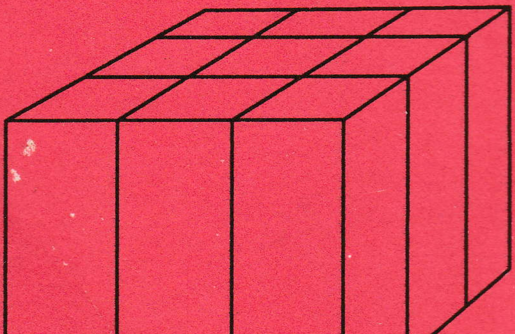
IBM

VSE/Advanced Functions

Diagnosis Reference
LIOCS Volume 1
General Information and Imperative Macros



VSE/Advanced Functions

Diagnosis Reference LIOCS Volume 1 General Information and Imperative Macros

Program Number 5666-301

Order Number LY33-9116-0
File No. S370/4300-30

First Edition (March 1985)

This edition applies to Version 2 Release 1 of IBM Virtual Storage Extended/Advanced Functions, Program Number 5666-301, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest _IBM System/370, 30XX and 4300 Processors Bibliography,_ GC20-0001, for the editions that are applicable and current.

PREFACE

This publication, although a 0-edition, is a major revision of the previously available publication LY24-5209-0.

This manual is the first in a series of four manuals providing detailed information about the VSE/Advanced Functions Logical IOCS programs. The four manuals are:

Volume 1: General Information and Imperative Macros, LY33-9116.

Volume 2: SAM, LY33-9117.

Volume 3: DAM and ISAM, LY33-9118.

Volume 4: SAM for DASD, LY33-9119.

This first volume is mainly intended for persons involved in program maintenance and for systems programmers who are altering the program design. The volume contains general information about Logical IOCS as well as descriptive text and flowcharts about commonly used transients. Included in this manual are:

1. The functions of logical IOCS, including a short description of the available access methods.
2. The modular-tabular system.
3. A short description of the declarative macros.
4. Complete description of the imperative macros.
5. File initialization and termination.
6. A detailed description of the open and close routines.
7. A detailed description of DASD file protect routines.
8. A detailed description of VTOC Display and Dump routines.
9. Charts.

In addition, this volume contains appendixes with information that is either supplementary to LIOCS or is an aid for information retrieval. To the first category belongs the EBCDIC - ASCII conversion tables. To the second category belong the error message list, master error message list, and master index.

Volumes 2, 3, and 4 contain information relating to all the logical IOCS components necessary to process the file types described within those books. Exception to this approach is found in those routines that are either common to more than one access method or independent of file types. These routines, which include the open and close monitor, DASD file protect, and VTOC routines, are documented in this volume.

PREREQUISITE PUBLICATIONS

- _IBM System/370 Principles of Operation_, GA22-7000, in conjunction with
- _IBM System/360 Principles of Operation_, GA22-6821.
- _OS/VS - DOS/VSE - VM/370 Assembler Language_, GC33-4010.
- _VSE/AF Data Management Concepts_, GC33-6192.
- _VSE/Advanced Functions Macro User's Guide_, SC33-6196.
- _VSE/Advanced Functions Macro Reference_, SC33-6197.
- _VSE/Advanced Functions System Control Statements_, SC33-6198.
- _VSE/Advanced Functions Diagnosis Reference: Supervisor_, LY33-9107.

RELATED PUBLICATIONS

- _VSE/Advanced Functions Diagnosis Reference: Initial Program Load and Job Control_, LY33-9110.
- _VSE/Advanced Functions Messages_, SC33-6098.

For other related publications, refer to _IBM System/370, 30xx and 4300 Processors Bibliography_, GC20-0001.

CONTENTS

FIGURES

CHARTS

INTRODUCTION

The transfer of data between storage and the input/output devices attached to a system is controlled by the Input/Output Control System (IOCS). IOCS allows the problem programmer to specify:

- What data has to be transferred.
- Which I/O device is to be used.
- In which sequence data transfer is to take place.

VSE/Advanced Functions gives the problem programmer a choice of two input/output control systems:

- Physical IOCS (PIOCS)
- Logical IOCS (LIOCS).

Full details on physical IOCS can be found in _VSE/Advanced Functions Diagnosis Reference: Supervisor,_ LY33-9107.

LOGICAL IOCS

LIOCS performs the data management function required to locate and access logical records for processing. Some LIOCS routines are linked and executed as a part of the user's problem program. Others, notably SAM DASD and DAM LIOCS routines, are provided by IBM, loaded into the System Virtual Area (SVA) at IPL time, and are dynamically linked to the user's program. They provide an interface between the user's file processing routine and the PIOCS routines. Some of the data management functions performed by LIOCS are:

- Blocking and deblocking of logical records.
- Switching between I/O areas when two areas are specified for a file.
- Handling End-of-File (EOF) and End-of-Volume (EOV) conditions.
- Issuing requests to PIOCS to execute the appropriate channel programs.

LIOCS makes use of two types of macro instructions to perform the required functions: _imperative_ macro instructions and _declarative_ macro instructions. Imperative macro instructions supply the facilities for reading, writing, blocking and deblocking, file labeling, and error checking. These instructions can be used only for data files that have been defined by declarative macro instructions. The declarative macro instructions specify the characteristics of a data file, such as the file name, I/O device type, or organization.

When LIOCS determines that a data area contains no logical record, it issues a physical IOCS macro instruction to execute the actual data transfer. Figure 1 on page 3 shows the relationship between logical and physical IOCS for a LIOCS imperative READ macro issued to an input file when one I/O area is used.

LOGICAL IOCS PROCESSING METHODS

Logical IOCS routines process records in any one of three ways:

1. Sequentially, through the use of the Sequential Access Method (SAM). This method can be used with all files on serial devices (such as card readers, tapes, and printers), and with sequentially organized files on disk and diskette.
2. Randomly, through the use of the Direct Access Method (DAM). This method can be used with files on disk only.
3. Both sequentially and randomly, through the use of the Indexed Sequential Access Method (ISAM) or the Virtual Storage Access Method (VSAM). These methods can be used with disk only. VSAM is available to VSE users through the VSE/VSAM program product.

Sequential Access Method (SAM)

Sequential processing reads/writes and processes successive records in a logical file. For example, card records are processed in the order the cards are fed; tape records are processed starting with the first record following the header labels and ending with the last record before the trailer labels. DASD records are processed starting with the beginning DASD address and continuing in order through the records on successive tracks and cylinders up to the ending address.

Diskette records are processed starting with the beginning diskette address and continuing in order through the records on successive tracks up to the ending address.

Volumes 2 and 4 contain a detailed discussion on sequential processing.

Direct Access Method (DAM)

The Direct Access Metho⁴ processes records contained on IBM disk devices that are usually organized in a random manner. DAM is a method for processing records rather than an organizational method.

The location reference required by LIOCS for processing a file in a random manner consists of two parts: a track reference and a record reference. The record reference may be the record key, or, if no key areas are present, the record ID which is in the count area of each DASD record. Volume 3 contains a detailed description of random processing through DAM.

Indexed Sequential Access Method (ISAM)

The Indexed Sequential Access Method can process records on a DASD device in a random and/or sequential order. Both orders use the control information that is in the key field of each record. The user supplies ISAM with the key (control information) of the desired record. ISAM searches for the record and makes it available for processing.

In sequential processing, a series of records is made available. The first record to be processed is specified by the user. ISAM retrieves the succeeding records (on demand) from the logical file, in key order, until the problem program terminates the operation.

ISAM creates an organized file and then adds to, reads from, and updates records in that file. The file is organized from records that are presorted by control information. As the DASD records are loaded, ISAM constructs indexes for the logical file. If records are added to the file at a later stage, ISAM updates the indexes to reflect the new records. Volume 3 describes ISAM in detail.

Virtual Storage Access Method (VSE/VSAM)

The Virtual Storage Access Method can process records on a DASD device. It differs from the access methods mentioned so far in that:

- It allows three different ways of data organization, each of which allows different ways of processing.
- It includes a facility for automatic space allocation.
- It includes a set of service programs that allow for the execution of a number of specialized functions.
- It allows ISAM files that have been converted to the VSAM format to be processed using ISAM macros.
- It offers device independence due to the special format of its physical blocks.
- It offers data integrity control and access control by means of design, and integrity and access control options.

In VSE/VSAM, a user may choose between three types of data organization:

- Key-sequenced data organization.
- Entry-sequenced data organization.
- Relative-record data organization.

In a key-sequenced organization, logical records are stored on the basis of a collating sequence determined by the content of the primary keys of those records. This key collating sequence is kept at all times. The key-sequenced organization is basically similar to the organization of an ISAM file.

Key-sequenced data organization allows for the following types of processing:

- Keyed-direct processing.
- Keyed-sequential processing.
- Addressed-direct processing.

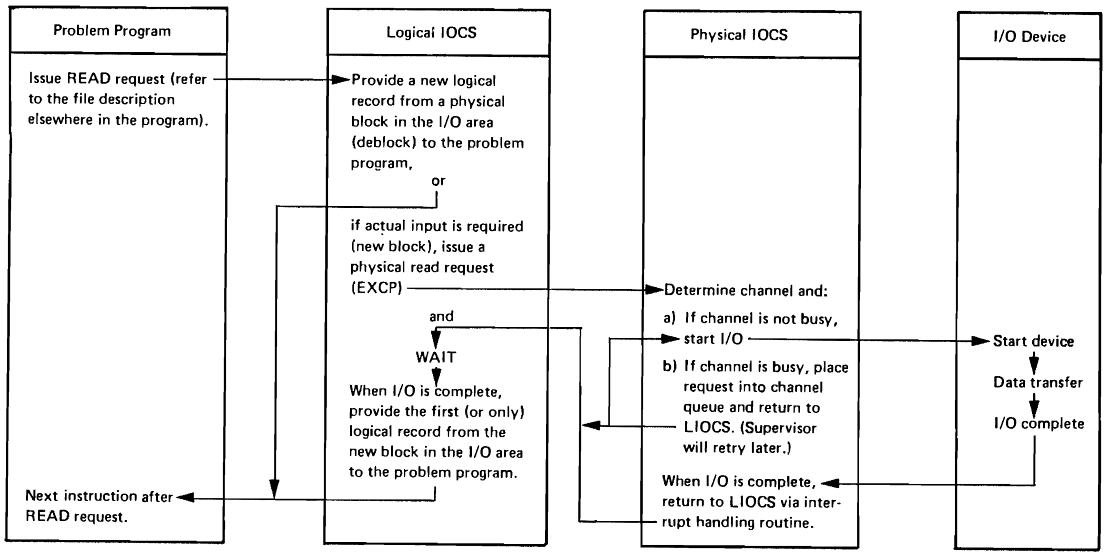

Problem Program
Issue READ request (refer to the file description elsewhere in the program).
Next instruction after READ request.
Logical IOCS
Provide a new logical record from a physical block in the I/O area (deblock) to the problem program,
or
if actual input is required (new block), issue a physical read request (EXCP)
and
WAIT
When I/O is complete, provide the first (or only) logical record from the new block in the I/O area to the problem program.
Physical IOCS
Determine channel and:
a) If channel is not busy, start I/O
b) If channel is busy, place request into channel queue and return to LIOCS. (Supervisor will retry later.)
When I/O is complete, return to LIOCS via interrupt handling routine.
I/O Device
Start device
Data transfer
I/O complete

Figure 1. Example of LIOCS and PIOCS Interrelationship

- Addressed-sequential processing.

In an entry-sequenced organization, logical records are stored physically in the same sequence in which they are entered. Newly added logical records are stored at the physical end of the file. This organization is basically similar to that of the SAM file.

Entry-sequenced data organization allows for the following types of processing:

- Addressed-direct processing.
- Addressed-sequential processing.

In a relative-record organization, logical records are stored in a string of fixed-length slots, each of which has a relative-record number, starting from one up to the maximum number of relative records that can be stored in the file. No index is built.

A slot may be empty or it may be occupied, in which case the record is identified by the number of the slot. For example, a record in the tenth slot of the file gets relative-record 10; it will always be the tenth record of the file regardless of whether or not records have been written into the preceding nine slots. A record is retrieved by its relative-record (that is, slot) number, the number being treated as a key.

Virtual and Basic Telecommunications Access Methods (ACF/VTAM and BTAM-ES)

VSE/Advanced Functions communicates with remote terminals with Advanced Communications Function/VTAM or Basic Telecommunications Access Method - Extended Support.

These processing methods are not documented, beyond an occasional reference, in this set of Diagnosis Reference Manuals. Specific information concerning ACF/VTAM and BTAM-ES is found in the ACF/VTAM and BTAM-ES publications.

Storage Requirements

Some logical IOCS routines are generated as part of the problem program, others (supplied by IBM) reside in the System Virtual Area and are dynamically linked to the user program. Imperative macro expansions, which serve as linkage to the logical or physical IOCS routines, are generated inline at the point the macro is used in the problem program. The open, close, EOF/EOV, and other special purpose routines are called into the B-transient (logical transient) area as required. The physical IOCS routines used by logical IOCS are generated as part of the supervisor program.

MODULAR-TABULAR SYSTEM

The term tabular and modular indicate that the system uses tables in conjunction with data handling modules to implement its functicns.

The modular-tabular system has the advantages of:

- Saving assembly time by allowing the data handling modules to be generated separately and to be stored in the relocatable library for subsequent use.

- Using one module with many files if the device types are the same and the files are similar.

The modular-tabular combination for a specific file is generated by two declarative macros: the file definition macros (DTFxx) and the module generation macros (xxMOD).

The <u>file definition</u> macros describe the logical file, indicate the type of processing to be used for the file, and specify storage areas (work area, I/O area) for the file. A number of file definition macros define the files processed by logical IOCS, and one defines files processed by physical IOCS (DTFPH). The file to be processed determines the type of file definition macro to be used.

The <u>module generation</u> macros generate the data handling logic modules. These modules contain generalized routines needed to perform the functions of the logical IOCS imperative macros. The generalized routines in the logic modules are altered and made more specific through various parameters (specified by the problem programmer) included in the xxMOD macro statements. It is possible, therefore, to generate many variations of a particular type of logic module, each specifically suited to the need of the problem programmer. For sequential DASD and DAM files, the data handling logic modules are provided by IBM. If the user provides a module in these cases, it is overridden by the IBM-supplied version.

<u>DECLARATIVE MACROS</u>

DTF (Define the File) Macros

Whenever logical IOCS imperative macro instructions are used in a problem program to control the transfer of records in a file, that file must be defined by a declarative DTF macro instruction. The DTF macro instruction describes (through various parameters specified by the problem programmer) the characteristics of the logical file, indicates the type of processing for the file, and specifies the main storage areas and routines. Figure 2 on page 6 summarizes the various DTF table types supported by VSE. Detailed descriptions of the logical IOCS file definition (DTF) macros and their parameters appear in <u>VSE/Advanced Functions Macro Reference.</u>

In general, the IBM-supplied file definition declarative macros are device-oriented. In addition, three declarative macros, DTFSR, DTFBG, and DTFEN are supported by VSE/Advanced Functions to provide upward compatibility from the IBM Basic Operating System (8K system). A brief description follows for each of the DTF macros available to users of VSE/Advanced Functions.

<u>DTFCD.</u> Define The File for a Card Device. To define a file associated with the records on a card unit or on the 3881 Optical Mark Reader.

<u>DTFCN.</u> Define The File for a CoNsole. To define a file associated with the console printer-keyboard (3210 or 3215) or with a Display Operator Console.

<u>DTFCP.</u> Define The File for a ComPiler. To provide limited device independence for IBM-written programs (COBOL, FORTRAN, PL/I). Because the DTFCP macro is written specifically to handle the needs of IBM internal programs, it is not documented in any System Reference Library publications.

<u>DTFDA.</u> Define The File for Direct Access method. To determine a file when DASD (Direct Access Storage Device) records are to be processed by the Direct Access Method.

<u>DTFDI.</u> Define The File for Device Independent system files. To define files assigned to the device independent system logical units SYSRDR, SYSIPT, SYSPCH, and SYSLST to provide DOS/VSE Assembler users with the same capabilities extended by DTFCP.

<u>DTFDR.</u> Define the File for the 3886 Optical Character Reader. To define a file associated with a 3886 Optical Character Reader.

<u>DTFDU.</u> Define the File for a Diskette Unit. To define a file associated with a 3540 Diskette Input/Output Unit.

<u>DTFIS.</u> Define The File for Indexed Sequential file management system. To define a file organized and processed by the Indexed Sequential File Management System.

<u>DTFMR.</u> Define The File for Magnetic Recognition. To define a file associated with a Magnetic Ink Character Recognition (MICR) device (1255/1259/1419) or Optical Reader/Sorter (1270-1275*).

<u>DTFMT.</u> Define The File for Magnetic Tape. To define a file associated with a magnetic tape device.

<u>DTFOR.</u> Define The File for an Optical Reader. To define a file associated with an Optical Character Reader device (1287).

<u>DTFPH.</u> Define The File for processing by PHysical IOCS. To define a magnetic tape, diskette, or DASD file with standard labels that is processed by physical IOCS when the user wishes to use the OPEN and CLOSE macros for label processing. DTFPH parameters define the magnetic tape, diskette, and DASD files. No other files processed by physical IOCS require definition.

* These devices are not available in the United States of America.

Only the following logical IOCS functions can be performed for files defined by a DTFPH macro.

- Check the header labels on input files, and close these files when requested.
- Create header labels on output files, and create trailer labels when the file is closed.
- Force end-of-volume on an output file when requested. (Force end-of-volume is not supported on diskettes.)

When a DTFPH macro instruction is encountered at assembly time, the assembler builds a DTF table that includes only the parameters needed for the OPEN, CLOSE, and FEOV routines. The OPEN, CLOSE, and FEOV macro expansions call the open and close routines into the supervisor B-transient area at object time.

DTFPR. Define The File for a PRinter. To define a file associated with a printer device, or a 2560 MFCM or 3525 Card Punch with the print feature.

DTFPT Define The File for Paper Tape. To define a paper tape file.

DTFSD. Define The File for Sequential DASD. To define sequential files on a Direct Access Storage Device (DASD).

DTFSR. Define The File in a SeRial type file device. To define a file for sequential processing of records on any IOCS supported I/O device.

The VSE DTFSR macro definition accepts either the BOS or BPS DTFSR macro as valid input. After determining the device type required, the VSE DTFSR macro calls, from the source statement library, the appropriate VSE DTF macro. The DTF macro called by the VSE DTFSR then sets up a DTF table in the usual manner.

The VSE macro definition is used only to allow upward compatibility and DTFSR should not be used as a statement in the user's VSE source deck.

DTFBG. The BeGin-definition must be punched with DTFBG in the operation field and DISK in the operand field. The name field is left blank. DTFBG is included in VSE to provide compatibility with the BOS DTFSR macro instruction.

DTFEN. Define The Field ENd. To show there are no more DTF source statements to process. Only to allow upward compatibility for BOS and BPS users.

ACB. The ACB macro produces an Access Method Control Block (ACB) for a VSE/VSAM file. The control block identifies the key-sequenced file and its index or the entry-sequenced file that is to be processed, and indicates the types of requests that are to be made. The ACB is similar to a DTF in that it identifies the file to be processed. However, most information about the file, such as key length and record format, is specified in the DEFINE command of the access method services. Information supplied in this command resides in the VSAM catalog and is read into storage when the ACB is opened.

DTF Type Code (Byte 20) of DTF Table	DTF	Description
X'00'	DTFCD	Combined files
X'01'	DTFPT	Paper tape files
X'02'	DTFCD	Reader and 3881 Optical Mark Reader files
X'03'	DTFCN	Console
X'04'	DTFCD	Punch files
X'05'	DTFCD	Reader files on 2560, 5424/5425
X'07'	DTFPR	Printer files on 2560
X'08'	DTFPR	Printer files
X'09'	DTFOR	Optical Reader files except 3881 and 3886 files
X'0A'	DTFOR	Optical Reader files (HEADER=YES)
X'0B'	DTFMR	Magnetic Ink Character Recognition (MICR) and Optical Reader/Sorter files
X'0C'	DTFDR	3886 Optical Character Reader files
X'10'	DTFMT	Unlabeled tape work files
	DTFCP	Unlabeled tape work files (compiler). (Note 1)
X'11'	DTFMT	Nonstandard or unlabeled tape files
X'12'	DTFMT	Standard labeled, output tape files
	DTFPH	Standard labeled, output tape files (physical IOCS)
X'13'	DTFMT	Standard labeled, input tape files (read backward)
X'14'	DTFMT	Standard labeled, input tape files (read forward)
X'15'	DTFMT	Standard labeled tape work files
X'1A'	DTFDU	Diskette Input/Output Unit files
X'20'	DTFSD	Sequential DASD work files and data files
	DTFCP	DASD work files (compiler)
X'21'	DTFPH	Sequential DASD files, MOUNTED=SINGLE (physical IOCS)
X'22'	DTFDA	Direct access files
X'23'	DTFPH	Direct access files, MOUNTED=ALL (physical IOCS)
X'24'	DTFIS	Indexed sequential, LOAD file
X'25'	DTFIS	Indexed sequential, ADD file
X'26'	DTFIS	Indexed sequential, RETRVE file
X'27'	DTFIS	Indexed sequential, ADDRTR file
X'28'	ACB	Access Method Control Block for VSE/VSAM
X'30'	DTFCP	Compiler file for DOS Version 1 (Note 1)
X'31'	DTFCP	Compiler file for DOS Versions 2 onward
X'32'	DTFCP	Compiler file for DOS Versions 2 onward (Note 2)
X'33'	DTFDI	Device independent system unit files
X'40'	DTFBT	Basic Telecommunications Access Method - Extended Support (BTAM-ES) file (Notes 3 and 4)
X'60' - X'67'		

Figure 2. DTF Table Types

Notes:

1. DTF type is X'30' except for tape or DASD assigned to units SYS000 to SYSnnn. In this case, the DTFCP open phases change the DTF type to X'10' for tape work files, or X'20' for DASD work files.

2. DTF type is X'32' except for DASD assigned to units SYS000 to SYSnnn. In this case, the DTFCP open phases change the DTF type to X'20' for DASD work files.

3. The following control unit codes are ORed into the low-order 4 bits of the DTF type code.

Control Unit	Code
7770	1
2848	3
2701	4
2702	5
2703	6

4. The DTF tables for BTAM-ES files are not documented in this manual. They are documented in the BTAM-ES publications.

MOD (Module Generation) Macros

Each DTF (except DTFCN, DTFPH, DTFSR, DTFDA, DTFSD, and DTFDI or DTFCP files residing on DASD and except DTFPR, DTFDI, DTFCP with DISK=YES for PRT1 and 3800 files, which use logic modules in the SVA) is linked to a logical IOCS module generated by an xxMOD macro instruction. These modules provide the necessary instructions to perform the input/output functions required by the problem program. For example, the module can read or write data, test for unusual input/output conditions, block or deblock records, or place logical records in a work area.

Some of the module functions are provided on a selective basis, according to the parameters specified in the xxMOD macro instruction. The problem programmer has the option of selecting (or omitting) some of these functions according to the requirements of his program. The omission of some of these functions results in smaller main-storage requirements for a particular module.

There are two options for MOD macros. The user can:

1. Insert the MOD macro instruction with its file parameters in the problem program source deck. In this case, the logic module is assembled in line with the problem program.

2. Choose to generate the logic modules needed for his file formats and system configuration. To do this, source decks using macro parameters to describe the file attributes are punched for each MOD macro statement. The logic module macro definition generates its own unique name, or the <u>user</u> can name the module in the name field of the MOD macro statement. The user name overrides the name the macro definition normally generates.

For each type of xxMOD macro, the problem programmer can generate, by issuing the macro with varying parameters for each required module, many logic modules. The logic modules must be cataloged in the relocatable library. The CATALR control cards are automatically generated when the module is assembled.

At assembly time, the Assembler produces an EXTRN (External Symbol) card for every V-type constant, or EXTRN statement, in the user program. At the time this program is link edited, the Linkage Editor resolves these EXTRN symbols. When these are resolved, the program is cataloged into the core image library, from which it is called for execution.

TRACK HOLD FUNCTION

The track (or control interval) hold function provides DASD track protection when the parameter HOLD=YES is specified in the operand of the module generation macro (DAMOD/ISMOD) and the DTFSD/DTFDA/DTFIS macro. If a task has previously accessed a DASD track and is currently modifying a record from that track, DASD track protection prevents another task in storage from accessing that track. The task attempting to access the held track is put in the wait state until the track has been released.

For DAM and ISAM, the problem program must issue the FREE macro to release a track held on READ operation. The module automatically holds and releases all tracks for WRITE operations.

For sequential DASD, the track is held and freed implicitly by the logic modules.

The track (or CI, for FBA) hold function is applicable to four situations:

1. Sequential DASD update files (data).

2. Sequential DASD work files with the UPDATE=YES parameter specified.

3. DAM files.

4. ISAM files.

REENTERABLE MODULES

A <u>reenterable</u> module is a logic module that can be used <u>asynchronously,</u> or shared, by more than one file. Including the RDONLY=YES parameter in the module generation macro generates a reenterable logic module. The RDONLY (read-only) parameter implies and assures, regardless of the processing requirements of any file(s) using the module, that the generated logic module is never modified in any way. To provide this feature, unique save areas, external to the logic module, are established, one for each task using the module. Each save area must be 72 bytes and double-word aligned. A task must provide the address of its unique save area in register 13 before an imperative macro is issued to a file and a logic module entered by the task.

The IBM-supplied logic modules used for DAM and sequential DASD (DTFCP, DTFDA, DTFDI, DTFSD) files are read-only and re-entrant, but do <u>not</u> require the user to provide a save area address in register 13 and will ignore such an address if provided. The same is true for logic modules used for tape support with DTFMT.
Reenterable modules include: CDMOD, CPMOD, DIMOD, DUMOD, and ISMOD

INTERRELATIONSHIPS OF THE DECLARATIVE MACRO INSTRUCTIONS

The DTFCD, DTFCP, DTFDA, DTFDI, DTFDR, DTFDU, DTFIS, DTFMR, DTFMT, DTFOR, DTFPR, DTFPT, and DTFSD declarative macros are similar in one respect. They each generate a DTF table that references an IOCS logic module. The first 20 bytes of each table have the same format; that is, a Command Control Block (CCB) and a logic module address. The remainder of each table is tailored to the particular device and file type.

When one of these DTF macro instructions is encountered at assembly time, the assembler builds a DTF table tailored to the DTF parameters. The table contains:

- Device CCB.
- A V-type statement used by the Linkage Editor to resolve the linkage to the logic module with this DTF. For DTFMT, the referenced logic module (IJJTCTL) processes only the CNTRL commands for not opened magnetic tape files. For DTFSD, DTFMT, DTFDA and, DTFPR/DI/CP if actual device is a PRT1 or 3800 printer, OPEN will dynamically fill in this field with the address of the IBM-supplied logic module. (Therefore for DTFSD and DTFDA zeros are generated.)
- Logic indicators; that is, one I/O area, two I/O areas, device type, and so on.
- Addresses of all (except work files) of the areas and controls used by this device.

Regardless of the method of assembling logic modules and DTF tables (that is, with the main program or separately), a symbolic linkage results between the DTF table and the logic module. Normally, the linkage editor resolves these linkages at edit time. However, for logic modules that support SAM and DAM files on DASD, the linkages are resolved at open time.

To accomplish the linkage between the DTF table and the logic module, the assembler generates a V-type address constant in the DTF table and a named CSECT in the logic module. To resolve this linkage, the linkage symbols (module names) must be identical. Figure 3 shows the relationship of the program (the imperative macro), the DTF, and the logic module. Imperative macros initiate the action to be performed on the file by branching to the logic module entry point generated in the DTF table. CRD is the name of the file; IJCFAOZO the name of the logic module.

IMPERATIVE MACROS

The problem programmer issues imperative logical IOCS macro instructions to initiate such functions as opening a file, making records available for processing, writing records that have been processed, controlling physical device operations, etc. Figure 4 on page 9 summarizes the macro instructions provided by IBM for logical IOCS. Figure 5 on page 11 further defines the general function of each of the macro instructions and indicates the devices with which they are used.

