc by use of its address, rather than by matching an argument to a portion of the record contents. Contrast with *search on key*. See also direct addressing. Synonymous with *random access*.

direct access method (DAM)

A disc file access technique within OS/3 data management which allows processing of logical records at random, accessing each record directly by its *relative disc address*.

direct addressing

The ability to use numeric values given in a field to retrieve a specific block or record from disc storage by a single access. See also *direct access relative disc address*.

discontinuous binary

A method of coding a numeric field such that each subfield is treated as a distinct binary entity. For example, a 1-byte straight binary field containing 11111111 would have the single decimal value 255; if it were considered to comprise two 4-bit subfields, in discontinuous binary, each half word would represent the decimal value 15.

disc

A type of direct access storage device. The SPERRY UNIVAC disc types supported by OS/3 are the following:

- 8411 disc subsystem
- 8413 Diskette Subsystem
- 8414 Disc Subsystem
- 8415 Disc Subsystem
- 8416 Disc Subsystem
- 8418 Disc Subsystem
- 8424 Disc Subsystem
- 8425 Disc Subsystem
- 8430 Disc Subsystem
- 8433 Disc Subsystem

Refer to Appendix C for disc capacity tables and record transmission times.

disc label processing

In the nonindexed disc file processing system, the user may process optional *user header labels* (UHL) or *user trailer labels* (UTL) in a routine whose address he furnishes to data management with the *LABADDR* keyword parameter of the *DTFSD, DTFDA,* or *DTFNI* declarative macro. Labels are not maintained at the file partition level, nor are user labels permitted in OS/3 *ASAM* or *ISAM*. For the format of the UHL and UTL, refer to Appendix D. Each user label is an 80-byte block.

In his *LABADDR* routine, the user codes ordinary *BAL* instructions and issues the *LBRET* imperative macro to create, retrieve, or retrieve and update his labels or to transfer control. No other imperative may be issued in the label routine. The user may have a maximum of eight optional user labels of each type.

When the *LABADDR* keyword is specified in the *DTF*, data management reserves the first track of *each* volume of a *DTFSD* file, or the first track of the *first* volume of a *DTFDA* or *DTFNI* file, for the UHL and UTL, which it writes or reads when the user issues the *LBRET* macro.

In these DTF declarative macros, the TYPEFLE keyword specifies to data management the mode of label processing to be performed on the file. User labels in DTFSD files and in sequentially processed DTFNI files may be updated only if TYPEFLE=INOUT is specified, or if the file processing direction has been reset to update mode with the SETF imperative macro. The UPDATE keyword parameter in the DTFs for disc files is unrelated to label processing.

When the user has specified the LABADDR keyword, his label routine is always accessed when the file is opened and, if he has also specified the TRLBL keyword, again when the file is closed. At file open time, register 0 contains the EBCDIC alphabetic character O in the least significant byte, which the user's LABADDR routine accesses to determine that it is to process UHL. The EBCDIC character F, placed in this register at file close, indicates that the user may process his UTL. Unlike some other systems, OS/3 data management does *not* read a user label into the I/O area before the OPEN transient gives control to the LABADDR routine. The user must issue the LBRET 2 macro to retrieve each user label.

Labels retrieved are made available to the user in the I/O buffer whose address data management loads into general register 1; it is not possible to process them in a work area. Likewise, the user must create user labels in the buffer pointed to by register 1.

disc prep

OS/3 uses one disc prep utility routine for making a surface analysis of disc volumes and preformatting them: DSKPRP, which services the 8415, 8416, and 8418 fixed-sector discs and the variable-sector 8411, 8414, 8424, 8425, 8430, and 8433 disc subsystems. DSKPRP is described in the system service programs (SSP) programmer reference, UP-8209 (current version). In OS/3 data management, the OPEN transients perform automatic preformatting of discs for new DTFDA, DTFNI, DTFSD, and DTFIS files, writing zero records (containing all binary 0's) at the user's specified block size on these volumes. The data management user may also have OS/3 job control preformat disc volumes for him, via a parameter of the job control EXT statement. Refer to the job control programmer reference, UP-8217 (current version).

disc space management

In OS/3, a set of routines of the supervisor, providing automatic space accounting and maintenance of the VTOC; most are called automatically for the data management user. Refer to the supervisor programmer reference, UP-8241 (current version). See also *rename*; *scratch*; *volume table of contents*.

disc track size

In OS/3, disc blocks or records may not cross a track boundary and therefore may not exceed in length the following track sizes:

<u>SPERRY UNIVAC Disc Subsystem</u>	<u>Track Size, in Bytes</u>
8411	3625
8413	3328
8414	7294
8415	10,240 (fixed 256-byte sectors, 40 sectors/track)
8416	10,240
8418	10,240
8424	7294
8425	7294
8430	13,030
8433	13,030

diskette

A type of direct access storage device intended as a rapid replacement medium for card processing devices. The diskette is divided into 128-byte sectors which correspond to records. Refer to Appendix C for diskette capacity tables and record transmission times.

double buffering

A technique useful with OS/3 data management to speed processing overall by overlapping sequential I/O operations with record processing through the use of two areas in main storage for I/O data. It usually involves the specification of two I/O buffers and an I/O register to point to current data, but it may instead entail the use of one I/O area and one or more work areas (requiring multiple-parameter forms of the GET and PUT macros).

No additional processing speed is gained by specifying both a secondary I/O area and a work area. For fastest processing, the user should specify either:

- A primary I/O area and one or more work areas; or
- A primary and a secondary I/O area.

Refer to the IOAREA1, IOAREA2, and IOREG keyword parameters of the DTFCD, DTFPR, DTFMT, DTFPT, DTFSD, and DTFNI/DPCA macros. Work area double buffering is specified by the WORKA keyword parameters of the declarative macros listed. Double buffering is not used in OS/3 DAM. Refer also to Table A—1, noting that it does not apply to DTFIS files.

OS/3 ISAM also allows double buffering via the IOAREA1/IOAREA2/IOREG keywords; more benefit accrues for load and sequential retrieval than for random operations. ISAM's uses of work areas, however, are distinct from those of the other access methods: refer to the WORK1 and WORKS keywords of the DTFIS declarative macro.

DTF

Define the file.

dualing characters

In a printer's *load code buffer*, alternate characters which the printer will accept to correspond to a given print band location. Specifiable in OS/3 job control via the DUAL keyword of the *LCB statement*. Refer to the OS/3 job control programmer reference, UP-8217 (current version), and to Appendix K.

dummy record

In OS/3 ISAM, a *prime data* record created and installed automatically by data management at the beginning of the file during the initial *load*. Its *key* contains all binary 0's, and the content of the 5-byte *record pointer* following it is hexadecimal FO AA AA AA AA. This pattern indicates to data management that the next record to it physically is also the next in logical sequence: the record that had the lowest key in the user's initial load.

The purpose of installing such a dummy record at file start is to enable the user to subsequently *add* records to the *overflow* data area having keys lower than the lowest key in the initial load.

For fixed-length record format, the length of the dummy record is the same as the user's specified record size. When the user's records are variable-length, the size of the dummy record (as contained in its own 2-byte record length field) is the sum of the specified key length and the number of bytes that precede the keys in the user's records. (Refer to the RECSIZE, KEYLEN, and KEYLOC keywords of the DTFIS declarative macro).

If the dummy record has a data portion, this also contains all binary 0's; however, the dummy record is not available to the user for storing data, and he may not retrieve it. For this reason, he should generally not present ISAM a *search key* comprising all binary 0's (refer to KEYARG keyword).

If, however, the user issues the SETL,BOF macro, or if he issues the SETL,GKEY imperative macro with an all-zero search key, data management prepares to give the user the first record *beyond* the dummy, and no exception/error reporting ensues.

On the other hand, to issue a READ,KEY or SETL,KEY macro with a search key of all binary 0's results in error processing. Data management issues error message DM32 (RECORD NOT FOUND FOR SEQUENTIAL FUNCTION), sets the *record not found* error flag in *filenameC* of the DTFIS file table, and branches to the user's ERROR routine. Refer to Appendix B.

Similarly, error processing results if the user issues:

- the WRITE,NEWKEY or ADD macro with a search key of all binary 0's;
- the READ,ID macro with the *relative disc address* of the dummy record specified in the KEYARG field; or
- the SETL,ID macro with the dummy's *relative disc address* specified in the KEYARG field.

In these instances, data management issues error message DM24 (INVALID REQUEST (ID) — OUTSIDE FILE LIMITS), sets the *invalid ID* error flag in *filenameC*, and branches to the user's ERROR routine. Refer to Appendix B.

dynamic extension

See *extend* (3).

E

ellipsis

In OS/3 data management documentation, a symbol comprising three periods (. . .) and used in format delineations to indicate the omission of a variable number of similar entries which may be coded serially by the programmer. The ellipsis itself is never coded.

end of data (EOD)

For sequentially processed disc input files defined by the DTFSD or DTFNI macros, the programmer must supply in his DTF the address of a routine coded to handle end-of-data processing, using the EOFADDR keyword. He need not specify this keyword in the DPCA macro unless he has special end-of-data requirements for the partition that differ from those for its parent DTFNI file. In addition to performing normal file termination procedures by issuing the CLOSE macro in his EOFADDR routine, the user may extend the file or partition beyond the end-of-data record, provided he has specified UPDATE=YES in the DTF or reset file processing direction to update mode with the SETF imperative macro. See also *extend* (2) and (3); *end-of-data* (/*) .

end-of-data (/*)

An OS/3 job control statement. OS/3 card SAM checks for an end-of-data card when reading a card file. When the /* configuration is sensed, data management transfers control to the end-of-file routine specified by the user with the EOFADDR keyword parameter in the DTFCD declarative macro for the file. User must supply this card for input and *combined card files*; data management does not punch the /* card for an output file. See also *end of data* (EOD).

end of file (EOF)

1. Generally, the *logical end-of-file*.
2. A hardware EOF is a unique marker generated by the hardware to denote EOF.
3. A software EOF is a software-generated unique record to denote a logical EOF, which may occur only in the block preceding the hardware EOF.

end-of-file label group

Same as *file trailer label group*.

end of partition

In the nonindexed disc file processor system, an entry made by data management in the DTFNI and DPCA file tables (and in the disc *format 2 label*), containing the 3-byte end-of-data pointer for the *partition*. This is the relative block address (plus 1) of the last block written into the partition; it is recorded in the file table or *partition control appendage* when, in the process of creating partitions sequentially, the user issues the SETP imperative macro to select the next partition. It may also be recorded when the user issues the WRITE,AFTER,EOF imperative macro while creating a partition; the result is to mark the logical end of the partition. See also *logical end-of-file*. Refer to Appendix D.

Within a *subfile* of a partition, however, the user must maintain his own end-of-subfile data record, which data management does not record in the *partition control appendage* nor in the subfile table. It is therefore possible to process through the end of a subfile to the end of the partition or the *logical end-of-file* (EOF).

end-of-record stop

In paper tape data management, a character specified by the user and punched on tape at the end of an undefined record to act as a delimiter of the record. Used only in *character mode*.

When the *0920 paper tape subsystem* is reading an input file containing undefined words, it automatically stops tape motion on encountering this delimiter at the end of a record. The delimiter, sometimes called the "wired stop" character, is specified to the hardware by wiring the *program connector* board and is the last character transferred to main storage by a read.

The user specifies the end-of-record stop character to data management for output files with the EORCHAR keyword parameter of the DTFPT macro.

end of subfile

See *end of partition*.

end-of-volume label group

Same as *file trailer label group*.

end-of-volume procedures

In both OS/3 tape SAM and the nonindexed disc file processing system, procedures initiated by data management when an EOVL label or sentinel is sensed at the end of the current volume of a sequentially processed file. For an output file, any data yet unwritten is first sent to the current volume as either a full or a truncated block, and the current volume is then closed. System standard labels (and optional UTL, if processing the latter is specified) are written. For an input file, end-of-volume labels, including UTL if present, are read and checked. Volume swapping is performed immediately, and the next volume is opened and checked; the next PUT or GET imperative macro involves I/O on the new volume.

EOD

End of data.

EOF

End of file.

EOF1

In OS/3 magnetic tape SAM, the first label in the *file trailer label group* of a standard-labeled file with an end-of-file condition; refer to Appendixes E and J.

EOF2

The second label in the *file trailer label group* of a standard-labeled magnetic tape file with an end-of-file condition. See *EOF1*.

EOV1

In OS/3 magnetic tape SAM, the first label in the *file trailer label group* of a standard-labeled file with an end-of-volume condition; refer to Appendixes E and J.

EOV2

The second label in the *file trailer label group* of a standard-labeled magnetic tape file with an end-of-volume condition. See *EOV1*.

error in OPEN

An error flag (bit 4, byte 0) set by data management in *filenameC*. Refer to Appendix B.

expiration date

In OS/3 magnetic tape SAM, the date on which a file may be overwritten or used for scratch. Contained in a field of the *HDR1*, *EOF1*, and *EOV1* labels of both EBCDIC and ASCII files; specified to data management in the third positional parameter of the job control *LBL statement* and used to prevent inadvertent writing on an unexpired tape file.

For output files, the expiration date in the existing *HDR1* label is checked against the current or actual calendar date to determine whether the associated file has expired. If it has, the tape is positioned so that the old *HDR1* label is written over. However, if the user attempts to write over an unexpired file (one whose expiration date is later than today's date), data management issues error message DM57 (ACCESS ATTEMPTED TO UNEXPIRED FILE RIC) to the operator at the system console. The operator replies "R" to retry the job; "I" to continue processing; and "C" to cancel the job.

In a multifile volume, if the user gives a new file a *later* expiration date than he has given to the files already recorded, the additional protection he intends 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.