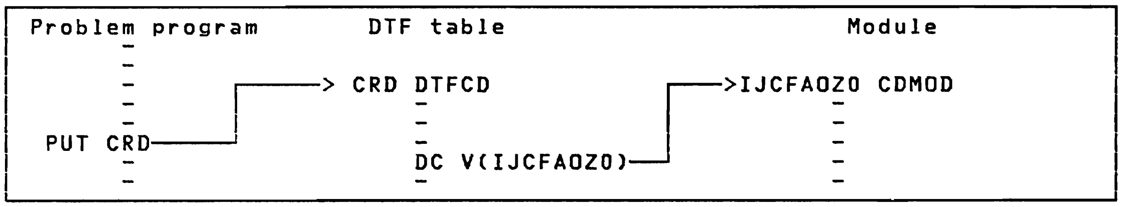


Figure 3. The Relationship Between Imperative and Declarative Macros

MACROS	DTFCD	DTFCN	DTFCP	DTFDA	DTFDI	DTFDR	DTFDU	DTFIS	DTFMR	DTFMT	DTFOR	DTFPH	DTFPR	DTFSD	DTFSR
CHECK									x	x[1]				x[1]	
CLOSE(R)	x		x	x	x	x	x	x	x	x	x	x	x	x	x
CNTRL	x			x		x		x		x	x		x[2]	x	x
DISEN									x						
DSPLY											x				
ENDFL								x							
ERET				x			x	x		x				x	
ESETL								x							
FEOV										x					x
FEOVD				x										x	
FREE				x										x[1]	
GET	x	x	x		x		x	x	x	x[3]	x			x[3]	x
LBRET				x						x				x	x
LITE									x						
NOTE										x[1]				x[1]	
OPEN(R)	x		x	x	x	x	x	x	x	x	x	x	x	x	x
POINTR										x[1]				x[1]	
POINTS										x[1]				x[1]	
POINTW										x[1]				x[1]	
PRTOV													x[2]		x
PUT	x	x	x		x		x	x		x[3]			x	x[3]	x
PUTR		x													
RDLNE											x				
READ				x		x		x	x	x[1]				x[1]	
RELSE										x				x	x
RESCN											x				

Figure 4 (Part 1 of 2). Logical IOCS Imperative Macros and DTFs

MACROS	DTFCD	DTFCN	DTFCP	DTFDA	DTFDI	DTFDR	DTFDU	DTFIS	DTFMR	DTFMT	DTFOR	DTFPH	DTFPR	DTFSD	DTFSR
SEOV										x					
SETDEV						x									
SETFL								x							
SETL								x							
TRUNC										x				x	x
WAITF				x		x		x	x		x				
WRITE				x				x		x[1]				x[1]	

[1].Work files only. [2].Not for 2560 work files. [3].Data files only

Figure 4 (Part 2 of 2). Logical IOCS Imperative Macros and DTFs

Macro Instruction		TYPE OF PROCESSING WITH LOGICAL IOCS																		
		Sequential															Indexed Sequential			
		3210/3215 Printer-Keyboards	1287 Optical Reader	1403/1443/3203/3211[13]/3800/ 3289-4, 5203 Printer, 2560, 5424/5425 with print feature	1419/1255/1259 Magnetic Ink Character Reader	1270/1275 Optical Reader/Sorter	3881 Optical Mark Reader	3886 Optical Character Reader	1442/2520/2540/3525 Punch 2560 MFCM, 5424/5425 MFCU	1442/2501/2520/2540/ 2596/3504/3505 Reader, 2560 MFCM, 5424/5425 MFCU	CKD/FBA DASD	3540 Diskette Input/Output Unit	Magnetic Tape Units	2671/1017 Paper Tape Reader	1018 Paper Tape Punch	Direct Access	Load File	Add Records	Random Retrieve	Sequential Retrieve
Initialize	OPEN(R)		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	LBRET[1]										X		X							
Process	GET	X	X		X	X	X[2]			X[2]	X	X	X	X						X
	PUT	X		X					X	X[4]	X[3]	X	X		X					X
	PUTR[11]	X																		
	READ		X		X	X		X			X		X						X	
	WRITE										X		X			X	X	X	X	
	CHECK				X	X					X		X							
	RELSE[5]										X		X							
	TRUNK[6]										X		X							
	WAITF		X		X	X		X								X	X	X	X	
	RDLNE		X																	
	RESCN		X																	
	DSPLY		X																	
Set Mode	SETFL																X			
	ENDFL																X			
	SETL																			X
	ESETL																			X
	SETDEV							X												
Non Data Operations	CNTRL[7]		X	X			X	X	X	X	X[12]		X			X				
	CHNG												X							
	PRTOV			X[10]																
	DISEN				X	X														
	LITE				X[9]	X[9]														
	ERET										X	X	X				X	X	X	X
Work Files for DASD and Magnetic Tape	READ										X		X							
	WRITE										X		X							
	CHECK										X		X							
	NOTE										X		X							
	POINTR										X		X							
	POINTW										X		X							
	POINTS										X		X							
Complete	CLOSE(R)		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	FEOV												X							
	FEOVD										X									
	FREE										X[8]				X					
	LBRET[1]										X		X							
	SEOV												X							

Notes:
1. Applies only if DTFSR, DTFMT, DTFDA, or DTFPH LABADDR or XTNTXIT is specified.
2. In the 2520 or 2540, GET normally reads cards in the read feed. If TYPEFLE=CMBND is specified, GET reads cards at the punch—feed—read station. For the 3881, the WORKNAME operand is invalid.
3. Put rewrites on input DASD records if UPDATE is specified.
4. In the 1442, 2520, or 2540, PUT punches an input card with additional information if TYPEFLE=CMBND is specified; PUT is specified by the 2560, 3525, and 5424/5425, if read/punch associated files are specified.
5. Applies only to blocked input records.
6. Applies only to blocked output records.
7. Provided only for upward compatibility for BPS and BOS.
8. Work files only.
9. Applicable to 1419 and 1275 with the Pocket Light Feature.
10. Not for 2560 or 5424/5425 with print feature.
11. Display Operator Console only.
12. CNTRL is treated as a no-op for FBA.
13. Applies also to 3211 compatable printers (with device type code of PRT1).

Figure 5. Logical IOCS Imperative Macros and Devices

IMPERATIVE MACRO EXPANSIONS

For each imperative macro issued by the problem programmer, the Assembler program generates an in-line expansion that links the instruction to the DTF table (and thus the logic module) for the specified file. The filename used for the DTFxx macro describing the file must always be an operand of the imperative macro instruction.

Typical expansions and brief descriptions of the function and procedure of each of the logical IOCS imperative macro instructions follow.

CHECK Macro

Label	CHECK	filename,PARAM*	
	L	1,=A(filename)	Loads address of DTF table.
	L	0,=A(PARM)	Loads address of control field. *
	L BAL	15,16(1) 14,8(15)	Loads address of logic module. Branch to CHECK routine in logic module.

* Optional

Function: This macro instruction forces the program to wait for completion of the I/O operation started by a READ or WRITE macro for the data file specified.

Procedure: This macro instruction waits for the completion of the input/output operation, started by a READ or WRITE, for the device associated with the filename. If the I/O operation is completed without an error or other exceptional condition, CHECK returns control to the next sequential instruction. If the operation results in an unusual condition (EOV, EOF, overflow, errors), CHECK processes the user's option specified in the DTF. Then, if the unusual condition is resolved, control returns to the user. Generally, if the unusual condition is not resolved, the routine posts a bit in some area set aside to indicate the condition, or issues a message to the operator on the system console printer.

CLOSE Macro

Label	CLOSE	FILEA,FILEB,...,FILEn	
	CNOP	0,4	
	DC	0F'0'	
	LA	1,=C'$$BCLOSE'	Initializes to fetch Close Monitor.
IJJCxxxx	BAL	0,*+4+4*(&N-1)	Register 0 points to the address of the DTF table for the first file in the file list. The second operand causes a branch to the SVC 2 instruction.
	DC	A(FILEA)	Start of file list.
	DC	A(FILEB)	(The file list contains the addresses of the DTF tables for all the files specified in the CLOSE macro operand.)
	.		
	.		
	.		
	DC	A(FILEn)	Address of the DTF table for the last file specified in the CLOSE macro operand.
	SVC	2	Fetches the Close Monitor, $$BCLOSE.

FILEn =	Symbolic address of the DTF table for the last file specified in the CLOSE macro operand.
N =	Sequence number of a file (1, 2, 3) in the order it appears in the CLOSE macro operand.
&N =	N of last file +1.

Function: The CLOSE macro instruction deactivates any file previously opened on any input/output unit in the system. The symbolic name of the logical file, assigned in the DTF header entry, is required in this instruction. Up to 16 files can be closed by one instruction by entering additional filename parameters. CLOSE is required whenever logical IOCS macro instructions have been used to transfer data, and the file has been previously opened.

Procedure: The CLOSE macro instruction calls the Close Monitor, $$BCLOSE, into the logical transient area to determine the device type assigned to the file.

for PRT1 or 3800 printers with DTFPR, DTFDI or DTFCP with DISK=YES CLOSE frees the DTF extension created by OPEN and indicates in the DTF that the file is no longer available for processing.

For other printers, card readers, card punches, CLOSE simply sets a bit off in the DTF table to indicate that the file is no longer available for processing. For magnetic tape, DASD, and MICR devices, the monitor calls the appropriate device-oriented close logical transient. For magnetic tape and DASD files, the CLOSE macro instruction causes trailer label processing for an input file, and trailer label creation for an output file, if necessary. If a magnetic tape file is being closed, the rewind option selected is executed. The file is deactivated by setting a bit off in the DTF table to indicate that the file is no longer available for processing.

For Diskette I/O Unit input files, the diskette is fed out and the file is deactivated. For output files, the HDR1 label is updated to reflect the proper end-of-data, the diskette is fed out, and the file is deactivated. The following table defines feed control:

	Input -- Programmer Logical Unit	Output -- Programmer Logical Unit	Input -- System Logical Unit	Output -- System Logical Unit
DTFCP	A	A	N	A
DTFDI	NA	NA	N	A
DTFDU	S	S	N	N
DTFPH	A	A	N	A

A -- always feed at close
S -- user can suppress feed at close
N -- never feed at close
NA-- not applicable

If physical IOCS is used, CLOSE is required only when standard labels are to be checked or written.

CLOSER Macro

Label	CLOSER	FILEA,FILEB,...,FILEn	
	CNOP	0,4	
	DC	0F'0'	
	B	*+8	
	DC	A(*)	Address used by CLOSER for relocation.
	LA	1,*-4	Loads actual location address.
	MVI	*-4,X'58'	Disable subsequent relocation.
	L	0,*-12	Loads relocation factor.
	SR	1,0	Finds displacement value.
	L	0,IJJCxxxx+N*4	Gets address of DTF table for file to be opened.
	AR	0,1	Adds displacement value.
	ST	0,IJJCxxxx+N*4	Returns new DTF table address to file list.
			(The three instructions listed are repeated for each file specified in the OPENR macro operand starting with FILEA.)
	LA	1,=C'$$BCLOSE'	Initializes to fetch $$BCLOSE.
	CNOP	0,4	
IJJCxxxx	BAL	0,*+4+4*(&N-1)	Register 0 points to the address of the DTF table for first in the file list. The second operand causes a branch to the SVC 2 instruction.
	DC	A(FILEA)	Start of file list.
	DC	A(FILEB)	(The file list contains the addresses of the
	.		DTF tables for all files specified in the
	.		CLOSER macro operand.)
	DC	A(FILEn)	Address of the DTF table for the last file in the CLOSER macro operand.
	SVC	2	Fetches Close Monitor, $$BCLOSE.

FILEn = Symbolic address of the DTF table for the last file specified in the CLOSER macro operand.

N = Sequence number of a file (1, 2, 3), in the order it appears in the CLOSER macro operand

&N = N of the last file +1.

Function: The CLOSER macro instruction deactivates files used by self-relocating programs.

Procedure: The CLOSER macro instruction performs its function in the same manner as the CLOSE macro.

CNTRL Macro

Label	CNTRL	filename,code, n¹, n²		1	2	3
	L	1,=A(filename)	Loads address of DTF table.	*	*	*
	MVI	23(1),code	Puts control code in the DTF table if delayed printer control.		*	*
	LA	0,code	Loads control code.	*		*
	L	15,16(1)	Loads address of logic module.	*		*
	BALR	14,15	Branch to CNTRL routine in logic module.	*		*

1. Instruction assembled if skip or space immediate is specified.
2. Instruction assembled if delayed skip or space is specified.
3. Instruction assembled if both delayed and immediate skip and space are specified.

Label	CNTRL	filename,code, n^1, n^2	
	L	1,=A(filename)	Loads address of DTF table.
	MVI	72(1),code	Puts command code for DMK, LMK, and ESP on a 3886 to the CCW in the DTF.
	MVC	*+11(1),fldname	Generated if code is ESP and n^2 is fldname. Move byte at fldname to second DC of parameter list (DC generated later).
	MVC	*+11(1),0(r)	Same, but n^2 is a register.
	BAL	0,*+6	Generated if n^1 is a number.
	DC	AL1(n^1)	Always generated if BAL is generated.
	DC	AL1(n^2)	Generated if code is LMK or ESP and n^2 is a number.
	DC	AL1(0)	Generated if code is LMK or ESP and n^2 is fldname or a register. The value is filled in by one of the MVC instructions described above.
	L	0,=A(fldname)	Generated if code is DMK or LMK and n^2 is fldname.
	LR	0,r	Same, but n^2 is a register other than register 0.
	L	15,16(1)	Loads address of logic module.
	BALR	14,15	Branch to CNTRL routine in logic module.

CNTRL expansion for the 3886 Optical Character Reader.

Function: The CNTRL (control) macro instruction provides commands for these input/output units: magnetic tape units, card reader-punches, punches, DASD, printers, and 3881 and 3886 optical readers. Commands apply to physical nondata operations of a unit and are peculiar to the unit involved. They specify such functions as rewinding tape, stacker selection of cards and documents, line spacing on a printer, etc. When a CNTRL macro is executed, the routine waits for completion of the operation before returning control to the user. On DASD, however, control returns at channel end.

Whenever CNTRL is to be issued in the problem program, the DTF entry CONTROL=YES must be included in the file definition (except in DTFDR and DTFMT).

The CNTRL macro instruction <u>must not</u> be used for printer or punch files, if the data records contain control characters and the entry CTLCHR= is included in the file definition (DTF) macro.

The CNTRL macro may also be used to process sequential DASD (DTFSD TYPEFLE=WORK,RECFORM=FIXUNB) work files backwards. The Backspace (BSL) function is invoked as follows:

```
          CONTRL {filename|(1)},BSL
```

<u>Operands</u>

filename (1)	The name of the file specified as a symbol or in register notation. It must be the name of the file specified in the DTF.
BSL	Mnemonic code for backspace.

As a result of the function code BSL, LIOCS sets the current position pointer back to the previous sequential record, unless one of the restrictions below applies. (i.e. assume record n has been handled by the last request, then the positioning of the file after BSL will be the same as after a POINTR or POINTW to record n-1, unless one of the restrictions below applies.) BSL out of the End-of-Extent routine specified by the EOXPTR parameter results in a POINTR or POINTW to the last record successfully written. The BSL function code is the only one recognized, all others will be ignored.

Violation of the following processing restrictions for BSL will result in a return code in register 0:

Return code 8:
For non-control-interval-format CKD files
- BACKSPACE cannot cross EXTENT-limits or boundaries. The file was already or is now positioned at the beginning of an EXTENT (not 1st EXTENT).

Return code 4:
For all files
- A BACKSPACE request was issued and the pointer was already

positioned at the beginning of the file, for instance immediately after OPEN or after POINTS, or the pointer is now positioned at the beginning.

Restriction: A WRITE UPDATE should not follow a BSL with return code 4 or 8, task gets cancelled, if the file is positioned at the beginning of an extent (POINTS like situation).

Procedure:;The control routine waits for completion of any previous operation of the file. Then the device symbolic address is moved to the CCB. The command code is moved to the CCW, and the CCB address is loaded into register 1. Next an SVC 0 is issued to perform the control function indicated by the CNTRL macro instruction. Then control returns to the problem program. CNTRL is treated as a no-op for sequential (DTFSD) files, unless BSL (backspace one logical record) and both RECFORM=FIXUNB and TYPEFLE=WORK are specified.

DISEN Macro

Label	DISEN	filename	
	L	1,=A(filename)	Loads address of DTF table.
	L	15,16(1)	Loads address of logic module.
	BAL	14,12(15)	Branch to DISEN routine in logic module.

Function: The DISEN (disengage) macro stops the feeding of documents through a magnetic ink character reader (MICR) or Optical Reader/Sorter.

Procedure: The DISEN macro modifies the instructions in the CCW chain and sets the disengage bit (bit 0 of byte 21) in the DTF table. Control returns to the problem program at the next sequential instruction following the DISEN macro expansion without waiting for completion of the disengage operation.

DSPLY Macro

Label	DSPLY	filename,r^1,r^2	
	L	1,=A(filename)	Loads address of DTF table.
	MVC	88(8,1),0(r^2)	Puts Load Format CCW for document coordinates of field to be displayed in DTF table.
	MVC	96(16,1),0(r^1)	Puts Load Format CCW for document coordinates of reference mark for field to be displayed in DTF table.
	L	15,16(1)	Loads address of logic module.
	BAL	14,20(15)	Branch to DSPLY routine in logic module.

Function: The DSPLY macro displays a specific field on the display scope of the IBM 1287 Optical Reader for entering the field from the keyboard. The DSPLY macro should be used in Document Mode only.

The macro requires three parameters, none of which can be omitted. The first parameter is the symbolic name of the 1287 file as specified in the DTFOR header entry. This parameter may also be a register that contains the address of the file. The second parameter must be a register that contains the address of the CCW defining the coordinates of the field to be displayed. The third parameter must also be a register that contains the address of the landmark defining CCW.

Procedure: If the reader cannot scan a complete field due to specific characters or fields running together, the field containing the error is retried by PIOCS. If still unsuccessful, the user is informed of the condition via his error correction routine (specified in the DTFOR COREXIT entry). The DSPLY macro is then issued to display the field in question on the 1287 display scope. The operator can then key in the correct characters. If an error is made in keying in the characters, the operator should press the cancel key and then the enter key, and the field will be redisplayed.

ENDFL Macro

Label	ENDFL	filename	
	L	0,=A(filename)	Loads address of DTF table.
	LA	1,C'$$BENDFL'	Loads Address of B-transient phase name.
	SVC	2	Fetches phase $$BENDFL.

Function: The ENDFL (END File Load mode) macro instruction ends the ISAM mode initiated by the SETFL macro. The name of the file that has been loaded is the only parameter required, and must be the same as the name specified in the file definition (DTF) macro.

Procedure: The ENDFL macro instruction performs a close operation for a file that was just loaded. It writes the last block of data records, if necessary, and then writes a DASD end-of-file record after the last record written. The EOF record is a DASD record with a data length of zero. The routine also updates the index entries as required, and writes dummy index entries for the unused portion of the prime data extent. Control then returns to the problem program.

ERET Macro

Label	ERET		
	B	0(14)	If operand is SKIP.
	B	4(14)	If operand is IGNORE.
	B	8(14)	If operand is RETRY.

Function: The ERET (Error RETurn) macro returns control to a logic module from an error routine in the problem program when ERREXT=YES is specified in the DTF macro. The choice of one of the three operands provided (SKIP, IGNORE, or RETRY) allows the problem programmer to select the subsequent action of the logic module. The problem programmer should select his operand based on the nature of the error as analyzed within his routine.

Procedure: An ERET macro issued in the problem program error routine generates a branch instruction to return control to the logic module. Register 14 in the generated branch instruction contains the address of the return point in the module. The macro operand (SKIP, IGNORE, or RETRY) supplies the displacement (0, 4, or 8 bytes respectively) from the return point of an instruction that returns control to the desired reentry point in the logic module.

ESETL Macro

Label	ESETL	filename	
	L	1,=A(filename)	Loads address of DTF table.
	L	15,16(1)	Loads address of logic module.
	BAL	14,20(15)	Branch to ESETL routine in logic module.

Function: The ESETL (End SET Limit) macro instruction ends the sequential mode initiated by the SETL macro.

Procedure: If blocked records are specified, ESETL writes the last block if a PUT macro was issued.

FEOV Macro

Label	FEOV	filename	
	L	1,=A(filename)	Loads address of DTF table.
	L	15,16(1)	Loads address of logic module.
	BAL	14,16(15)	Branch to FEOV routine in logic module.

Function: The FEOV (Force End-of-Volume) macro instruction is for either input or output files on magnetic tape devices to force an end-of-volume condition when neither an EOF indicator nor a reflective marker has been sensed. It indicates that processing of records on one volume is considered finished, but that more records for the same logical file are to be read from, or written on, the following volume.

Procedure: The FEOV macro fetches the proper phases to close the current volume and open the new volume.

FEOVD Macro

Label	FEOVD	filename	
	LA	1,=CL8'$$BOSDEV'	Loads address of B-Transient name.
IJJOxxxx	BAL	0,*+8	
	DC	A(filename)	
	SVC	2	Fetch phase $$BOSDEV.

Function: The FEOVD (Forced End-of-Volume for Disk) macro instruction is used for either input or output files in sequential disk processing to force an end-of-volume condition before end-of-volume has actually been reached. It indicates that record processing on one volume is finished, but that more records for the same logical file are to be read from, or written on, the following volume. If no extents are available on the new volume, the job is canceled.

The FEOVD macro fetches $$BOSDEV to close the current volume and open a new volume.

Procedure: When FEOVD is issued, an end of extent switch is set in the DTFSD. When the next GET or PUT is issued, end of extent is detected and the open transients are called.

FREE Macro

Label	FREE	filename	
	L	1,=A(filename)	Loads address of DTF table.
	L	15,16(1)	Loads address of logic module.
	BAL	14,44(15)	Branch to FREE routine in the logic module.

Function: The FREE macro instruction releases a protected track (Track hold function included for Asynchronous Processing) on a direct access storage device.

Procedure: The FREE routine in the logic module determines the seek address of the protected (held) track, and loads the address of the control seek CCB into general register 1. The routine then issues an SVC 36 to free the track. For sequential DASD files, FREE is treated as a no-op since the holding and freeing of tracks (or control intervals) is done implicitly by the logic modules.

GET Macro

Label	GET	filename,PARAM*	
	L	1,=A(filename)	Loads address of DTF table.
	L	0,=A(PARAM)	Loads address of work area if specified. *
	L BAL	15,16(1) 14,8(15)	Loads address of logic module. Branch to GET routine in logic module.

* Optional

Function: This instruction makes the next sequential logical record from an input file available for processing in either an input area or a specified work area. It is used for any input file in the system, and for any type of record: blocked or unblocked, spanned or unspanned, fixed or variable length, and undefined. When the GET routine detects an end-of-volume or an end-of-file condition, it calls in the EOV/EOF monitor, which initiates the correct file termination procedures.

The GET macro instruction is written with one or two parameters, depending on the area where the records will be processed. Either form, but not both, can be used for one logical file. If records are to be processed directly in the input area(s), the GET macro instruction requires only one parameter. This parameter specifies the name of the file from which the record is to be retrieved. The file name must be the same as the one specified in the DTF header entry for the file.

The second parameter is optional, and if used, specifies the address (or a register containing the address) of the work area. This parameter is used if records are to be processed in a work area defined by the user. The second parameter causes the GET routine to move each logical record from the input area to the work area.

Procedure: Two input areas permit an overlap of data transfer and processing operations. Whenever two input areas are specified, the LIOCS routines transfer records alternately to each area (except when combined files are specified). The LIOCS routines completely handle the switching of I/O areas so that the next sequential record is always available to the problem program for processing. If the file is blocked, it is not necessary to transfer data from the input device to main storage on every GET instruction. Only when the first record of a block is required (blocked records), is it necessary to transfer data.

If overlap is possible, the transfer of data required for the current GET was initiated on a previous GET. If overlap is not possible, it is necessary to start data transfer, read data, and wait for completion of the I/O operation. The handling of the data is done after a test for unusual condition is made. Unusual conditions are: end of reel, wrong-length record, irrecoverable error, no record found, etc.

LBRET Macro

Label	LBRET	1	
	SR SVC	1,1 9	Zero register 1. Return to logical IOCS.

Label	LBRET	2	
	SVC	9	Return to logical IOCS.

Label	LBRET	3	
	LNR	1,1	Put negative value in register 1.
	SVC	9	Return to logical IOCS.

Function: The LBRET (LaBel RETurn) macro instruction provides the return from:

1. Your routine for the processing of additional user labels or nonstandard labels that you want to check or write.

2. Your routine for any examination or processing of extent information during the direct access open of a DASD file.

To return from a label processing routine (specified by the DTF entry LABADDR), issue the LBRET macro after each user's header or trailer label is processed. Tape files need an operand of 1 or 2, while DASD label routines use all three operands as required.

To return from an extent processing routine (specified by the DTF entry XTNTXIT), issue the LBRET macro after handling each extent. An operand of 2 passes the next extent to your routine. After processing the last extent, an operand of 1 signifies to LIOCS that all user extent processing has been completed.

Procedure for Tape and DASD Labels:

1. Input Files. The LBRET macro checks for an operand of 1. If one, the user label processing is terminated and any additional labels are skipped. If all the labels on an input file are to be processed, the LBRET 1 macro is not needed. That is, IOCS ends processing when the DASD end-of-file record or the tapemark is sensed.

2. Output Files. LBRET 1 is required to return to logical IOCS when all user labels have been created and written. Otherwise, LIOCS terminates label processing after a maximum of 8 header or (where allowed) 8 trailer labels.

Operand 1 is invalid for tape input files that contain nonstandard labels (FILABL=NSTD).

Operand 2 (input file) returns to LIOCS after each additional user standard label has been checked. LIOCS makes the next label, if present, available for checking in the label input area. When IOCS senses the end of the label set (DASD end-of-file record or tapemark), it terminates label processing.

Operand 2 (output file) returns to LIOCS after each additional user standard label except the last has been built. LIOCS writes the label from the label output area and returns to the user's label routine to permit him to build his next label. LBRET 1 terminates the label set or it is terminated after 8 header or 8 trailer labels have been written.

For nonstandard tape labels, LIOCS branches to the user's label routine only once, and the problem program must read or write every required label before issuing LBRET 2 to return to LIOCS.

Procedure for DASD Extents:;The LBRET macro checks for an operand of 2 to determine if the user desires any additional extents for examination. Control passes between LIOCS and the user's routine for each extent requested until an operand of 1 terminates extent processing for this file.

Operand 3 causes LIOCS to write an updated label onto a DASD input file. After writing the updated label, LBRET 2 procedures are followed.

Note: If register 15 is required in your routine, save the contents of it, and restore the contents before returning to LIOCS via the LBRET macro instruction.

LITE Macro

Label	LITE	filename,PARAM	
	L	1,=A(filename)	Loads address of DTF table.
	L	0,=A(PARAM)	Loads address of user's 2-byte pocket light indicator.
	L	15,16(1)	Loads address of logic module.
	BAL	14,16(15)	Branch to pocket light routine in the logic module.

Function: The LITE macro turns on the 1275/1419 pocket lights specified by the problem programmer.

Note: The problem program must issue a DISEN macro before issuing a LITE macro.

Procedure: The LITE macro turns on the pocket lights that are specified by setting indicators (bits) in a 2-byte field identified in the macro operand. When all the specified pocket lights are turned on, control returns to the problem program at the next sequential instruction following the LITE macro expansion.

NOTE Macro

Label	NOTE	filename	
	L	1,=A(filename)	Loads address of DTF table.
	L	15,16(1)	Loads address of logic module.
	BAL	14,12(15)	Branch to NOTE routine in logic module.

Function: The NOTE macro instruction retains the identification of a physical record just read or written in a specified file.

The user must ensure that the previous operation was completed satisfactorily by using the CHECK macro before issuing a NOTE. The record identification is placed in register 1.

Procedure: For a tape file, this routine loads the physical record count into register 1, and control returns to the user.

For DASD, register 1 is loaded with the four bytes identifying the cylinder, head, and record number (CCHR) or BBBn for control interval format, where:

BBB = physical Block Number of the Control interval and

n = the logical block number within the control interval.

If NOTE follows a WRITE macro, the unused space remaining on the track or control interval is loaded into register 0.

Label	OPEN	FILEA,FILEB,...,FILEn	
	CNOP	0,4	
	DC	0F'0'	
	LA	1,=C'$$BOPEN'	Initializes to fetch the OPEN processor.
IJJOxxxx	BAL	0,*+4+4*(&N-1)	Register 0 points to the address of the DTF table for the first file in the file list. The second operand causes a branch to the SVC 2 instruction.
	DC	A(FILEA)	Start of the file list.
	DC	A(FILEB)	(The file list contains the addresses of the DTF tables for all of the files specified in the operand of the OPEN macro.)
	.		
	.		
	DC	A(FILEn)	Address of the DTF table for the last file specified in the OPEN macro operand.
	SVC	2	Fetches the OPEN processor, $$BOPEN.

FILEn = Symbolic address of the DTF table for the last file specified in the OPEN macro operand.

N = Sequence number of a file (1, 2, 3, etc.), in the order it appears in the OPEN macro operand.

&N = N of the last file +1.

Function: The OPEN macro instruction activates each file in the problem program. The symbolic name of the logical file (assigned by the DTF header entry) is entered in the operand field of this instruction. Up to 16 files may be opened with an OPEN macro instruction by entering the filenames in the operand field. If physical IOCS is used, OPEN is required only when standard labels are to be checked or created.

Procedure: The OPEN macro instruction calls the OPEN processor, $$BOPEN, into the logical transient area. The monitor checks for the device type assigned to the file, and calls the appropriate device-oriented open logical transient. The tape, diskette, and DASD open transients do all processing required to check or create standard labels for their respective files. For devices other than magnetic tape, diskette, or DASD an indicator is set in the DTF table to show that these files have been opened. For PRT1 or 3800 printers with DTFPR, DTFDI or DTFCP with DISK=YES OPEN creates a DTF extension in a getvised area.

OPENC Macro

Label	OPENC	SYSxxx¹,SYSxxx²,.....SYSxxxn	
	LA	1,=C'$$BOPENC'	Loads address of B-transient name.
	BAL	0,IJJOxxxx	Branch to fetch B-transient.
	DC	AL1(class)¹	Logical unit class for SYSxxx¹.
	DC	AL1(number)¹	Logical unit number for SYSxxx¹.
	DC	AL1(class)²	Logical unit class for SYSxxx².
	DC	AL1(number)²	Logical unit number for SYSxxx².
	.		
	.		
	DC	AL1(class)n	Logical unit class for last SYSxxx in list.
	DC	AL1(number)n	Logical unit number for last SYSxxx in list.
IJJOxxxx	SVC	2	Fetches phase $$BOPENC.

n = a maximum of 16 symbolic units (either system or programmer) can be included in the macro operand.

Function: The OPENC macro instruction determines if a physical device is assigned to more than one of the symbolic units specified in the macro operand. A maximum of 16 symbolic units can be checked with a single macro instruction.

Procedure: The OPENC macro instruction calls the logical transient, $$BOPENC, which checks each symbolic unit specified in the macro operand in turn. $$BOPENC determines the PUB entry address specified in the LUB for the corresponding symbolic unit, and compares it to the PUB entry addresses of each of the remaining symbolic units in the macro operand. If an equal comparison results between the PUB addresses of any two symbolic units, an error message is printed and the job is canceled.

OPENR Macro

Label	OPENR	FILEA,FILEB,...,FILEn	
	CNOP	0,4	
	DC	0F'0'	
	LA	1,IJJOxxxx+4	Loads actual location address.
	MVI	*-4,X'58'	Disable subsequent relocation.
	L	0,IJJOxxxx+4	Loads relocation factor.
	SR	1,0	Finds displacement value.
	L	0,IJJOxxxx+4+4*N	Gets address of DTF table for file to be opened.
	AR	0,1	Adds displacement value.
	ST	0,IJJOxxxx+4+4*N	Returns new DTF table address to file list. (The three instructions listed are repeated for each file specified in the OPENR macro operand starting with FILEA.)
	LA	1,=C'$$BOPENR'	Initializes to fetch $$BOPENR.
	CNOP	0,4	
IJJOxxxx	BAL	0,*+8+4*(&N-1)	Register 0 points to the address used for relocation. The second operand causes a branch to the SVC 2 instruction.
	DC	A(*)	Address used by OPENR for relocation.
	DC	A(FILEA)	Start of file list.
	DC	A(FILEB)	(The file list contains ADCONS for the addresses of the DTF tables for all the files specified in the operand of the OPENR macro.)
	.		
	.		
	DC	A(FILEn)	ADCON for last file in file list.
	SVC	2	Fetches $$BOPENR.

FILEn = Symbolic address of the DTF table for the last file specified in the operand of the OPENR macro.
N = Sequence number of a file (1, 2, 3, etc.), in the order it appears in the OPENR macro operand.
&N = N of the last file +1.

Function: The OPENR macro instruction activates files used by self-relocating programs. In addition to the basic function performed by the OPEN macro, the OPENR macro relocates all the address constants within the DTF tables for the files specified in the operand field. A maximum of 16 files can be specified in the operand of a single OPENR macro instruction.

Procedure: The OPENR macro instruction calls the logical transient $$BOPENR to perform the relocation of the DTF table address constants for each individual file. After the DTF address constants for all the files specified in the macro operand have been relocated, $$BOPENR calls the OPEN processor ($$BOPEN), then the Open Monitor ($$BOPEN1) to perform the actual open function. After all the specified files are opened, control returns to the problem program.

POINTR Macro

Label	POINTR	filename,PARAM	
	L	1,=A(filename)	Loads address of DTF table.
	L	0,=A(PARAM)	Loads address of field containing record identification.
	L	15,16(1)	Loads address of logic module.
	BAL	14,16(15)	Branch to POINTR routine in logic module.