When a new file is created, if the expiration date is not specified in the *LBL statement*, data management makes the expiration date that it writes in the *HDR1* label the same as the date on which the file is created.

extend

1. In OS/3 ASAM and ISAM, to extend a file is to *load* additional records into the *prime data* area of an existing file, by resuming the load process from the point where the initial load left off and an ENDFL imperative macro has terminated the initial load sequence. The file must have been closed and reopened, with the same specification of all DTFIS keywords as when it was created. Records may have already been placed in *overflow*, by an *add* operation.

IOROUT=LOAD must be specified, explicitly or by default, and a new file load sequence initiated with the SETFL macro. The user issues the WRITE,NEWKEY macro to extend the file with a new logical record that he has preloaded into the work area. As for load, keyed logical records must be presented in ascending order of keys, and all must have keys higher than the current highest key, whether this key is in a prime record or in an overflow record already on disc. Contrast with *add*; *load*; *reload*; *update*.

2. In the nonindexed file processor system, a DTFSD file or a sequential DTFNI file or partition may be extended beyond its current *end-of-file* (EOF) record when the file is being processed in the *update* mode. The file may not be extended beyond its current volume by this method, which requires the issuance of appropriate job control statements at the time of file creation and is effected by coding in the user's end-of-file routine. Refer to the EOFADDR keyword. See also *LFD statement*.
3. A DTFSD file may be dynamically extended when being processed in the output mode, and may be extended to additional volumes if desired. The user specifies TYPEFLE=OUTPUT in the DTFSD declarative macro or resets the file-processing direction to OUTPUT with the SETF imperative macro. The LFD statement in the *device assignment set* for the current last volume of the file must contain the EXTEND positional parameter. When the user issues an OPEN macro to the file, data management positions the file to its current *logical end-of-file* (EOF) address. Subsequent PUT macros add records beyond the current EOF. If additional volumes are required, the user allocates these and may continue to extend the file beyond its current last volume to subsequent volumes. All LFD names within a job step must be unique; if more than one file is given the same name, only the last to be specified is available for any operation.
4. A DTFNI file partition may be extended dynamically, but not beyond the volumes on which the file resides, by means of the UOS keyword parameter of the DPCA declarative macro (or of the DTFNI macro defining the first or only partition of the file). The user must have specified a *unit of store* greater than zero, and the job control *extent specification* (EXT) statement in the device assignment set must also specify an increment greater than zero. Subsequent PUT or WRITE imperative macros referencing a block lying beyond the current maximum relative block address that data management keeps in the *partition control appendage* causes data management to act automatically to extend the partition. The percentage of secondary disc space specified by the UOS keyword is suballocated to the partition each time. Data management keeps a record of suballocation information developed for all partitions in the *format 2 label* for the DTFNI file.

Data management sets the *invalid ID* flag in *filenameC* and branches to the user's error routine or returns control to him inline:

- if the user has not specified the UOS keyword;
- if he has specified UOS=00;
- if there is no space available on the disc to extend the partition; or
- if, after extending the partition by one unit of store, data management finds that the requested block still lies beyond the new maximum relative block address.

Error message DM24 is also issued; refer to Appendix B.

5. The OS/3 supervisor EXTEND macro is issued automatically by data management for the user when required; the data management user never issues this macro directly in his program. Refer to the supervisor programmer reference, UP-8241 (current version).
6. Magnetic tape SAM will automatically extend an output file beyond the volumes initially specified in the VOL statement, provided that the user has not specified the (PREP) parameter in the VOL statement and provides sufficient pre-prepped back-up volumes.

On the first PUT macro issued after end-of-volume has been detected (or forced by the user) on the last of the tapes specified in the VOL statement, the operator receives a mount message (with an override option) to mount a scratch tape. If he does so and responds "R" to the mount message, processing may continue. This procedure is repeated, one volume at a time, each time that the user specifies or reaches end-of-volume and then issues a PUT macro, until he issues a CLOSE macro to the file. Refer to the tape SAM FEOV imperative macro. See also *end-of-volume procedures; tape prep facilities*.

extent

A reserved contiguous physical area on a DASD volume, defined—for disc—in terms of tracks and cylinders by means of the starting and ending locations allocated to an individual file.

A set of contiguous tracks on disc, assigned exclusively to one file. Several extents may be required to provide space enough for a file. The maximum number of extents any OS/3 data management disc file may have on a single volume is 16; 13 extents is the maximum for a *split-cylinder* file.

extent map

Same as *extent table*.

extent table

A table of extents, normally used to define the areas on a disc volume or volumes which have been allocated to a particular file. Synonymous with *extent map*.

F**FCB**

File control block.

figure scan table

See *scan table*.

figure shift

See *shift character*.

file control block (FCB)

A repository for information required to control a specific file and the device on which it resides. Information in the FCB is compiled by job control from the *device assignment set* of job control statements for the file. One of the six basic control blocks in OS/3, the FCB is input to the OPEN imperative macro. Described fully in the supervisor programmer reference, UP-8241 (current version).

file deallocation

See *scratch*.

file definition

See *define the file (DTF)*; *LFD statement*.

file deletion

See *scratch*.

file header label group

In OS/3 standard-labeled magnetic tape files, a pair of required labels following the *VOL1 label* and followed by the *tape mark* that precedes the data blocks of each file. The first of these is the first file header label, HDR1; it identifies the file and is written at the beginning of each file. It is extended and followed by the second file header label, HDR2. The fields of these labels may be specified in the job control stream. Refer to Appendix E for format and content, and to Appendix J for the reel organization of a standard-labeled tape volume. See also *file trailer label group*.

file identifier

In OS/3 disc files a 44-byte field in the Format 1 label identifies the file. In OS/3 tape SAM, a field of the *HDR1*, *EOF1*, and *EOV1* labels; a unique 17-byte configuration that identifies the file. Refer to Appendix D and Appendix E. Specified to data management in the first positional parameter of the job control *LBL statement* in the *device assignment set* for the file. If no *LBL statement* is specified for an output file, data management uses the logical *file name* specified in the *LFD statement*. Refer to the job control programmer reference, UP-8217 (current version).

file lock

A facility common to all OS/3 disc data management methods whereby the user may control the extent to which disc files may be shared between tasks and jobs, and the mode in which sharable files may be processed. The user specifies that a disc file is lockable by assigning 1 of 16 unique lock IDs as a 6-byte prefix to the file name through the *LBL job control statement*. Thereafter, when the file is opened, a write or a read lock is set automatically on the file, according to the user's specification of the *LOCK keyword* parameter in the *DTF* defining it. One purpose of the file lock facility is to prevent other tasks from reading a sharable file while it is being updated; another is to prevent simultaneous updating of the same file by two programs. Refer to Appendix L and to *LOCK keyword* parameters of the *DTFDA*, *DTFIS*, *DTFNI*, and *DTFSD* declarative macros.

file name

In OS/3 data management, the label of a *DTF* declarative macro must be a 7-character file name, designated *filename* in the format delineation. The first of its all-alphanumeric characters must be alphabetic; *filename* must be the same as used in the last preceding valid job control *LFD statement* for the *LFD statement* file. The *filename* is the logical name by which the user program addresses the file, for example, in the operands of data management *imperative macro instructions*. When a number of file names may be specified (as in the operands of the *GET* and *PUT* imperative macros), the maximum number is 16. See also *rename*.

filenameA

A 2-byte addressable field in the *DTFIS* file table, half-word aligned, in which data management keeps the number of *prime data* cylinders having full *overflow* areas. The *filenameA* field is set to zero if the user has not specified cylinder overflow with the *PCYLOFL* keyword. It is updated by the *ENDFL* macro which ends a file *load* or *extend* sequence, as well as by the *WAITF* macro following the *ADD* macro (or its equivalent: *WRITE,NEWKEY*) used to add new records to the overflow area of the *ASAM* or *ISAM* file.

filenameB

The *NOTE* imperative macro of the nonindexed file processor system returns the partition-*relative disc address* of the current block or record in a nonpartitioned *DTFNI* file, or in the first partition of a partitioned *DTFNI* file, to a 6-byte field of the file table, full-word aligned, and addressable as *filenameB*. Address is in discontinuous binary.

For sequentially processed *DTFNI* files, the *NOTE* macro returns this address in the form:

Obbbdd

where:

Obbb

Is the number of the current block, relative to the first block of the file.

dd

Is the displacement in bytes of the current record within that block.

For randomly processed DTFNI files, the form in which the NOTE macro returns the current record address is governed by the user's specification of the RELATIVE keyword parameter in the DTF:

If RELATIVE=R is specified, this form is:

rrrdd

where:

rrr

Is the relative record address of the current record.

dd

Is binary 0.

If RELATIVE=T is specified, the address is returned in the form:

Ottrdd

where:

Ottr

Is the relative track address of the current record.

dd

Is binary 0.

When the DTFNI file is multipartitioned, the corresponding fields of the *partition control appendages* to which the NOTE macro returns the current relative block address are accessible as *partitionnameB*.

filenameC

A user-addressable field in each DTF file table, full-word aligned, in which data management sets certain bits on the execution of each imperative macro to indicate to the user the detection of errors or of exceptions to the exact performance of the specified function.

In the DTFCD, DTFPT, and DTFPR file tables, *filenameC* is one byte long; it is four bytes long in the DTFMT, DTFSD, DTFDA, DTFNI, DTFMI, DTFIR, and DTFIS tables. *filenameC* is addressable by concatenating the character "C" to the 7-character *file name*.

Data management sets the error or status flags in this field before transferring control to the user's error routine, or to him inline. Refer to Appendix B for the significance of each bit in *filenameC*.

filenameD

1. In the nonindexed disc file processor system, a 2-byte addressable field of the DTFNI file table, half-word aligned, into which data management returns the displacement into its block of a blocked input record retrieved by the READ, ID imperative macro. Return is made after execution of the mandatory WAITF imperative macro. Displacement, measured in hexadecimal bytes, is right-justified in *filenameD*.
2. In paper tape data management, a 4-byte addressable field of the DTFPT file table, full-word aligned, that is loaded by data management with the address of an input record containing a parity error. Used only when the user has specified more than one I/O buffer but has not specified an I/O index register by which he can address the record. Refer to the IOREG keyword parameter of the DTFPT declarative macro. See also *double-buffering*.

filenameE

In all data management access methods, a 1-byte field (decimal byte 56) of each DTF that is loaded by data management with a hexadecimal code corresponding to the general error condition. This code is the same as the numeric component of the numbered data management error message issued to the log; for example, X'10' would correspond to error message DM10.

FilenameE is more useful to examine in a program dump listing than to access dynamically; it can be readily located by the address listed for it in the definitions dictionary or the allocation map.

filenameG

A 5-byte addressable field in the DTFIS file table, unaligned, into which data management loads the *relative disc address* from which the last record was retrieved. Loaded by WAITF macro after random retrieval functions. Content is in the same format as the ASAM/ISAM *record pointer*.

filenameH

A 5-byte addressable field in the DTFIS file table, unaligned, into which data management loads the *relative disc address* to which a record has been written in an ASAM or ISAM file, by the execution of an ADD, PUT, UPDT, or WRITE,KEY or WRITE,NEWKEY macro.

The field is loaded by the WRITE,NEWKEY macro itself in the *extend*, *load*, and *reload* operations, and by the WAITF macro which follows each of the output macros in the *add* and random *update* operations. The PUT macro loads *filenameH* in a sequential *update*.

The user may access this address, which is in the same format as contained in the ISAM/ASAM *record pointer*, and save it for later use in direct accessing.

filenameO

A 2-byte addressable field in the DTFIS file table, half-word aligned, in which data management keeps the total number of *overflow* records added to the ASAM or ISAM file. Updated by the WAITF macro instruction which must follow each ADD macro (or its equivalent: WRITE,NEWKEY) issued to the file.

filenameP

A 3-byte addressable field of the DTFIS file table, unaligned, in which data management keeps the total number of *prime data* records in an ASAM or ISAM file. Maintained dynamically by *load*, *reload*, or *extend* operation.

filenameR

A 2-byte addressable field of the DTFIS file table, half-word aligned, in which data management maintains a count of the number of retrieved *overflow* records that were not first in the chain.

filenameS

1. In OS/3 card SAM, an addressable field of the DTFCD file table into which data management places the image of a punched card containing a hole count error if user has specified the CRDERR keyword parameter and six successive attempts to punch have failed. *filenameS* is an 80-byte field for all modes but binary (image) mode, and a 160-byte field for that mode; it is full-word aligned.
2. In OS/3 ISAM, a 2-byte addressable field of the DTFIS file table, half-word aligned, into which data management places the number of bytes required to hold the *top index* of the ISAM file structure in main storage. May be accessed by user after file is closed. Not used in ASAM.

filenameT

A 2-byte addressable field of the DTFIS file table, half-word aligned, which data management maintains for the life of an ASAM or ISAM file, in which the user may keep his count of the number of records he has, according to his own conventions, tagged for *record deletion*.

file processing mode

The "direction" of processing a magnetic tape or sequential disc file (i.e., in the input or output processing mode); specified with the TYPEFLE keyword parameter of the DTFMT, DTFSD, or DTFNI declarative macro. The file processing mode can be reset, but only when a file is closed, by issuing the SETF imperative macro to it.