Function: The POINTR macro instruction repositions the file to read a magnetic tape or DASD record previously identified by a NOTE macro instruction.

Procedure: If the file is on tape, this routine spaces tape either forward or backward until the block count in the DTF table reaches the value provided as a parameter of the POINTR macro. Then the file is backspaced so the record may be read.

For DASD files, the POINTR macro instruction logic flow is the same as POINTW except track space is not considered. The POINTR macro is only used with IBM disk devices.

POINTS Macro

Label	POINTS	filename	
	L	1,=A(filename)	Loads address of DTF table.
	L	15,16(1)	Loads address of logic module.
	BAL	14,24(15)	Branch to POINTS routine in logic module.

Function: The POINTS macro instruction repositions a magnetic tape or DASD file to the beginning of the file.

Procedure: For a magnetic tape file, a POINTS macro instruction rewinds the tape associated with the filename. If any header labels are present, they are bypassed on the next READ or WRITE instruction. The tape is positioned to the first data record following the label set.

For a DASD file, a POINTS macro instruction positions the file to the lower limit of the first extent. The first record on the file is read or written when the next READ or WRITE macro instruction is issued for the file.

POINTW Macro

Label	POINTW	filename,PARAM	
	L	1,=A(filename)	Loads address of DTF table.
	L	0,=A(PARAM)	Loadss address of field containing record identification.
	L	15,16(1)	Loads address of logic module.
	BAL	14,20(15)	Branch to POINTW routine in logic module.

Function: The POINTW macro instruction repositions the file to write a magnetic tape or DASD record following the one previously identified by a NOTE macro instruction.

Procedure: If the file is on magnetic tape, this routine spaces tape either forward or backward until the block count in the DTF table reaches the value provided as a parameter of the POINTW macro.

For a DASD file, the DASD address of the record to be written is calculated. The POINTW routine determines if the record can be contained in the same extent used by the preceding record (the preceding record is the one identified by the NOTE macro). If not, the Sequential DASD Open routine is called to open the required extent. When the correct extent is obtained, the CCW seek address is modified and the space remaining on the extent is updated in the DTF table. Control then returns to the problem program.

PRTOV Macro

Label	PRTOV	filename,CHAN,routine*	
	L	1,=A(filename)	Loads address of DTF table.
	L SR	0,=A(routine) 0,0	Loads address of user's overflow routine if specified. * Zero register 0 if no user routine specified. *
	L	15,16(1)	Loads address of logic module.
	OI	21(1),1	Sets channel 9 bit in DTF table if CHAN is 9; otherwise, channel 12 assumed. **
	BAL	14,4(15)	Branch to PRTOV routine in logic module.

* Optional
** Only if CHAN=9

Function: The PRTOV (PRinTer OVerflow) macro instruction specifies the operation to be performed when an overflow condition is reached on a printer. Whenever this macro instruction is to be issued in a problem program, the DTFPR or the DTFSR entry PRINTOV must be included in the file definition.

Procedure: The program performs the functions specified by the problem programmer. That is, skip to channel 1 on a 9 or 12, or perform his own functions when a 9 or 12 is sensed. If skip to channel 1 on a 9 or 12 is desired and a 9 or 12 is sensed, skip to channel 1 is placed in the CCW chain. Then, an SVC 0 executes the skip and resets the channel 9 and 12 indicators.

If a user routine is specified in the macro instruction, the problem programmer may issue any logical IOCS macro instructions (except another PRTOV) in his routine to perform whatever functions are desired. For example: print total lines, skip to channel 1, and print overflow page headings. The user routine must return to LIOCS by a branch to the address in register 14. Logical IOCS supplies this address upon entry to the user's routine. Therefore, if LIOCS macros are used in the routine or if register 14 is used, the return address must be saved.

PUT Macro

Label	PUT	filename,PARAM,control*	
	L	1,=A(filename)	Loads address of DTF table.
	L L L	0,=A(STLSP) 0,=A(STLSK) 0,=A(PARAM)	Loads address of control field, if control = STLSP. * Loads address of control field, if control = STLSK. * Loads address of work area, if specified. *
	OI	48(1),X'80'	Sets indicator in DTF table if control = STLSK. *
	L BAL	15,16(1) 14,12(15)	Loads address of logic module. Branch to PUT routine in logic module.
	NI	48(1),X'7F'	Resets control = STLSK indicator in DTF table. *

* Optional

Function: This instruction writes or punches logical records that have been built directly in the output area or in a specified work area. It is for any output file in the system (except work file), and for any type of record: blocked or unblocked, spanned or unspanned, fixed or variable length, and undefined. It operates much the same as GET but in reverse. It is issued after a record is built.

Similar to GET, the PUT macro instruction is written with one or two parameters, depending on the area where the records are built. Either form, but not both, can be used for one specified logical file. If records are built directly in the output area(s), the PUT macro instruction requires only one parameter. This parameter specifies the name of the file to which the record is to be transferred. The filename must be the same as the one specified in the DTF entry for the file.

The second parameter is optional and if used, specifies the address (or a register containing the address) of the work area. This parameter is used if records are to be built in a work area defined by the user. The second parameter causes the PUT routine to move each logical record from the work area to the output area.

A third (optional) parameter, CONTROL=, is included in the macro operand for files assigned to printers with the Selective Tape Lister (STL) feature.

Procedure: Two output areas permit an overlap of data transfer and processing operation. Whenever two output areas are specified, the LIOCS routines transfer records alternately from each area (except for combined files). The LIOCS routines completely handle the switching of I/O areas so that the proper area is available to the program for the next sequential output record.

If a work area is specified, the output record is moved from the work area to the output area.

With blocked files specified, it is not necessary to transfer information from main storage to the output device on each PUT instruction. Only if the logical record is the last record of a block is it necessary to transfer a physical record to the output device. If overlap is possible, the transfer of information need not be completed before another PUT requiring data transfer is issued. When overlap is not possible, the transfer of data must be completed before another PUT is issued.

Tests are made for unusual conditions, which include: end of reel, wrong length record, irrecoverable error, no record found, etc.

PUTR Macro

Label	PUTR	filename,workout*,workinp*	
	L	1,=A(filename)	Loads address of DTF table.
	L L	0,=A(workout) 2,=A(workinp)	Loads address of output work area. * Loads address of input work area. *
	OI L BAL	2(1),X'08' 15,16(1) 14,4(15)	Set action message indicator in CCB. Load address of logical module. Branch to PUTR routine in logic module.

* Optional

Function: The PUTR (PUT with Reply) macro handles action messages that appear on the screen of the Display Operator Console. PUTR used with the 3210 or 3215 performs the same functions as a PUT followed by a GET. Moreover, the message non-deletion code for the Display Operator Console is then provided.

Procedure: The PUTR macro is issued after a record has been built. It processes fixed-length records only. The PUTR macro is written with either one or three parameters, depending on the area in which the records must be built. Either form, but not both, can be used for a logical file. If the records are built in the I/O area, only the filename parameter is required. If the records are to be built in a user-specified work area, both workout and workinp must be specified. In this case, the record is moved from the work area to the I/O area.

In the case of overlap, information transfer need not be completed before the next PUTR requests new data to be transferred. If overlap is not possible, the next PUTR must wait for the completion of the previous PUTR. Tests are made for unusual conditions such as end-of-reel, wrong length record, irrecoverable error, no record found, etc.

PUTR sets bit 5 of byte 3 in the CCB to '1' to indicate an action message; it then passes control to logical IOCS, which executes a PUT immediately followed by a GET.

RDLNE Macro

Label	RDLNE	filename	
	L	1,=A(filename)	Loads address of DTF table.
	L	15,16(1)	Loads address of logic module.
	BAL	14,4(15)	Branch to RDLNE routine in logic module.

Function: The RDLNE macro provides selective online correction when journal tapes are being processed on an IBM 1287 Optical Reader. This macro reads a line in the online correction mode while processing is in the offline correction mode.

Procedure: If the reader cannot read a character, logical IOCS retries the line containing the unread character. If still unsuccessful, the user is informed of the condition via his error correction routine (specified in the DTFOR COREXIT entry). The RDLNE macro causes another attempt to read the line. If the character in the line cannot be read during this attempt, the character is displayed on the 1287 display scope. The operator may key in the correct character, if possible. If the defective character cannot be readily identified by the operator, he may enter a reject character in the error line. This condition is posted in byte 80 of the DTF table for user examination. Wrong length records and lost line conditions are also posted to byte 80 of the DTF table. RDLNE should be used in COREXIT only; otherwise the line following the one in error will be read in online correction mode.

The macro requires only one parameter, the symbolic name of the file from which the record is to be retrieved. This name is the same as that specified in the DTFOR header entry for this file. The filename can be specified as a symbol or in special or ordinary register notation.

READ Macro

Label	READ	filename,TYPE,PARAM,length	
	L	1,=A(filename)	Loads address of DTF table.
	L	0,=A(PARAM)	Loads address of input area.
	L	15,16(1)	Loads address of logic module.
	BAL	14,28(15)	If TYPE=ID. *
	BAL	14,24(15)	If TYPE=KEY. *
	BAL	14,0(15)	If TYPE=MR. *
	LA	14,IJJRSYSNDX+10	Loads return address for TYPE=SQ.
IJJRSYSNDX	BAL	0,4(15)	Branch to READ routine in the logic module if TYPE=SQ.
	DC	A(PARAM)	Address of input area.
	DC	H'length'	Length of record to be read.

* Portion of macro expansion determined by TYPE= parameter.

Label	READ	filename,DR,PARAM1,PARAM2	
	L	1,=A(filename)	Loads address of DTF table.
	L	0,=A(fldname)	Loads the field name specified by PARAM1.
	LR	0,r	Loads the register specified by PARAM1.
	BAL	0,*+6	
	DC	AL1(PARAM1)	Generated if PARAM1 and PARAM2 are numbers.
	DC	AL1(PARAM2)	
	L	15,16(1)	Loads address of logic module.
	BAL	14,8(15)	Branch to read routine in logic module.

READ macro expansion for the 3886 Optical Character Reader.

Function: The READ macro instruction causes part or all of the next sequential physical record (or the next logical block for control interval format) to be read from the file associated with the filename into the area of storage indicated. If the file is on a 3886 Optical Character Reader, the storage area is indicated in the DTF.

Procedure: The READ macro instruction must always be followed by either a CHECK macro (MICR and work files) or a WAITF macro (DAM, ISAM, and 3886 files) to ensure the completion of the READ instruction.

The read logic sets up the channel program, modifies the CCW, inserts the address and number of bytes to be read, and issues an SVC 0. For control interval format the READ may not cause physical I/O.

The read logic does not provide for deblocking of records. If the user wishes to use blocked records, he must provide this function in the problem program.

RELEASE Macro -- Dynamic Device Release

Label	RELEASE	SYSxxx,...	
	STM	0,1,SAVE	Saves registers 0 and 1.
	LA	1,=C'$$BRELSE'	Loads address of B-Transient name.
	BAL	0,*+4+6	Branches to fetch and skip table.
	SVC	2	Fetches $$BRELSE.
	LM	0,1,SAVE	Restores registers 0 and 1.
	SVC	14	Normal end of job.

Function: This macro releases a unit table as specified by the problem program and fetches $$BRELSE.

The 'savearea' parameter is optional. If it is provided, it should be the name of an 8-byte area where registers 0 and 1 are saved for the user. If it is not provided, the contents of registers 0 and 1 are destroyed.

Procedure: The macro checks all of the units provided in the operand sublist to assure that no system logical units are requested for release. If system logical units are specified, an MNOTE is issued and the unit is ignored.

After all checking is done, a unit table is set up, register 0 is loaded with the table address, and $$BRELSE is fetched. If the 'savearea' option is specified, registers 0 and 1 are saved, and code is generated to restore them after the transient returns control to the RELEASE macro.

RELSE Macro

Label	RELSE	filename	
	L	1,=A(filename)	Loads address of DTF table.
	L	15,16(1)	Loads address of logic module.
	BAL	14,4(15)	Branch to RELSE routine in logic module.

Function: The RELSE (release) macro instruction is used in conjunction with blocked input records. It allows the programmer to skip the remaining records in a block. If the record spans multiple physical blocks, the entire logical spanned record is bypassed. Processing continues with the first record of the next block when the next GET macro instruction is issued.

Procedure: The GET routine is modified to make the current record being processed look like the last record of the block. With this indication, the next GET transfers information from the input device to main storage and makes the first record of the new block available to the problem program.

RESCN Macro

Label	RESCN	filename,r^1,r^2	
	L	1,=A(filename)	Loads address of DTF table.
	LA	0,18	
	MVC	88(8,1),0(r^2)	Puts Load Format CCW for reference mark in DTF table.
	MVC	96(16,1),0(r^1)	Puts Load Format CCW for field to be read in DTF table.
	L	15,16(1)	Loads address of logic module.
	BAL	14,16(15)	Branch to RESCN routine in logic module.

Function: The RESCN macro provides the capability of rereading a field that has a defective character. This macro pertains only to the document mode and rereads into the portion of IOAREA1 corresponding to the original read. Online correction can also be forced by this macro.

The macro requires from three to five parameters. The first parameter specifies the symbolic name of the 1287D file given in the DTFOR header entry for the file. The second parameter specifies a general purpose register (2-12) which must contain the address of the Load Format CCW giving the document coordinates for the field to be read. The third parameter specifies a general purpose register (2-12) that must contain the address of the Load Format CCW giving the coordinates of the reference mark. The fourth parameter specifies a number (n), which is the number of retries to be given. The fifth parameter specifies one more retry with forced online correction. This parameter must be the letter F.

Procedure: When a character cannot be read, logical IOCS retries the line containing the unread character. If the character still cannot be read, the user is informed of the condition in his error correction routine specified in the DTFOR COREXIT entry. The user can then issue the RESCN macro to reread the field with the unreadable character. If the character still cannot be read, it is retried up to nine times depending on what the user specified. If the error still exists on the last retry, online correction is forced if the user specified this.

SEOV Macro

Label	SEOV	filename	
	LA	1,=C'$$BCEOV1'	Loads addreess of B-transient name.
	L	0,=A(filename)	Saves filename for B-transient phase.
	SVC	2	Fetches phase $$BCEOV1.

Function: The SEOV (System Units End-of-Volume) macro instruction allows automatic volume switching to occur if the reflective spot is reached on a magnetic tape output file assigned to either SYSLST or SYSPCH.

Procedure: An SEOV macro, issued after the physical end-of-volume has been detected on a tape file, fetches phase $$BCEOV1 to determine the file type, and to select the proper tape close routine. The selected tape close routine performs the appropriate close functions and determines if an alternate tape is available. If an alternate tape is available, it is opened and made ready for processing.

SETDEV Macro

Label	SETDEV	filename,phasename	
	L	1=A(filename)	Loads address of the DTF table.
	BAL	0,*+12	Generated if the phasename is an actual phasename.
	DC	CL8'phasename'	
	LR	0,r	If phasename is specified in a register (r) other than register 0.
	L	15,16(1)	Loads address of logic module.
	BAL	14,16(15)	Branch to SETDEV routine in logic module.

Function: The SETDEV (SET DEVice) macro instruction loads a format record into the 3886 Optical Character Reader.

Procedure: The SETDEV macro generates code which sets up parameters and branches to the 3886 logic module. The logic module gets the format record from the core image library and loads it into the 3886 device control unit.

SETFL Macro

Label	SETFL	filename	
	L	0,=A(filename)	Loads address of DTF table (DTFIS Load).
	LA	1,=C'$$BSETFL'	Loads address of B-transient name.
	SVC	2	Fetches phase $$BSETFL.
	LR	1,0	Saves address of DTF table for the problem program.

Function: The SETFL (SET File Load mode) macro instruction sets up the ISAM file so that the load function can be performed.

Procedure: The SETFL macro instruction preformats the last track index of each cylinder of a file with zero entries, and initializes for a WRITE. Control then returns to the problem program.

SETL Macro

Label	SETL	filename,PARAM	
IJJS&SYSNDX	ST	PARAM(1),IJJS&SYSNDX+8	Saves parameter.
	LA	1,=C'$$BSETL'	Loads address of B-transient name.
	BAL	0,*+12	Branch to fetch B-transient.
	DC	A(filename)	Address of DTF table.
	DC	A(PARAM(1))	Address of field containing starting (or lowest) reference if PARAM=ID name. *
	DC	CL4'PARAM'	If PARAM = BOF, KEY, or GKEY. *
	SVC	2	Fetches phase $$BSETL.
	L	1,IJJS&SYSNDX+4	Loads address of DTF table.

*Optional

Function: The SETL (SET Limits) macro instruction initiates the mode for sequential retrieval and initializes the ISAM routines to begin retrieval at a specified starting address.

Procedure: If KEY is specified in the DTFIS table, the SETL routine searches the indexes to find the track and record address of the keyed record. The GET/PUT constants are initialized to begin with the address of the keyed record. When BOF (beginning of the file) is specified, SETL initializes the GET/PUT logic to begin retrieval with the first record in the file. If ID is specified in the DTF, the GET/PUT logic is initialized to start with the record in the prime data area corresponding to the specified ID.

TRUNC Macro

Label	TRUNC	filename	
	L	1,=A(filename)	Loads address of DTF table.
	L	15,16(1)	Loads address of logic module.
	BAL	14,20(15)	Branch to TRUNC routine in logic module.

Function: The TRUNC (TRUNCate) macro instruction is used with blocked output records. It allows the programmer to write a short block of records. (Blocks do not include padding.) Thus, the TRUNC macro is used for a function similar to the RELSE (release) instruction for input records, but in reverse. That is, when the end of a group of logical records is reached, that block is written and a new group is started at the beginning of a new block.

Procedure: If (as a result of the previous PUT) the block has already been transferred to the output device, the TRUNC macro requires no additional handling. If physical I/O is needed, the PUT routine is modified to handle the truncated record. Control then returns to the problem program.

WAITF Macro

Label	WAITF	filename¹,filename²,...filenamen	
IJJW&SYSNDX	ST	SYSLIST(n,1), IJJW&SYSNDX+n*4	Stores end of list code, n+1, in last entry in file list.
	L	1,=A(filenamen)	Loads address of DTF table.
	L	15,16(1)	Loads address of logic module.
	BAL	14,4(15)	Branch to WAITF routine in logic module.
	DC	A(SYSLIST(n))	Address of file list.

n = a maximum of 16 files can be specified in the macro operand.

Function: The WAITF macro tests the condition of MICR device(s) and tests for I/O complete when used with DAM or ISAM files.

Procedure: For MICR files, if any one of the devices tested is operative and ready (that is, has records or error conditions to be processed), control returns to the problem program at the next sequential instruction following the macro expansion. On the other hand, if all the devices tested are not operational (that is, they are all waiting for documents to process), the system enters the wait state.

For DAM or ISAM files, the WAITF macro makes the system enter the wait state until a previously started I/O operation is complete.

Note: Only that partition in which the device(s) tested is operating enters the wait state. This allows processing to continue in another partition.

WRITE Macro

Label	WRITE*	filename,TYPE,PARAM	
	L L L	1,=A(filename) 0,=A(PARAM) 15,16(1)	Loads address of DTF table. Loads address of output area. Loads address of logic module.
	BAL	14,32(15)	Branch to WRITE routine in logic module if TYPE=SQ. **
	BAL	14,28,(15)	Branch to WRITE routine in logic module if TYPE=UPDATE. **

* For RECFORM = FIXUNB.
** Optional

Function: The WRITE macro instruction writes a record from the indicated area in main storage to the file associated with the file name.

Procedure: The WRITE macro sets up the channel program, modifies the CCW command code to write, inserts the address and number of bytes to be written, and issues an SVC 0. For control interval format, physical I/O may or may not occur.

The write logic does not provide for blocking of records. If the user wishes to block records, he must provide for it in the problem program.

The WRITE macro instruction must always be followed by either a CHECK macro (work files) or a WAITF macro (DAM and ISAM files) to ensure the completion of the WRITE instruction before another instruction is issued.

FILE INITIALIZATION AND TERMINATION

File initialization and termination routines open files required by the problem programmer, and close the files when they are no longer needed. These routines, called into the B-transient (logical transient) area by the corresponding OPEN and CLOSE macros, consist of:

1. TES Processor ($$BOESTV).
2. Open Monitor ($$BOPEN, $$BOPEN1, $$BOPEN2, $$BOPEN4, and $$BOPLBL).
3. Close Monitor ($$BCLOSE, $$BCLOS2, $$BCLOS4, $$BCLRPS, and $$BCLLBL).
4. EOF/EOV Monitor ($$BCEOV1).
5. Device or file-processing method oriented open and close transients.

OPEN ROUTINES CHARTS 01-04

The open routine opens each file needed in the problem program. Up to 16 files can be opened with each OPEN macro instruction by entering their filenames as parameters.

To open a particular file, the Open Monitor (Chart 02) examines the DTF table specified by the filename to determine the file type and/or the file processing method. This information is obtained from byte 20 of the DTF table. Figure 2 on page 6 summarizes these DTF type codes. In addition, the Open Monitor performs some initialization and checking, and reads any necessary label information into main storage. The Open Monitor then calls the appropriate open transient(s) to handle the file open.

Unit Record and 3881 Optical Mark Reader Files

When opening unit record devices (readers, punches, consoles, printers, paper tapes, and the 3881 Optical Mark Reader), the Open Monitor calls $$BOUR01 to determine if the device is in the ready condition. If the device is ready, the open indicator in the DTF table is set to a 1 (bit 0 of byte 21) to indicate the file is open.

The Open Monitor calls $$BOMRCE if the device is a 3505 with OMR and RCE or a 3525 with RCE.

Magnetic Ink Character Recognition Files

When opening MICR type devices (IBM 1255, 1259, 1270, 1275, and 1419), the Open Monitor calls $$BOMR01, which clears the document buffer area and initializes the document buffer pointer within the DTF. The address of the DTF is inserted into the correct entry of the supervisor PDTABB table. The unit exception bit in the CCB is turned on, and the device address is calculated and moved into the DTF. The OPEN indicator in the DTF table is set to indicate that the file is open.

Optical Reader Files (Except 3881)

When opening the IBM 1287 Optical Reader, the Open Monitor calls $$BOOR01, which determines if the device is ready, and if so, further determines if a header is to be read (HEADER=YES specified in the DTF). If it is, the open routine waits for the operator to manually key in a header. When the header has been read, the OPEN indicator in the DTF table is set to 1 to indicate that the file is open.

When opening the 3886 Optical Character Reader file, the Open Monitor calls $$BOOR01, which determines if the device is ready and if so, loads a format record from disk into the format area of the DTF. If the length of the format record is found to be within the required limits, it is loaded into the 3886 control unit. If no errors occur on the load, the open bit in the DTF is set on and control is returned to the Open Monitor. If the format record length is incorrect or if an error occurs on the load, the open routine is canceled by an illegal SVC.

Magnetic Tape Files (DTFMT, DTFPH-MT)

When opening magnetic tape files, the Open Monitor calls $$BOTSVA to link to the $IJJTTOP SVA phase to complete the OPEN processing.

DASD Files

When opening DASD files, the Open Monitor checks the label information to determine the type of processing used for the file: SAM, DAM, ISAM, or VSAM. The monitor then calls the appropriate transient to complete the open. If an ISAM DTF is linked with a VSAM file, IIPOPEN is called.

Diskette Files

When opening diskette files, the Open Monitor checks the DTF type code (byte 20 of the DTF table) and the device code (byte 29 of the DTF table) to determine if the Diskette Input/Output Unit transients are needed. The monitor then fetches the appropriate transient to complete the open (see Charts 07 and 08).

CLOSE ROUTINES CHARTS 05, 06

The close routine closes any file that was previously opened in the system. Up to 16 files can be closed by each CLOSE macro instruction by entering their filenames as parameters.

Unit Record Files (Except MICR)

For unit record devices, the Close Monitor sets the close indicator in the DTF table (bit 0 of byte 21) to a 0 to indicate that the file is closed.

MICR (Magnetic Ink Character Recognition) Files

For MICR type files, the Close Monitor calls $$BCMR01 to complete the close function.

Magnetic Tape Files (DTFMT, DTFPH-MT)

For magnetic tape files, the CLOSE Monitor calls $$BOTSVA to link to the $IJJTTOP SVA phase to complete the CLOSE processing.

DASD Files

For DASD files processed by SAM the Close Monitor calls $$BOSFBL to link to the $IJJGTOP SVA phase to complete the close function. For DASD files processed by ISAM, the Close Monitor calls $$BCISOA to update and rewrite the format-1 and format-2 standard file labels, and to set the close indicator in the DTF table. If an ISAM DTF is linked with a VSAM file, ISCCLOSE is called. For DASD files processed by DAM, $$BCLRPS is called to free storage that was obtained for the DTF extension.

Diskette Files

For Diskette Input/Output Unit files, the Close Monitor calls $$BODIO4 to complete the close function.

FILE LABELING

VSE/Advanced Functions can identify and protect DASD, diskette, and magnetic tape files by recording labels on each volume (DASD pack, diskette, or magnetic tape reel). These labels ensure that the correct volume is used for input and that no current information is destroyed when a volume is used for output.

DASD, diskette, and magnetic tape files processed by logical IOCS must conform to certain standards regarding the use of labels. Although it is possible to process files with physical IOCS macros such as EXCP and WAIT, without processing labels, any file processed this way that is defined by a DTFPH macro must also conform to the same label standards established for files processed by logical IOCS.

The standard label set processed by logical IOCS includes one <u>volume</u> label for each volume, and one or more <u>file</u> labels for each logical file contained within the volume. Optional user labels can be included in the label set but these must be processed by an independent user routine. (Logical IOCS routines pass control to the user's label routine in the problem program if the LABADDR= parameter is specified in the file definition, DTF, macro.) Additional volume and file labels can also be included in the label set but these labels can only be processed by the user, and only if nonstandard labels are specified in the file definition macro.

User labels are not supported for diskette files.

LABEL PROCESSING

Creation of Tape Volume Labels

The IBM or American National Standards Institute, Inc. standard volume label 1, and any additional EBCDIC volume labels, are written by an IBM-supplied utility program at the time a reel is prepared for use. The information in the standard volume label is checked, but never altered, during file processing. Logical IOCS bypasses all additional volume labels when building output files.

Standard Tape File Labels

Standard file labels are written before and after every logical file on a reel. These labels are referred to as file header labels or file trailer labels, depending on their position and use. They are always 80 bytes long and always have the same format and content, with the following exceptions:

1. The label identifier field (bytes 1-3) contains:
   a. HDR to indicate a header label (precedes the data file).
   b. EOV to indicate an End-of-Volume (end of reel) trailer label (written at the end of a reel, indicating that the file is continued on another reel).
   c. EOF to indicate an End-of-File trailer label (written at the end of the logical file).
2. The block count field is used only in the EOF and EOV trailer labels. This field is set to zero in the HDR label.

Additional File Labels on Tape

Each standard file label (one header and one trailer) can be followed by up to seven additional file labels for EBCDIC tape files, or by up to eight additional file labels for ASCII tape files. The labels are 80 bytes long and must contain the label identifier HDR, EOV, or EOF in the first three bytes. The fourth byte should contain a character 2, 3,...n, indicating the second, third,... and up to the last file label. These labels are not processed by LIOCS. If required, these labels must be written in the user's LABADDR routine by use of physical I/O macro instructions.

LIOCS bypasses additional header labels on input files. For ASCII ouput files a HDR2,EOF2 or EOV2 label is written by LIOCS following the HDR1, EOF1 or EOV1 label.

User Header and Trailer Labels on Tape

The user can include additional header and trailer labels to further define his file, if he desires. Each additional label in the set is 80 characters long. EBCDIC label identifiers are numbered from UHL1 and UTL1 through UHL8 and UTL8, maximum, for user header and trailer labels, respectively. American National Standards Institute, Inc., user header and trailer labels are identified by UHLa and UTLa, respectively, wherein "a" represents the range 2/0 through 5/14 except 2/7 (quotation mark). The remaining 76 characters can contain any information and arrangement desired by the user.

Tapemarks with Standard Tape Labels

The sequence of items on the tape that uses standard label sets is:

1. No tapemark preceding the header label set.
2. Header label set:
   a. Standard volume label (required).
   b. Additional volume labels (0-7, optional: EBCDIC only).
   c. Additional user volume labels (0-9, optional: American National Standards Institute, Inc., only).
   d. Standard file header label (required).
   e. Additional file labels (0-7, EBCDIC: 0-8, American National Standards Institute, Inc., optional).
   f. User header labels (0-8, EBCDIC: or range 2/0-5/14 except 2/7, American National Standards Institute, Inc., optional).
3. Tapemark between header label set and first data record.
4. Physical data records for file.
5. Tapemark between last data record and trailer label set.
6. Trailer label set:
   a. Standard file trailer label (required at end-of-file and end-of-volume).
   b. Additional file labels (0-7, EBCDIC: 0-8, American National Standards Institute, Inc., optional).
   c. User trailer labels (0-8, EBCDIC: range 2/0-5/14 except 2/7 (quotation mark), American National Standards Institute, Inc., optional).
7. Tapemark after trailer label set.
8. If multifile reel (EOF label), next standard file header label follows here. If single-file reel (EOF label) or if last file of a multifile reel, another tapemark follows here. If multireel file (EOV label), one tapemark follows the EOV label on an EBCDIC file. Two tapemarks follow the EOV label on a multireel ASCII file.

Standard Tape Label Processing

Standard tape label processing is performed by the LIOCS transient label-processing (Open, Close, EOF/EOV) routines. These routines use the information supplied in the job control card (// TLBL) that was stored in the label information area.

The actual label processing consists of the following checks:

Tape Input File:

- The volume serial number in the standard volume label on the first or only reel is compared to the file serial number in the TLBL card. All other volume labels on all reels of the file are bypassed.
- The contents of the TLBL card are compared to the corresponding fields in the standard file header label on the first reel. Fields 1-10 are required. Fields 11-14 are optional. For successive reels of a multireel file, the volume sequence number (EBCDIC file) or file section number (ASCII file) is increased by 1 for each reel.
- If user labels are indicated, they are read into main storage by the open routine for processing by the user's label routines. The user labels are read one at a time, until all have been processed.
- When a standard file trailer label is read, the block count is compared to a count accumulated by IOCS.
- If user trailer labels are indicated, they are read into main storage by the close routine for processing by the user's label routine. The user trailer labels are read one at a time until all have been processed.

Tape Output File:

- The volume serial number in the standard volume label on the first or only reel is compared to the file serial number in the // TLBL card. All other volume labels on all reels are bypassed.
- The expiration date in the standard file header label is checked against the <u>today's</u> <u>date</u> in the communications

region. If the expiration date has passed, the reel is backspaced to write the new standard file label. If not, the operator is notified of the condition. This check is performed on each reel of a multireel output file. If no file label is present, the tape is considered expired. For an expired 9-track tape, the user-specified density is compared to the VOL1 density of the mounted tape. If a discrepancy is found, and if the tape is at load point, the volume label(s) is rewritten according to the user-specified density.