For DTFMT, DTFSD, and sequentially processed DTFNI files for which TYPEFLE=INOUT is specified, data management expects that the file will first be used for output when it is opened, and it sets the DTF file table accordingly. Therefore, if the user intends first to use an input file for *input*, he must issue the SETF imperative to reset its mode to input *before* he opens the file.

file sequence number

In OS/3 tape SAM, a field of the *HDR1*, *EOF1*, and *EOV1* labels; containing the relative position of this file with respect to the first file of the group. The first file is numbered 0001. Refer to Appendix E. Specified in the fifth positional parameter of the job control *LBL statement*; used by data management in positioning the tape to the correct file. See also *open-and-rewind options*. Contrast *file serial number*.

file serial number (FSN)

In OS/3 tape SAM, a field of the *HDR1*, *EOF1*, and *EOV1* labels; the content of this field (a 1- to 6-character alphanumeric string) must be the same as the *VSN* field of the *VOL1 label* for the first reel of the file or of a group of multifile volumes. Refer to Appendix F. Specified in the second positional parameter of the job control *LBL statement* for checking or creating by data management; refer to Table A-11. Contrast *file sequence number*.

file sharing

See *file lock*.

file space requirements

To calculate the number of cylinders required for data in an ASAM or ISAM file, the programmer may proceed as follows:

- Calculate *r*, the number of logical records per data block (*r* is an integer):

$$r = \frac{(\text{block size}) - 2 \text{ bytes}}{(\text{average record size}) + 5 \text{ bytes}}$$

- From this, calculate *b*, the number of *prime data* blocks required for the initial ISAM load:

$$b = \frac{\text{number of records to be loaded}}{r}$$

- Then calculate *d*, the total number of data blocks:

$$d = \frac{100 b}{100 - (\text{percent overflow})}$$

- Finally, convert *d* into *C_d*, the number of cylinders required for data:

$$C_d = \frac{d}{(\text{number of data blocks each cylinder can hold})}$$

See also *index space calculation*.

file trailer label group

In OS/3 standard-labeled magnetic tape files, 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, identical in format to the file HDR1 label; the second label is the EOF2 label. Its format is identical to the file HDR2 label. In the end-of-volume condition, these labels are the EOVI and EOVI2 labels; their formats are identical to their counterparts in the *file header label group*, HDR1 and HDR2. Refer to Appendix E for format and content, and to Appendix J for positions on standard-labeled tape volumes. Synonymous with *end-of-file label group*; *end-of-volume label group*.

file updating

See *file lock*.

first character control

In OS/3 printer SAM, a method for specifying *line spacing* or *skipping* when a printer file is printed, by including a control character in each record of the file and specifying in the DTFPR macro that all records contain this character.

fixed-sector disc

Preformatted by definition prior to file open, and not automatically by data management. Contrast with *variable-sector disc*.

format label

Any of a series of seven standard disc labels, contained in the *volume table of contents*, that describe a particular volume or the files residing on that volume. Refer to Appendix D. Any format label may be retrieved with the disc space management OBTAIN macro; refer to the supervisor programmer reference, UP-8241 (current version).

format 0 label

A disc format label that represents available space within the *extent* of the VTOC itself. When a format 1, 2, 3, 5, or 6 label is deleted from the VTOC, a format 0 label is written in its place. Contents: all binary 0's.

format 1 label

A standard disc label required for each file residing on the volume. It identifies the file and details certain of its characteristics, provides limited extent information, and (for ISAM) points to the *format 2 label*. In disc SAM, it points to the *format 3 label* if one is required.

format 2 label

A disc format label that provides additional description for a format 1 label. If present, a format 2 label is chained from a format 1 label.

format 3 label

An optional disc file label in OS/3 ASAM or ISAM which describes any disc extents of the file not described in the associated *format 1 label*.

format 4 label

A standard disc label which describes the *volume table of contents* (VTOC) itself.

format 5 label

A disc format label describing up to 26 *extents*, contiguous or noncontiguous, that are available on the volume. Always the second record in the VTOC. Several format 5 labels may be chained together, if required to describe more than 26 extents on the same volume.

format 6 label

A disc format label describing up to 26 *split cylinder* extents. Chained from the format 4 label; several format 6 labels may be used, chained together, if necessary to describe more than 26 split cylinder extents.

forms advance

Same as *paper advance*.

forms control

See *line skipping; vertical format buffer; 0768 printer; line spacing (2)*.

forms indexing

Advancing a printer form to a predetermined line, or by a predetermined number of lines. Controlled on the *0768 printer* by codes punched in a paper forms control loop, and on the *0770, 0773, and 0778 printers* by codes placed in the *vertical format buffer*. In OS/3, the data management user may control forms indexing on any of these printers using a buffer specified via the job control *VFB statement*. Refer to the OS/3 job control programmer reference, UP-8217 (current version), and to Appendix M.

forms overflow

In printer operations with OS/3 data management, the forms overflow condition occurs when a CNTRL or PUT imperative macro is issued that requires positioning the form to a print line beyond the line designated by the user as the lowest line on the form to be used for printing. This lowest print line is designated by coding it in the *vertical format buffer* (or on the *forms control* loop of the 0768 printer) with the appropriate forms overflow (OV) code; refer to Tables A—4 and A—5. OS/3 data management executes one user-issued CNTRL or PUT macro after the immediate CNTRL or PUT macro on which forms overflow is detected.

The PRINTOV keyword parameter of the DTFPR declarative macro is used to specify the action to be taken by data management when the forms overflow code is detected during a space or print-and-space command. The data management user has three options: He may specify an automatic skip to the *home paper* position; he may indicate that he is controlling overflow detection and action with the PRTOV imperative macro; or he may have data management transfer control to an overflow routine of his own coding. Because the forms overflow code cannot be detected on skip or print-and-skip commands, the user's overflow options will not be executed if the imperative macro executed after the macro on which overflow is detected results in forms advance to a skip code.

If the user opts to use the PRTOV imperative macro in his program, he must have loaded the proper forms overflow code in the VFB for the *0770, 0773, and 0778 printers*, using the OS/3 job control *VFB statement*, or have punched the overflow code in the paper forms control loop of the 0768 printer. If he is using the PRTOV macro with the *0770 printer*, each code required must be loaded in the VFB. Refer to the OS/3 job control programmer reference, UP-8217 (current version), and to Appendix K.

If the user provides his own overflow routine, data management transfers control to this without automatically advancing the form to the home paper position. In his routine, he may print total lines, skip to home paper, and print page headings. To branch back in his program to the point where processing would have continued if overflow had not occurred, he may use the address contained in general register 14. If he issues imperative macros (CNTRL, PUT) in his overflow routine, he should store register 14 before issuing them and restore it after macro execution.

After executing any carriage movement through an imperative macro instruction (PUT or CNTRL macro), the user should issue a PRTOV macro instruction before issuing another PUT or CNTRL macro.

The PRTOV macro instruction can be used with the 0770 printer to selectively check for a code 9 or code 12 forms overflow condition. When the PRTOV macro is issued, data management either skips to the home paper channel or branches to the forms overflow routine when it detects a forms overflow condition on the preceding print or space command. It does not recognize the forms overflow code during a skip operation.

FSN

File serial number.

H**hardware delete**

See delete character.

hardware error flag

Bit 3, byte 0, of *filenameC*. Set by OS/3 card SAM when six successive attempts to punch a card have failed. See also *filenameS*.

HDR1

In OS/3 magnetic tape SAM, the first label in the *file header label group* of a standard-labeled file; refer to Appendixes E and J.

HDR2

The second label in the *file header label group*; see *HDR1*.

Hollerith punched card code

Refer to Appendix I.

home paper

The home paper position is the line of the form where the first line of printed output is to be placed, usually near the top of each page of printout. The home paper instruction is issued by the user immediately before the command is issued to print this line, for example, after printing the last line to be printed on the preceding page. Refer to Table A—7 and to Appendix K. See also *line spacing* (1).

I**ID**

See relative disc address.

image code

Same as *column binary*.

imperative macro instruction

A macro instruction that generates machine instructions which may be executed subsequently to establish linkages or to perform processing.

The format of the file-processing imperative macro instructions included in the user program to communicate with data management transient routines and logical IOCS modules is:

LABEL	△ OPERATION △	OPERAND
[name]	XXXXXX	YYYY,...,ZZZZ

A symbolic name may appear in the label field and may have a maximum of eight alphanumeric characters, but it must begin with an alphabetic character. The operation field contains the mnemonic macro call. Positional parameters must be coded in the specified order in the operand field and must be separated by commas. Except for omitted trailing parameters, each parameter not coded must be represented by a comma. General registers may be specified to OS/3 data management imperative macros by coding the register number in the proper position, within or without parentheses.

independent overflow (IOF)

In OS/3 ASAM and ISAM, a special pointer that data management writes and maintains in the DTF file table and VTOC to identify the cylinder that is currently used to accept overflow records from other cylinders.

index

An ordered reference list of the contents of a file, with keys or reference notations for the identification or location of these contents.

index register

1. In all OS/3 access methods but DAM, the general register the user specifies with the IOREG keyword parameter in the DTF to be used to point to current data. Refer to Table A—1. See also *double-buffering*.
2. In printer SAM, data management loads the index register after each line is printed and before returning control to the user with one of the following, depending on record format:
 - The address where the first character of the next record to be output should be placed when fixed-length, unblocked records or undefined records are used. This is either a control character or the character in print position 1.
 - The address of the location where the 4-byte record-length field, followed by the control character (if any) and data to be printed, should be placed for variable-length, unblocked records.

index space calculation

To calculate the disc space required for the *block index* structure of an ISAM file, the programmer proceeds as follows:

1. Calculate r , the number of logical records per data block (r is an integer):

$$r = \frac{\text{blocksize} - 2}{\text{average record size} + 5}$$

2. Calculate b , the number of prime data blocks required:

$$b = \frac{\text{total number of records to be loaded}}{r}$$

3. Calculate e , the number of entries data management will make per index block (e is also an integer):

$$e = \frac{256}{\text{keysize} + 3}$$

4. The next calculation gives i , the number of index blocks the ISAM index structure will contain:

$$i = \frac{b}{e}$$

5. Dividing this by the number of index blocks each cylinder may hold (a constant depending on the disc subsystem) gives C_i , the number of cylinders the index will require:

$$C_i = \frac{i}{\text{number of index blocks per cylinder}}$$

These are the numbers of 256-byte ISAM index blocks that each cylinder may hold on the disc subsystems used by OS/3:

<u>SPERRY UNIVAC Disc Subsystem</u>	<u>Number of 256-Byte Index Blocks per Cylinder</u>
8411	100
8414	340
8415	120
8416	280
8418	280
8424	340
8425	340
8430	551
8433	551

See also *filenameS*; *file space requirements*; *top index*.

indexed random access method (IRAM)

An access technique within OS/3 data management which allows processing of logical records by relative record number in a random or a consecutive order or by key (indexed) in a random or a sequential order.

indexed sequential access method (ISAM)

An access technique within OS/3 data management which allows processing of logical records in either a random or a sequential order, through the use of an index. See also *alternate sequential access method (ASAM)*.

inline return

Register 14 contains the inline return address in OS/3 data management.

input/output control system (IOCS)

Generally, any set of macro instructions and routines that facilitate I/O operations. This term includes both the *physical input/output control system (physical IOCS)* and the *logical input/output control system (logical IOCS)*, or simply, OS/3 data management.

integrated card punch

Same as *0605 card punch*.

integrated card reader

Same as *0717 card reader*.

integrated peripheral channel (IPC)

In the SPERRY UNIVAC 90/30 Data Processing System, the IPC is an I/O channel which is an integral part of the processor. It coordinates all of the information transferred between main storage and its attached unit record subsystems. These include the following of interest to the OS/3 data management user:

0717 card reader

0605 card punch

0773 printer

0778 printer

integrated printer

Same as *0773* or *0778 printer*.

interlace

The record interlace feature, unique to OS/3 data management, is a data-mapping and tuning technique for increasing throughput with disc files by reducing the effect of rotational delay on record input and output. Appendix H describes the calculation of the *lace factor*, for specification in the DTFSD, DTFDA, DTFNI, or DPCA declarative macro. Not available in ISAM or ASAM.

interlock

See *rewind with interlock*.

intermediate index

In OS/3 ISAM, a component of the index structure which is generated by ISAM as need arises: usually when key length and file size are such that a *search on key* cannot reach the data-pointer level in the *block index* in two steps. See also *top index*.

interrecord gap

In paper tape files, the user may provide a certain fixed number of *null* or *delete characters* at the end of each record to help distinguish between them and aid in error recovery. He must punch these record gap characters on output and program to deal with them as necessary on input. On output files and in *binary mode* input files, any character to be placed by the user at the end of a record to serve as an interrecord gap must be included in calculating his block size and (for undefined records) in their record size. Refer to the BLKSIZE and RECSIZE keywords of the DTFPT macro.

For *character mode* (MODE=STD) input files, record gap characters are not transferred into main storage and consequently are not included in the BLKSIZE specification, nor (for undefined records) in the record length placed in the optional RECSIZE register by data management.

invalid ID

Error flag (byte 0, bit 1) set in *filenameC* by all disc access methods. Refer to Appendix B. ISAM sets this flag when the user addresses a record by a *relative disc address* that lies beyond the limits of the file. The nonindexed disc file processor system sets it under these circumstances when the user has not provided for file extension, when it is impossible to extend the file, or when, after extending a DTFNI partition by one *unit of store*, data management finds that the ID requested still lies beyond the new maximum relative block address. See also *extend*.

invalid macro/macro sequence

Error flag (byte 0, bit 6) set in *filenameC* by all OS/3 data management systems when the user issues an imperative macro that is invalid for the DTF or not supported by the logical IOCS processor configuration, or when the sequence of macros issued is invalid (including the failure to issue the WAITF imperative macro when required). Error message DM14 is also issued; refer to Appendix B.

invalid subfile

An error flag (byte 3, bit 3) set in *filenameC* of a DTFNI file table when the *subfile* number specified in a SETS imperative macro issued to an output file exceeds the maximum (71) or, to an input file, when the subfile entry in the DTF is zero (that is, SUBFILE=YES has not been specified). Error messages DM51 and DM52 are issued; refer to Appendix B.

IOCS

Input/output control system.

IOF

Independent overflow.

IPC

Integrated peripheral channel.

IRAM

Indexed random access method.

ISAM

Indexed sequential access method.

J**job**

A total processing application comprising one or more processing steps. Each job is divided into job steps, executed serially. With the exception of disc space, OS/3 allocates hardware resources to jobs; disc space is allocated to *job steps*. OS/3 is capable of concurrently processing a maximum of 7 jobs.

job step

The unit of work associated with one processing program; an executable program comprising one or more *tasks* and requiring a specific amount of hardware resources (e.g., control of the processor for a finite duration of time), or a specified amount of disc storage space. In OS/3, each job step may have as many as 256 tasks.

K**key**

1. In the nonindexed file processor system, a character string, unrelated to the disc location of a physical block, which the user specifies and writes with each block to uniquely distinguish it from other blocks in the search area of the file. May be used with DTFDA and DTFNI files; not used with DTFSD files. Minimum key length is 3 bytes; maximum is 255. No byte of the key may contain the hexadecimal value FF, nor may the key consist entirely of binary 0's.

In a DTFDA file, all keys must have the same length; in DTFNI files, all keys within the same *partition* must have the same length. Blocks without keys may not be included in a file of keyed blocks. All keys must be at head of block. Refer to the KEYARG and KEYLEN keyword parameters. See also *search on key*.

2. In OS/3 ASAM and ISAM, a user-specified character string, associated with each logical record to uniquely identify that record. Required for ISAM files; optional for ASAM. No byte of the key may contain the hexadecimal value FF, nor may the key contain all binary 0's. All keys in the same file must have the same length; the maximum key length is 253 bytes and the minimum is 3. All keys in a file must start at the same location in each record, but may be internal to the record. Refer to the KEYARG, KEYLEN, and KEYLOC keyword parameters. See also *dummy record*; *search on key*.

key argument

See *search key*.

key search

See *search-on-key*.

L

label processing

See *disc label processing*; *tape user label processing*; *tape volume label processing*.

label standard level field

In ASCII magnetic tape labelling practice, a field occurring in the VOL1 label. When writing ASCII output tapes, OS/3 data management writes "1" in this field to indicate that the tape is in full compliance with *American National Standard Magnetic Tape Labels for Information Interchange, X 3.27—1969*.

On input tapes, a "1" in this field ensures correct processing; tapes created under later versions of the ASCII standard may also be accepted, but the user is not assured of correct processing.

lace factor

In the OS/3 nonindexed file processor system, an integer specified with the LACE keyword parameter of the DTFNI, DPCA, or DTFSD declarative macro when creating a file with the record *interlace* technique, or accessing a file so created. The lace factor is determined by the programmer who creates the file; Appendix I provides the formulas for calculating it.

last block on track accessed

A status flag (byte 0, bit 0) set by data management in *filenameC* of the DTFDA, DTFIS, DTFSD, and DTFNI file tables after the execution of certain functions to these disc files has read or written a block or record in the last position on the current track. The user should check this flag and be prepared to take appropriate action in his program; he should test inline because control is not transferred to his ERROR routine by the setting of this flag.

In DTFDA output file tables, set by the WAITF imperative macro following the execution of the WRITE,AFTER macro when the record just written occupies the last record position on the current track. The user should be prepared to move to another track. Refer to the CNTRL or the WRITE,RZERO imperative macro.

latency

On disc, a time delay occurring between the positioning of a read/write head for a read or write operation and the arrival of the desired block or the track area sought under the head for that operation. Latency is a function of the rotational speed and the radius of the disc; it always exceeds the time delay for electronic head switching and is always less than *seek time*.

LBL statement

In OS/3 job control, the *file label information* statement; immediately precedes the *LFD statement* in the *device assignment set* for a standard-labeled magnetic tape file and is required if volume checking is to be done by data management. Refer to Table A—11.

The LBL statement provides data management with information required for writing and checking the *file header label group* and the *file trailer label group*. It has seven positional parameters; only the first, which specifies the *file identifier*, is required. Refer to the job control programmer reference, UP-8217 (current version).

LCB statement

In OS/3 job control, the *load code buffer* statement, by which the data management user may specify and load into the printer a unique buffer, overriding his installation's system standard default. If used, the LCB statement is included in a valid DVC-LFD *device assignment set* for the printer file to which it applies. With the LCB statement, the user may specify the type of printer for which the load code is intended and identify the printer cartridge to be mounted. He specifies the actual load code buffer in the sequence of the characters on the cartridge and may specify the *space code* and the *mismatch character*. He may also require reporting of *character mismatch* error conditions and specify *dualing characters*. Refer to the OS/3 job control programmer reference, UP-8217 (current version) and to Appendix K. See also *vertical format buffer; VFB statement*.

letter scan table

See *scan table*.

letter shift

See *shift character*.

level

See *data level*.

LFD statement

In OS/3 job control, the *logical file definition statement*; always required and the last statement in the *device assignment set* for every file. The first positional parameter of the LFD statement is the logical name of the file, by which the user's program refers to it; it is the same as the label of the DTF declarative macro instruction that defines the file. Note that, although OS/3 job control provides for an 8-character file name, data management requires that the *file name* not exceed seven alphanumeric characters, the first of which is alphabetic. Refer to the job control programmer reference, UP-8217 (current version). See also *file lock*.

line rate

Same as *line spacing* (1).

line skipping

Advancing the paper or form to be printed to a line on the form specified by a code the user places in the *vertical format buffer*, or by a punch the user makes in the paper forms control loop on the *0768 printer*.

In OS/3 printer SAM, the user may specify immediate or delayed skipping via the CNTRL imperative macro. Refer to Tables A—4 and A—5. See also *home paper*.

line spacing

1. The number of lines (6 or 8) printed per vertical inch on the form. The OS/3 data management user may specify line spacing with the DENSITY keyword parameter of the job control *VFB statement*, which loads the *vertical format buffer* for OS/3 printers. Refer to the OS/3 job control programmer reference, UP-8217 (current version) and to Appendix M. Line spacing on the 0773 and 0778 printers is also controlled through an operator switch. On the 0768 printer, the home paper code on the paper forms control loop sets the line spacing. Refer to Table A-5, Note 3. Synonymous with *density*; *line rate*.
2. Advancing the paper or forms to be printed a specified number of lines, under the control of a line counter. In OS/3 printer SAM, specified by the user through the CNTRL imperative macro or with the CTLCHR or PRAD keyword parameter of the DTFPR declarative macro. Refer to Tables A-4 and A-5.

line truncated

An error flag (bit 0, byte 0 of *filenameC*), set in the DTFPR file table by OS/3 printer SAM when printing of a truncated line is detected. See also *truncation* (1).

load

In OS/3 ASAM and ISAM, to load a file is to write the initial set of records into the *prime data* area of the file. The load sequence is initiated by the SETFL (set file load) macro instruction and terminated by the ENDFL (end file load) macro; the WRITE,NEWKEY macro is issued to transfer each logical record from the work area to the I/O area. The DTFIS keyword parameter IOROUT must be equated to LOAD, explicitly or by default, and (for an ISAM file only) the INDAREA and INDSIZE keywords must be specified. The PCYLOFL keyword is specified if the file is to be allocated overflow area.

Before issuing the WRITE,KEY macro, the user must preload a logical record into the work area, with its key (if the file is to contain keyed records) in ascending order of keys. Data management checks the key in the work area against the key of the last record transferred into the I/O area. If it is not greater, either a key duplication or a sequence check error has occurred; data management sets the appropriate flag in *filenameC* of the DTFIS file table and transfers control to the user's error routine or to him inline. The current record is not transferred to the I/O area, and the user may resume loading with the next logical record.

If the key of the current record is greater, the record and key are transferred from the work area to the I/O area, where data management blocks the records before writing them to disc. When the I/O area cannot accommodate the current record, data management writes a block to the prime data area on disc, and starts the next block with the current record.

When a block is written to an ISAM file, a key-pointer entry is formed for the *block index*. This action is omitted for an ASAM file, which has no index (the user having specified INDEX=NO in the DTFIS declarative macro).

Records loaded to an ASAM or ISAM file are chained in logical sequence by means of the 5-byte *record pointers* which data management writes in the dividers between them. After the initial load, each record-pointer contains the hexadecimal pattern F0 AA AA AA AA, which indicates that the next record in physical sequence is also the next in logical sequence.

Following each execution of the WRITE,NEWKEY macro, *filenameH* contains the 5-byte relative disc address to which the record has been written; the user may save this address during load and present it later for direct addressing. Similarly, *filenameP* contains the count of the total number of records in the prime data area.

When an ISAM file has been closed upon completion of the load operations, the user may access *filenameS* to ascertain the number of bytes required to hold the *top index* in main storage.

During load operations, data management monitors continuously to ensure that it does not exceed the prime data area allocated. If a load fills the prime space on one volume, data management seeks to continue the load on the succeeding volume, if there is one. If there is none, it seeks additional space on the current volume and notifies the user if there is no additional space available by issuing error messages DM41. Refer to Appendix B.

A file-loading operation may end with the final data block only partly full. When this occurs, data management stores the exact location of unused file space in the disc *volume table of contents* (VTOC) for later recovery and use as needed.

Contrast with *add; extend; reload; update*.

load code buffer

A buffer, located in the 0768, 0770, 0773, 0776, and 0778 printers, that allows the user to specify any 8-bit code for any graphical symbol on the print band or drum. In OS/3 job control, the load code buffer is loaded by the user via the *LCB statement*, which allows him to specify a unique load code and to override the default system standard at his installation. Refer to the OS/3 job control programmer reference, UP-8217 (current version), and to Appendix K.

load point

On magnetic tape volumes, the point at which the recording of information begins. See also *open and rewind options*.

lock ID

See *file lock*.

lockable file

See *file lock*.

logical end-of-file

All OS/3 data management files are terminated by a logical end-of-file (EOF) pointer. In the nonindexed file processing system, the EOF address is one block higher than the highest used relative block address and is partition relative; that is, the EOF address is the address (relative to the physical block that begins the file partition) of the block next in physical sequence to the last block into which the user has placed a data record.

In DTFSD files, the EOF address also indicates the logical end-of-volume (EOV) for the volume which is online. When the user issues the CLOSE macro on completing file creation, data management stores the address of the current block as the EOF/EOV address. On encountering the EOF address during file input, it transfers control to the user's end-of-file routine. Refer to the EOFADDR keyword parameter of the appropriate DTF. See also *end of data (EOD)*.

logical input/output control system (logical IOCS)

A comprehensive set of macro instructions and routines provided to allow a user to create, process, and maintain a data file on a *logical record* level. OS/3 data management, essentially, is a logical IOCS.

Contrast with *physical IOCS*.

logical record

A user-defined character string handled as an entity in storage and retrieval.

A collection of items, considered independently of their physical environment. A record considered from the standpoint of its content, function, and use rather than its physical attributes, that is, defined in terms of the information it contains.

M

maximum block size

In OS/3 tape SAM, the upper limit set by system requirements on block size (see Table C—5). The user's BLKSIZE specification in the DTFMT declarative macro may not exceed these limits, and must be four bytes less than the maximum for the device if he specifies *block numbering*. See also *disc track size*.

mismatch character

In a printer's *load code buffer*, a specification of the character which is to be printed when a data byte is transmitted which is not in the load code. Specifiable in OS/3 job control via the MISMCHAR keyword of the *LCB statement*. Refer to the OS/3 job control programmer reference, UP-8217 (current version), and to Appendix K. See also *character mismatch*.

move mode

Referring to processing in which logical records are not processed in an I/O area but are moved from or to a specified work area.

multijobbing

In OS/3, the concurrent scheduling, loading, or execution of more than one job at a time.

Contrast with *multitasking*; *multiprogramming*.

multiplexer channel

In the SPERRY UNIVAC 90/30 Data Processing System, a channel which provides I/O capability between the processor and as many as eight low-speed peripheral subsystems. Those of concern to the OS/3 data management user include:

0716 card reader
0604 card punch
0776 printer
0770 printer
0768 printer
0920 paper tape subsystem

MIRAM

Multiple indexed random access method.

multiprogramming

The concurrent execution of two or more programs. Contrast with *multijobbing*; *multitasking*.

multitasking

In OS/3, the concurrent processing of many *tasks* asynchronously, by switching processor control among two or more tasks on a priority or rotational basis. Contrast with *multijobbing*; *multiprogramming*.

MUX

See *multiplexer channel*.

N

nonbinary mode

See *character mode*.

nondirectory ISAM

Same as *alternate sequential access method (ASAM)*.

nonindexed file processor system

In OS/3 data management, a generalized *logical IOCS* which enables the user to create and maintain data files on disc devices and to process these files in sequential order, in a random order, or by a combination of sequential and random file processing techniques. The processor, defined through the NAMIO declarative macro, may be used with disc files defined by the DTFNI, DTFDA, and DTFSD declarative macros.

nonindexed ISAM

Same as *alternate sequential access method (ASAM)*.

null character

In paper tape files, the null character is represented by the sprocket or feed hole alone, with no other punches. Its usual representation in main storage is binary 0's (hexadecimal 00).

The null character is used to constitute the *paper tape leader* and *trailer*; it may also be used to denote an *interrecord gap*. It is transferred to main storage from input paper tapes only in *binary mode*.

O

open-and-rewind options

In OS/3 tape SAM, the options concerning rewinding tape reels at open time, specifiable by the OPRW and REWIND keyword parameters of the DTFMT declarative macro. Refer also to the CLRW keyword, which may be used to specify rewind action at file close or end-of-volume condition.

The basic default action taken by tape SAM when none of these three keywords is specified is to rewind the tape volume to the *load point* when the file is opened, and to *rewind it with interlock*, causing it to be unloaded, when the file is closed or the end-of-volume condition is reached.

The keywords are mutually exclusive: the REWIND keyword should not be specified if either of the others is. The REWIND keyword takes precedence in the event of multiple specification, and the others are ignored.

optional file

An input or output file which will not invariably be produced or required for processing by every program execution. Specified in the appropriate DTF by the OPTION keyword parameter. See also *optional file processing*.

optional file processing

In OS/3 card SAM, processing performed by data management when the OPTION keyword parameter is specified in the DTFCD declarative macro, depending on job control statements and device availability:

- For an input or *combined card file*, data management branches to user's *end of file* routine (EOFADDR), without reading cards, when a GET imperative macro is issued. User should close the optional file.
- For an output or *combined card file*, when a PUT or CNTRL imperative macro is issued, data management disables these and immediately returns control to the user program at the normal return point. No I/O is performed.

output parity check

See *parity check*.