- The new standard file label is written with the information supplied in the // TLBL card. For multireel files, the volume sequence number (EBCDIC file) or file section number (ASCII file) is increased by 1 for each successive reel.
- If user header labels are indicated, the user's label routine is entered to furnish the labels as each reel is opened. This can be done for as many as eight user header labels per EBCDIC file and for an unlimited number of user header labels per ASCII file.
- If end of reel is sensed before completing the file, an EOV trailer label is written with all fields presented in the // TLBL card plus a block count.
- When end of file is reached, an EOF trailer label is written identical to the EOV label previously mentioned.
- If user trailer labels are indicated, the user's label routine is entered to furnish the labels after each trailer (EOV or EOF) label is written. This can be done for as many as eight user trailer labels for EBCDIC files and an unlimited number of trailer labels for ASCII labels.

Nonstandard Tape Labels

Any tape labels that do not conform to the standard label specifications are considered nonstandard. Nonstandard labels are not supported in ASCII files. If nonstandard labels are to be read, checked, or written, it must be done by the user. On input files, the nonstandard labels may or may not be followed by a tapemark. Therefore, four conditions are possible:

1. Nonstandard label(s), followed by a tapemark, to be checked.
2. Nonstandard label(s), _not_ followed by a tapemark, to be checked.
3. Nonstandard label(s), followed by a tapemark, _not_ to be checked.
4. Nonstandard label(s), _not_ followed by a tapemark, _not_ to be checked.

For conditions 1 and 2, the DTFMT or DTFSR entries must specify nonstandard labels and the address of a user-written routine to do the reading or writing.

For condition 3, nonstandard labels must be specified, but the address of a user routine is omitted. IOCS skips all labels, passes the tapemark, and positions the tape at the first data record to be read.

For condition 4, nonstandard labels and a user address are specified. IOCS cannot distinguish labels from data records because there is no tapemark to indicate the end of the labels. Therefore, to position the tape at the first data record, the user must read all labels.

With nonstandard labels when an end-of-file or an end-of-volume condition exists, the user indicates to IOCS which condition it is. On end-of-file, IOCS branches to the user's end-of-file address. On end-of-volume, IOCS initiates the end-of-volume procedures to close the completed volume and open the next volume for processing.

On output files, nonstandard labels are written by the user's routine by using physical IOCS. The OPEN routine writes a tapemark between the user's nonstandard header labels and his first data record unless the DTF macro instruction has the entry: TPMARK=NO. The close routine writes a tapemark after the user's _last_ data record before he writes his nonstandard trailer labels, and after the trailer labels.

Unlabeled Tape Files

The DTF macro instruction specifies whether the first record of an unlabeled file is a tapemark.

Unlabeled IBM EBCDIC _input_ tape files may or may not have a tapemark as the first record. (If the first record is not a tapemark, IOCS assumes it is a data record.) Any tape that is to be read backward may have a tapemark as the first record on tape. Unlabeled _output_ tape files (written by IOCS) may be written with a tapemark as the first record. ASCII unlabeled tapes do not contain leading tapemarks. A read backward operation is performed to load point for these files by special error recovery procedures.

Note:;Seven-track tapes may be read backward only if they were written in EBCDIC, and they must not have been written in the conversion mode.

When an unlabeled output file is specified, the open routine assumes the mounted scratch tape is also unlabeled. No checking of expiration date is performed. Therefore, _any existing labels, including the volume label, are destroyed._

DASD Label Processing

When a DASD file is processed by logical IOCS, the file must be opened before any transfer of data can be made. The open routines check the DASD labels identifying the file. The open routines also compare information from the actual file labels in the VTOC against the label information supplied by the user in job control cards, and stored in the label information area by job control.

Note: References made in this manual to the // DLBL and // EXTENT job control statements also apply to the // VOL, // DLAB, and // XTENT statements for the 2311, and 2314/2319.

The DTFSD and DTFSR routines process the labels of a sequential file (input or output) one volume at a time. For DTFSR, as each extent is checked, IOCS can pass control to a user's extent exit routine. When the end of the last extent on a volume is reached, an automatic open is issued for the next volume. The DTFDA and DTFIS routines require that all volumes be online for the initial OPEN. DTFPH can be used to process SAM or DAM files. The actual label processing consists of the following operations:

DASD Input Files:

- The volume serial numbers in the volume labels are compared to the volume serial numbers in the DLBL/EXTENT cards.
- The file identification, format identifier, and the file serial number in the format-1 label are compared to the corresponding fields in the DLBL card. The volume sequence number, the creation and expiration dates are then checked against their EBCDIC equivalents in the DLBL card.
- Each of the extent definitions in the format-1 and format-3 labels is checked against the limit fields supplied in the EXTENT cards.
- If user header labels are indicated (when DTFSD, DTFSR, DTFPH, or DTFDA are used), they are read as each volume is opened. After reading each label, the open routine branches to the user's label routine to perform any processing necessary.
- If user trailer labels are indicated (when DTFSD or DTFSR are used), they are read after reaching the end of the last extent on each volume or an end-of-file read by logical IOCS. As with the user header labels, the trailer labels are processed by the user's routine.

DASD Output Files:

- The volume serial numbers in the volume labels are compared to the volume serial numbers in the DLBL/EXTENT cards.
- The extent definitions in all labels in the VTOC are checked to determine whether any extend into those defined in the EXTENT cards. If any do overlap, the expiration date is checked against the <u>current date</u> in the communication region. If the expiration date has passed, the old labels are deleted. If not, the operator is notified of the condition.
- The file names of all entries in the VTOC are compared with the filename in the DLBL statement. If a match is found with an expired file, the expired file is deleted. If a match is found with an unexpired file, the operator is notified.
- The new format-1 label is written with information supplied in the DLBL card. If an indexed sequential file is being processed, the DTFIS table supplies information for the format-2 label.
- The information in the EXTENT cards is placed in the format-1 labels, and (if necessary) additional format-3 labels.
- If user header labels are indicated (when DTFSD, DTFSR, DTFPH, or DTFDA are used), the user's label routine is entered to furnish the labels as each volume is opened. This can be done for as many as eight header labels per volume. As each label is presented, IOCS writes it out on the first track of the first extent of the volume.
- If user trailer labels are indicated (when DTFSD or DTFSR are used), the user's label routine is entered to furnish the labels when the end of the last extent on each volume is reached. This can be done for as many as eight user trailer labels. As each label is presented, IOCS writes it out on the first track of the first extent of the volume. The CLOSE macro instruction must be issued to create trailer labels for the last volume of a file.

Diskette Label Processing

When a diskette file is processed by logical IOCS, the file must be opened before any transfer of data can be made. The open routines check the diskette labels (which identify the file) against the label information supplied by the user in the control cards (stored in the label information area by job control).

A diskette file can be identified by two job control statements: // DLBL and // EXTENT. When the extent limits on a volume are exhausted, an automatic open is issued for the next volume (for DTFDU and DTFPH). DTFPH can be used to process diskette files, feed the diskettes out for a multivolume file, and issue an open to get the new extent limits for the new diskette (both for input and for output).

Diskette Input Files

- The volume serial numbers in the labels are compared to the serial numbers in the DLBL/EXTENT cards.
- If 'file ID' is supplied on the DLBL card, then that file on the diskette is

processed (if found). If 'file ID' is omitted, the DTF name is used.

- Both volume and file security label fields are examined and handled to ensure data integrity.
- All symbolic units specified in the EXTENT cards are checked to ensure that only one physical unit is being addressed. This is necessary to ensure that only one file is open on a diskette.
- The extent limits in the file label are checked for validity; if they are found to be correct, the DTF is initialized.
- For multivolume diskette input files using DTFDU, the extent cards and the multivolume indicator are used in conjunction by the OPEN transients to determine when end-of-file has occurred. If three extents were provided by the user, the following multivolume indicator combination could occur:

Multivolume Indicator	Action by OPEN Transients
, anything	Process first volume and issue warning message.
L, anything	No volumes are processed; issue permanent error message.
C,	Process first volume and issue permanent error message.
C, x	Process first volume and issue permanent error message because file not found.
C, L, anything	Process through the 'L' and issue warning message.
C, C, C	Process through the number of extents. No message.
C, C, L	Process through the 'L'. No message.

In summary, for DTFDU the number of diskettes can be less than the number of extents provided. For all other supported DTF's, processing continues until the number of extents is exhausted. Regardless of the DTF type, for system files processing continues until all extents are exhausted.

Diskette Output Files

- The volume serial numbers in the labels are compared to the serial numbers in the DLBL/EXTENT cards.
- If 'file ID' is supplied on the DLBL card, it will become the name of the new file on the diskette. If 'file ID' is omitted, the DTF name is used.
- Extent limits are determined by OPEN; any expired files that are overlapped by the file to be created are deleted. The operator is informed of any overlap with an unexpired file.
- All file names are compared with the name of the file to be created. If a match is found with an expired file, the file is deleted. The operator is informed of a match with an unexpired file.
- The new HDR1 label is created and written back out onto the diskette.
- If a secured file is being created, the volume label is updated to indicate a secured volume.
- A CLOSE macro instruction must be issued to ensure that all records are written and to update the HDR1 label for the last volume of the file.

COMMON AND SPECIAL PURPOSE LOGICAL IOCS ROUTINES

This section contains detailed descriptions of certain routines generic to logical IOCS. In general, these routines cannot be related to a specific file type or file processing method. Describing LIOCS in four volumes has made it necessary to include details of these routines in Volume 1 even though they may relate to file processing described in other volumes.

Included in this section are:

- TES Processor ($$BOESTV)
- Open Monitor ($$BOPEN, $$BOPEN1, $$BOPEN2, $$BOPEN4, and $$BOPLBL)
- Close Monitor ($$BCLOSE, $$BCLOS2, $$BCLOS4, $$BCLLBL, and $$BCLRPS)
- Open for self-relocating programs ($$BOPENR, and $$BOPNR2, and $$BOPNR3)
- RPS SVA initialization routine ($$BOPENS) and RPS phase loading routine ($$VOPENT).
- DASD File Protect and VTOC Display and Dump routines.
- DASD RPS Common Close ($$BCLRPS)
- Check Duplicate Device Assignments for Logical Units ($$BOPENC)
- Enqueue and Dequeue for VSE/VSAM Routines ($$BENDQB)
- SD Close Input and Output ($$BOSDC1)
- Close, Free Track Function ($$BOSDC2)
- Forced End of Volume for Disk ($$BOSDEV)
- Remove Extents from Extent Block ($$BODQUE)
- Device Release ($$BRELSE)

$$BOESTV: Error Statistics by Tape Volume

Objective: For tape, record TES information from the PUB2 table onto SYSREC as applicable, post the new tape open, and pass control to the next transient.

Entry:

1. From $$BOPEN1 or $$BPCP01 when tape unit ready.
2. From $$BOPEN for job control tape OPEN.
3. From a message writer routine to post OPEN and process new volume label.

Exit:;To next transient.

Method: $$BOESTV tests the device type of the device to be opened. It does the following:

1. The tape label is read and compared with the label currently stored in the PUB2 table for that device.
2. Control is passed to the appropriate exit phase if the tape was previously opened.
3. The tape open bit is posted, the volume serial number in the PUB2 table is saved, and control is passed to the appropriate exit phase if this is the first tape on the device.
4. The tape open bit is posted and control is passed to the appropriate exit routine if the tape is unlabeled, there is no volume ID in the PUB2 table (the previous tape was also unlabeled), and individual recording was not specified.
5. The TES record is written onto SYSREC, the tape open bit is posted, and control is passed to the appropriate exit phase if the tape is unlabeled and either individual recording was specified or the previous tape was labeled.
6. The TES record is written onto SYSREC, the tape open bit is posted, the new volume ID is stored in the PUB2 table, and the appropriate phase is fetched if the tape label read is different from the label in the PUB2 table.

$$BOPEN: Open Monitor

Objective:

1. Initialization of the Logical Transients Common Area and the Fetch RPS Initialization Routine.

2. Tape Error Recording Routine for Job Control open.

Entry:

1. From an OPEN macro expansion in the problem program.
2. From a successfully completed open routine.
3. From the $$BOPENR or $$BOPNR2, DTF relocation routines.
4. From a message writer routine.
5. From the open routine for DTFCP or DTFDI files.

Exits: To $$BOPEN1, $$BOESTV, and $$BOPENS.

Method:

1. If RPS is not yet initialized, $$BOPENS is fetched to do so.
2. $$BOPEN tests the device type of the device to be opened. If the device is a tape, the logical transients common area is initialized for tape open. If $$BOPEN was fetched by job control, an exit is taken to $$BOESTV to do recording. If the open is not for job control, $$BOPEN1 is fetched. If the device is not a tape, initialization of the logical transients common area takes place and $$BOPEN1 is fetched.

$$BOPEN1: Open Monitor Phase 1

Objective: To determine, initialize for, and fetch the proper open routine for DASD, diskette, magnetic ink character recognition (MICR), magnetic tape, optical reader, unit record, and telecommunications files.

Entry: From $$BOPEN, or return from another logical transient.

Exits:

- To $$BOSFBL for DTF type code X'20' and X'21'.
- To $$BOPLBL, and then to $$BOPEN2 for ISAM files.
- To $$B35400 for diskette files.
- To an appropriate open routine if other files are to be opened.
- To a message writer routine if an error has occurred.
- To the problem program if no more files are to be opened.
- To $$BOCP01 for DTFCP printer files.
- To $$BOUR01 for DTFPR/DI printer files.

Method: The $$BOPEN1 phase begins the initialization of the open table located at the end of the logical transient area. The open table is initialized for all file types and passes information to the successive open phases. Next, the type of entry into the $$BOPEN1 phase is determined. If entry was made directly from an OPEN macro, the monitor prepares to open the first file specified in the macro operand. If access control is in the system, the monitor links first to the access control module residing in the SVA. If entry was made from another open phase, the monitor prepares to open the next file specified in the macro operand. If entry to the $$BOPEN1 phase was from a message writer phase or from a device independent file (CP or DI) open phase, processing continues on the current file. At this point $$BOPEN1 checks whether the control block is a DTF or a VSAM ACB by testing the type code (byte 20 of the control block). If the code is X'28', the file being opened is a VSAM file with an ACB control block. In this case, phase $$BOVSAM is called. If the code is anything other than X'28', X'20', X'21', X'22', and X'23', $$BOPEN1 loads and branches to $$BOPIGN.

When $$BOPIGN returns control, $$BOPEN1 determines the type of file being opened from byte 20 of the DTF table. If an invalid file type is detected, message 4880I is printed and the job is canceled. The file type governs the functions that the open monitor must perform to open a particular file:

- Console (DTFCN) files are ignored.
- Unit record (DTFCD, DTFPR, and DTFPT), optical reader (DTFOR), magnetic ink character recognition (DTFMR), compiler (DTFCP), and basic telecommunication access method - extended support (BTAM-ES) files are checked to validate the address limits of the respective DTF tables and the proper open phase is fetched.

For all DTFMT and DTFPH-MT files, $$BOPEN1 fetches the SVA link phase $$BOTSVA to link to the $IJJTTOP SVA phase to complete the OPEN processing.

For DAM and sequential DASD files defined by DTFCP, DTFDA, DTFDI and DTFSD $$BOPEN1 fetches the SVA link phase $$BOSFBL to link the $IJJGTOP SVA phase to complete the OPEN processing.

For diskette files, $$BOPEN1 prepares to read sequential DASD labels from the label area into the logical transient area and fetches diskette open phase $$B35400.

$$BOPEN4: DASD DTF DEV Type Update OPEN Phase

Objective:

1. To locate the PUB for the DASD, using the corresponding LUB pointer.
2. To test the PUB to make sure it is used for a 3340.

3. To check the VOL ID to make sure that the corresponding 3340 is ready and the VOL ID is correct.

Entry: From $$BOPLBL and reentry from $$BOMSG1.

Exits: To $$BOPEN2 to continue OPEN processing for ISAM or to $$BOMSG1 for operator communication.

Method:;The logical unit address in the first type-1 label extent information of an ISAM file defines the correct size for all 3340 data modules containing prime data and/or overflow areas of an ISAM multivolume file. The logical unit address of the first (or only) type-4 label extent information defines the size of the 3340 data module containing the index area(s).

$$BOPIGN: Open Ignore

Objective: To check for the COBOL open ignore option.

Entry: From $$BOPEN1.

Exits:

- To $$BOPEN1 to continue opening the files.
- To $$BOMSG1 if an error occurs.

Method: $$BOPIGN determines if the COBOL _open ignore_ option is specified for the file by testing bit 2 in byte 16 of the DTF table. If the bit is on, a second test determines if the file is either unassigned or assigned _ignored._ If this is the case, the open for the file is bypassed, and control returns to $$BOPEN1 to open the next file. In all other cases, $$BOPIGN validates the address limits of the DTF table, and returns to $$BOPEN1 which continues opening the file.

$$BOPEN2: Open Monitor, Phase 2

Objective: To read label information from the label area for ISAM files, and to fetch the required open phase for the file being opened.

Entry: From $$BOPLBL, $$BOPEN4, or from a message writer phase ($$BOMSG1).

Exits:

- To the required open phase determined by $$BOPEN1.
- To $$BOMSG1 if an error is detected.
- To phase IIPOPEN if an ISAM DTF is linked with a VSAM file.
- To phase $$BOCISC if CDLOAD for IIPOPEN was not successful.

Method: This phase of the Open Monitor reads the label information (stored by Job Control on the label area) into the area obtained by $$BOPLBL through a GETVIS macro.

For ISAM files, $$BOPEN2 of the Open Monitor reads a single DLBL/EXTENT record. This record can contain more than one EXTENT card image. The DLBL label type indicator is checked. If it contains 'V', the file is a VSAM file. In this case the open-active indicator is reset and phase IIPOPEN is loaded using the CDLOAD function. IIPOPEN is part of the ISAM interface program, IIP. The user return address is stored from the user save area into the DTF. The file list pointer is stored into register 0 of the user's save area, control is given to IIPOPEN, and the B-transient area is released. If the DLBL label type indicator contains 'C' or 'F', indicating an ISAM file, the file type is checked against the DTF type. Then the DASD address limits of each extent are checked. Any extent errors cancel the job. When checking of the extent address limits is complete, $$BOPEN2 fetches the appropriate open phase determined by $$BOPEN1.

$$BOPLBL: Open Monitor Label Space Processor

Objective: To determine the size of the read-in area required to process the DLBL/EXTENT information and to issue a GETVIS for the required space
.

Entry: From $$BOPEN1.

Exit:

- To $$BOPEN4 for ISAM.
- To $$BOMSG1 if an error occurs.

Method: $$BOPLBL, at open time, builds a parameter list and calls Symbolic Label Access to determine the amount of DLBL/EXTENT information to be processed. If the space obtained by a previous OPEN or CLOSE in this job step is not sufficient to meet the label processing requirements, a FREEVIS macro is issued to release this space and a GETVIS macro is issued to obtain the required space. Pointers and channel programs are then updated and an exit is taken to the next phase.

$$BOPENR: Relocate DTF Address Constants

Objective: To relocate all DTF address constants from the assembled address into executable main storage addresses.

Entry: From the OPENR macro to the label START.

Exits:

- To $$BOPNR3.

- To the Open Monitor, $$BOPEN, when the last DTF table is processed.

Method: The $$BOPENR routine first determines if modification (relocation) of the DTF address constants is necessary by subtracting the assembled DTF table address from the relocated DTF table address. The relocation factor in register RELOCREG is the result of this operation. If the relocation factor is 0, no relocation is necessary.

If relocation is required and if the DTF has not already been relocated, the relocation indicator in the DTF is turned on. The CCW address in the CCB and the logic module address in the common portion of the DTF are then modified. If the required relocation was accomplished by a previous opening of the file, the entire relocation routine is bypassed for the file.

Following the modification of addresses in the common portion of the DTF, the individual DTF type is determined and the address of the corresponding address modification table is obtained. When the remaining addresses in the DTF have been modified, a branch is made to the ending routine.

The ending routine determines the next operation. If there are more DTFs to be processed, a branch is made to the beginning of the relocation routine to repeat the procedure for the next DTF. If the last DTF has been relocated, the Open Monitor, $$BOPEN, is fetched.

$$BOPENC: Check Duplicate Device Assignments for Logical Units

Objective: To determine if a physical device is assigned to more than one of the logical units specified in the operand of the OPENC macro.

Entry: From an OPENC macro expansion to the label OPENCNAM.

Exits: To the problem program if no error is detected, or to CANCEL if a physical device is assigned to more than one logical unit.

Method: The $$BOPENC phase begins by building a table, called the OPENC table, containing the 2-byte LUB entry for each logical unit specified in the OPENC macro operand. Because the first byte of a LUB entry contains a pointer to a specific PUB (physical device), the byte can be compared to the corresponding byte of any other LUB to determine if a duplicate assignment exists. (Refer to _VSE/Advanced Functions Diagnosis Reference: Supervisor_ for additional information pertaining to LUB and PUB entries.)

The comparison is carried out in the following manner. Byte 0 of the first LUB entry in the OPENC table is compared to the corresponding byte in the second, third, fourth, etc., until the end of the table is reached. Then, byte 0 of the second LUB entry in the OPENC table is compared to the corresponding byte in the third, fourth, fifth, etc., until the end of the table is reached. The procedure is repeated until all of the LUB entries are similarly checked. If an equal comparison is made at any point in the procedure, checking is discontinued, error message 4885I is printed, and the job is canceled.

$$BENDQB: Enqueue and Dequeue for VSE/VSAM Routines

Objective: To enable the VSE/VSAM routines to enqueue and dequeue their OPEN and CLOSE routines in the B-transient area of the supervisor, although these routines are not themselves B-transient routines.

Entry: From a VSE/VSAM routine that issues the ENQB macro.

Exit: To the calling routine that issued the ENQB macro.

Method: When a VSE/VSAM routine issues the ENQB macro, $$BENDQB is fetched (via SVC 2) from the core image library and put into the B-transient area. Control is transferred to $$BENDQB, which temporarily returns control (via SVC 8) to the routine that issued the ENQB macro. (The B-transient area is _not_ released.) When the DEQB macro is issued, control is returned (via SVC 9) to the B-transient routine $$BENDQB, which has been previously loaded into the transient area by the ENQB macro. $$BENDQB now executes an SVC 11 to release the B-transient area and to return to the highest-priority program ready to run. (Note: The ENQB and DEQB macros destroy the original contents of registers 0 and 1.

$$BOPNR2: Relocate DTF Address Constants, Phase 2

Objective: To relocate the address constants in DTFCP, DTFPT, DTFDI, DTFDR, and DTFDU tables.

Entry: From $$BOPNR3.

Exit: To $$BOPEN.

Method: This phase is an extension of $$BOPENR and performs the same function in the same manner.

$$BOPNR3: Relocate DTF Address Constants, Phase 3

Objective: To relocate the address constants of DTFs connected with unit record files.

Entry: From $$BOPENR.

Exits:

- To $$BOPNR2 if other than unit record files still have to be relocated.

- To the Open Monitor, $$BOPEN, if no more files have to be relocated.

MODLOOP (Address Modification) Subroutine

The MODLOOP subroutine performs the actual address modification using an address modification table. The following example of the relocation of a unlabeled work file DTFMT table (see Figure 6) illustrates the operation of the MODLOOP subroutine and the use of the address modification table.

Modification of the address constants starts with those in the common portion of the DTF table. At this time the following registers are loaded:

- BASEREG - with the address of byte 0 of the DTF table (this register is used as a pointer within the DTF table).
- MODREG - with the address of byte 0 of the address modification table at the label COMMON.
- CCWREG - with the address of byte 0 of the DTF table.

The address modification table at the label COMMON contains three hexadecimal bytes, X'020808'. The first byte is a count of the number of address constants (ADCONs) to be modified; two in this case. This count controls the number of times the modification loop is used. The succeeding bytes contain displacement values to update the register, BASEREG.

The first time through the address modification loop, the second byte of the modification table (X'08') is added to the starting address of the DTF (BASEREG) to obtain the location of the CCW address in the CCB to which the relocation factor (RELOCREG) is added. The count of address constants to be modified is then reduced by 1, and the modification loop is entered a second time. Upon reentering the modification loop, the BASEREG contains the starting address of the DTF+8 to which is added the third byte of the modification table (X'08'). As a result, BASEREG then points to byte 16 in the DTF table, that is, to the logic module address. The relocation factor is added to this address and the count of address constants to be modified is again reduced by 1. Since the count now goes to 0, an exit is made from the modification loop.

After determining that the DTF type is a DTFMT work file, the MODLOOP subroutine is again used. This time the register MODREG is loaded with the address of byte 0 of the address modification table at the label MAGWORK which contains four hexadecimal bytes, X'030C040C'. This means that three address constants (the address of the EOF routine, the data address in the CCW, and the address of the error routine) are to be modified. The register BASEREG contains the starting address of the DTF+16 (carried over from the modification of addresses in the common portion of the DTF). To this is added the second byte of the MAGWORK address modification table (X'0C'). As a result, BASEREG contains the location of the EOF routine address (that is, 16 + 12), or byte 28.

Byte	Bits	Function
0-15 (0-F)		CCB.
16(10)		X'08' indicates DTF relocated by OPENR.
17-19 (11-13)		Address of logic module.
20(14)		DTF type (X'10')
21(15)	0	1 = No rewind.
	1	1 = Unload rewind.
	2	1 = Work file.
	3	1 = Read backward.
	4	1 = Write.
	5	1 = POINTW.
	6	1 = Force checking of read or write.
	7	1 = Forward space before next operation.
22-23 (16-17)		Not used.
24-25 (18-19)		Record length.
26-27 (1A-1B)		Maximum BLKSIZE.
28(1C)		Read op code.
29-31 (1D-1F)		EOF address.
32-39 (20-27)		CCW.
40-43 (28-2B)		Block count, initialized 00000000 for read forward, 00400000 for read backward.
44(2C)	0	1 = Error routine.
	1	1 = Ignore.
	2	1 = Read next record switch
	3	1 = Record fixed unblocked.
	4-7	Not used.
45-47 (2D-2F)		Address of error routine.
Numbers in parentheses are displacements in hexadecimal notation.		

Figure 6. DTFMT Unlabeled Workfile Format

Note: Register BASEREG points to the start of a 4-byte field, the last three bytes of which contain the address of interest.

The relocation factor (RELOCREG) is then added to the address constant. This procedure is repeated for the remaining two address constants in the DTF table.

$$BOPENS: RPS SVA Initialization Routine

Objective: To load the RPS local directory list and phase loading routine into the SVA, if this routine was called by $$BOPEN during the first DASD open.

Entry: From IPL and $$BOPEN.

Exits: To IPL and $$BOPEN.

Method: When called by IPL, the SVA initialization routine returns immediatly.

If this routine was called by $$BOPEN during the first open of a DASD file, space is obtained from the SVA, and the local directory list and the phase loading routine are loaded into the GETVIS area of the SVA. A SYSCOM indicator (displacement X'FC') is set when all operations are completed successfully, or when either the GETVIS or load operations fail.

$$BOPENS exits back to IPL with an SVC 11 or to $$BOPEN with an SVC 2.

$$VOPENT: RPS Phase Loading Routine

Objective: To locate in or load into the SVA the RPS phases for all access methods, when called by an open transient. To remove RPS phases and release SVA space for a terminating job, when called by $IJBEOT.

Entry: From open transients when RPS support is provided for a DTF. From $IJBEOT when a job terminates.

Exit: To the calling transient.

Method: When called by an open transient, the RPS phase loading routine issues a load to search the RPS local directory list for the required phase. If the phase is not in the SVA, a GETVIS is issued to acquire space and the phase is loaded. Exit is taken to the calling transient with the load address of the phase or an unsuccessful condition code set.

When called by $IJBEOT, the routine searches the RPS local directory list for phases that were loaded into the GETVIS area of the SVA for a terminating job. If this is the last job requiring the phase, the SVA space is released and the directory entry is set to inactive. On return to $IJBEOT no condition codes are set.

$$BCLOSE: Close Monitor, Phase 1

Objective: To determine the DTF file type and to fetch the proper close phase.

Entry: From a problem program CLOSE macro expansion, or from a successful CLOSE if more than one file is specified by the same CLOSE macro instruction. In addition, $$BPCLOS enters $$BCLOSE at EOJ to close any unclosed 3800 printer extended buffering DTFs.

Exits:

- To the appropriate close phase.
- To the message writer if an error is detected.
- To the problem program if no files remain to be closed.
- To phase 2 of the Close Monitor, $$BCLOS2.
- To $$BPCLOS when $$BCLOSE was originally invoked by $$BPCLOS.

Method: The first phase of the Close Monitor begins the initialization of a table, located at the end of the logical transient area, for the close operation. This table is called the <u>open table</u> even though it is used by both initialization (open) and termination (close) phases. Files requiring label processing, except for sequential DASD, also enter information into the GETVIS label area.

Next, the $$BCLOSE phase validates the address of the first 44 bytes of the DTF table for all file types except VSE/VSAM files; for VSE/VSAM files, phase $$BCVSAM is called. For magnetic tape (DTFDI, and DTFCP), unit record (DTFCD, DTFPT, DTFCN, and DTFPR), optical reader (DTFOR), and magnetic ink character recognition (DTFMR) files, $$BCLOSE fetches the second phase of the Close Monitor, $$BCLOS2.

For all sequential DASD files $$BCLOSE fetches the SVA link phase $$BOSFBL to link to the $IJJGTOP SVA phase to complete the close processing. For ISAM and DAM DTFs $$BCLLBL is called which in turn calls $$BCLOS4 for ISAM DTFs and $$BCLRPS for DAM DTFs.

For all DTFMT and DTFPH-MT files $$BCLOSE fetches the SVA link phase $$BOTSVA to link to the $IJJTTOP SVA phase to complete the close processing.

For diskette files, $$BCLOSE reads label information into the transient label area at the beginning of the open table, saves address of this area in the open table for use by the next close phase, and fetches the diskette close phase $$BODIO4.

$$BCLOS2: Close Monitor, Phase 2

Objective: To initiate the proper close procedure for unit record, optical reader, MICR, and Optical Reader/Sorter files.

Entry: From phase 1 of the Close Monitor, $$BCLOSE.

Exits:

- To phase 1 of the Close Monitor, $$BCLOSE, to handle next DTF if any.
- To $$BCLOSP for punch and paper tape files.
- To $$BCTC01 for BTAM-ES telecommunication files.
- To $$BCCPT1 for magnetic tape (DTFCP, DTFDI) files.
- To $$BCMR01 for magnetic ink character recognition (MICR) type files.
- To the message writer phase, $$BOMSG1, if an invalid file type is detected.
- To IJDPR3 for printer files opened in extended buffering mode.
- To IJDPRT for PRT1 or 3800 printer files opened with DTFPR/CP/DI.

Method: The function performed by the second phase of the Close Monitor depends upon the file type:

- For files opened to a 3800 printer, $$BCLOS2 enters module IJDPR3 (residing in the SVA) at offset 32 to perform close processing related to the 3800 printer. The address of IJDPR3 is obtained from the Anchor Table Extension (ATX). The address of the ATX is obtained by issuing a CDLOAD for phase IJDANCHX. IJDPR3 is called only if the OPN3800 bit in COMRG is on, indicating that one or more files were opened in 3800 printer extended buffering mode.
- For optical reader and unit records files, except paper tape and DTFCD punch files, the only function performed by phase $$BCLOS2 is to turn off the open indicator in the DTF table for the file being closed.
- For DTFCD punch files, after turning off the open indicator, $$BCLOS2 fetches phase $$BCLOSP if error recovery is possible.
- For DTFCP and DTFDI magnetic tape files, $$BCLOS2 fetches phase $$BCCPT1 after first checking to determine whether or not tape error statistics by volume are being collected. For DTFCP and DTFDI punch files, phase $$BCLOSP is fetched.
- For BTAM-ES telecommunication files, $$BCLOS2 fetches phase $$BCTC01.
- For 3505 or 3525 with OMR or RCE specified, $$BCLOS2 resets the device to the normal mode.

$$BCLOS4: Close Monitor, Phase 4

Objective: To determine the DTF file type and to fetch the proper close phase for ISAM files.

Entry: From $$BCLOSE.

Exits:

- To the appropriate close phase.
- To the message writer if an error is detected.
- To the problem program if no files remain to be closed.
- To phase IIPCLOSE if an ISAM DTF is linked with a VSE/VSAM file.
- To phase $$BOCISC if CDLOAD for IIPCLOSE was not successful.

Method: This phase of the Close Monitor begins the initialization of a table, located at the end of the logical transient area, for the close operation. This table is called the <u>open table</u> even though it is used by both initialization (open) and termination (close) phases. Files requiring label processing, except for sequential DASD, also enter information into the GETVIS label area.

For ISAM DTFs, byte 16 bit 0 of the DTF table is checked. This bit is set to one by phase ISCOPEN if the ISAM DTF is linked with a VSE/VSAM file. In that case the close-active indicator is reset and phase IIPCLOSE is loaded using the CDLOAD function. IIPCLOSE is a part of the ISAM Interface program, IIP. The user return address is stored from the user save area into the DTF, the file list pointer is stored into register 0 of the user save area, control is given to ISCCLOSE, and the B-transient area is released.

For all ISAM DTFs not linked to a VSE/VSAM file, $$BCLOS4 reads label information from the label information area into the open table for use by the next phase, and fetches the ISAM close phase $$BCISOA.

$$BCLLBL: Close Monitor Label Space Processor

Objective: To determine the size of the read-in area required to process the DLBL/EXTENT information and to issue a GETVIS for the required space.

Entry: From $$BCLOSE.

Exit:

- To $$BCLOS4 for an ISAM file.
- To $$BOMSG1 if an error occurs.

Method: $$BCLLBL, at close time, builds a parameter list and calls Symbolic Label Access to determine the amount of DLBL/EXTENT information to be processed. If the space obtained by a previous OPEN or CLOSE in this job step is not sufficient to meet the label processing requirements, a FREEVIS macro is issued to release this space and a GETVIS macro is issued to obtain the required space. Pointers and channel programs are then updated and an exit is taken to the next phase.

$$BCLRPS: DASD RPS Common Close

Objective: To reestablish the original DTF that was modified for ISAM/RPS or for DAM DASDs.

Entry:

- From $$BCLLBL for DAM or ISAM DTFs.

Exits:

- To $$BCLOSE for direct access or IOCS type DTFs.
- To $$BODACL for direct access type DTFs with user trailer labels.
- To $$BOISOA for indexed sequential access type DTFs.

Method: This routine is called when the DTF for the device being closed was modified to support RPS.

All access methods use this routine. Therefore, it is necessary to first determine the DTF type, since the displacements are different in each case. Refer to Figure 7 and Figure 8 on page 47.

The addresses of the original logic module and channel program are restored in the DTF. The bits indicating an RPS DTF and that it has been extended into the virtual area are turned off. The user save area that was obtained for the DTF extension is freed, and the use count for the RPS logic module is decremented.

$$BOSDC1: SD Close Input and Output

Objective: To restore the DTF to its original state in the event the file was not opened.

Entry: From $$BCLOSE.

Exits:

- To the CLOSE Monitor, $$BCLOSE.

Method:;This routine is only entered if the file was not opened successfully. It restores the DTF to its original state and returns to $$BCLOSE to process another DTF.

$$BOSDC2: Close: Free Track Function

Objective: To free any tracks held by the file being closed.

Entry: From ISAM CLOSE.

Exits:

- To the close monitor, $$BCLOSE.
- To $$BCISOA for ISAM files.
- To the problem program.

Method: This routine searches the track hold table to determine whether a track is being held by the file being closed. If so, an SVC 36 is issued to free the track. If another file remains to be closed, control returns to the close monitor, $$BCLOSE. If ISAM files are being processed, control returns to $$BCISOA. Otherwise, control returns to the problem program.

$$BOSDEV: SD Close

Objective: When FEOVD has been specified, $$BOSDEV closes the current volume and opens a new volume.

Entry:

- From the FEOVD macro.

DTF	DTFDA no trailer labels and DTFPH	DTFDA with trailer labels	DTFIS (all)
Type Code	22,23	22	24,25 26,27
Byte	32(20)	32(20)	64(41)
Dev Type Bit[1]	1	1	4,7[3]
DTF Type Bit[2]	7	7	5
Exit to	$$BCLOSE	$$BODACL	$$BCISOA
Numbers in parentheses are displacements in hexadecimal notation.			

[1] If this bit is set on, the device supports RPS.
[2] If this bit is on, the DTF extends into the partition virtual area.
[3] Bit 4 on indicates prime data. Bit 7 on indicates index.

Figure 7. Use of Different DTF Types by $$BCLRPS

Exits:

- To $$BOPEN.
- To the problem program.

Method: An interface to the OPEN/CLOSE SVA phase is established allowing the FEOVD request to be processed.

$$BODQUE: Remove Extents from Extent Block

Objective: To delete all entries for a particular logical unit from the extent block.

Entry: From the ISAM DASD open phase.

Exit: To the problem program if no files remain to be opened, or to $$BOPEN, unless the name of the phase to be returned to is supplied by the calling phase.

Method: After storing the contents of registers 3 through 8 and, if it is specified, the name of the phase to which control is to be returned, phase $$BODQUE builds the EXTENT macro parameter list. All extent block entries for the logical unit of the current DTF are erased by issuing an EXTENT macro for this logical unit.

Phase $$BODQUE then fetches the calling phase or $$BOPEN, if the name of the calling phase was not supplied and there is another file to be opened. If the name of the calling phase was not supplied and there are no other files to be opened, phase $$BODQUE returns control to the problem program via an SVC 11.

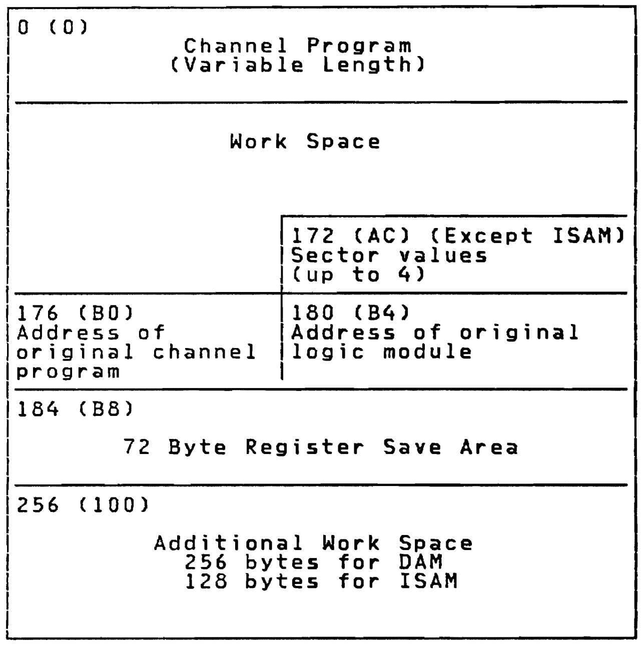


Figure 8. ISAM RPS or DAM DASD Device Independent Extension Work Area

$$BRELSE: Device Release

Objective: To perform the actual device release of the units in the table released by the RELEASE macro.

Entry: From the RELEASE macro.

Exit:

- To the problem program via SVC 11.

Method: To perform the actual device release, the transient sets the unit to the permanent assignment, if one exists. Otherwise, the device is unassigned. If the device is at permanent assignment level, the transient takes no action on the unit.

The PUBOWNER bits of all requested units, for which no other assignments exist, are reset.

COMMONLY USED LOGICAL TRANSIENTS

The logical transients included in this section of the manual are those that pertain to sequential, indexed-sequential, and direct access DASD files.

$$BOFLPT: DASD File-Protect

Objective: To place the upper and lower extent limits into the Extent Block to provide file protection for DASD files.

Entry:;

- From phase $$BOISO7 for ISAM files.

Exits:

- To the open monitor, $$BOPEN, if more files are to be opened and a specific phase name is not supplied.
- To the problem program if a specific phase name is not supplied and no more files remain to be opened.
- To the transient phase specified by the calling phase.

Method: The $$BOFLPT phase provides file protection for DASD files by storing extent limit information in the extent block. Further information pertaining to the extent block and LUBs is found in _VSE/Advanced Functions Diagnosis Reference: Supervisor._

The $$BOFLPT phase begins by determining:

- The number of extents to be processed.
- The addresses of the DLBL-EXTENT card image.
- The file type.
- The device type.

When these factors are known, the phase determines the maximum number of extents per logical unit and the required GETVIS space if the workarea is too small to hold all extents. It loads the extents per logical unit into the workarea and sorts them according to disk addresses. Contiguous extents are combined. The EXTENT macro is used to add extent entries into the extent block. After all logical units are processed, a FREEVIS is issued for the workarea, if necessary.

From information passed by the calling phase, $$BOFLPT determines the next action required and issues either an SVC 2 to fetch the proper transient phase, or an SVC 11 to return to the problem program.

$$BODSPV: VTOC Display, Phase 1

Objective: To determine the logical unit (SYSLOG or SYSLST) on which the operator wants the VTOC displayed, and to print an error message if SYSLST is the unit selected but not assigned to a printer.

Entry: From phases $$BODMS2, $$BODIO8, $$BODSMO, or $$BOMSG2 when the operator's response is DSPLYV.

Exit:

- To the second phase of VTOC display, $$BODSPW. (If a diskette is being displayed, exit is to phase $$BODSPO.)
- To job control via an SVC 11 if the operator's response to message 4V95A is

END or CANCEL and the open was for job control.

- To phase $$BCNCL via an SVC 6 to cancel the job if the operator's response to message 4V96A is END or CANCEL and the open was not for job control.

Method: The first phase of VTOC display issues a message on SYSLOG to determine whether the operator wants the VTOC displayed on SYSLOG or on SYSLST. If the operator's reply is SYSLST, a check is made to ensure that SYSLST is a printer. If SYSLST is not a printer, error message 4V96A is issued. If the VTOC is to be displayed on SYSLST, preparation is made to start the display on a new page. Phase $$BODSPV then fetches phase 2 of VTOC display, $$BODSPW (or, if a diskette is being displayed, $$BODSPV fetches $$BODSPO).

$$BODSPW: VTOC Display, Phase 2

Objective: To display, on either SYSLST or SYSLOG, the VTOC for the volume currently being opened.

Entry: From the first phase of VTOC display, $$BODSPV.

Exit: To $$BOMSG1 or $$BODSMW.

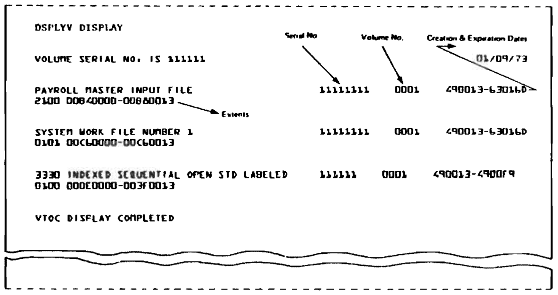
Method: The volume label on the current volume being opened is read to retrieve the pointer (CCHHR address) to the VTOC and the volume serial number. A header line is printed to indicate the date and identify the volume by the volume serial number. Next, the first label in the VTOC (format-4 label) is read to determine the limits of the VTOC, and the CCW chain is initialized to read the file labels (format-1) contained in the VTOC.

The file label for each file on the volume is displayed by printing the contents of the label. The first line printed for each format-1 label contains the first 59 bytes of the label and includes:

- filename
- format identifier
- file serial number
- volume sequence number
- creation date
- expiration date.

Succeeding lines printed for a format-1 label contain extent information. Each line contains a maximum of three extents. (If more than three extents are specified for the file, the additional extents are contained in a format-3 label.) When all extents for a file have been printed, phase $$BODSPW initializes to process the next format-1 label in the same manner.

When all format-1 labels in the VTOC have been processed, the message 'VTOC DISPLAY COMPLETED' is printed and control is passed to $$BOVDMP. Figure 9 is a sample of the VTOC display printed by this phase.


Serial No
Volume No.
Creation & Expiration Dates
Extents

```
DSPLYV DISPLAY

VOLUME SERIAL NO. IS 111111                                        01/09/73

PAYROLL MASTER INPUT FILE                      11111111   0001   490013-630160
2100 00840000-00880013

SYSTEM WORK FILE NUMBER 1                      11111111   0001   490013-630160
0101 00C60000-00C60013

3330 INDEXED SEQUENTIAL OPEN STD LABELED       111111   0001   490013-4900F9
0100 000E0000-003F0013

VTOC DISPLAY COMPLETED
```

Figure 9. VTOC Display of Disk Pack (DSPLYV Response)

$$BODSPO: Diskette VTOC Display

Objective: To display, on either SYSLST or SYSLOG, the VTOC for the diskette currently being opened.

Entry: From the first phase of VTOC display, $$BODSPV.

Exit: To $$BODIO8, $$BODMSG, or $$BODSMO.

Method: The volume label on the volume currently being opened is read to retrieve the volume serial number. A header line is printed to indicate the date and identify the volume by the volume serial number. Next, the CCW chain is initialized to read the file labels (HDR1) contained in the VTOC.

The file label for each file on the volume is displayed by printing the contents of the label. The printed line includes:

- file name
- beginning extent
- end extent
- volume sequence number
- creation date
- expiration date

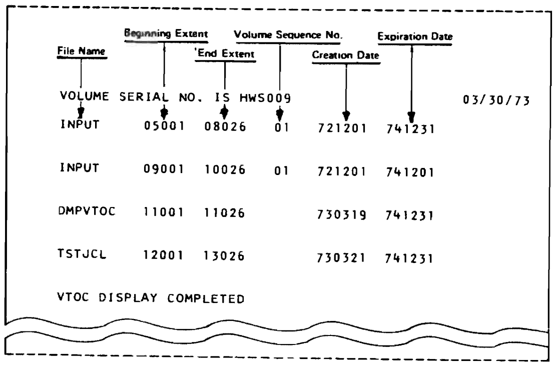
When extents for a file have been printed, phase $$BODSPO initializes to process the next label in the same manner.

When all HDR1 labels in the VTOC have been processed, control is returned to the calling transient. Figure 10 on page 49 is a sample of the VTOC display printed by this phase.

$$BOVDMO: Diskette VTOC Dump

Objective: To provide a list of all the labels in the VTOC for the diskette being opened.

Entry: From phase 2 of the Diskette Open Message Writer, $$BODMS2, or $$BODIO8, when the operator's response is CANCELV, or from the problem program.



```
VOLUME SERIAL NO. IS HWS009                              03/30/73
INPUT       05001   08026   01   721201   741231

INPUT       09001   10026   01   721201   741201

DMPVTOC     11001   11026        730319   741231

TSTJCL      12001   13026        730321   741231

VTOC DISPLAY COMPLETED
```

Figure 10. VTOC Display of Diskette (DSPLYV Response)

Exits: To phase $$BCNCL via SVC 6 to cancel the job if $$BOVDMO is entered from the message writer phase $$BODMS2, or to the problem program, or to $$BOWDMO to continue CANCELV.

Method: Phase $$BOVDMO reads the VOL1 label to retrieve the volume serial number for the volume being opened. A header line is then printed on SYSLST to indicate the date and identify the volume with the volume serial number. If SYSLST is not assigned to a printer, the VTOC Dump is ignored.

$$BOWDMO: Diskette List VTOC

Objective: To provide a listing of all the labels in the VTOC for the diskette.

Entry: From phase 1 of the VTOC dump, $$BOVDMO.

Exits: Control returns to job control or to the user's program.
Figure 11 on page 50 is a sample of the VTOC Dump printed by this phase.

Method: All the VTOC labels for unsecured files (except blank labels) and the file being accessed (whether secured or unsecured) are listed. Any other secured files are not listed. When all labels have been printed, an EOJ message is printed and control returns to the user or to job control.

Note: NB, NS, NP, NE, or NV indicate that a label field is blank. B,S, P, E, or V indicate that the label field was found to be not blank.

$$BODMSG: Diskette Open Error Message Writer Phase 1

Objective: To initialize the message output area, SYSLOG CCB and CCWs, and to fetch phase 2 of the message writer, $$BODMS2.

Entry:

- From the diskette VTOC display phase, $$BODSPO.
- From a diskette open or close phase.
- From the DTFCP open phase, $$BODUCP.

Exit: To phase 2 of the open error message writer, $$BODMS2.

Method: The calling phase supplies the following information to the message writer:

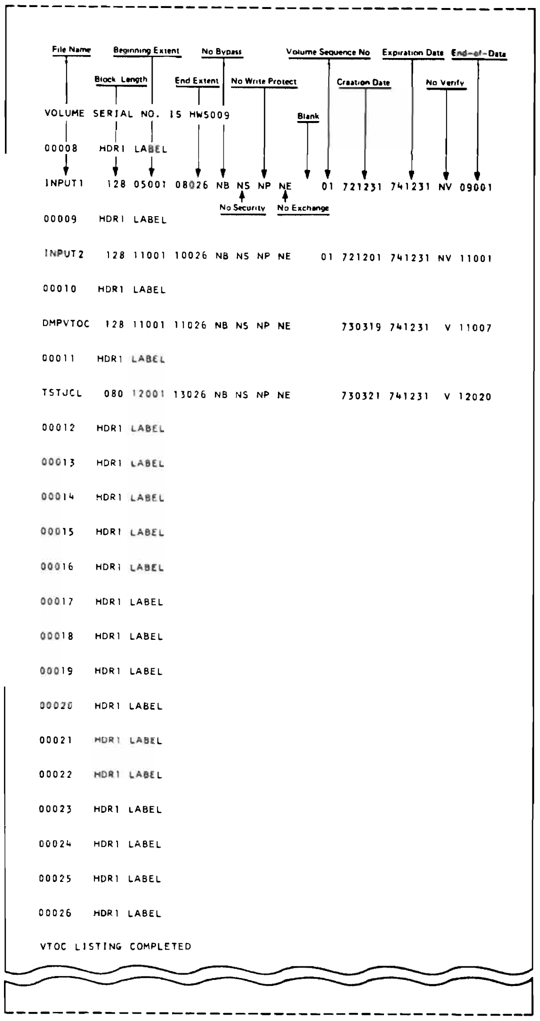
- _Register 0_ contains the last four characters in the name of the phase requesting the message. On cancel messages, register 0 need not be initialized. $$BO is assumed for the first four characters of the phase name.
- _Register 2_ contains the address of the DTF table for the current file.
- _Register 3_ contains the message code (in binary) for the message to be printed. This code is converted to the last two digits of the message number (XX in the example 4nXXI).
- _Transient region + 1185_ contains the numeric decimal value assigned to the various open/close phases for message numbering (X in the example 4XnnI).
- _Transient region + 1000_ contains the start of the CCB.

The message writer overlays the first 888 bytes of the transient region. Therefore, any information that the calling phase needs to save is located beyond this point.

This phase first saves the last four characters in the name of the phase requesting the message. It initializes the SYSLOG message output area with the organization type numeric code, DTF file name, and symbolic unit and constant. It builds the SYSLOG CCWs for writing the message and reading the response, and determines if the required message is in this phase of the message writer. If it is not in this phase, the routine determines which overlay phase contains the message (either $$BOMSG3, $$BOMSG4, $$BOMSG5, $$BOMSG6, or $$BOMSG7) and fetches $$BODMS2 to load the required overlay phase.

$$BODMS2: Diskette Open Error Message Writer, Phase 2

Objectives: To issue an error message to the operator, read the operator's reply (if an IBM 1052 Printer-Keyboard is assigned to SYSLOG) or exit to the phase that requested the message (after ensuring the validity of the operator's response). Also, to cancel the job either by operator request or, if the message type indicates this, by end-of-job.


File Name
Beginning Extent
No Bypass
Volume Sequence No
Expiration Date
End-of-Data
Block Length
End Extent
No Write Protect
Creation Date
No Verify
Blank
No Security
No Exchange

```
VOLUME SERIAL NO. IS HW5009

00008   HDR1 LABEL

INPUT1    128 05001 08026 NB NS NP NE    01 721231 741231 NV 09001

00009   HDR1 LABEL

INPUT2    128 11001 10026 NB NS NP NE    01 721201 741231 NV 11001

00010   HDR1 LABEL

DMPVTOC   128 11001 11026 NB NS NP NE       730319 741231  V 11007

00011   HDR1 LABEL

TSTJCL    080 12001 13026 NB NS NP NE       730321 741231  V 12020

00012   HDR1 LABEL

00013   HDR1 LABEL

00014   HDR1 LABEL

00015   HDR1 LABEL

00016   HDR1 LABEL

00017   HDR1 LABEL

00018   HDR1 LABEL

00019   HDR1 LABEL

00020   HDR1 LABEL

00021   HDR1 LABEL

00022   HDR1 LABEL

00023   HDR1 LABEL

00024   HDR1 LABEL

00025   HDR1 LABEL

00026   HDR1 LABEL

VTOC LISTING COMPLETED
```

Figure 11. VTOC Dump of Diskette (CANCELV Response)

Entry: From phase 1 of the Diskette Open Error Message Writer, $$BODMSG.

Exit:

- To the VTOC dump phase, $$BOVDMO.
- To phase 1 of the VTOC display routine, $$BODSPV.
- To the diskette open/close organization phase requesting the message (if a cancel was not encountered).

Method: $$BODMSG supplied the following information to this phase:

- Register 1 contains the name (last four characters) of the message overlay phase to fetch if the required message appears in some other phase than $$BOMSG1.
- Register 3 contains the address of the message to be written on SYSLOG.

This phase determines the message type. It can be either a file overlap pack, wrong pack, or other.

For wrong-pack type, the message is initialized with the pack number and the wrong-pack switch is turned on. This switch is interrogated later in the routine to test if the operator has mounted the correct pack.

Next, the routine determines if the message to be written on SYSLOG is in main storage. If the message is not in main storage, the message overlay phase containing the required message is loaded into main storage. The message overlay phases consist of $$BOMSG3, $$BOMSG4, $$BOMSG5, $$BOMSG6, and $$BOMSG7. These phases contain messages only. The message is then moved to the SYSLOG output area and an SVC 0 is issued to type the message and read the reply.

If the message indicates the job is not to be canceled, the routine determines if the user wants a VTOC display. If a VTOC display is wanted, the routine issues an SVC 2 to fetch $$BODSPV, the VTOC display phase. If the user does not want a VTOC display, the routine tests for a D-type message.

If the message is a D-type, the message return indicator is set, the address of the next phase name is retrieved, and an SVC 2 is issued to fetch the return phase. If the message is not a D-type, the routine tests the wrong-pack switch as previously mentioned.

The message writer issues an illegal response message for the following conditions:

1. Operator reply of IGNORE for a D-type message.
2. Equal file ID message.
3. No EXTENT to be bypassed.
4. Next pack not mounted.

If the job is to be canceled, a test determines if the job control open switch (in communications region) is on. If so, an SVC 11 is issued to return to job control. If the switch is not on, the routine checks to determine if a request has been made for a VTOC dump. If yes, an SVC 2 is issued to call the VTOC dump transient, $$BOVDMO. If a VTOC dump has not been requested, an SVC 6 is issued and the job is canceled.

Figure 13 on page 54 shows the message code (passed via register 3) together with the last two digits and action indicator of the associated number. For reference purposes, the text of the message is also included.

$$BODSMO: Diskette Data Security Message Writer

Objective: To issue message 4n99D and read the reply from the operator.

Entry: From $$BODSPO, $$BODIO1, $$BODIO5, and return from $$BODSPV.

Exits: The exit depends on the operator's reply to message 4n99D.

- If reply is YES, control returns to the problem program.
- If the reply is EOB, NO, CANCEL, or CANCELV, the problem program is canceled. If a VTOC dump is requested, $$BOVDMO is fetched. If $$BODSMO was fetched by job control, an exit is made to job control.
- If the reply is DSPLYV, $$BODSPV is fetched.

Method: After gathering preliminary data about the calling routine, $$BODSMO issues message 4n99D, 'DATA SECURED FILE/VOLUME ACCESSED'. If the operator types YES on SYSLOG, the file is made available.

$$BOVDMP: VTOC Dump

Objective: To provide a list of all the labels in the VTOC, for the volume being opened.

Entry: From phase 2 of the Disk Open Message Writer, $$BOMSG2, when the operator's response is CANCELV, or from the problem program.

Exits: To phase $$BCNCL via an SVC 6 to cancel the job if $$BOVDMP is entered from the message writer phase $$BOMSG2, or to the problem program, or to $$BOWDMP to continue CANCELV.

Method: Phase $$BOVDMP reads the VOL1 label to retrieve the volume serial number and the CCHHR address of the VTOC for the volume being opened. A header line is then printed on SYSLST to indicate the date and identity of the volume with the volume serial number. If SYSLST is not assigned to a printer, the VTOC Dump is ignored.

$$BOWDMP: List VTOC

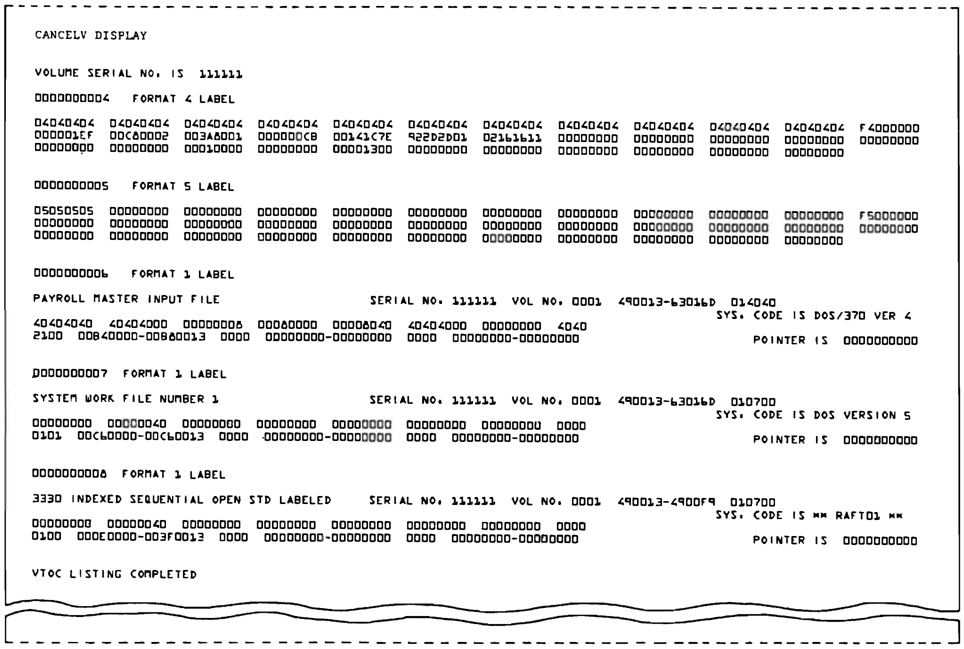
Objective: To provide a listing of all the labels in the VTOC.

Entry: From phase 1 of the VTOC dump, $$BOVDMP.

Exits: If no record if found, exit is to the disk message writer, $$BOMSG1. Otherwise, control returns to job control or to the user's program.

Method: All the VTOC labels for unsecured files (except blank labels) and for the file being accessed (whether secured or unsecured) are listed. Any other secured files are _not_ listed. A maximum of five extents are printed on a line. When all labels have been printed, an EOJ message is printed, and control returns to the user or to job control.

Figure 12 on page 52 is a sample of the VTOC Dump printed by this phase.



```
CANCELV DISPLAY

VOLUME SERIAL NO. IS  111111

0000000004   FORMAT 4 LABEL

04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  F4000000
000001EF  00C80002  003A8001  000000CB  00141C7E  922D2D01  02161611  00000000  00000000  00000000  00000000  00000000
00000000  00000000  00010000  00000000  00001300  00000000  00000000  00000000  00000000  00000000  00000000

0000000005   FORMAT 5 LABEL

05050505  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  F5000000
00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000

0000000006   FORMAT 1 LABEL

PAYROLL MASTER INPUT FILE                 SERIAL NO. 111111  VOL NO. 0001  490013-63016D  014040
                                                                                     SYS. CODE IS DOS/370 VER 4
40404040  40404000  00000008  00080000  00008040  40404000  00000000  4040
2100  00840000-00880013  0000  00000000-00000000  0000  00000000-00000000             POINTER IS  0000000000

0000000007  FORMAT 1 LABEL

SYSTEM WORK FILE NUMBER 1                 SERIAL NO. 111111  VOL NO. 0001  490013-63016D  010700
                                                                                     SYS. CODE IS DOS VERSION 5
00000000  00000040  00000000  00000000  00000000  00000000  00000000  0000
0101  00C60000-00C60013  0000  00000000-00000000  0000  00000000-00000000             POINTER IS  0000000000

0000000008  FORMAT 1 LABEL

3330 INDEXED SEQUENTIAL OPEN STD LABELED   SERIAL NO. 111111  VOL NO. 0001  490013-4900F9  010700
                                                                                     SYS. CODE IS ** RAFT01 **
00000000  00000040  00000000  00000000  00000000  00000000  00000000  0000
0100  000E0000-003F0013  0000  00000000-00000000  0000  00000000-00000000             POINTER IS  0000000000

VTOC LISTING COMPLETED
```

Figure 12. VTOC Dump of Disk Pack (CANCELV Response)

$$BOMSG1 Disk Open Error Message Writer, Phase 1

Objective: To initialize the message output area, SYSLOG CCB and CCWs, and to fetch phase 2 of the message writer, $$BOMSG2 for informational messages. For messages requiring operator action/response, $$BOMSVA is fetched, which in turn transfers control to the SVA.

Entry:

- From a DASD open or close phase.
- From the DTFCP open phases, $$BOCP01, $$BOCP02, $$BOCP11, or $$BOCP12.
- From IJDPRT OPEN routine.