OV

See *forms overflow*.

overflow

1. See *forms overflow*.
2. That portion of an ASAM or ISAM file into which data may be deposited but which is not considered a part of the primary data storage area, and which is normally used during the record insertion process. See also *add; cylinder overflow control record; independent overflow*.

overlap mode

In OS/3 card SAM, a method of processing cards in a *combined card* file, using the *read/punch* feature of the 0604 or 0605 card punch. Requires specification of the ORLP keyword parameter of the DTFCD declarative macro; offers three possible combinations of the GET and PUT imperative macros:

1. Alternating GET and PUT macros, with alternating prepunched and blank cards. Each GET macro applies to a prepunched input data card, and each PUT macro punches data into a blank card.
2. Multiple GET macros, issued between single PUT macros. Produces valid results when the number of GET macros corresponds to the number of prepunched data cards between the single blank cards referenced by the PUT macros.
3. Multiple GET and multiple PUT macros, used with multiple prepunched cards between multiple blanks. Produces valid results when the numbers of GET and PUT macros and the numbers of prepunched and blank cards are consistent throughout.

See also *stacker selection*.

overlap processing

In OS/3 disc SAM, the ability to read a physical record into one I/O area while retrieving records from another I/O area or a work area; conversely, to write a physical record from one I/O area while reading records into another I/O area or a work area. Refer to Table A-1. See also *double-buffering*.

oversized I/O buffers

In paper tape data management, the use of oversized I/O buffers for processing input or output files containing fixed, unblocked records with *shifted data* is recommended for increased efficiency.

When fixed, unblocked record format is specified, the user's block size specification equals the length of each logical record, but this length does not allow room for the insertion of *shift codes* by data management. To specify oversized buffers for input files allows data management to read in more characters than the fixed length specified; it removes shift codes and *delete characters* from the larger block until the length of compressed record thus formed in the buffer equals the specified block size. If it cannot form a logical record of this size from the data read into the buffer, it reads in more. If *double-buffering* via work areas is specified, data management moves the compressed record to the work area, the defined length of which need not exceed the specified block size.

For output files, the reverse takes place. Data management utilizes the additional space in the oversized buffer to insert the required shift codes into the fixed-length record the user has supplied, and continues to do so until the buffer is filled. When the buffer is full, it is written out to tape. If necessary, data management issues additional I/O orders until the entire logical record has been punched.

If oversized buffers are specified, but work areas are not, the user should avoid accessing data in an I/O area at a displacement greater than his block size specification. Refer to the OVBLKSZ and BLKSIZE keyword parameters of the DTFPT declarative macro.

P

padding

In *ASCII* magnetic tape practices, padding may be used (but only with the agreement of the interchange parties) whenever it is necessary or advisable to extend the recorded area of a block beyond the end of the last (or only) record in it. The *ASCII* circumflex character (5/14) is used for padding.

OS/3 magnetic tape SAM, however, provides no facility to specify that padding should be used to fill out unused space at block end; padding therefore cannot be written automatically for output files. If the user needs to read *ASCII* input files that contain padding, moreover, his block size specification must include the necessary space, and his program must take care of identifying and detaching the string of padding characters before processing his data.

In other systems, provision has sometimes been made to allow label blocks to be padded out, using any *ASCII* character. This is not possible in OS/3; if the labels in an *ASCII* input file are not exactly 80 bytes in length, they cannot be processed properly. The only exception to this rule occurs when BKNO=YES is specified: in this case, label length should be exactly 81 bytes.

Refer to *American National Standard Magnetic Tape Labels for Information Interchange, X3.27—1969*.

paper advance

Vertical movement of the form or paper in the printer either after printing or without printing. Refer to the CTLCHR and PRAD keyword parameters of the DTFPR declarative macro and to Appendix K. See also *line skipping*; *line spacing*; *vertical format buffer*. Synonymous with *forms advance*.

paper tape leader

If the optional tape spooler is used with the *0920 paper tape subsystem*, a tape leader at least three feet (360 characters) long must precede the first data character on an input tape. If the tape spooler is not used, the leader may be as short as 2 inches (20 characters). The user must either generate the leader by punching characters on output or splice a previously punched leader to his tape file. In *binary mode*, he may use only the *null character*; in character mode (MODE=STD) he may punch either the null or the *delete character*.

paper tape trailer

A tape trailer at least 12 inches (120 characters) long must follow the last data character on an input tape. This length prevents a false indication of a broken tape error (broken tape being indicated when data characters are under the read photodiodes and the end-of-tape switch is activated). The user must generate the trailer by punching characters or must splice a trailer to his tape. In *binary mode*, he may use only the *null character*; in character mode (MODE=STD), he may use either nulls or *delete characters*.

parity check

A process for verifying or determining the accuracy of data transfer that tests whether the number of 1's or 0's in the array of bits is odd or even.

In OS/3 ISAM, ASAM, and the nonindexed disc file processing system, data management performs parity check verification on output files only if the programmer has specified VERIFY=YES in the appropriate DTF. Verification of output records necessarily increases execution time for PUT or WRITE macros; the programmer weighs this overhead against the value to him of immediate notification of problems with his file. If a parity check is conducted and reveals an error, data management normally sets the *output parity check* error flag (byte 2, bit 2) in *filenameC* and branches to the user's error routine, or transfers control to him inline if no error routine is provided.

The ERROPT keyword parameter, however, is available for use in the DTFSD, DTFNI, and DPCA declarative macros, and also in the DTFMT declarative macro of OS/3 tape SAM, for specifying to data management certain actions it is to take when informed of an input or output parity error from which the physical IOCS has not been able to recover. This keyword is not available in OS/3 ASAM or ISAM.

In paper tape data management, parity generation and checking, even or odd, may be performed only in *character mode* and requires specific wiring of the *program connector* board on the *0920 paper tape subsystem*. Refer to Table B—2 for error flag and recovery actions when parity errors are detected on input files. See also *filenameD* (2).

partition

A subdivision of a DTFNI file, required to have uniform block specifications. OS/3 data management provides partition-relative addressing, and individual partition extension capabilities.

partition control appendage (PCA)

1. In OS/3 data management, a table generated by the DPCA *declarative macro instruction* that defines a single *partition* of a file defined by the DTFNI declarative macro. The PCA is an addendum to the DTFNI table.
2. In OS/3 data management, "PCA" is also a keyword parameter of the DTFNI declarative macro.
3. "PCA" is also the mnemonic for a declarative macro instruction of the OS/3 supervisor that defines a single partition of a file defined by the supervisor DTFPF declarative macro. Refer to the supervisor programmer reference, UP-8241 (current version).

partitionnameB

In the nonindexed disc file processor system, a 6-byte field of a DPCA *partition control appendage* to which the NOTE imperative macro returns the *partition-relative disc address* of the current block or record. The field is full-word aligned. When the DTFNI file is nonpartitioned, or when the user is processing in the first partition (PCA1) of a multipartitioned file, the corresponding field is accessed as *filenameB*; *partitionnameB* is used to access the address of the current record or block in the second and succeeding partitions (PCA2 through PCA7) only.

See the entry for *filenameB* in this glossary for details on the form in which the NOTE macro returns the address.

PCA

Partition control appendage.

physical block

Same as *block*.

physical input/output control system (physical IOCS)

In OS/3, a set of function-oriented supervisor modules resident in main storage at all times, which controls the dispatching, queueing, and interrupt processing for all I/O devices directly connected to the system. Physical IOCS schedules and monitors execution of channel programs, thus controlling the actual transfer of physical records (*blocks*) between the external storage devices and main storage; it also provides device I/O error recovery.

physical IOCS

Physical input/output control system.

physical record

Same as *block*.

pointer

A field containing a value for *direct addressing*. In OS/3 ASAM and ISAM files, data management introduces *record pointers* between records to maintain their logical sequence.

PREP

A parameter of the OS/3 job control *VOL statement* (specified within parentheses) by which the user may call on the dynamic *tape prep facility* of data management. Refer to Table A—11, note 3. See also *disc prep*.

prepunch read station

Same as *read/punch*.

prime data

In OS/3 ASAM and ISAM, the collection of user records stored on disc during *load*.

print position options

Optional features on the 0770, 0773, and 0778 printers. When installed, the maximum block size specifiable for DTFPR files is 169 bytes for the 0770 printer, 153 for the 0773 printer and 145 for the 0778 printer; these maximums may be specified if CTLCHR=DI and RECFORM=VARUNB are specified. If not installed, the maximum block sizes for the 0770, 0773, and 0778 printers are 141, 129, and 141 bytes, respectively. The 0768 and 0776 printers have no print position options; the maximum block sizes specifiable to DTFPR files using these devices are 141 and 145 bytes, respectively.

printer overflow

See *forms overflow*.

program connector

The *0920 paper tape subsystem* uses a printed circuit patch card, or plug board, with small patch wires with which the user makes appropriate connections for control of the subsystem for his program: this is the program connector board. It has a section for the paper tape reader and a section for the punch, and it is used only for processing in *character mode*; it is completely bypassed for *binary mode* processing.

By means of the program connector board, the user may specify the correspondence between holes punched in the *data levels* on paper tape and the bits of the byte in main storage that represents the character punched on tape. He may also wire the program connector board so as to specify one *delete* character and an *end-of-record stop* character to be recognized by the hardware in input records. A third use of the program connector board in character mode is to control the generation of parity signals or the checking of parity.

R**random access**

Same as *direct access*.

RDW

Record descriptor word.

read-backward processing

In OS/3 tape SAM, magnetic tape files may be read forward or backward; data management assumes that input files are to be read in a forward direction unless the user specifies READ=BACK in the DTFMT declarative macro. This assumption applies both to files for which TYPEFLE=INPUT is specified, explicitly or by default, and to those for which TYPEFLE=INOUT is specified if the user resets *file processing mode* with the SETF imperative macro. If the user specifies READ=BACK, data management assumes that the volume is not to be rewound before its labels are checked during the processing of the OPEN macro instruction. Because no VSN checking can be performed (as no VSN is present in the volume trailer labels), read-backward processing may not be specified for a multivolume file. Refer to Table A—11. See also *open-and-rewind options*.

read lock

See *file lock*.

read/punch

A special hardware feature, optional on both the 0604 and 0605 *card punch* subsystems, that permits the user program to read and punch cards in the same file on a single pass. Files processed in this manner are termed *combined card files*; they may be processed in *overlap* or *nonoverlap* mode. Synonymous with *prepunch read station*.

record deletion

In OS/3 ASAM and ISAM, the user may, as he updates his files, tag records by his own conventions for deletion when he decides to reorganize his file. There is no way to avoid retrieval of records tagged for deletion; therefore, the user must include in his program the coding necessary to detect the presence of this tag. If he uses *filenameT* to store his count of the number of records tagged, this information is available to him for the life of the file to aid in deciding when to reorganize it. Data management also stores his count in the *format 2 label*; refer to Appendix D. See also *reload*.

record descriptor word (RDW)

In variable-length record format, the first four bytes of each logical record, the first two bytes of which contain the length of the logical record.

record gap

See *interrecord gap*.

record insertion

In OS/3 ASAM and ISAM, new records are never actually inserted into a file, in the sense of being physically written to a position on disc between the records they logically relate to. This procedure, utilized in other ISAM systems, usually involves the track pushdown process, which OS/3 avoids. In OS/3, all ASAM and ISAM records remain in the original positions where first written to the *prime data* area by the file *load* or *extend* operation. The same is true of new records written to the overflow area in *add* operation. Records rewritten by an *update* operation are placed on disc in the same locations occupied by the former records and may not differ in length or key. It is by means of the *record pointer* that OS/3 ISAM produces the logical effect of insertion of records into one long orderly series.

record interlace

See *interlace*; *lace factor*.

Refer to Appendix H.

record not found

An error flag (byte 1, bit 3) set in *filenameC* of the DTFIS file table when the starting point specified by the user for a retrieval sequence to be initiated by the SETL macro cannot be satisfied. This occurs when the key or the *relative disc address* supplied by the user in the KEYARG keyword parameter of the DTFIS declarative macro lies outside the limits of the file. Refer to Appendix B. See also *dummy record*.

record pointer

In OS/3 ASAM and ISAM, a 5-byte pointer located in the dividers that occur at the end of each prime and overflow record, and containing in binary the 3-byte block number followed by a 2-byte displacement value for the associated record in the block. A record that is preceded by 125 other bytes in the block requires a displacement value of 125. After an initial *load*, each record pointer contains the hexadecimal pattern FO AA AA AA AA, signifying that the next record in physical sequence is also the next in logical sequence. See also *relative disc address*.

record transmission time

For a disc record of a given size, the product of record length, measured in bytes, multiplied by the data rate of the disc subsystem, measured in milliseconds per byte.

reentrant

Referring to code which does not modify itself nor contain storage which is modified: that is, is not self-modifying. *Shared data management* modules are reentrant code. Contrast with *serially reusable*. See also *common code*.

relative disc address (ID)

In the nonindexed disc file processing system, all block or record addressing in direct access DTFDA and DTFNI files is *relative*. The relative disc address, or ID, may be used in either of two forms selected by the user and specified with the RELATIVE keyword parameter of the DTFDA and DTFNI declarative macros: the relative record form of addressing, or the relative track form. Both forms use a 4-byte address.

In relative *record* addressing (RELATIVE=R), the 4-byte ID has the following construction:

rrrr

where:

rrrr

Is a hexadecimal number, right-justified in its 4-byte field, that represents the sequential position of the block or record, relative to the beginning of the current file or *partition*. The number of the first record of a file or partition is 1.

On the other hand, in relative *track* addressing (RELATIVE=T), the ID is constructed as follows:

tttr

where:

ttt

Is a 3-byte hexadecimal number, relative to the first track in the file or partition, of the track on which the record or block occurs. Track numbering begins with 001.

r

Is the sequential number of the block or record on this relative track; record numbering begins with 1.