Exit: To phase 2 of the open error message writer, $$BOMSG2, or to $$BOMSVA (see _VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 2_).

Method: The calling phase supplies the following information to the message writer:

- _Register 0_ contains the last four characters in the name of the phase requesting the message. On cancel messages, register 0 need not be initialized. $$BO is assumed for the first four characters of the phase name.
- _Register 2_ contains the address of the DTF table for the current file.
- _Register 3_ contains the message code (in binary) for the message to be printed. This code is converted to the last two digits of the message number (xx in the example 4nxxI).
- _Transient region + 1185_ contains the numeric decimal value assigned to the various open/close phases for message numbering. (x in the example 4xnnI.)
- _Transient region + 1000_ contains the start of the CCB.

The message writer overlays the first 888 bytes of the transient region. Any information that the calling phase needs to save is located beyond that point.

This phase first saves the last four characters in the name of the phase requesting the message. It then checks the message type. For action type messages, $$BOMSVA is fetched in order to transfer control to the SVA. For information type messages, it initializes the SYSLOG message output area with the organization type numeric code, DTF filename and symbolic unit and constant. It builds the SYSLOG CCWs for writing the message and determines if the required message is in this phase of the message writer. If it is not in this phase, the routine determines in which overlay phase the message is located (either $$BOMSG3, $$BOMSG4, $$BOMSG6, $$BOMSG7, or $$BOMSG8) and fetches $$BOMSG2 to load the required overlay phase.

$$BOMSG2: Disk Open Error Message Writer, Phase 2

Objectives: To issue informational error message to the operator, and to cancel the job if the message indicates end of job.

Entry: From phase 1 of the disk open error message writer, $$BOMSG1.

Exit:

- To the DASD open/close organization phase requesting the message.

Method: $$BOMSG1 supplied the following information to this phase:

- Register 1 contains the name (last four characters) of the message overlay phase to be fetched if the required message appears in some phase other than $$BOMSG1.
- Register 3 contains the address of the message to be written on SYSLOG.

This routine determines if the message to be written on SYSLOG is in storage. If the message is not in storage, the message overlay phase containing the required message is loaded into storage. The message overlay phases consist of $$BOMSG3, $$BOMSG4, $$BOMSG5, $$BOMSG6, $$BOMSG7, $$BOMSG8, and $$BOMSG9. These phases contain messages only. The message is then moved to the SYSLOG output area, and an SVC 0 is issued to type the message.

Then, a test determines if the job control open switch (in communications region) is on. If so, an SVC 11 is issued to return to job control. If the switch is not on, an SVC 6 is issued and the job is canceled.

Figure 13 on page 54 shows the message code (passed via register 3) together with the last two digits and action indicator of the associated message number. For reference purposes, the text of the message is also included.

$$BODSMW Data Security Message Writer

Objective: To issue message 4n99D and read the reply from the operator.

Entry: From $$BODSPW, $$BOIS06, $$BORTV1, and return from $$BODSPV.

Exit: To $$BOMSVA (see _VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 2_).

Method: After gathering preliminary data about the calling routine, $$BOMSVA is fetched to transfer control to the SVA.

Message Code	Message Number	Message
0	44A	OVERLAP ON UNEXPRD FILE
1	55A	WRONG PACK, MOUNT nnnnnn
2	40A	EXTENT OVERLAPS ANOTHER
3	41A	EXTENT OVERLAP ON VTOC
4	42A	NO MATCHING EXTENT
5	33A	EQUAL FILE ID IN VTOC
6	66A	1 TRACK USER LBL EXTENT
7	59A	INVALID EXTENT
15	84D	NEED FILE PROTECT RNG
16	31D	VOLUME SEQUENCE ERROR
17	38D	USER HDR LBL IS NOT STD
18	39D	USER TRL LBL IS NOT STD
19	08D	NO UTLO FILE MARK FOUND
20	47A	EXTENTS NOT ON SAME UNIT
21	86D	TAPE UNIT NOT READY
22	00I	NO RECORD FOUND
23	01I	NO RECORD FOUND
24	02I	NO RECORD FOUND
25	03I	NO RECORD FOUND
26	04I	NO RECORD FOUND
27	05I	NO RECORD FOUND
28	06I	NO RECORD FOUND
29	07I	NO RECORD FOUND
31	09I	NO RECORD FOUND
32	00I	NO LABEL SPACE IN VTOC
33	01I	NO FORMAT 1 LABEL FOUND
34	02I	NO FORMAT 2 LABEL FOUND
35	03I	NO FORMAT 3 LABEL FOUND
36	04I	NO FORMAT 4 LBL IN VTOC

Note: A- and D-type messages are not issued by $$BOMSG1 or $$BOMSG 2, but by $IJJGMSG from the SVA.

Figure 13 (Part 1 of 3). Message Code for Disk Open Error Message Writer

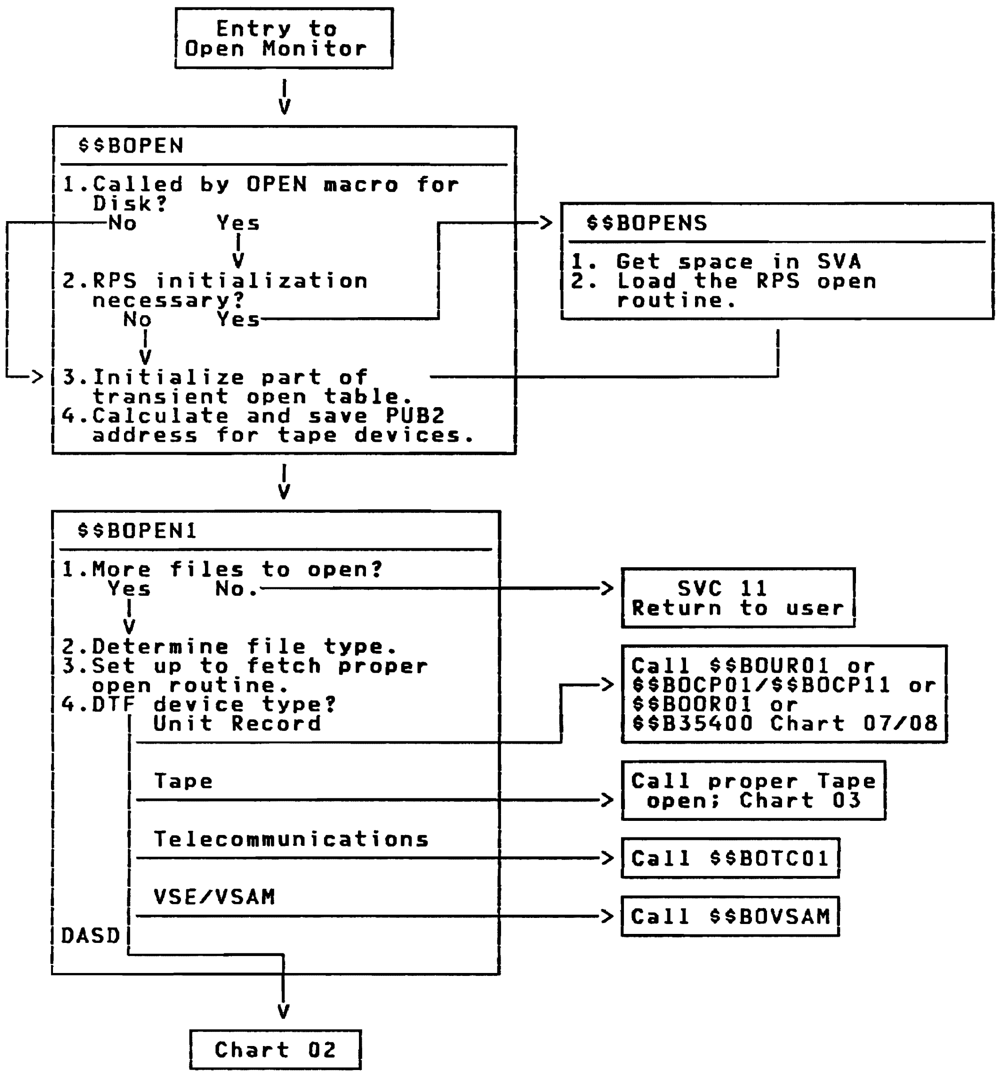
Message Code	Message Number	Message
37	06I	NO STANDARD VOL1 LABEL
38	41I	EXTENT OVERLAP ON VTOC
39	46I	DISCONT INDEX EXTENTS
40	51I	SYSUNITS NOT IN SEQUENCE
41	52I	DISCONT TYPE 1 EXTENTS
42	54I	DSKXTN ENTRY TABLE FULL
43	62I	NO PRIME DATA EXTENT
44	45I	TOO MANY EXTENTS
45	49I	DATA TRACK LIMIT INVALID
46	59I	INVALID EXTENT
47	60I	NO EXTENTS, ALL BYPASSED
48	61I	INVALID DLBL FUNCTION
49	63I	LOAD FILE NOT CLOSED
50	80I	INVALID FILE TYPE
51	81I	NO LABEL INFORMATION
52	83I	INVALID LOGICAL UNIT
53	90I	SVA EXTENT AREA EXHAUSTD
54	87I	SYS FILE EXTENT EXCEEDED
55	35I	DELETED WORKFILE LABEL
56	34I	CURRENT FILE LBL DELETED
57	40I	EXTENT OVERLAPS ANOTHER
58	36I	NO MORE AVAIL/MATCH XTNT

Figure 13 (Part 2 of 3). Message Code for Disk Open Error Message Writer

Message Code	Message Number	Message
59	48I	SYSIN/SYSOUT UNSUPPORTED
60	70I	1ST XTNT CD NOT INDX VOL
61	71I	EXTENT INFO NEEDED
62	72I	MOD AND DTF INCOMPATIBLE
63	58I	NO EXTENT FOR OUTPUT FILE
64	88I	EOF ON SYSTEM FILE
68	98I	OVLAP UNEXPRD SECRD FILE
69	69I	FILE IS OPEN FOR ADD
70	97I	OVLAP EXPIRED SECRD FILE
71	85I	INVALID FORMAT RECORD
74	30I	INVALID HDR1 LABEL
75	33I	EQUAL FILE ID VTOC
76	37I	CHAINING TO SYSTEM UNIT
77	31I	VOLUME SEQUENCE ERROR
80	82I	ISAM NULL FILE
82	74I	BLKSIZE OPEN FAILURE
83	75I	BLKSZ NOT MULT OF RECSZ
85	78I	NO LOGIC MODULE ...
86	79I	GETVIS FAILED
87	05I	UNRECOVERABLE I/O ERROR

Figure 13 (Part 3 of 3). Message Code for Disk Open Error Message Writer

Chart 01. Open Monitor


Entry to Open Monitor
$$BOPEN
1.Called by OPEN macro for Disk?
No
Yes
2.RPS initialization necessary?
No
Yes
3.Initialize part of transient open table.
4.Calculate and save PUB2 address for tape devices.
$$BOPENS
1. Get space in SVA
2. Load the RPS open routine.
$$BOPEN1
1.More files to open?
Yes
No.
2.Determine file type.
3.Set up to fetch proper open routine.
4.DTF device type?
Unit Record
Tape
Telecommunications
VSE/VSAM
DASD
SVC 11 Return to user
Call $$BOUR01 or $$BOCP01/$$BOCP11 or $$BOOR01 or $$B35400 Chart 07/08
Call proper Tape open; Chart 03
Call $$BOTC01
Call $$BOVSAM
Chart 02

Note:
Telecommunications and VSAM are not documented in VSE/AF LIOCS Manuals.

Chart 02. Open Monitor

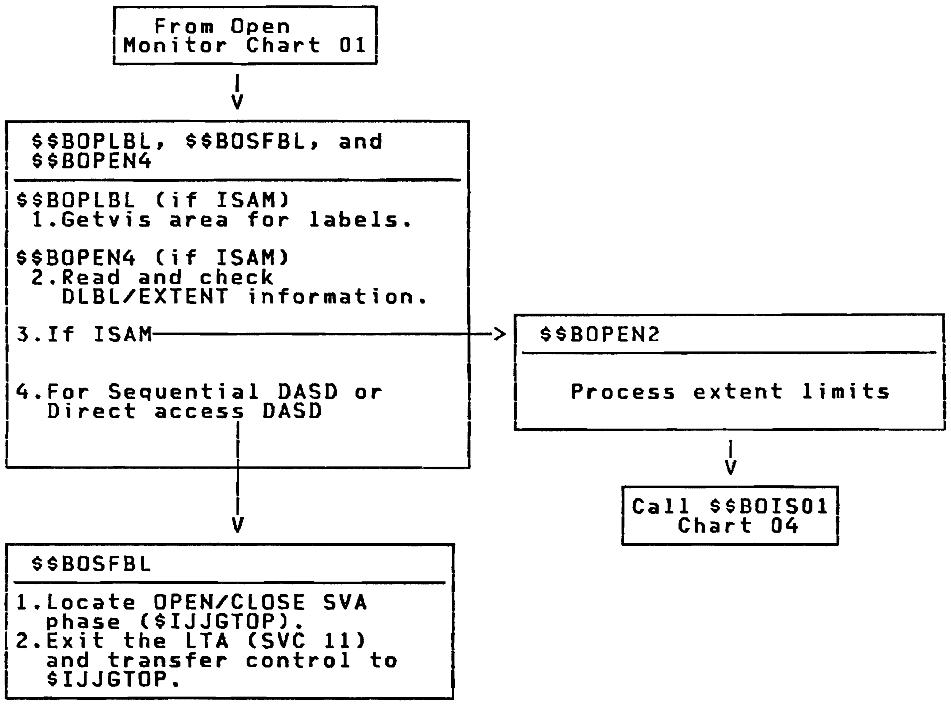

From Open Monitor Chart 01
$$BOPLBL, $$BOSFBL, and $$BOPEN4
$$BOPLBL (if ISAM)
1.Getvis area for labels.
$$BOPEN4 (if ISAM)
2.Read and check DLBL/EXTENT information.
3.If ISAM
4.For Sequential DASD or Direct access DASD
$$BOPEN2
Process extent limits
Call $$BOIS01 Chart 04
$$BOSFBL
1.Locate OPEN/CLOSE SVA phase ($IJJGTOP).
2.Exit the LTA (SVC 11) and transfer control to $IJJGTOP.

Chart 03. Open Magnetic Tape

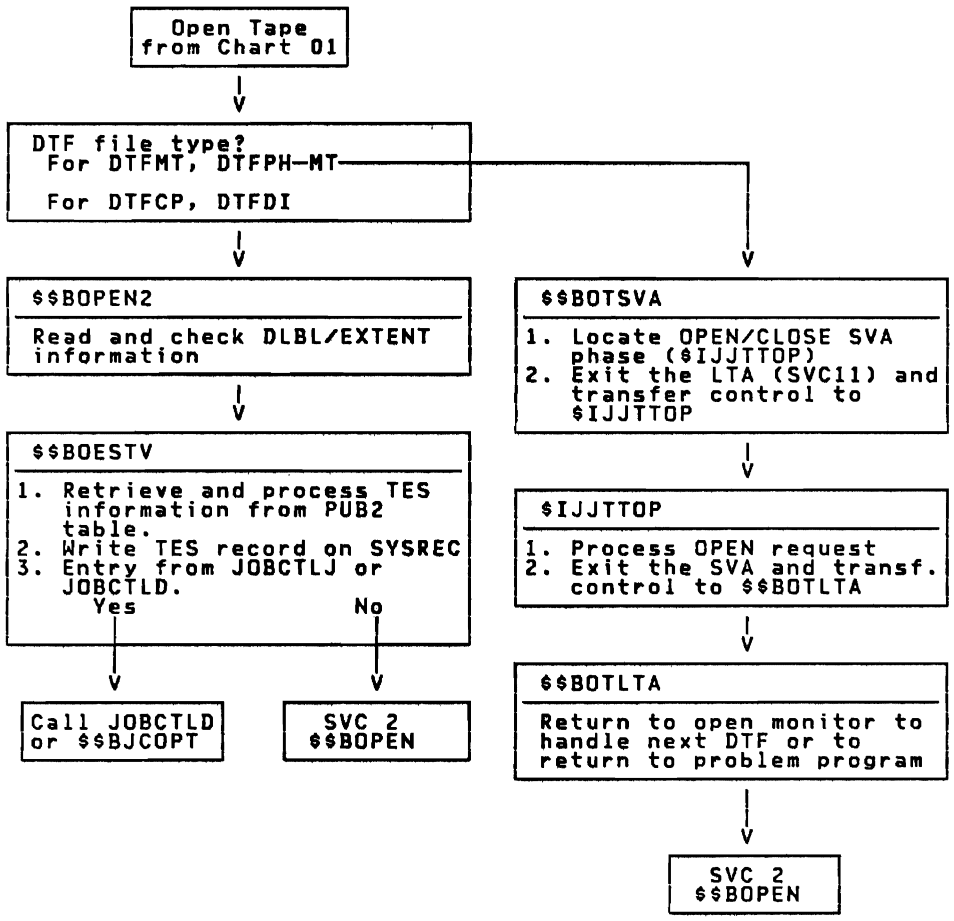

Open Tape from Chart 01
DTF file type?
For DTFMT, DTFPH-MT
For DTFCP, DTFDI
$$BOPEN2
Read and check DLBL/EXTENT information
$$BOESTV
1. Retrieve and process TES information from PUB2 table.
2. Write TES record on SYSREC
3. Entry from JOBCTLJ or JOBCTLD.
Yes
No
Call JOBCTLD or $$BJCOPT
SVC 2 $$BOPEN
$$BOTSVA
1. Locate OPEN/CLOSE SVA phase ($IJJTTOP)
2. Exit the LTA (SVC11) and transfer control to $IJJTTOP
$IJJTTOP
1. Process OPEN request
2. Exit the SVA and transf. control to $$BOTLTA
$$BOTLTA
Return to open monitor to handle next DTF or to return to problem program
SVC 2 $$BOPEN

Chart 04. Open ISAM

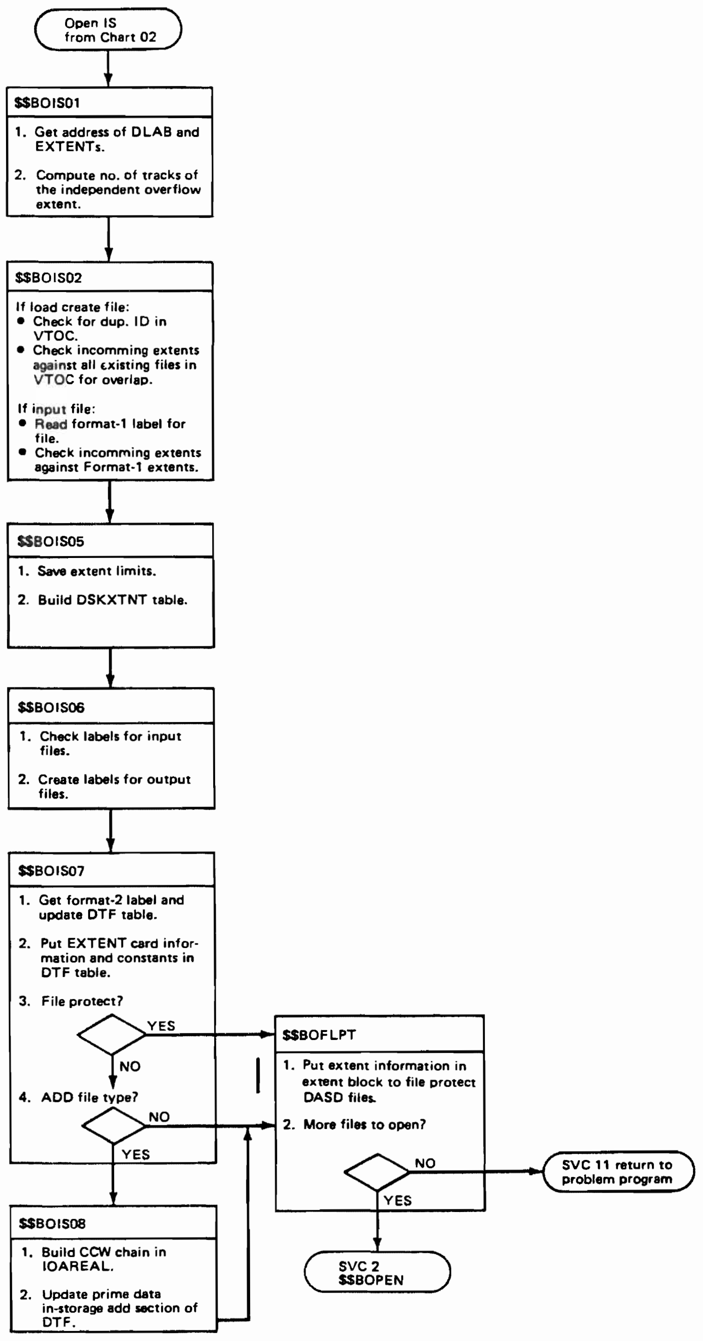
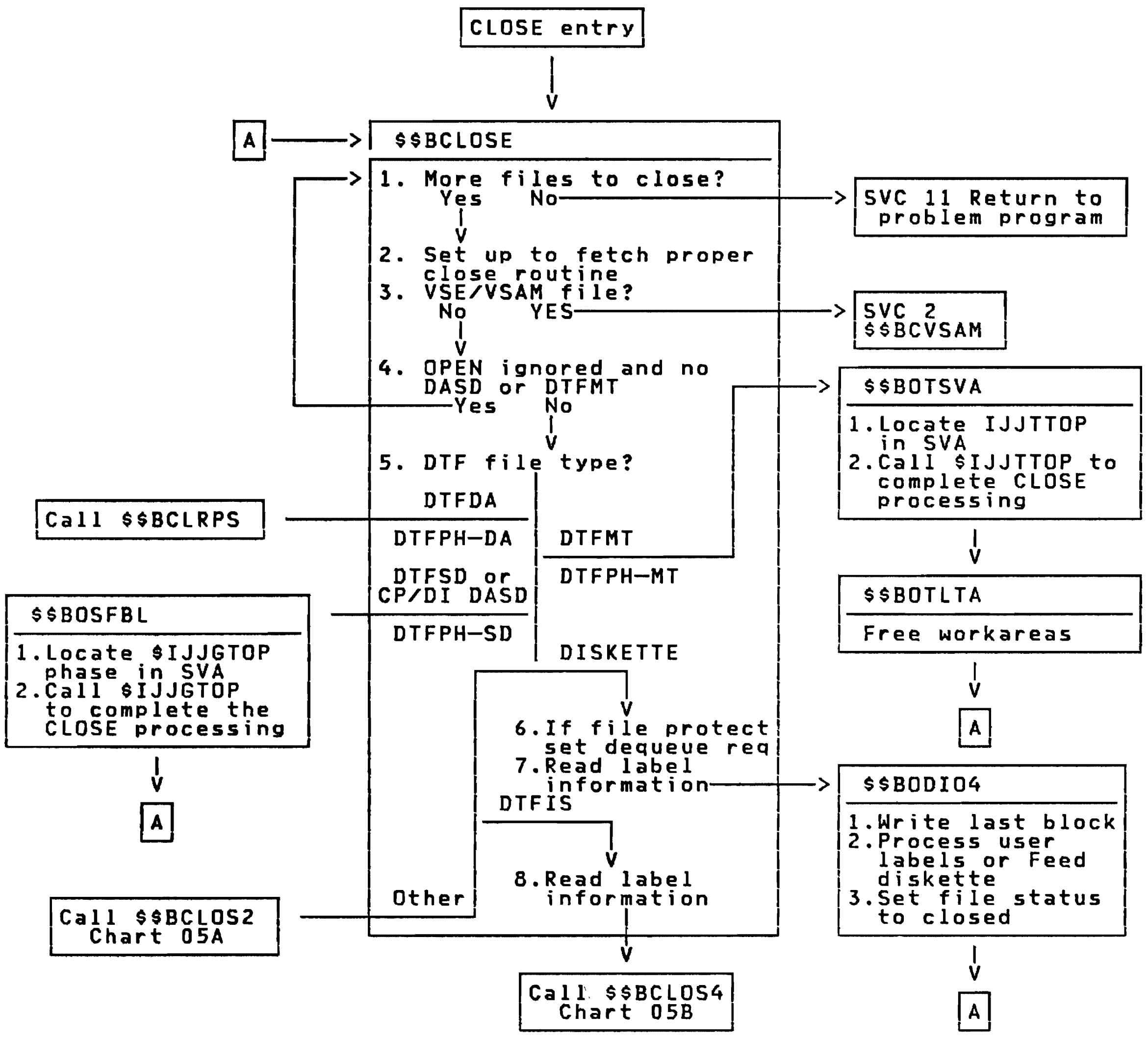
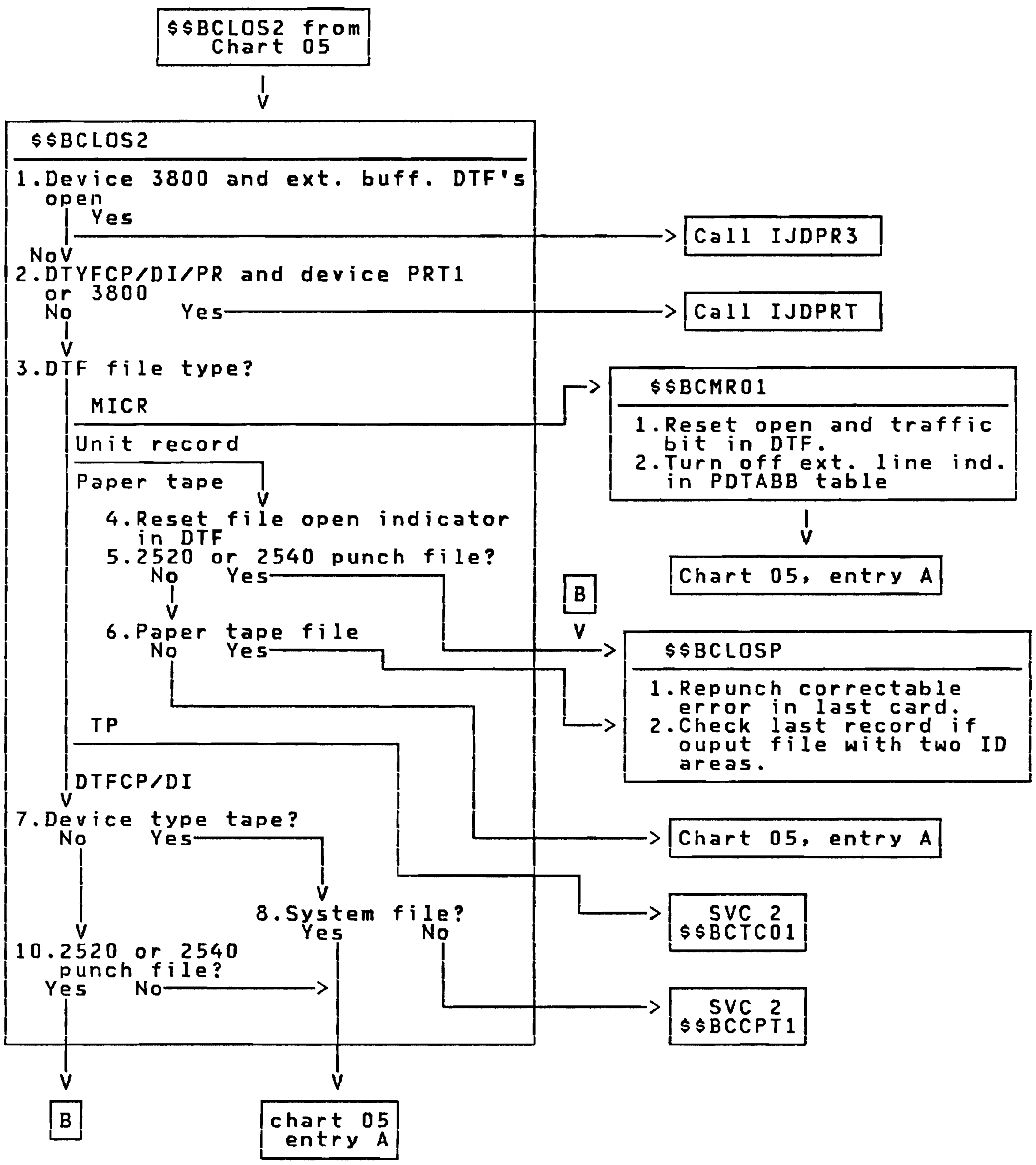

Open IS
from Chart 02
$$BOIS01
1. Get address of DLAB and EXTENTs.
2. Compute no. of tracks of the independent overflow extent.
$$BOIS02
If load create file:
● Check for dup. ID in VTOC.
● Check incomming extents against all existing files in VTOC for overlap.
If input file:
● Read format-1 label for file.
● Check incomming extents against Format-1 extents.
$$BOIS05
1. Save extent limits.
2. Build DSKXTNT table.
$$BOIS06
1. Check labels for input files.
2. Create labels for output files.
$$BOIS07
1. Get format-2 label and update DTF table.
2. Put EXTENT card information and constants in DTF table.
3. File protect?
YES
NO
4. ADD file type?
NO
YES
$$BOFLPT
1. Put extent information in extent block to file protect DASD files.
2. More files to open?
NO
YES
SVC 11 return to problem program
SVC 2
$$BOPEN
$$BOIS08
1. Build CCW chain in IOAREAL.
2. Update prime data in-storage add section of DTF.

Chart 05. Close Monitor



Note:
$$BCLLBL is called to GETVIS the label buffer and to read the label information for DASD files.

Chart 05A. Close Monitor Part 2


$$BCLOS2 from Chart 05
$$BCLOS2
1.Device 3800 and ext. buff. DTF's open
Yes
No
Call IJDPR3
2.DTYFCP/DI/PR and device PRT1 or 3800
No
Yes
Call IJDPRT
3.DTF file type?
MICR
Unit record
Paper tape
$$BCMR01
1.Reset open and traffic bit in DTF.
2.Turn off ext. line ind. in PDTABB table
Chart 05, entry A
4.Reset file open indicator in DTF
5.2520 or 2540 punch file?
No
Yes
B
6.Paper tape file
No
Yes
$$BCLOSP
1.Repunch correctable error in last card.
2.Check last record if ouput file with two ID areas.
TP
DTFCP/DI
7.Device type tape?
No
Yes
Chart 05, entry A
8.System file?
Yes
No
SVC 2 $$BCTC01
10.2520 or 2540 punch file?
Yes
No
SVC 2 $$BCCPT1
B
chart 05 entry A

Note:
$$BCTC01 is not documented in VSE/Advanced Functions LIOCS

Chart 05B. Close Monitor Part 3

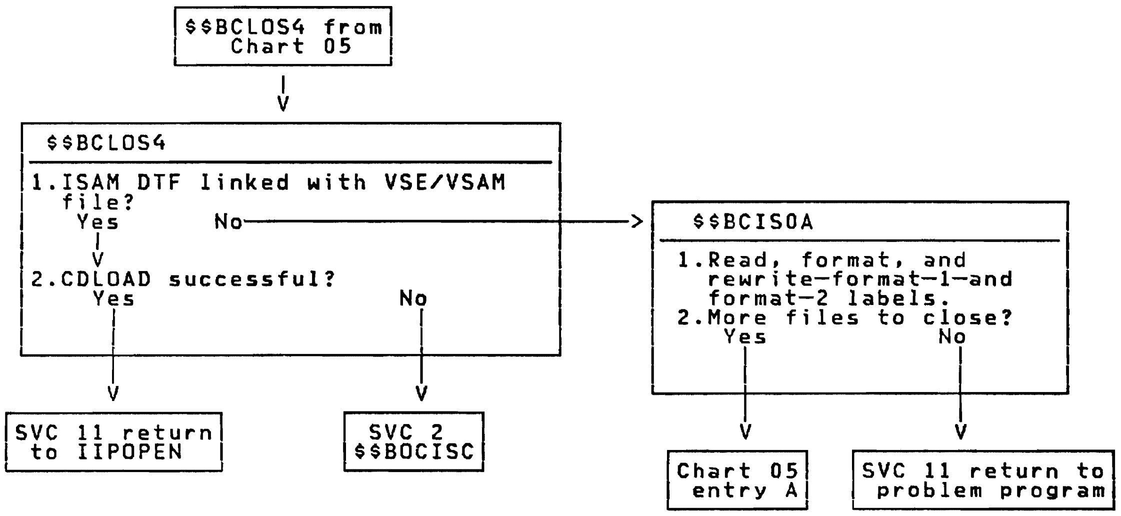


Chart 06. EOF/EOV Routine

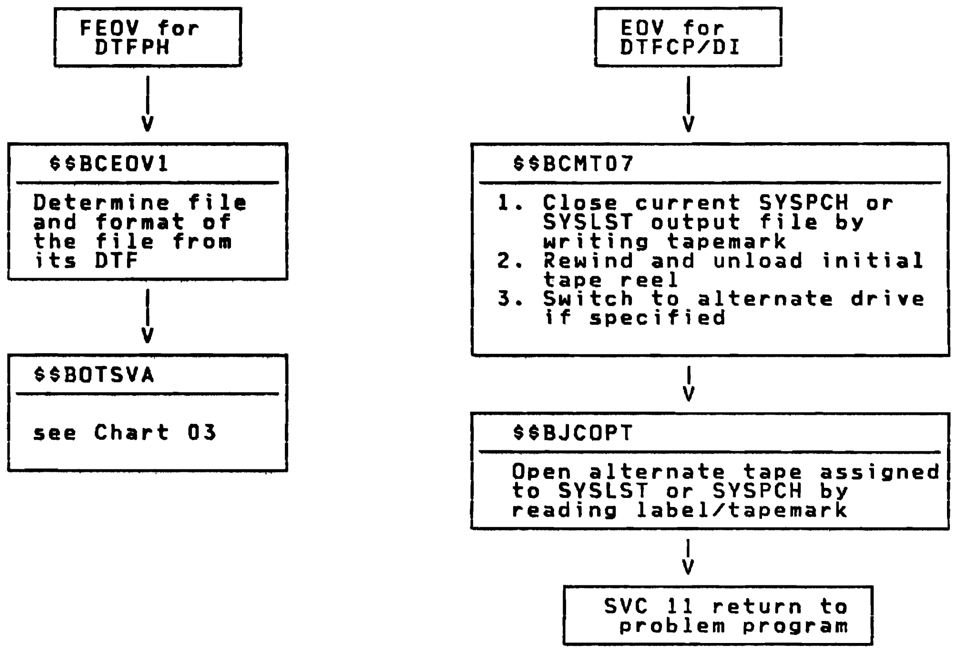

FEOV for DTFPH
$$BCEOV1
Determine file and format of the file from its DTF
$$BOTSVA
see Chart 03
EOV for DTFCP/DI
$$BCMT07
1. Close current SYSPCH or SYSLST output file by writing tapemark
2. Rewind and unload initial tape reel
3. Switch to alternate drive if specified
$$BJCOPT
Open alternate tape assigned to SYSLST or SYSPCH by reading label/tapemark
SVC 11 return to problem program

Chart 07. Open Diskette, Input

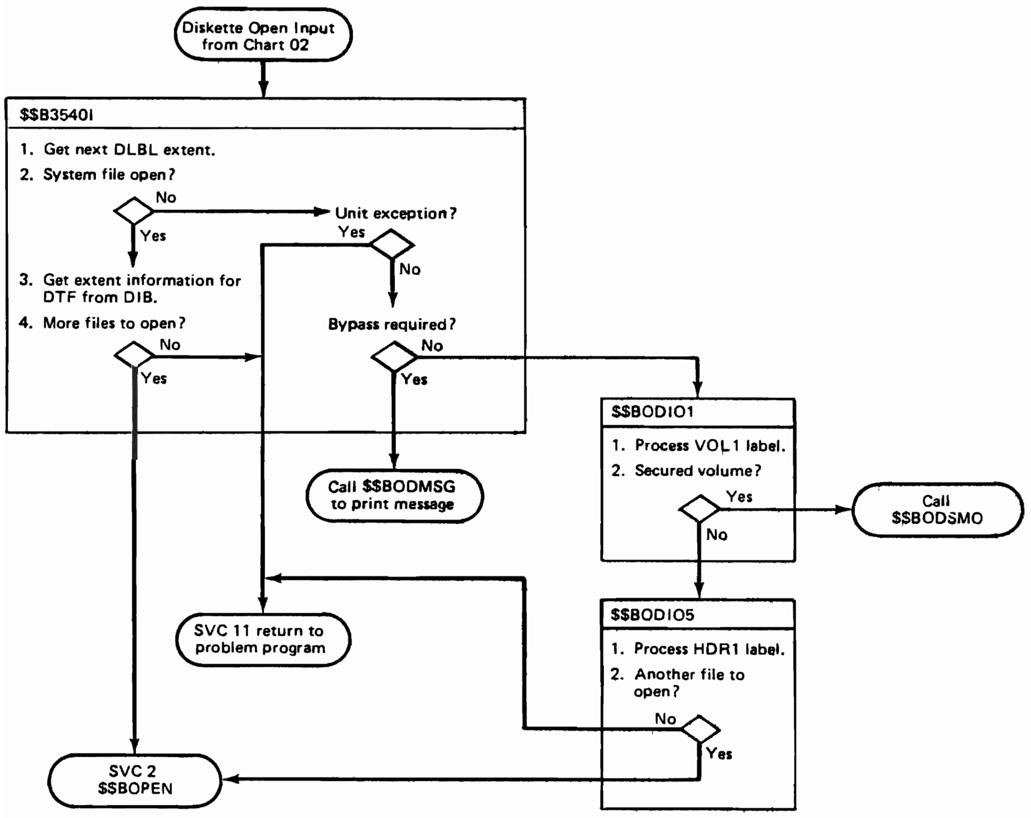
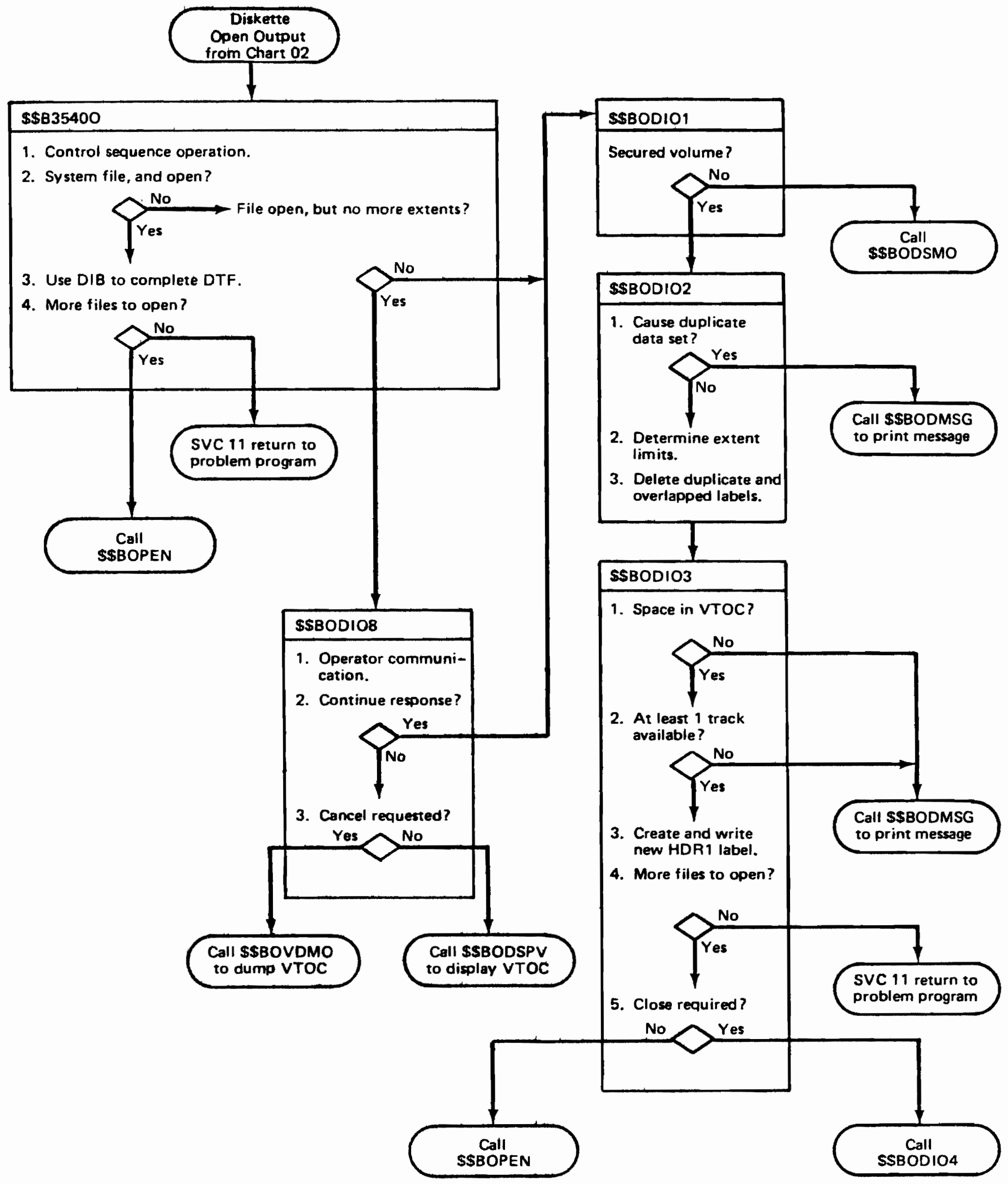


Chart 08. Open Diskette, Output


Diskette Open Output from Chart 02
$$B3540O
1. Control sequence operation.
2. System file, and open?
No
File open, but no more extents?
Yes
3. Use DIB to complete DTF.
4. More files to open?
No
Yes
No
Yes
SVC 11 return to problem program
Call $$BOPEN
$$BODIO1
Secured volume?
No
Yes
Call $$BODSMO
$$BODIO2
1. Cause duplicate data set?
Yes
No
Call $$BODMSG to print message
2. Determine extent limits.
3. Delete duplicate and overlapped labels.
$$BODIO3
1. Space in VTOC?
No
Yes
2. At least 1 track available?
No
Yes
Call $$BODMSG to print message
3. Create and write new HDR1 label.
4. More files to open?
No
Yes
SVC 11 return to problem program
5. Close required?
No
Yes
Call $$BOPEN
Call $$BODIO4
$$BODIO8
1. Operator communication.
2. Continue response?
Yes
No
3. Cancel requested?
Yes
No
Call $$BOVDMO to dump VTOC
Call $$BODSPV to display VTOC

APPENDIX A: MASTER ERROR MESSAGE LIST

The messages in this list are arranged in sequence by message number. The message numbers of all logical IOCS messages start with the digit 4. The second digit of the message number indicates the type of file or routine issuing the message. The indicators are:

0 = Punch file
1 = Magnetic tape file
2 = ISAM
3 = Sequential DASD, diskette - open input
4 = Sequential DASD, diskette - open output
5 = Sequential DASD, diskette - close
6 = DAM - input
7 = DAM - output
8 = Common open/close routines
9 = Sequential DASD - work file
V = VTOC display routines

The alphabetic character after the message number is the action indicator. These indicators are:

Action Indicator	Meaning
A - Action	The operator must perform a specific manual action before the program can continue. For example, mount a tape or ready an I/O device.
D - Decision	The operator must make a choice of alternative courses of action.
I - Information	The message does not require immediate operator action. For example: This type of message can indicate successful completion of a problem program.

The number(s) in the volume column refers to the documentation of the message issuing routine(s) in the following VSE/Advanced Functions Diagnosis Reference manuals:

1. _LIOCS Volume 1: General Information and Imperative Macros,_

2. _LIOCS Volume 2: SAM,_

3. _LIOCS Volume 3: DAM and ISAM,_

4. _LIOCS Volume 4: SAM for DASD_

For further detailed information on these messages, see _VSE/Advanced Functions Messages._

Message Number	Module	Volume	Message
4110A	$$BOCPT3 IJJTOPN	2 2	NO VOL1 LBL FOUND TLBL=xxxxxx filename SYSxxx=cuu
4111I	$$BOCPT4 IJJTOPN	2 2	NO VOL1 LBL FOUND filename SYSxxx=cuu
4112A	$$BOCPT3 $$BOCPT4 IJJTOPN	2 2 2	VOL SERIAL NO. ERROR TLBL=xxxxxx filename SYSxxx=cuu
4113D 4113I	$$BOCPT4 IJJTOPN	2 2	NO HDR1 LBL FOUND filename SYSxxx=cuu
4114A	$$BOCPT4 IJJTOPN	2 2	FILE SEQ NO. ERROR filename SYSxxx=cuu
4115A	$$BCCPT4 IJJTOPN	2 2	FILE SER. NO. ERROR TLBL=xxxxxx filename SYSxxx=cuu
4116A	$$BOCPT4 IJJTOPN	2 2	VOLUME SEQ. NO. ERROR filename SYSxxx=cuu
4117D	IJJTOPN	2	NO TM FOUND ON READBK filename SYSxxx=cuu
4118D 4118I	IJJTOPN	2	FILE ID ERROR, READBK filename SYSxxx=cuu
4119A	$$BOCPT3 IJJTOPN	2 2	FILE UNEXPIRED filename SYSxxx=cuu
4120I	IJJTOPN IJJTSRV	2 2	TAPE POSITIONED WRONG filename SYSxxx=cuu
4122I	IJJTEOF	2	EOV ENCOUNTERED SYSxxx=cuu
4123D 4123I	IJJTOPN	2	WRONG POSITN, READBK filename SYSxxx=cuu
4124I	IJJTSRV	2	TOO MANY UHL'S filename SYSxxx=cuu
4125D 4125I	IJJTOPN	2	VOL1 LBL FOUND filename SYSxxx=cuu
4126I	IJJTSRV	2	EOV ENCOUNTERED filename SYSxxx=cuu
4128I	IJJTOPN IJJTSRV	2 2	ACCESS TO FILE NOT ALLOWED filename SYSxxx=cuu
4130A	IJJTEOF	2	EOF OR EOV INQUIRY filename SYSxxx=cuu
4131D	IJJTEOF	2 2	BLOCK COUNT ERROR filename SYSxxx=cuu DTF=xxxxxx LBL=xxxxxx
4132D	$$BOCPT4 IJJTOPN	2 2	ERROR IN FILE ID filename SYSxxx=cuu
4133D	$$BOCPT4 IJJTOPN	2 2	ERROR IN HDR LBL filename SYSxxx=cuu

Figure 14 (Part 1 of 8). Master Error Message List

Message Number	Module	Chart	Volume	Message
4140A	IJJTSRV		2	NO ALTERN DRIVE ASSGN filename SYSxxx=cuu
4151I	IJJTOPN		2 2	HDR1 LBL INFORMATION filename SYSxxx=cuu
4170A	$$BJCOPT		2	FILE PROTECTED TAPE filename SYSxxx=cuu
4171A	$$BJCOP1		2	UNEXPIRED FILE SYSxxx=cuu
4172A	IJJTOPN		2	INVALID LABEL SET SYSxxx=cuu
4183I	$$BJCOPT IJJTOPN		2 2	INVALID LOGICAL UNIT filename SYSxxx=cuu
4184D	$$BOCPT2 $$BOCPT3 IJJTSRV		2 2 2	NEED FILE PROTECT RNG filename SYSxxx=cuu
4185I	$$BOMRCE $$BOMRCE		2 2	INVALID FORMAT RECORD
4190I	IJJTOPN		2	LOG. UNIT NOT ASSIGNED TO A TAPE filename SYSxxx=cuu
4191I	IJJTLOG IJJTOPN IJJTSRV IJJTTOP		2 2 2 2	ERROR WHILE PROCESSING FILE filename SYSxxx=cuu RC=nn
4192I	IJJTSRV		2	VOLUME ACCESS DENIED filename SYSxxx=cuu
4193I	IJJTSRV		2	FILE ACCESS DENIED filename SYSxxx=cuu
4000I	CDMOD $$BCLOSP $$BCLOSP $$BERRTN		2 2 2 2	RETRY
4n00I	IJJGSDVH	3-93	4	NO LABEL SPACE IN VTOC
4400I	$$BODIO3		2	
4n01I	IJJGSDVH	3-93	4	NO FORMAT 1 LABEL or NO RECORD FOUND
4201I	$$BOIS02 $$BOISOA		3 3	
4301I	$$BOSIO5		2	
4202I	$$BCISOA		3	NO RECORD FOUND
4n03I	IJJGDAI1 IJJGDAI2 IJJGSDI3 IJJGSDI4 IJJGSDW3	3-66 3-67 3-42 3-43 3-48	4	NO FORMAT 3 LABEL FOUND
4n04I	IJJGSDVH	3-93	4	NO FORMAT 4 LBL IN VTOC
4204I	$$BOIS02		3	NO FORMAT 4 LBL IN VTOC or NO RECORD FOUND

Figure 14 (Part 2 of 8). Master Error Message List

Message Number	Module	Chart	Volume	Message
4n05I	$$BOPEN2 $$BOPLBL $$BCLLBL $$BOSDW1 $$BCCPT1 IJJGDARL IJJTSRV	 3-97 	1 1 1 2 2 4 2	UNRECOVERABLE I/O ERROR
4n06I	IJJGSDGC	3-97	4	NO STANDARD VOL 1 LABEL or NO RECORD FOUND
4206I	$$BOIS02 $$BCISOA		3 3	
4306I	$$BODI01		2	
4506I	$$BODI04		2	
4806I	$$BOPEN4		1	
4n07I	IJJGSDRL	3-96	4	NO RECORD FOUND
4307I	$$B35400		2	
4407I	$$B3540I		2	
4n08D/I	$$BOKUL1 $$BIKUL1 $$BOULI1 $$BOULO1	3-11 3-5 3-5.1 3-11.1	4	NO UTLO FILE MARK FOUND or NO RECORD FOUND
4608D	$$BODACL		3	
4329D	$$B3540I		2	EXTENTS NOT EXHAUSTED
4n31D	IJJGSDI2	3-41	4	VOLUME SEQUENCE ERROR
4332I	$$BODI05		2	
4n33D	IJJGDA03 IJJGSD04	3-62 3-30	4	EQUAL FILE IN VTOC
4433D	$$BODI02		2	
4n34I	IJJGSD06 IJJGSDW3	3-32 3-48	4	CURRENT FILE LBL DELETED
4n36I	IJJGSDW3	3-48	4	NO MORE AVAIL/MATCH EXTENT
4337I	$$B3540I $$BODI06		2	CHAINING TO SYSTEM UNIT
4437I	$$B35400 $$BODI07		2	
4n38D	$$BIKUL1 $$BOULI1	3-5 3-5.1	4	USER HDR LBL IS NOT STD
4639D	$$BODACL		3	USER TRL LBL IS NOT STD
4n40D	IJJGSD04 IJJGDA03	3-30 3-62	4	EXTENT OVERLAY ON ANOTHER

Figure 14 (Part 3 of 8). Master Error Message List

Message Number	Module	Chart	Volume	Message
4240I	$$BOIS02		3	EXTENT OVERLAPS ANOTHER
4n41D	IJJGDA03 IJJGSD04	3-62 3-30	4	EXTENT OVERLAP ON VTOC
4241I	$$BOIS02		3	
4n42D	IJJGSDI4	3-43	4	NO MATCHING EXTENT
4243I	$$BORTV1		3	INV EXTENT HI/LO LIMITS
4n44D	IJJGDA03 IJJGSD04	3-62 3-30	4	OVERLAP ON UNEXPIRED FILE
4n45I	IJJGDACX IJJGSDSF IJJGSDXT	3-58 3-23 3-34	4	TOO MANY EXTENTS
4245I	$$BOIS06		3	
4445I	$$BODI08		2	
4246I	$$BOIS07		3	DISCONT INDEX EXTENTS
4n47A	IJJGSDW1	3-46	4	EXTENTS NOT ON SAME UNIT
4n48I	IJJGSDSF	3-23	4	SYSIN/SYSOUT UNSUPPORTED code is still in the modules, but situation cann't occure anymore.
4348I	$$B35400		2	
4249I	$$BOIS05		3	DATA TRACK LIMIT INVALID
4n50D	IJJGSDXT	3-34	4	NO MORE AVAILABLE EXTENTS
4450D	$$BODI08		2	
4n51I	IJJGDACX	3-58	4	SYSUNITS NOT IN SEQUENCE
4252I	$$BOIS05		3	DISCONT TYPE 1 EXTENTS
4n54I	IJJGDART	3-68	4	DSKXTN ENTRY TABLE FULL
4254I	$$BOIS05 $$BORTV2		3	
4n55A	IJJGDAVC IJJGSDLP	3-59 3-98	4	WRONG PACK, MOUNT nnnnnn
4355A	$$BODI01		2	
4855A	$$BOPEN4		1	
4856A	$$BOPEN4		1	WRONG MODULE SIZE
4n58I	IJJGDA01 IJJGSDRL	3-60 3-95	4	NO EXTENT FOR OUTPUT FILE
4358I	$$B35400		2	

Figure 14 (Part 4 of 8). Master Error Message List

Message Number	Module	Chart	Volume	Message
4n59D	IJJGSD02 IJJGSDI4 IJJGDACX IJJGSD05 IJJGSDLP	3-29 3-43 3-58 3-31 3-98	4	INVALID EXTENT
4n59I	IJJGSDRL	3-95	4	
4359I	$$BODIO5 $$BODIO6		2	
4459I	$$BODIO3		2	
4859I	$$BOPEN2		1	
4n60I	IJJGSD01 IJJGSDI1 IJJGSDSF IJJGSDW1 IJJGDA01 IJJGDA02 IJJGDA04	3-28 3-39 3-23 3-46 3-60 3-61 3-63	4	NO EXTENTS, ALL BYPASSED
4360I	$$B3540I		2	
4n61I	IJJGSDRL IJJGDARL	3-95 3-57	4	INVALID DLBL FUNCTION
4261I	$$BOIS01 $$BORTV1		3	
4361I	$$B35400		2	
4861I	$$BOPEN2		1	
4262I	$$BOIS05 $$BORTV1		3	NO PRIME DATA EXTENT
4263I	$$BOIS07		3	LOAD FILE NOT CLOSED
4364I	$$BODIO5 $$BODIO6		2	INVALID HDR1 LABEL
4465I	$$BODIO2		2	EQUAL FILE LABEL IN VTOC
4n66D	IJJGSDI4 IJJGSD05 IJJGDA02	3-43 3-31 3-61	4	1 TRACK USER LBL EXTENT
4266I	$$BOIS05		3	
4n67I	IJJGSD04 IJJGSDVH IJJGVD00 IJJGVD10 IJJGDA03	3-30 3-93 3-77 3-85 3-62	4	CVH PROCESSING FAILURE
4n68D	IJJGSDI4 IJJGSD05	3-43 3-31	4	USER LBLS EXHAUST FIRST EXTENT
4n68D	IJJGDA02	3-61	4	
4269I	$$BOIS07		3	FILE IS OPEN FOR ADD
4270I	$$BORTV2		3	1ST XTNT CD NOT INDX VOL

Figure 14 (Part 5 of 8). Master Error Message List

Message Number	Module	Chart	Volume	Message
4271I	$$BOIS01		3	EXTENT INFO NEEDED
4272I	$$BOIS08		3	MOD AND DTF INCOMPATIBLE
4n74I	IJJGSDBS IJJGSDW1 IJJGDAMX	3-37 3-46 3-52	4	BLKSIZE OPEN FAILURE
4n75I	IJJGSDBS IJJGSDI2	3-37 3-41	4	BLKSIZE NOT MULT OF RECSIZE
4n76D	IJJGSDLP IJJGDAVC	3-98 3-59	4	VOL SER NOT XXXXXX
4n77D	IJJGSDXT	3-34	4	EXTENT ENTRY ERROR - RETRY
4n79I	$$BOFLPT $$BOPLBL $$BCLLBL $$BOPEN2		1	GETVIS FAILED
	$$BCEOV1 $$BCCPT1 $$BOTSVA IJDPRT IJJTSRV		2	
	$$BOSVLT $$BOSFBL IJJGMFBA IJJGSDVH IJJGSDUL IJJGDARL IJJGDAI2 IJJGVD00 IJJGVD10 IJJGSDFP IJJGMMBF IJJGDAMX	3-4 3-3 3-18 3-93 3-100 3-57 3-67 3-77 3-85 3-102 3-21 3-52	4	
4n80I	$$BOSFBL	3-3	4	INVALID FILE TYPE
4880I	$$BOPEN1 $$BCLOS2		1	
	$$BCEOV1		2	
4n81I	IJJGSDRL IJJGSDMO IJJGSDSF IJJGDARL	3-95 3-22 3-23 3-57	4	NO LABEL INFORMATION
4881I	CLOSE $$BCLLBL $$BOPEN2 $$BOPLBL		1	
	$$BCCPT1 $$BCEOV1 $$B35400 IJJTOPN		2	
4282I	$$BOIS07		3	ISAM NULL FILE

Figure 14 (Part 6 of 8). Master Error Message List

Message Number	Module	Chart	Volume	Message
4n83I	IJJGSDGC IJJGSDNV IJJGSDRL IJJGDACX	3-97 3-99 3-95 3-58	4	INVALID LOGICAL UNIT
4383I	$$B3540I		2	
4483I	$$B3540O		2	
4883I	$$BOPEN4		1	
	$$BOCP01 $$BOCP02 $$BOCP11 $$BOCP12 $$BOUR01		2	
4884D	$$BOPEN1		1	NEED FILE PROTECT RNG filename SYSxxx=cuu
	$$BOCP02 $$BOCP11 $$BOCP12		2	
4885I	$$BOPENC		1	SYSxxx AND SYSyyy ARE ASSIGNED TO THE SAME PHYSICAL UNIT
4n86D	$$BOPEN1 IJJTSRV		1 2	TAPE UNIT NOT READY
4887I	$$BERRTN		2	SYS FILE EXTENT EXCEEDED
4888I	$$BERRTN		2	EOF ON SYSTEM FILE
4n89I	IJJGSDSF	3-23	4	WORKFILE NOT SUPPORTED FOR SYSFIL
4n90I	IJJGSDVH IJJGVD00 IJJGVD10 IJJGSDFP	3-93 3-77 3-85 3-102	4	SVA EXTENT AREA EXHAUSTED
4890I	$$BOFLPT		1	
4n93I	IJJGSDRL	3-96	4	UNRECOVERABLE I/O ERROR
4n94I	IJJGSDCI IJJGMIOI	3-94 3-24	4	CISIZE INCORRECT
4n95I	IJJGSDRL $$BOSFBL IJJGMLLM IJJGDAMX	3-96 3-3 3-19 3-52	4	(PHASENAME) NOT IN SVA
	$$BOTSVA IJJTSRV		2	
4n96I	IJJGSDSF	3-23	4	IMPROPER DTFSD SYSFIL OPEN
4n97I	IJJGDA03 IJJGSD04	3-62 3-30	4	OVLAP EXPIRED SECRD FILE
4n98I	IJJGSD04 IJJGDA03	3-30 3-62	4	OVLAP UNEXPRD SECRD FILE
4n99D	IJJGSDI2 IJJGDAI1	3-41 3-66	4	DATA SECURED FILE ACCESSED

Figure 14 (Part 7 of 8). Master Error Message List

Message Number	Module	Chart	Volume	Message
4H01I	IJDPRT		2	INVALID ASA CONTROL CHAR nn filename SYSxxx
4H02I	IJDPRT		2	PRTOV USED BUT NO PRINTOV SPECIFIED filename SYSxxx
4H03I	IJDPRT		2	CNTRL USED BUT NO CONTROL SPECIFIED filename SYSxxx
4H04I	IJDPRT		2	PHASE IJDPRT INTERNAL ERROR RC=nn filename SYSxxx
4H05I	IJDPRT		2	INVALID RECORD LENGTH filename SYSxxx
4H06I	IJDPRT		2	DTF INCORRECT RC=01 filename SYSxxx
4MR1I	MRMOD		2	EXTERNAL INTERRUPT I/O ERROR filename SYSxxx
4MR2I	MRMOD		2	SCU NOT OPERATIONAL filename SYSxxx
4P01I	$$BERPTP		2	DATA CHECK SYSxxx=cuu
4P02D	$$BERPTP		2	DATA CHECK SYSxxx=cuu
4V03I	$$BODSPW		1	NO RECORD FOUND filename SYSxxx
4V04I	$$BODSPW $$BOVDMP		1	NO RECORD FOUND filename SYSxxx, or NO FORMAT 4 LBL IN VTOC filename SYSxxx
4V06I	$$BOVDMO $$BOVDMP		1	NO STANDARD VOLUME LABEL filename SYSxxx
4V09I	$$BODSPW $$BOWDMP		1	NO RECORD FOUND filename SYSxxx
4V67I	IJJGVD00 IJJGVD10	3-83 3-90	4	CVH PROCESSING FAILURE
4V95A	$$BODSPV IJJGVD10	 3-85	1 4	SYSLOG OR SYSLST
4V96A	$$BODSPV IJJGVD10	 3-85	1 4	SYSLST NOT A PRINTER
P200I	$$BOPR3		2	3800 PRINTER EXTENDED BUFFERING MODE NOT USED REASON CODE = nn

Figure 14 (Part 8 of 8). Master Error Message List

Note: A- and D- type messages are not issued by the B-transient message writer. The respective message writers call $$BOMSVA which in turn transfers control to the SVA message writers in order to issue the message from the SVA.