Any disc address the user generates or provides to the imperative macros of the nonindexed disc file processing system must be in one of these two forms, and he must specify the form of addressing with the RELATIVE keyword. Conversely, when data management returns to him the ID of a record or block after the execution of an imperative macro, it will be in the form he has specified. Refer to the IDLOC and SEEKADR keywords of the DTFDA and DTFNI macros. See also *filenameB*.

OS/3 ISAM also uses relative disc addresses; however, in ISAM these are 5-byte addresses in the form:

bbbdd

where:

bbb

Is a 3-byte hexadecimal number representing the sequential position of the current block, relative to the first prime data block of the file; and

dd

Is a 2-byte hexadecimal number representing the displacement, measured in bytes, of the current record into this relative block. The first record in a block begins after the block header, at a displacement of 2 bytes into the block.

Relative disc addresses in this form are used in the 5-byte ISAM *record pointers*; and the user supplies addresses in this form to the READ, ID and SETL, ID imperative macros. Refer to the KEYARG keyword parameter of the DTFIS declarative macro. See also *block pointer*; *chaining*; *cylinder overflow control record*; *dummy record*; *filenameG*; *filenameH*.

In certain other data management systems, the term "record ID" denotes an absolute disc identification address: a 5-byte address containing the cylinder number, head number, and record number that defines the physical location of a record on disc. OS/3 data management does not use absolute disc addressing.

relative record address

See *relative disc address (ID)*.

relative track address

See *relative disc address (ID)*.

reload

In OS/3, an ASAM or ISAM file which has been reorganized may be reloaded without reformatting by specifying the RELOD parameter of the job control LFD statement in the *device assignment set*. This erases the *format 2 label* when the file, which has been offloaded by a sequential retrieval operation and reorganized, is again written to disc with an ISAM *load* operation. All DTFIS keyword parameters must be unchanged from their original specifications.

rename

In OS/3, the user is offered the capability for renaming his files by the RENAME imperative macro of the disc space management facilities of the supervisor. A file may not be renamed while it is open. Refer to the supervisor programmer reference, UP-8241 (current version).

residual space

When the user is outputting variable-length, blocked records to a magnetic tape file or to a sequentially processed disc file or partition, and is building these in an I/O area without a work area, he specifies the VARBLD keyword to designate a register into which data management loads the number of bytes of residual space remaining in the current I/O area. Data management updates the VARBLD register after each PUT macro is executed.

Before issuing his next PUT macro, the user must access the VARBLD register to test whether enough residual space remains to accommodate the current record. If so, he issues the PUT; if not, he first issues a TRUNC macro to write a short block to the device. His next PUT macro then starts off the next block in the I/O area with the current record. VARBLD keyword is used only with DTFMT and DTFSD macros and sequentially processed DTFNI files and DPCA partitions. The TRUNC macro resets the IOREG index register to point to the new current record address in the next available I/O area. Refer to Table A—1.

rewind with interlock

To rewind a magnetic tape volume (reel) past the *load point*; causes tape to be dismounted. See also *open-and-rewind*.

roundup

In files residing on the 8416 disc, records are written in fixed physical sectors 256 bytes; all data transfers must be in multiples of this length. When the user's data block size is not a multiple of 256 bytes, data management will pack into each data block as many complete records as will fit into the user-specified length. Before transferring the data block to the 8416 disc, it will round its length up to the next higher multiple of 256 bytes. The length of the I/O area supplied must therefore be a multiple of 256 bytes and may in fact be larger than the specified BLKSIZE.

rub-out character

See *delete character*.

S**SAM**

Sequential access method.

SAT

System access technique.

save area

Before issuing an imperative macro for processing any OS/3 data management file, the user should generally first load register 13 with the address of a 72-byte, labeled save area, full-word aligned, in which data management expects to save the contents of the general registers. As an alternative, the user may add to his program a 72-byte, labeled save area, aligned as described, and specify its address with the optional SAVAREA keyword parameter in the DTF declarative macro for each file the program will process. Only one save area is required per program. When data management does not encounter the SAVAREA keyword in a DTF, it assumes that the user has preloaded register 13 with the address of a 72-byte save area, aligned on a full-word boundary.

The storage area specified with the SAVAREA keyword parameter is often useful to examine in snaps or dumps of the user's program region. It comprises 18 full words, the first three of which are used by data management. Following these are displayed full words for the contents of 15 of the general registers, presented in the following order: 15, 14, 0, 1, 2, and so on, through 12. Register 13 is not included.

When the user specifies the SAVAREA keyword, register 13 is available to him for his own use. Since its contents are not displayed in the general register save area, if he wants to see them in a snap or dump, he must make special provisions for them.

scan table

In paper tape data management, the user codes scan tables, specifying their labels with the SCAN, FSCAN, and LSCAN keyword parameters of the DTFPT declarative macro, for data management to use in selecting input or output data for translation as "figures" or "letters", in character mode (MODE=STD).

The SCAN keyword parameter is specified, along with FTRANS and LTRANS keywords, for input files with *shifted codes*, to provide the label of an input file shift code scan table. Data management uses this scan table to identify the *figure shift code*, the *lettershift code*, those input characters that are to be translated, and those characters that are to be deleted. The user inserts certain nonzero entries in the positions in the SCAN table that correspond to the shift and delete characters, and places the hexadecimal value 00 in all other positions. Characters in the input data whose table entries are hexadecimal 00 are translated by his FTRANS or LTRANS table, according as they follow the *figure shift* or *lettershift code* on tape.

To produce output files with shifted codes in character mode (MODE=STD), the user must specify the labels of two scan tables via the FSCAN and LSCAN keywords, and must code a figure scan table and a letter scan table. The FSCAN table contains the letter shift code in each position that corresponds to a "letter" code and hexadecimal 00 in all others. The LSCAN table contains the figure shift code in each position that corresponds to a "figure" code and hexadecimal 00 in the rest.

In applying these tables to the user's output, data management submits his data as operand 1 and one scan table or the other in turn as operand 2 of the BAL *translate and test* (TRT) instruction. Output characters encountered whose positions in the scan table contain hexadecimal 00 are selected for punching on tape after the appropriate shift code is punched (and after translation, if a TRANS table is also specified). On the other hand, when an output character is encountered whose result byte in the scan table is nonzero, the scanning process stops. The appropriate shift code is punched, and the current character is then punched on tape after it (being translated if a TRANS table is specified). Scanning then resumes, data management having switched to the other scan table.

All three scan tables are therefore prepared in the form the assembler expects for operand 2 of the BAL *translate and test* (TRT) instruction. Refer to the assembler programmer reference, UP-8227 (current version). Because the assembler generates an EXTRN for the labels specified by the SCAN, FSCAN, and LSCAN keywords in expanding the DTFPT declarative macro, the user may assemble these scan tables separately from his coding containing the macro call.

scratch

OS/3 data management does not offer a facility for dynamically scratching or deallocating a disc file; this capability is provided by the disc space management SCRATCH macro of the OS/3 supervisor, which may be issued only while the file is closed. Refer to the supervisor programmer reference, UP-8241 (current version).

OS/3 job control also has an SCR statement useful for deallocating a file from a disc pack; refer to the job control programmer reference, UP-8217 (current version).

A tape volume may be designated as a scratch tape by coding SCRATCH as the first positional parameter of the job control *VOL statement* in the *device assignment set*. Refer to Table A—11 for the related actions of the OPEN transient. The file *expiration date* is used by data management to prevent inadvertent use of an unexpired file for scratch. Refer to Appendix E.

search argument

Same as *search key*.

search key

Data to be compared to specific parts of each item for the purpose of conducting a *search-on-key*. No key may consist entirely of binary 0's, nor may any byte of the key contain the hexadecimal value FF. In OS/3 ISAM, and the nonindexed disc file processor system, the user presents the search key to data management in a field of his DTF specified by the KEYARG keyword parameter. Synonymous with *search argument*.

search-on-key

1. In the nonindexed disc file processor system, a function performed by the READ,KEY macro on a DTFDA or DTFNI file (the WRITE,KEY macro does not conduct a search). Retrieves a record or, in a blocked DTFNI file, a block of records by searching for one with a key matching the key that the user supplies in the KEYARG field of his DTF. Search begins at the relative disc address preloaded into the SEEKADR field of the DTF and continues to the end of the track, unless the user has specified SRCHM=YES in the DTF, when it continues to end of cylinder.

If the search is successful, the data portion only is read into the current I/O area. If the search is unsuccessful, data management sets the *record not found* flag (byte 1, bit 3) in *filenameC* and either the *end-of-track* flag (byte 1, bit 6) or if SCRCHM=YES is specified, the *end-of-cylinder* flag (byte 1, bit 7).

2. In OS/3 ISAM, a function which permits random retrieval of keyed items in disc files by means of a hierarchy of 256-byte index blocks, formatted for hardware search. Not used in ASAM.

An ISAM search starts at the beginning of the *top index* with a hardware search that finds a block containing an entry pointing to another index track. When this is searched, a second index block is read; the second block usually contains the entry pointing to the desired data block in the data area of the file.

In some cases, when keys and the file are unusually large, one additional index-track search is needed to reach the data-pointer level in the index.

No data are transferred to main storage from the data area on disc until a hit in the *block index* has pointed to the data block sought. When this data block is read into main storage, data management searches there for the logical record the user has requested.

seek time

In disc operations, the amount of time required for the mechanical movement of a read/write head to reposition it from one track to another. Refer to Table C—4.

sequential access

Retrieval of records in the sequence that has been established by original load and subsequent insertion, if any.

sequential access method (SAM)

A group of programs which are part of OS/3 data management and which allow processing of records sequentially. OS/3 offers SAM for card, printer, magnetic tape, paper tape, and disc files.

serially reusable

Referring to code which is self-initializing. Contrast with *reentrant*.

SFT

Software facility specification.

sharable files

See *file lock*.

shift character

A code punched on paper tape before a character or group of characters to indicate which of two *translation tables* data management is to use for translate input data. Used only in *character mode* processing, shift characters permit the user to represent more characters on 5- to 7-level tape than otherwise possible. Data management automatically inserts shift characters in output records before punching them in paper tape, and automatically removes them, translating the intervening data, from input records before making them available to the user in his data area.

Shift characters are specified by the user via *scan tables*; he may use any two of the codes that can be punched in the number of data levels in his tape, except for the *null character*. One shift character is designated as the letter shift and is always punched on tape before those characters that represent "letters"; the other is the figure shift and usually precedes those characters on tape that represent "figures". (Not invariably, however; data management assumes that the first character of the *first record* on paper tape is either a "figure" or the letter shift code.)

"Figures" and "letters" may be any characters of the user's choice; they need not be numerics or alphabetic codes, respectively, and may include punctuation or other special characters.

The entirely arbitrary assignments are made by the user in his scan and translation tables.

The shift codes extend the length of logical records but are not included in the user's block size specification. Refer to the following keywords of the DTFPT declarative macro: BLKSIZE, OVBLKSZ, SCAN, FSCAN, LSCAN. See also *oversized I/O buffers*.

shift code

See *shift character*.

shift code scan table

See *scan table*.

skipping

See *line skipping*.

software delete

See *delete character*.

space code

In a printer's *load code buffer*, specified to give the character representation which is to be printed as blank. Specified in OS/3 job control with the SPACE parameter of the *LCB statement*. Refer to the OS/3 job control programmer reference, UP-8217 (current version), and to Appendix K.

spacing

See *line spacing*.

split cylinder allocation

Except for ISAM and ASAM files, disc files in OS/3 may be allocated on a split-cylinder basis; i.e., several member files may be included in the set sharing the same *extent* area. Allocations are made on the job control EXT (extent specification) statement in the *device assignment set*; they are controlled by the *disc space management* routines of the supervisor, which create and maintain appropriate entries in the *format 6 label*. A split-cylinder file may not have more than 13 extents on the same volume.

Split cylinder allocation minimizes access arm movement and is used if two or more related sequential disc files are to be accessed by the same program. A split cylinder set comprises a primary member and one or more subsequent members; each member may be specified as a percentage of the whole set. Refer to the job control programmer reference, UP-8217 (current version).

stacker selection

In OS/3 and SAM, the CNTRL macro allows the user to control stacker selection on the *0604 card punch* for output or *combined card files*. In processing a combined file, for example, he is enabled to read a card, process the data read from it, and then select an output stacker to receive the card, based on the result of his processing.

He issues the CNTRL macro immediately after the GET or PUT imperative macro that referred to the same card whose destination — either the normal stacker or the select (or error) stacker — is to be specified by the CNTRL macro. The automatic deflection of error cards to the select stacker of the 0604 card punch is not affected by issuing the CNTRL macro.

See also *overlap mode*.

standby processing

See *overlap processing*.

start-of-data (/S)

An OS/3 job control statement. OS/3 card SAM does not check for a start-of-data card. If the user desires, he may use the /S card as a card file identifier, for consistency; if he does so, his program should include a check for the presence of the card.

stop character

See *end-of-record stop*.

stub card read feature

An optional hardware feature which may be installed on 0716 and 0717 card readers to enable them to read 51- or 66-column stub cards. In OS/3 card SAM, when this feature is to be used on an input card file, the user must specify the appropriate form of the STUB keyword parameter in the DTFCD declarative macro defining the file; if he is using the 51-column cards, his I/O areas must be at least 52 bytes long. The keyword is not used for output or *combined card files*.

SU

See *system utility (SU) symbiont*.

subfile

Within each of the seven optional partitions of a DTFNI file, the user may create as many as 71 subfiles. The subfiles maintain the same characteristics as the partition; the user accesses these serially within the partition via the SETS imperative macro, having selected the partition with the SETP macro.

Data management reserves one track of the first volume of the file on which to maintain subfile tables, when subfiles are to be created by the user. The subfile tables, which may not be accessed by the user, are written by data management on the first track of the volume unless the user has optional *user header* or *trailer* labels for his file; in this case, the UHLs and UTLs are written on the first track, and the subfile tables on the second. The user must specify the SUBFILE keyword parameter in the declarative macros (DTFNI or DPCA) defining the partitions in question. See also *end of partition*.

system access technique (SAT)

In OS/3, a generalized IOCS comprising a set of supervisor programs which data management uses to interface with *physical IOCS* routines for processing tape and disc files. SAT controls record *interlace* completely and maintains a considerable measure of *device independence* for the data management user. In addition, SAT also includes a set of declarative and imperative macros for defining and processing tape and disc files without OS/3 data management. See also *tape system access technique* (tape SAT).

system utility (SU) symbiont

A generalized utility for use under OS/3, described in the system service programs (SSP) programmer reference, UP-8209 (current version) and in the handbook for operators, UP-8072 (current version). It offers 35 functions: 10 for disc files, 16 for magnetic tape, and 9 for card files. With it, for example, the data management user may obtain a listing of his disc files, or a printout of certain *VTOC* information, including (for an ISAM file) the number of bytes required to hold his *top index* in main storage. See also *filenameS*.

system code field

In OS/3 magnetic tape SAM, a 13-byte field in the *HDR1* and *EOF1/EOV1* labels of both EBCDIC and ASCII files. In writing output tapes, data management writes "OS3" left justified in the field; the remainder of the field is left blank (space-filled). This field is ignored in processing input tapes.

T**tape mark**

A special bit-pattern or control block recorded on magnetic tape, essentially indicating the boundary between files and file labels and also certain label groups. In OS/3 EBCDIC standard labeled, nonstandard labeled, and unlabeled tape volumes, the second of two tape marks following the data blocks of the last file on the volume signifies that no valid information follows.

On nonstandard-labeled input files, data management expects a tape mark between the last header label and the first data block. On unlabeled input files, a tape mark is expected before the data blocks and is *required* when the user specifies *read-backward processing*. When a tape mark is not found at a file boundary, data management issues error message DM60; the assumption is that the wrong file is mounted.

For nonstandard output files, the user may specify the TPMARK keyword of the DTFMT declarative macro to prevent writing of the tape mark that separates labels from data; when he does so, however, he assumes the responsibility for distinguishing between labels and data on input.

Refer also to the READ keyword parameter of the DTFMT declarative macro and to the CNTRL imperative. For locations of tape marks in OS/3 magnetic tape files, refer to Appendix J. See also *block numbering*.

tape prep facilities

In OS/3, two facilities exist for prepping magnetic tape volumes. Volumes of a standard-labeled file may be prepped dynamically, as a preliminary part of the job step creating the file, by tape SAM itself; the user specifies the PREP parameter in the job control *VOL statement* of the file's *device assignment set*. Refer to Table A—11, note 3, and to the job control programmer reference, UP-8217 (current version). The alternative is the stand-alone TPREP utility routine, which does not prep a tape dynamically; refer to the system service programs (SSP) programmer reference, UP-8209 (current version). See also *extend* (6).

tape SAT

See *tape system access technique* (TSAT).

tape system access technique (TSAT)

A generalized IOCS designed to provide a standard interface to the physical IOCS for the systems user of SPERRY UNIVAC tape subsystems. It provides block-level I/O for sequential tape files. Not a part of OS/3 data management, which provides record-level I/O, TSAT is documented in the supervisor programmer reference, UP-8241 (current version).

tape user label processing

In OS/3 magnetic tape SAM, the user may process optional *user header labels* (UHL) and *user trailer labels* (UTL) in a routine whose symbolic address he furnishes to data management with the LABADDR keyword of the DTFMT declarative macro. In a standard-labeled file, his UHL and UTL are limited in number to eight of each type; if he uses them, they must have the standard format and content described in Appendix E. In a nonstandard-labeled file, their number, format, and content are entirely at his option. Appendix J describes the location of UHL and UTL in OS/3 magnetic tape files.

ASCII magnetic tape labeling practice allows the user to have his own volume labels (user volume labels (UVL)). However, OS/3 magnetic tape SAM does not provide a facility for creating these on output ASCII tapes; if they exist in ASCII input tapes, they are accepted but bypassed.

When the user has specified *block numbering*, optional user labels must be 83 bytes long in standard-labelled EBCDIC input files, and 81 bytes long in ASCII files, to ensure correct processing.

When the user is creating his own header or trailer labels for a nonstandard-labeled output file, his LABADDR routine must place the length of each label in the two least significant bytes of register 0. Conversely, when reading user labels in a nonstandard-labeled input file, data management places the length of each user label in the two most significant bytes of register 0 before making it available to the user.

The user must specify the LABADDR keyword in order to create user labels for an output file or to process user labels on an input file; if he omits the LABADDR keyword in the DTF for an input file containing user labels, these labels are bypassed. When he specifies the LABADDR keyword, his label processing routine receives control after he issues each OPEN or CLOSE imperative macro to the file, or when he issues the FEOV imperative macro to force *end-of-volume* procedures on the current volume of a multivolume *output* file. Before passing control to the user's LABADDR routine, data management preloads general register 0 with a code indicating whether the file is being opened or closed, or whether end-of-volume has been reached (or forced). General register 1 is preloaded with the address of the I/O buffer for processing user labels.

For an input file, when the user's LABADDR routine receives control from the OPEN transient, register 0 contains the EBCDIC alphabetic character "O" in its least significant byte. Unlike some other systems, OS/3 data management does *not* read a user label into the I/O area before the OPEN transient gives control to the user's LABADDR routine. The user must issue the LBRET 2 macro to retrieve each user label. For an output file, the code preloaded by the OPEN transient into register 0 is the same; the user moves his first UHL into the I/O buffer whose address is contained in register 1.

When the user issues a CLOSE macro to an input file, his LABADDR routine receives control after the least significant byte of general register 0 has been loaded with either the character "F" (signifying that an end-of-file condition has been reached) or the character "V" (indicating end-of-volume).

The user's LABADDR routine does not receive control when he issues the FEOV imperative macro to force end-of-volume procedures on the current volume of a multivolume *input* file; in this event, all end-of-volume label checking is bypassed.

For an output file, the codes preloaded by the CLOSE transient are the same; the user moves his first UTL into the buffer whose address is contained in general register 1. See also *tape volume label processing*.

tape volume label processing

In OS/3 magnetic tape SAM, the OPEN transient performs a number of functions related to the system standard labels contained on each volume of a standard-labeled file. Table A—11 summarizes those functions having to do with checking information in these labels against information provided by certain job control statements. Appendix E describes these labels; Appendix J shows their positions in standard-labeled files.

When the user issues an OPEN imperative macro to an output 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* and, checking the *VSN*, saves this for use in writing or reading the *file header labels* (HDR1 and HDR2). It then positions the tape so that the volume labels are not destroyed, and no further volume label processing is performed.

If the output tape is not at the load point after the open-and-rewind options are executed, magnetic tape SAM assumes that it is positioned between the two ending tape marks of the previous file, or just prior to the HDR1 label of the existing file. In either case, no volume label checking or creation is performed.

For an input tape, the OPEN transient first executes the open-and-rewind options and then checks to see whether the tape 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 job control *LBL statement*, and no further volume label processing is performed. If the input tape is not at the load point after execution of the open-and-rewind options, tape SAM assumes 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 checking is performed.

When any volume label is encountered during the processing of a CLOSE macro instruction for an input file for which the user has specified READ=BACK, the label is bypassed without processing.

For input tapes, all fields up to and including the *system code* field in the HDR1 label are checked by the OPEN transient against the values specified in the job control *LBL statement*. If READ=BACK is specified, tape SAM bypasses the HDR1 label without processing it: unless the HDR2 label was created by OS/3, as indicated in the *system code* field of the HDR1 label, tape SAM ignores it on input tapes.

For output files, the tape must be positioned properly before the file may be opened. The OPEN transient checks the *expiration date* in the HDR1 label against the current or actual calendar date to determine whether the associated file has expired. If it has not, data management issues error message DM57, which may be overridden at the system console. If the file has expired, the tape is positioned so that the old HDR1 label is overwritten. The HDR1 label for the new file, set up from values specified on the LBL statement, is written on the tape.

When the user issues an OPEN macro instruction to an input tape file, with READ=BACK specified in the DTFMT declarative macro, the OPEN transient checks the fields in the EOF1 or EOVI label against the values specified in the LBL statement. This processing is similar to that of the HDR1 label.

task

A unit of work capable of competing with other tasks for control of the processor. Each *job step* has at least one task, and many have more. A maximum of 256 tasks can be specified by the OS/3 user.

top index

In OS/3 ISAM, that part of the index structure that is first to be searched. It contains keys and 3-byte pointers to the traks of the lower index levels. Unless an *intermediate index* is also needed because of unusually large key or file size, there is only one other part of the ISAM index structure: the *block index*.

The user may speed random retrieval or record insertion operations against his ISAM file by reserving adequate space in main storage with the INDSIZE keyword parameter and causing data management to load as much of his top index as will fit into this area in table form by specifying the INDAREA keyword. The following must also be specified:

- the KEYARG and KEYLEN keywords;
- IOROUT=ADD, IOROUT=ADDRTR, or IOROUT=RETRVE; and
- TYPEFLE=RANDOM or TYPEFLE=RANSEQ.

The INDSIZE, INDAREA, and KEYLEN parameters define the size and address of the index buffer and the amount of top index that is brought into main storage automatically when the file is opened. OS/3 ISAM determines the size of the top index table entries from the length of keys specified by the KEYLEN keyword, ensuring that the length of the INDAREA table accommodates at least one block of top index entries. If the user's INDSIZE specification is less than the minimum (KEYLEN+3 bytes), his attempt to load the top index is negated, and appropriate diagnostics are listed in his DTFIS macro expansion. ISAM trims any excess specification by rounding it down to an integer multiple of one block of top index entries. In loading the top index, ISAM begins with the last block (the entry containing the all-FF₁₆ high key) and works towards the beginning of the index.

A random function issued to the file begins with a search of that part of the top index that is in main storage; a hit here results in an access to the next level of index on disc. An unsuccessful search of the index in storage moves the search to the beginning of the top index on disc, but that part which has been copied into main storage is not searched again on disc. It is not possible to load the *intermediate* or *block index* into main storage.

If the ISAM file resides on an 8415, 8416, or 8418 fixed sector disc, the size of the top index is limited to two tracks; on the 8411, 8414, 8424, 8425, 8430, or 8433 disc, the maximum size is four tracks. These maximums are always reserved on disc; the number of bytes required to hold the actual top index in main storage may be obtained by accessing *filenameS*, reading the *format 2* label, or obtaining a VTOC print. See also *index space calculation*; *system utility symbiont*.

track capacity

Refer to Table C—4.

track pointer

In OS/3 ISAM, a 3-byte pointer located in the *top index* entries and containing in binary the block number, relative to the index partition, that describes the block which starts out each track to be searched. Not used in ASAM.

track pushdown

See *record insertion*.

track size

See *disc track size*.

transient

An OS/3 data management routine which is effectively nonresident but which may be retrieved automatically from auxiliary storage and assigned to an area of main storage for execution.

translation table

1. In OS/3 card SAM, the user may code a 256-byte table, assigning a bit-pattern to each position, to control translation of input or output card files to or from EBCDIC. He specifies *MODE=TRANS* and the *ASCII*, *ITBL*, or *OTBL* keyword parameters, as appropriate, in the *DTFCD* declarative macro for the file. His table requires no specific alignment.

Appendix I is provided to facilitate coding the translation table. On reading a byte or card column in the record to be translated, OS/3 card SAM places into the receiving byte of the user's I/O area the bit-pattern it finds in the position of his translation table that corresponds to the position which the bit- or hole-pattern to be translated occupies in the table of Appendix I.

2. In paper tape data management, the user may provide translation tables in order to work with a more convenient code, or one that comprises more individual characters, than can be punched in the number of *data levels* available in his tape. If he wishes to process paper tapes in *ASCII*, moreover, he must provide an appropriate translation table in his program.

If he does not provide a translation table, he must work entirely with the data characters that can be punched directly on the tape; although in *character mode* he may wire the *program connector* board to change the correspondences between data levels and the bits in the byte of main storage that represents each character, he cannot create more data characters to work with this way.

All translation tables used in paper tape data management must be prepared in the form the assembler expects as operand 2 of the BAL *translate* (TR) instruction; they show the correspondence between a character on tape and an 8-bit code in the user's data area. Refer to the assembler programmer reference, UP-8227 (current version). The user codes and labels his translation tables, and specifies the label of each with the TRANS or FTRANS and LTRANS keyword parameters of the DTFPT macro. Because the assembler generates an EXTRN pseudo-op code for each such label, the user may assemble his translation tables separately from the coding that contains his DTF call. Refer to *American National Standard Perforated Tape Code for Information Interchange, X3.30—1971*. See also *scan table; shift character*.

3. In order to process magnetic tape files in ASCII, the user automatically or specifically includes a 516-byte EBCDIC/ASCII translation table module provided by OS/3 data management. The module for specific inclusion is maintained in the system object library (\$Y\$OBJ) under the name DT\$ETA; it contains two translation tables for use by data management and SAT in processing.

For output files, all systems labels are written in ASCII; refer to Appendixes F and L. The user must present his data and optional user labels in EBCDIC; each EBCDIC character in his data must correspond to one of the 128 ASCII characters shown in Table I—1. The record length fields in variable-length records must be in binary (as for EBCDIC files). Data management translates this and EBCDIC-coded fields into ASCII before writing these out to tape.

In translating ASCII input files, data management assumes that the tape being read fully complies with the *American National Standard Magnetic Tape Labels for Information Interchange, X3.27—1969*, and that there is no mixture of character codes. (The ASCII V-format record, in which the 4-byte record-length field is coded in binary, is not supported by OS/3 magnetic tape SAM and will not be processed correctly.) Before passing data and optional user header and trailer labels to the user, data management converts them to the form it expects to receive for output: the user's data and labels are in EBCDIC; the record length fields of variable-length records in binary. User volume labels (UVL1 through UVL9) are bypassed if present; they are not translated or passed to the user. See also *padding; tape user label processing*.

truncation

Generally, a data transfer involving less than complete transfer — usually an error.

1. In OS/3 printer SAM, when data management detects that the execution of a PUT imperative macro has resulted in the printing of a truncated line, it sets the *line truncated* flag (bit 0, byte 0 of *filenameC*). However, it does not pass control to the user's error routine until after the printer file has been closed, by which time other truncated lines may also have been printed. The user may find it useful to test for this flag after each execution of the PUT macro and provide for a flag character to be printed with the next line to speed visual location of all truncation errors on his printout.
2. In OS/3 ISAM, a key may be truncated if the key length is not accurately specified in the DTF. See also *wrong length found*.
3. A TRUNC macro is used in magnetic tape SAM and the nonindexed disc file processor system to output a block when the current record will not fit without exceeding block size; the current record is used to start the next block. Refer to Table A—1. See also *residual space; wrong length found*.

TSAT

Tape system access technique.

U

UHL

User header label.

See also *disc label processing; tape user label processing.*

unique unit error

An I/O error variously defined according to the peripheral device involved; the error flag in bit 2, byte 0 of *filenameC*. Data management issues error message DM30 (VALIDITY CHECK ERROR):

1. In OS/3 card SAM, a *validity check error*; i.e., the occurrence of more than one punch in rows 1 through 7 of any column of a card. Refer to the AUE keyword parameter of the DTFCD declarative macro.
2. In OS/3 printer SAM, the *character mismatch* error. Refer to the UCS keyword parameter of the DTFPR declarative macro.
3. In OS/3 paper tape data management, a *parity check* error detected in reading an input file in *character mode*. Refer to Table B—2.

unit of store (UOS)

In the OS/3 nonindexed disc file access methods, that percentage of the secondary space allocation, specified in the job control *extent specification* (EXT) statement, which data management is to use each time the disc file in question requires dynamic extension. The UOS is specified to data management with the UOS keyword parameter in the DTF; if UOS=00 is specified, or the parameter is omitted, or a zero increment is specified in the EXT statement, no dynamic extension is possible. See also *extend* (4).

UOS

Unit of store.

update

1. In OS/3 ASAM and ISAM, to update a record in an existing file is to retrieve the record from disc (using the READ,KEY; READ,ID; or GET imperative macro) and to write a new record in place of the old with the UPDT macro, or its equivalent (WRITE,KEY), or the PUT macro. The user must not change the record length nor alter the record key in any way. Updating requires specification of IOROUT=ADDRTR or IOROUT=RETRVE in the DTFIS declarative macro; UPDATE=NO must not be specified.

In response to the UPDT macro, when processing is random, data management moves the updated logical record from the work area to the correct location in the I/O area, and the block is rewritten to disc from there.

When file processing is sequential, the PUT macro must be part of a retrieval sequence initiated by the SETL macro and must follow a GET macro instruction. The PUT macro instruction sets an indicator in the DTFIS file table to ensure that a write is performed before the present block is abandoned; the next block is not accessed until all logical records in the current block have been processed.

If the user has specified the use of a work area, the PUT macro must specify the address of that work area from which an updated record will be transferred to the I/O area. This need not be the same area as specified in the preceding GET macro instruction. If the user has specified WORKS=NO in the DTF, he must supply the address of the logical record in the index register when he issues the GET macro; in this event, data management assumes that he has updated the record in the I/O area.

If, in updating his file, the user has tagged records for deletion by his own conventions, he may update his count of records to be deleted by accessing *filenameT*; he may update *filenameT* before or after issuing the PUT or UPDT macro. Data management maintains this addressable field of the DTFIS file table for the life of the file as an aid to the user in deciding when file reorganization is beneficial.

Contrast with *add*; *extend* (1); *load*; *reload*. See also *file lock*.

2. In the OS/3 nonindexed disc file processor system, a DTFSD or sequentially processed DTFNI file or partition may be updated with GET and PUT imperative macros. TYPEFLE=INOUT must be specified in the DTF and either UPDATE=YES specified or the file-processing direction reset to UPDATE with the SETF imperative macro. A sequential disc file may also be extended when processed in the update mode; see *extend* (2). See also *file lock*.

user header label (UHL)

Optional in OS/3 nonindexed disc file processing system and magnetic tape SAM, which allow a maximum of eight UHLs. Not used in OS/3 ASAM or ISAM. Each UHL is an 80-byte block; refer to Appendix D or E for format and content. See also *disc label processing*; *tape user label processing*; *user trailer label*.

user program

A program (including OS/3 system programs) requiring the services of OS/3 data management.

user trailer label (UTL)

Optional in OS/3 nonindexed disc file processing system and tape SAM, which allows a maximum of eight UTLs. Not used in OS/3 ASAM or ISAM. Each UTL is an 80-byte block; refer to Appendix D or E for format and content. See also *disc label processing*; *tape user label processing*; *user header label*.

UTL

User trailer label.

See also *disc label processing*; *tape user label processing*.

UVL

User volume label: permitted in *ASCII* magnetic tape files but bypassed by OS/3 data management if present in input files. Not created on output. See also *tape user label processing*; *translation table* (3).

V

validity check error

Same as *unique unit error* (1).

variable-sector disc

In OS/3, the 8411, 84148, 8424, 8425, 8430, and 8433 *disc subsystems*. Preformatted automatically in the nonindexed disc file processor system at open time by data management unless user specifies the AFTER keyword parameter in the DTFDA or DTFNI declarative macro. Contrast with *fixed-sector disc*.

vertical format buffer (VFB)

A buffer in the 0770, 0773, 0776, and 0778 printers used to control *forms indexing* containing a location for each line on the form. (On the 0768 printer, a paper forms control loop augments the function of the VFB). The user may load a code in the VFB location corresponding to a specific line and, in his program, advance the form to that line by issuing a skip command and specifying the code. In OS/3 job control, the VFB may be loaded via the *VFB statement*. Refer to the OS/3 job control programmer reference, UP-8217 (current version), and to Appendix K.

VFB

1. *Vertical format buffer.*
2. Acronym for OS/3 job control *vertical format buffer statement*. See *VFB statement*.

VFB statement

In OS/3 job control, the *vertical format buffer statement*, by which the data management user may specify and load a unique buffer to override his installation's system standard default buffer. If used, the VFB statement is included in a valid DVC-LFD *device assignment set* for the printer file to which it applies. With the VFB statement, the user may identify his form and specify its length in lines, specify the type of printer and the line rate or *line spacing* to be used, and indicate which lines of the buffer are to contain specific *skip*, *home paper*, and *forms overflow* codes. Refer to the OS/3 job control programmer reference, UP-8217 (current version), and to Appendix M. See also *LCB statement*; *load code buffer*.

V-format ASCII record

See *translation table* (3).

VOL statement

In OS/3 job control, the *volume specification statement*, which the magnetic tape SAM user specifies for standard-labeled volumes only to:

- supply VSNs for checking by data management (Table A—11);
- specify *block numbering*;
- request data management to prep the tape volumes dynamically;
- specify the tape density and other mode characteristics for the volumes; and
- specify that a scratch tape volume is to be used (Table A—11).

The VOL statement is part of the *device assignment set* for a file; it immediately follows the DVC statement. Refer to the job control programmer reference, UP-8217 (current version). See also *extend* (6) and *tape prep facilities*.

volatility

An inherent characteristic of a file's usage: the rate or percentage of addition and deletion of its records. A volatile file is one having a high rate of record additions and deletions. A static file, on the other hand, is one with a low rate of change. Volatility affects choice of file organization because adds/deletes are more readily handled by certain methods of organization than others. Contrast *activity*.

volume 1 label (VOL1)

The OS/3 standard volume label in the system for disc and tape files. Data management retrieves the *volume serial number* from the VOL1 label and, for disc, the address of the VTOC.

VOL1

Volume 1 label.

volume checking

For the actions of the OPEN transient in checking VSN and FSN on standard-labeled magnetic tape files, refer to Table A—11.

volume label group

In OS/3 magnetic tape SAM, the volume label group comprises a single label, the *volume 1 label* (VOL1), used to identify the tape reel and its owner. Refer to Appendixes E and J, and to Table A—11.

volume prepping

See *disc prep; tape prep facilities; VOL statement*.

volume sequence number

The position of the current tape or disc volume with respect to the first volume of a sequential, multivolume file. Contrast with *volume serial number*.

volume serial number (VSN)

An entry in the *VOL1* label for tape or disc volumes representing the unique code assigned by the user's installation when the volume enters the system. Should appear visually on the reel or pack for operator identification. Refer to Table A—11. Contrast with *volume sequence number*.

volume table of contents (VTOC)

A table associated with a direct access volume, which describes each file on the volume. Contains *disc format labels*, retrievable with the disc space management OBTAIN macro; described in Appendix D. Refer to the supervisor programmer reference, UP-8241 (current version).

VSN

Volume serial number.

VTOC

Volume table of contents.

W**WAITF required**

Error flag in *filenameC* (byte 0, bit 7) in the nonindexed file processing system. Set by data management when the user fails to issue the WAITF imperative macro, when required, to a DTFDA file, or a randomly processed DTFNI file or partition. The WAITF must be issued following a READ or WRITE imperative macro and before issuing another imperative macro to the same file or partition. Error message DM14 is also issued; refer to Appendix B.

Although the WAITF imperative macro must also be issued under certain circumstances in ISAM and ASAM, there is no corresponding error flag in *filenameC* of the DTFIS declarative macro. If the ISAM user fails to issue the WAITF macro when required, upon his issuing another imperative to the file ISAM sets the *invalid macro/macro sequence* error flag and issues error message DM14. Refer to Appendix B.

wired stop

See *end-of-record stop*.

work area

An unaligned area in main storage reserved by the user program and specified as the location into which data management will present logical records for processing. In ASAM and ISAM, the user presents his records to data management in the work area.

write lock

See *file lock*.

wrong length found

An error flag (bit 5, byte 1) of *filenameC*, set by data management in the DTFIS, DTFSD, DTFDA, and DTFNI file tables when the residual byte count is nonzero after data transfer to or from disc.

Also set when the key or data length of the referenced block (specified in the count field on disc) does not match the specifications of the DTF or DPCA table. For output operations, data management will either truncate or pad with zeros. Error message DM25 is also issued; refer to Appendix B. See also *truncation*.

/*

End-of-data.

/s

Start-of-data.

0604 card punch

The SPERRY UNIVAC 0604 Card Punch Subsystem, connected to the *multiplexer channel*. Refer to Table C—2.

0605 card punch

The SPERRY UNIVAC 0605 Card Punch Subsystem, connected to the *integrated peripheral channel*. Synonymous with *integrated card punch*. Refer to Table C—2.

0716 card reader

The SPERRY UNIVAC 0716 Card Reader Subsystem; connected to the *multiplexer channel*. Basic read mode is *column binary* (image); also available are *compressed card code*, *EBCDIC*, and *ASCII*. *Validity check* hardware is an optional feature. Refer to Table C—1.

0717 card reader

The SPERRY UNIVAC 0717 Card Reader Subsystem; connected to the *integrated peripheral channel*. Hardware to detect *validity check* errors is a standard feature. Refer to Table C—1. Synonymous with *integrated card reader*.

0768 printer

The SPERRY UNIVAC 0768 Printer Subsystem, connected to the *multiplexer channel*. Standard print line of 132 print columns. *Line spacing* (6 or 8 lines per inch) is controlled through both a punched paper tape loop (*forms control loop*) and the job control *VFB statement*; each is required. Refer to Appendix M. *Paper advance* (up to 15 lines) may be accomplished after printing or without printing. See Table C—3.

0770 printer

The SPERRY UNIVAC 0770 Printer Subsystem, connected to the *multiplexer channel*. Standard print line of 160 print columns. *Line spacing* (6 or 8 lines per inch) and *paper advance* (up to 15 lines) are controlled through the *vertical format buffer* and may be accomplished after printing or without printing. See Table C—3.

0773 printer

The SPERRY UNIVAC 0773 Printer Subsystem, connected to the *integrated peripheral channel*. Standard print line of 120 columns; 132 or 144 by hardware option. *Line spacing* (6 or 8 lines per inch) set by operator using a switch. *Paper advance* (up to 15 lines), controlled by the *vertical format buffer*, may be accomplished after printing or without printing. Refer to Table C—3. Synonymous with *integrated printer*.

0776 printer

The SPERRY UNIVAC 0776 Printer Subsystem, connected to the *multiplexer channel*. Standard print line of 136 print columns. *Line spacing* (6 or 8 lines per inch) and *paper advance* (up to 15 lines) are controlled through the *vertical format buffer* and may be accomplished after printing or without printing. See Table C—3.

0778 printer

The SPERRY UNIVAC 0778 Printer Subsystem, connected to the *integrated peripheral channel*. Standard print line of 120 columns: 136 by hardware option. *Line spacing* (6 or 8 lines per inch) set by operator using a switch. *Paper advance* (up to 15 lines), controlled by the *vertical format buffer*, may be accomplished after printing or without printing. Refer to Table C-3. Synonymous with *integrated printer*.

0920 paper tape subsystem

The SPERRY UNIVAC 0920 Paper Tape Subsystem, connected to the *multiplexer channel*. Reads perforated paper tape of five, six, seven, or eight channels at a rate of 300 characters per second and punches paper tape at the rate of 110 characters per second; characters are spaced 10 to the inch. Tape parity generation or checking, as well as data bit and character manipulation, are performed in *character mode* according to the wiring of a *program connector* board; parity cannot be checked or generated in *binary mode*. May be configured as a tape reader, tape punch, or both. Simultaneous reading and punching may be performed but not on same tape in same pass, as input and output tapes take separate physical paths. Refer to Table C-6.



USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

(Document Title)

(Document No.)

(Revision No.)

(Update No.)

Comments:

Cut along line.

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD



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

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424



CUT

FOLD

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

(Document Title)

(Document No.)

(Revision No.)

(Update No.)

Comments:

along line.

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD



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

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424



CUT

FOLD