APPENDIX B: ASCII CONVERSION TABLES

ASCII to EBCDIC Correspondence (0/0 to 3/15)

ASCII					EBCDIC				
Character	Col	Row	Bit Pattern		Col (in Hex)	Row	Bit Pattern		Comments
NUL	0	0	0000	0000	0	0	0000	0000	
SOH	0	1	0000	0001	0	1	0000	0001	
STX	0	2	0000	0010	0	2	0000	0010	
ETX	0	3	0000	0011	0	3	0000	0011	
EOT	0	4	0000	0100	3	7	0011	0111	
ENQ	0	5	0000	0101	2	D	0010	1101	
ACK	0	6	0000	0110	2	E	0010	1110	
BEL	0	7	0000	0111	2	F	0010	1111	
BS	0	8	0000	1000	1	6	0001	0110	
HT	0	9	0000	1001	0	5	0000	0101	
LF	0	10	0000	1010	2	5	0010	0101	
VT	0	11	0000	1011	0	B	0000	1011	
FF	0	12	0000	1100	0	C	0000	1100	
CR	0	13	0000	1101	0	D	0000	1101	
SO	0	14	0000	1110	0	E	0000	1110	
SI	0	15	0000	1111	0	F	0000	1111	
DLE	1	0	0001	0000	1	0	0001	0000	
DC1	1	1	0001	0001	1	1	0001	0001	
DC2	1	2	0001	0010	1	2	0001	0010	
DC3	1	3	0001	0011	1	3	0001	0011	
DC4	1	4	0001	0100	3	C	0011	1100	
NAK	1	5	0001	0101	3	D	0011	1101	
SYN	1	6	0001	0110	3	2	0011	0010	
ETB	1	7	0001	0111	2	6	0010	0110	
CAN	1	8	0001	1000	1	8	0001	1000	
EM	1	9	0001	1001	1	9	0001	1001	
SUB	1	10	0001	1010	3	F	0011	1111	
ESC	1	11	0001	1011	2	7	0010	0111	
FS	1	12	0001	1100	1	C	0001	1100	
GS	1	13	0001	1101	1	D	0001	1101	
RS	1	14	0001	1110	1	E	0001	1110	
US	1	15	0001	1111	1	F	0001	1111	
SP	2	0	0010	0000	4	0	0100	0000	
|	2	1	0010	0001	4	F	0100	1111	Logical OR
"	2	2	0010	0010	7	F	0111	1111	
#	2	3	0010	0011	7	B	0111	1011	
$	2	4	0010	0100	5	B	0101	1011	
%	2	5	0010	0101	6	C	0110	1100	
&	2	6	0010	0110	5	0	0101	0000	
'	2	7	0010	0111	7	D	0111	1101	
(	2	8	0010	1000	4	D	0100	1101	
)	2	9	0010	1001	5	D	0101	1101	
*	2	10	0010	1010	5	C	0101	1100	
+	2	11	0010	1011	4	E	0100	1110	
,	2	12	0010	1100	6	B	0110	1011	
–	2	13	0010	1101	6	0	0110	0000	Hyphen, Minus
.	2	14	0010	1110	4	B	0100	1011	
/	2	15	0010	1111	6	1	0110	0001	
0	3	0	0011	0000	F	0	1111	0000	
1	3	1	0011	0001	F	1	1111	0001	
2	3	2	0011	0010	F	2	1111	0010	
3	3	3	0011	0011	F	3	1111	0011	
4	3	4	0011	0100	F	4	1111	0100	
5	3	5	0011	0101	F	5	1111	0101	
6	3	6	0011	0110	F	6	1111	0110	
7	3	7	0011	0111	F	7	1111	0111	
8	3	8	0011	1000	F	8	1111	1000	
9	3	9	0011	1001	F	9	1111	1001	
:	3	10	0011	1010	7	A	0111	1010	
;	3	11	0011	1011	5	E	0101	1110	
<	3	12	0011	1100	4	C	0100	1100	
=	3	13	0011	1101	7	E	0111	1110	
>	3	14	0011	1110	6	E	0110	1110	
?	3	15	0011	1111	6	F	0110	1111	

Figure 15 (Part 1 of 2). ASCII to EBCDIC Conversion

ASCII to EBCDIC Correspondence (4/0 to 7/15)

ASCII					EBCDIC				
Character	Col	Row	Bit Pattern		Col (in Hex)	Row (in Hex)	Bit Pattern		Comments
@	4	0	0100	0000	7	C	0111	1100	
A	4	1	0100	0001	C	1	1100	0001	
B	4	2	0100	0010	C	2	1100	0010	
C	4	3	0100	0011	C	3	1100	0011	
D	4	4	0100	0100	C	4	1100	0100	
E	4	5	0100	0101	C	5	1100	0101	
F	4	6	0100	0110	C	6	1100	0110	
G	4	7	0100	0111	C	7	1100	0111	
H	4	8	0100	1000	C	8	1100	1000	
I	4	9	0100	1001	C	9	1100	1001	
J	4	10	0100	1010	D	1	1101	0001	
K	4	11	0100	1011	D	2	1101	0010	
L	4	12	0100	1100	D	3	1101	0011	
M	4	13	0100	1101	D	4	1101	0100	
N	4	14	0100	1110	D	5	1101	0101	
O	4	15	0100	1111	D	6	1101	0110	
P	5	0	0101	0000	D	7	1101	0111	
Q	5	1	0101	0001	D	8	1101	1000	
R	5	2	0101	0010	D	9	1101	1001	
S	5	3	0101	0011	E	2	1110	0010	
T	5	4	0101	0100	E	3	1110	0011	
U	5	5	0101	0101	E	4	1110	0100	
V	5	6	0101	0110	E	5	1110	0101	
W	5	7	0101	0111	E	6	1110	0110	
X	5	8	0101	1000	E	7	1110	0111	
Y	5	9	0101	1001	E	8	1110	1000	
Z	5	10	0101	1010	E	9	1110	1001	
[	5	11	0101	1011	4	A	0100	1010	
\	5	12	0101	1100	E	0	1110	0000	Reverse Slant
]	5	13	0101	1101	5	A	0101	1010	
¬	5	14	0101	1110	5	F	0101	1111	Logical NOT
_	5	15	0101	1111	6	D	0110	1101	Underscore
`	6	0	0110	0000	7	9	0111	1001	Grave Accent
a	6	1	0110	0001	8	1	1000	0001	
b	6	2	0110	0010	8	2	1000	0010	
c	6	3	0110	0011	8	3	1000	0011	
d	6	4	0110	0100	8	4	1000	0100	
e	6	5	0110	0101	8	5	1000	0101	
f	6	6	0110	0110	8	6	1000	0110	
g	6	7	0110	0111	8	7	1000	0111	
h	6	8	0110	1000	8	8	1000	1000	
i	6	9	0110	1001	8	9	1000	1001	
j	6	10	0110	1010	9	1	1001	0001	
k	6	11	0110	1011	9	2	1001	0010	
l	6	12	0110	1100	9	3	1001	0011	
m	6	13	0110	1101	9	4	1001	0100	
n	6	14	0110	1110	9	5	1001	0101	
o	6	15	0110	1111	9	6	1001	0110	
p	7	0	0111	0000	9	7	1001	0111	
q	7	1	0111	0001	9	8	1001	1000	
r	7	2	0111	0010	9	9	1001	1001	
s	7	3	0111	0011	A	2	1010	0010	
t	7	4	0111	0100	A	3	1010	0011	
u	7	5	0111	0101	A	4	1010	0100	
v	7	6	0111	0110	A	5	1010	0101	
w	7	7	0111	0111	A	6	1010	0110	
x	7	8	0111	1000	A	7	1010	0111	
y	7	9	0111	1001	A	8	1010	1000	
z	7	10	0111	1010	A	9	1010	1001	
{	7	11	0111	1011	C	0	1100	0000	
|	7	12	0111	1100	6	A	0110	1010	Vertical Line
}	7	13	0111	1101	D	0	1101	0000	
~	7	14	0111	1110	A	1	1010	0001	Tilde
DEL	7	15	0111	1111	0	7	0000	0111	

Figure 15 (Part 2 of 2). ASCII to EBCDIC Conversion

EBCDIC to ASCII Correspondence (X'00' to X'82')

	EBCDIC				ASCII				
Character	Col	Row	Bit Pattern		Col	Row	Bit Pattern		Comments
	(in Hex)								
NUL	0	0	0000	0000	0	0	0000	0000	
SOH	0	1	0000	0001	0	1	0000	0001	
STX	0	2	0000	0010	0	2	0000	0010	
ETX	0	3	0000	0011	0	3	0000	0011	
HT	0	5	0000	0101	0	9	0000	1001	
DEL	0	7	0000	0111	7	15	0111	1111	
VT	0	B	0000	1011	0	11	0000	1011	
FF	0	C	0000	1100	0	12	0000	1100	
CR	0	D	0000	1101	0	13	0000	1101	
SO	0	E	0000	1110	0	14	0000	1110	
SI	0	F	0000	1111	0	15	0000	1111	
DLE	1	0	0001	0000	1	0	0001	0000	
DC1	1	1	0001	0001	1	1	0001	0001	
DC2	1	2	0001	0010	1	2	0001	0010	
DC3	1	3	0001	0011	1	3	0001	0011	
BS	1	6	0001	0110	0	8	0000	1000	
CAN	1	8	0001	1000	1	8	0001	1000	
EM	1	9	0001	1001	1	9	0001	1001	
FS	1	C	0001	1100	1	12	0001	1100	
GS	1	D	0001	1101	1	13	0001	1101	
RS	1	E	0001	1110	1	14	0001	1110	
US	1	F	0001	1111	1	15	0001	1111	
LF	2	5	0010	0101	0	10	0000	1010	
ETB	2	6	0010	0110	1	7	0001	0111	
ESC	2	7	0010	0111	1	11	0001	1011	
ENQ	2	D	0010	1101	0	5	0000	0101	
ACK	2	E	0010	1110	0	6	0000	0110	
BEL	2	F	0010	1111	0	7	0000	0111	
SYN	3	2	0011	0010	1	6	0001	0110	
EOT	3	7	0011	0111	0	4	0000	0100	
DC4	3	C	0011	1100	1	4	0001	0100	
NAK	3	D	0011	1101	1	5	0001	0101	
SUB	3	F	0011	1111	1	10	0001	1010	
SP	4	0	0100	0000	2	0	0010	0000	
[	4	A	0100	1010	5	11	0101	1011	
.	4	B	0100	1011	2	14	0010	1110	
<	4	C	0100	1100	3	12	0011	1100	
(	4	D	0100	1101	2	8	0010	1000	
+	4	E	0100	1110	2	11	0010	1011	
|	4	F	0100	1111	2	1	0010	0001	Logical OR
&	5	0	0101	0000	2	6	0010	0110	
]	5	A	0101	1010	5	13	0101	1101	
$	5	B	0101	1011	2	4	0010	0100	
*	5	C	0101	1100	2	10	0010	1010	
)	5	D	0101	1101	2	9	0010	1001	
;	5	E	0101	1110	3	11	0011	1011	
¬	5	F	0101	1111	5	14	0101	1110	Logical NOT
-	6	0	0110	0000	2	13	0010	1101	Hyphen, Minus
/	6	1	0110	0001	2	15	0010	1111	
¦	6	A	0110	1010	7	12	0111	1100	Vertical Line
,	6	B	0110	1011	2	12	0010	1100	
%	6	C	0110	1100	2	5	0010	0101	
_	6	D	0110	1101	5	15	0101	1111	Underscore
>	6	E	0110	1110	3	14	0011	1110	
?	6	F	0110	1111	3	15	0011	1111	
`	7	9	0111	1001	6	0	0110	0000	Grave Accent
:	7	A	0111	1010	3	10	0011	1010	
#	7	B	0111	1011	2	3	0010	0011	
@	7	C	0111	1100	4	0	0100	0000	
'	7	D	0111	1101	2	7	0010	0111	
=	7	E	0111	1110	3	13	0011	1101	
"	7	F	0111	1111	2	2	0010	0010	
a	8	1	1000	0001	6	1	0110	0001	
b	8	2	1000	0010	6	2	0110	0010	

Figure 16 (Part 1 of 2). EBCDIC to ASCII Conversion

EBCDIC to ASCII Correspondence (X'83' to X'F9')

EBCDIC					ASCII				
Character	Col	Row	Bit Pattern		Col	Row	Bit Pattern		Comments
	(in Hex)								
c	8	3	1000	0011	6	3	0110	0011	
d	8	4	1000	0100	6	4	0110	0100	
e	8	5	1000	0101	6	5	0110	0101	
f	8	6	1000	0110	6	6	0110	0110	
g	8	7	1000	0111	6	7	0110	0111	
h	8	8	1000	1000	6	8	0110	1000	
i	8	9	1000	1001	6	9	0110	1001	
j	9	1	1001	0001	6	10	0110	1010	
k	9	2	1001	0010	6	11	0110	1011	
l	9	3	1001	0011	6	12	0110	1100	
m	9	4	1001	0100	6	13	0110	1101	
n	9	5	1001	0101	6	14	0110	1110	
o	9	6	1001	0110	6	15	0110	1111	
p	9	7	1001	0111	7	0	0111	0000	
q	9	8	1001	1000	7	1	0111	0001	
r	9	9	1001	1001	7	2	0111	0010	
~	A	1	1010	0001	7	14	0111	1110	Tilde
s	A	2	1010	0010	7	3	0111	0011	
t	A	3	1010	0011	7	4	0111	0100	
u	A	4	1010	0100	7	5	0111	0101	
v	A	5	1010	0101	7	6	0111	0110	
w	A	6	1010	0110	7	7	0111	0111	
x	A	7	1010	0111	7	8	0111	1000	
y	A	8	1010	1000	7	9	0111	1001	
z	A	9	1010	1001	7	10	0111	1010	
{	C	0	1100	0000	7	11	0111	1011	
A	C	1	1100	0001	4	1	0100	0001	
B	C	2	1100	0010	4	2	0100	0010	
C	C	3	1100	0011	4	3	0100	0011	
D	C	4	1100	0100	4	4	0100	0100	
E	C	5	1100	0101	4	5	0100	0101	
F	C	6	1100	0110	4	6	0100	0110	
G	C	7	1100	0111	4	7	0100	0111	
H	C	8	1100	1000	4	8	0100	1000	
I	C	9	1100	1001	4	9	0100	1001	
}	D	0	1101	0000	7	13	0111	1101	
J	D	1	1101	0001	4	10	0100	1010	
K	D	2	1101	0010	4	11	0100	1011	
L	D	3	1101	0011	4	12	0100	1100	
M	D	4	1101	0100	4	13	0100	1101	
N	D	5	1101	0101	4	14	0100	1110	
O	D	6	1101	0110	4	15	0100	1111	
P	D	7	1101	0111	5	0	0101	0000	
Q	D	8	1101	1000	5	1	0101	0001	
R	D	9	1101	1001	5	2	0101	0010	
\	E	0	1110	0000	5	12	0101	1100	Reverse Slant
S	E	2	1110	0010	5	3	0101	0011	
T	E	3	1110	0011	5	4	0101	0100	
U	E	4	1110	0100	5	5	0101	0101	
V	E	5	1110	0101	5	6	0101	0110	
W	E	6	1110	0110	5	7	0101	0111	
X	E	7	1110	0111	5	8	0101	1000	
Y	E	8	1110	1000	5	9	0101	1001	
Z	E	9	1110	1001	5	10	0101	1010	
0	F	0	1111	0000	3	0	0011	0000	
1	F	1	1111	0001	3	1	0011	0001	
2	F	2	1111	0010	3	2	0011	0010	
3	F	3	1111	0011	3	3	0011	0011	
4	F	4	1111	0100	3	4	0011	0100	
5	F	5	1111	0101	3	5	0011	0101	
6	F	6	1111	0110	3	6	0011	0110	
7	F	7	1111	0111	3	7	0011	0111	
8	F	8	1111	1000	3	8	0011	1000	
9	F	9	1111	1001	3	9	0011	1001	

Figure 16 (Part 2 of 2). EBCDIC to ASCII Conversion

APPENDIX C: DASD AND TAPE LABELS

This part contains information formarely provided in Chapters 2 and 3 of SC24-5212 (VSE/Advanced Functions Tape Labels) and SC24-5213 (VSE/Advanced Functions DASD Labels).

LABEL PROCESSING FOR SAM AND DAM FILES ON DASD OR DISKETTE

This section summarizes DASD label processing performed for sequential (consecutive) and direct access files. Processing performed for standard format-1 and format-3 labels, and for user-standard labels is described under the headings "Input File" and "Output File." This section also describes diskette label processing. Processing performed for standard HDR1 labels is described under the headings "Diskette Files: Input File" and "Diskette Files: Output File."

SAM AND DAM INPUT FILE

VOL1 Label

The standard volume label (VOL1) must be on cylinder 0, track 0, record 3 for CKD devices, and in block 1 for FBA devices. If it is not, the job is canceled.

The VOL1 label, written by the IBM-supplied program for initializing disks, contains a permanent Volume Serial Number.

Whenever a logical file is to be processed, IOCS reads and checks the VOL1 label against the Volume Serial Number that you supply in an EXTENT statement. For a multiextent, or multivolume multiextent file, IOCS performs this check for each EXTENT. If an error is detected, a message is issued to the operator. The operator may mount the correct volume and continue processing or he may terminate the job.

If you use EXTENT and omit the Volume Serial Number, IOCS checks the Volume Serial Number against the serial number, of the previous EXTENT. If there was no previous EXTENT, IOCS assumes that the correct volume is mounted and does not check the VOL1 label.

For a multivolume SAM file, only one extent is processed at a time, and thus, only one volume need be mounted at a time.

For a multivolume DAM file, all extents (and therefore all volumes) are opened before any data records are processed. Thus, all volumes containing the file must be on-line ready at the same time.

IOCS determines the location of the VTOC from the Data File Directory field of the VOL1 label.

If any additional volume labels (VOL2-VOL8) follow the VOL1 label, IOCS ignores them.

Format-1 Label

You must supply one DLBL or DLAB statement for the logical file to be processed, and one EXTENT statement for each separate area (extent) that the file occupies on the volume. The EXTENT statement may be omitted for a SAM file if the file is on a single volume and the DTF DEVADDR entry is included.

If you use EXTENT for a _single volume_ file and omit the Symbolic Unit field, IOCS uses the Symbolic Unit of the preceding EXTENT. If there is no preceding EXTENT, the Symbolic Unit specified in the DTF is used. If you also omit Symbolic Unit in the DTF, you get an error message. If you use EXTENT for a _multivolume SAM file_, you must supply, for each volume, at least the first EXTENT statement containing the Symbolic Unit. In a _multivolume DAM file_ you must supply a sequential set of Symbolic Unit numbers in EXTENT for the volumes required.

IOCS locates the format-1 label of the file to be processed by first reading the address of the VTOC in the VOL1 label and then searching the VTOC for the format-1 label that contains the File Identification that you specify in DLBL The File Identification (field K1) was written in the key area of the label record when the file was created. Thus, you must specify the same identification now as you did when the file was written as an output file.

If you use DLBL and omit the File Identification, IOCS searches for the label in the VTOC by using the DTF name that you specify in the DLBL Filename field.

For label fields D1-D21, IOCS OPEN routines check the appropriate fields against the corresponding information supplied by you in DLBL or in a DTF specification. Some fields provide information that is required during the processing of data, and other fields are not required by VSE and are ignored. See "Section Label Fields" (Figure 20 on page 105 and Figure 21 on page 107) for the details about each field of the label.

Label fields D22-D25 define the area (extent) of the volume where the data records are located (if user-standard labels have not been written for the file). The extent is one continuous area, and these fields contain the lower limit (starting address) and upper limit (ending address) of the area. They also contain a code for the type of records written in the area, and they provide the order in which this extent should be processed in a multiextent file.

If a file is scattered over separate areas (extents) of the volume, a separate definition is required for each extent. Fields D26-D29 define a second extent in the same way as fields D22-D25 define the first. Fields D30-D33 define a third extent.

If user-standard labels have been written for the file, IOCS previously established an area for them (the first track of the first extent specified for data records) and defined that area in the first Extent field (D22-D25). In this case, the second Extent field (D26-D29) defines the first area that contains data records.

In a <u>SAM file,</u> IOCS checks the starting and ending addresses you supply in EXTENT statements (for the data records) against the lower and upper limits in the corresponding Extent field of the label. If your specifications equal or fall within these limits, IOCS makes the area you specify (in EXTENT) available for processing. If not, a message is issued to the operator. If you omit EXTENT, IOCS does no checking and makes available the area defined by the label.

If you have a multiextent SAM file, IOCS checks your second EXTENT statement information against the second Extent field (or third with user-standard labels).IOCS performs this check after all the data records in the first extent have been processed, and then makes the second area available for processing.

If you have a multiextent DAM file, IOCS makes all the areas you specify in EXTENT statements available at the same time, when the file is first opened.

If more than three Extent fields are used, IOCS reads the Pointer field (D34) and searches for the format-3 label that defines the additional extents.

For a <u>DAM file,</u> you can determine the exact areas of the volume that were specified when the file was created by including the DTFDA entry SXTNTXIT=Name and supplying an extent-processing routine. IOCS branches to your routine after each EXTENT statement is processed. IOCS stores, in register 1, the address of the 14-byte field that contains the information from the Extent field of the label that corresponds to the EXTENT statement just processed. From this field you can obtain, for example, the lower and upper limit of each extent, and save them to check the address of data records. At the end of your routine, return control to IOCS by issuing a LBRET macro instruction.

If you include the DTF LABADDR entry to indicate that user-standard labels are to be processed, IOCS branches to your label routine after processing the standard labels. At the end of your routine, return control to IOCS by issuing a LBRET macro instruction.
You can control the processing of any remaining label fields by the operand in the LBRET instruction. A LBRET 3 instruction permits IOCS to update (rewrite) the label read and pass you the next label; a LBRET 2

instruction permits the processing of another label; a LBRET 1 instruction terminates the processing of user labels.

Format-3 Label

If more than three extent fields were required when the file was created, IOCS set up and created a format-3 label for the additional extents.

On input, IOCS searches for the format-3 label when it reads another EXTENT after the third extent of the format-1 label has been processed. IOCS reads the address of the format-3 label from the Pointer field (D34) of the format-1 label.

For a _SAM file,_ IOCS searches for a second format-3 label if it reads another EXTENT after the 13th extent of the first format-3 label has been processed. IOCS reads the address of the second format-3 label from the Pointer field (D38) of the first format-3 label.

A _DAM file_ permits the use of only one format-3 label.

IOCS processes the extent fields of the format-3 label in the same manner as those in the format-1 label.

User-standard Label

When user-standard labels (UHL/UTL) are to be checked and logical IOCS macros are used for the file, DTF LABADDR=Name must be specified. If it is not specified, IOCS bypasses all user-standard labels.

When physical IOCS macros are used for a file and DTFPH is specified, LABADDR=Name must be included if user-standard header labels (UHL) are to be checked. IOCS does not provide for user checking of user-standard trailer labels (UTL) with the DTFPH.

For a _SAM file,_ IOCS provides for checking user-standard '_header_ labels after it checks the standard VOL1 and format-1 labels. In a multivolume file, IOCS provides for checking user-standard header labels on each volume when that volume is ready to be processed.

For a _DAM file,_ IOCS reads the user-standard header label after it checks the standard VOL1 and format-1 labels of a single-volume file and makes the label available to you. In a multivolume file, IOCS processes all labels when the file is initially opened.
At that time, IOCS checks the standard VOL1 and format-1 labels on the first volume, and then reads the user-standard header labels on the first volume, and makes the label available to you.
Next, IOCS checks the standard labels on the second volume and reads the user-standard header labels on that volume. Label processing progresses in this manner through all on-line volumes, before any data records are processed.

IOCS provides for user checking of user-standard _trailer_ labels on an end-of-volume or end-of-file condition. IOCS indicates the status of the file through the low-order byte in register 0. The indication is O, V, or F; meaning open, end-of-volume, or end-of-file, respectively.

The input file (such as a card reader) that contains the user's information for checking user-standard labels must be opened before the file with the UHL labels. This is done by specifying the label-information file ahead of the labeled file in the same OPEN instruction, or by issuing a prior, separate OPEN instruction.

IOCS identifies the user-standard labels by UHL or UTL in the first three bytes of the label.

IOCS reads each user-standard label, one at a time, from the partition GETVIS area. IOCS supplies the address of the area into which the labels are read in register 1.

After a label is read in, IOCS branches to your label-checking routine. The same routine (specified by DTF LABADDR=Name) is used for checking both user-standard header (UHL) and user-standard trailer (UTL) labels. You can identify the type of label by the UHL or UTL in the first three positions of the label itself.

After you check a label, return to IOCS by issuing a LBRET macro instruction. You control the checking of any remaining user-standard labels by the operand in the LBRET instruction. A LBRET 3 instruction permits IOCS to update (rewrite) the label read and pass you the next label. A LBRET 2 instruction permits the checking of another label. A LBRET 1 instruction or an end-of-file record terminates label checking.

If the user, or an end-of-file record, does not terminate the label checking, IOCS reads in the next user-standard header label.

SAM AND DAM OUTPUT FILE

VOL1 Label

The standard volume label (VOL1) must be on cylinder 0, track 0, record 3, or in block 1 for FBA devices. If it is not, the job is canceled.

The (VOL1) label contains a permanent Volume Serial Number. IOCS neither rewrites nor alters the VOL1 label in any way.

Whenever a logical file is to be processed, IOCS reads and checks the VOL1 label against the Volume Serial Number that you supply in an EXTENT statement. For a multiextent, or multivolume multiextent file, IOCS performs this check for each EXTENT. If an error is detected, a message is issued to the operator. The operator may mount the correct volume and continue processing, or terminate the job.

If you use EXTENT and omit the Volume Serial Number, IOCS checks against the number of the previous EXTENT. If there was no previous EXTENT, IOCS assumes that the correct volume is mounted and does not check the VOL1 label.

For a multivolume _SAM file,_ only one extent is written at a time, and thus only one volume need be mounted at a time.

For a multivolume _DAM file,_ all extents (and therefore all volumes) are opened before any data records are written. Thus, all volumes that will contain the file must be on-line and ready at the same time.

IOCS determines the location of the VTOC from the Data File Directory field of the VOL1 label.

If any additional volume labels (VOL2-VOL8) follow the VOL1 label, IOCS ignores them.

Format-1 Label

You must supply one DLBL statement for the logical file, and one EXTENT statement for each separate area (extent) that the file will occupy on the volume(s).

An EXTENT statement defines the area (extent) of the disk pack where the data records are to be written. For a CKD device, an EXTENT statement provides the starting address (relative track) and the number of tracks, which indirectly gives the ending address of the extent. For an FBA device, an EXTENT statement specifies the address of the physical block with which the extent begins and the number of physical blocks within the extent.

An EXTENT statement also contains a code for the type of records that are to be written, and provides the order in which this extent should be processed in a multiextent file.

If you use EXTENT for a _single volume file_ and omit the Symbolic Unit field, IOCS uses the Symbolic Unit of the preceding EXTENT. If there is no preceding EXTENT, the Symbolic Unit specified in the DTF is used. If you also omit Symbolic Unit in the DTF, you get an error message. If you use EXTENT for a _multivolume SAM file,_ you must supply, for each volume, at least the first EXTENT statement containing the Symbolic Unit. In a _multivolume DAM file,_ you must supply a sequential set of Symbolic Unit numbers in EXTENT for the volumes required. Also, in a _multivolume DAM file_ a separate physical device must be assigned to each symbolic unit.

If a file is to be scattered over separate areas (extents) of the disk pack, a separate EXTENT statement is required for each extent. In that case, the same Symbolic Unit number must be used.

IOCS first validates the EXTENT statement specifications:

1. The extents must not overlap each other.
2. If user-standard labels are to be written (specified by DTF LABADDR), the first extent must be at least two tracks.
3. The valid extent types for a <u>SAM file</u> are:

   1 - data records

   8 - data records with split cylinder, in EXTENT (not valid for FBA)

   128 - data records with split cylinder, in XTENT (not valid for FBA)
4. The valid extent type for a <u>DAM file</u> is 1 (data records).
5. For a <u>DAM file,</u> the maximum number of EXTENT statements is 15 if user-standard labels are specified, or 16 if they are not.

IOCS OPEN routines locate the VTOC by reading its address in the VOL1 label.

IOCS checks the limits of an EXTENT against the limits of each extent field of each label already written in the VTOC. If the new extent overlaps any previously written extent, IOCS checks the expiration date of the old file to ensure that the data records are no longer active.

If the expiration date has passed, IOCS deletes the old label(s) by setting the File Identification Field of the format-1 label to binary zeros, which, in effect, removes the expired file from the volume and makes a record in the VTOC available. A format-3 label associated with the expired file is deleted at OPEN time along with the format-1 label.

If the expiration date has not passed, a message is issued to the operator. The operator can delete the unexpired file and continue processing, bypass this EXTENT, or terminate the job.

IOCS reads the format-4 label (first record in VTOC) to determine the limits of the VTOC.

IOCS searches the Key Identification Field (K1) for zeros, which indicate an available location. IOCS checks this location to verify that it is contained within the VTOC limits, and then writes the format-1 label. The process is repeated if a format-3 label is required.

For label fields K1 and D1-D21 of the format-1 label, IOCS writes the information supplied by you in DLBL or in a DTF entry, or generated by the system. See "Section Label Fields" (Figure 20 on page 105 and Figure 21 on page 107) for the details about each field of the label.

If you use DLBL and omit some specifications, IOCS writes predetermined default values (see Figure 21 on page 107).

Label fields D22-D25 define the area (extent) of the volume where the data records will be written. IOCS writes these fields from the first EXTENT statement (if user-standard labels are not specified for the file).

If a file is to be scattered over separate areas (extents) of the volume, a separate definition will be required for each extent. Fields D26-D29 are used to define a second extent, and fields D30-D33 to define a third. These fields are written from additional EXTENT statements you supply.

If user-standard labels are written for the file (specified by DTF LABADDR), IOCS establishes an area for them (the first track of the first extent you specify for the data records) and defines that area in the first extent field (D22-D25). In this case, IOCS writes your first EXTENT statement information for data records in the second extent field of the label.

After writing the label(s), IOCS makes the area(s) of the volume available for writing the data records. In a _SAM multiextent file,_ IOCS makes only the first specified extent available.
After that extent is filled, IOCS makes the next specified extent available. In a _DAM multiextent file,_ IOCS makes all the extents available at the same time.

If you include more than three EXTENT statements (without user-standard labels, or two with user-standard labels), IOCS writes a format-3 label, and writes the address of that label in the Pointer field (D34) of the format-1 label.

If you include the DTF LABADDR entry to indicate that user-standard labels are to be written, IOCS branches to your label routine prior to writing each standard label.

Format-3 Label

If more than three Extent fields are required for the file, IOCS sets up a format-3 label for the additional extents.

On output, IOCS writes the format-3 label when it reads another EXTENT statement after the three Extent fields of the format-1 label have been filled. IOCS writes the address of the format-3 label in the POINTER field (D34) of the format-1 label.

For a _SAM_ file, IOCS writes a second format-3 label if it reads another EXTENT statement after the 13 Extent fields of the first format-3 label have been filled. IOCS writes the address of the second format-3 label in the Pointer field (D38) of the first format-3 label.

A _DAM_ file permits the use of only one format-3 label.

IOCS processes the Extent fields of the format-3 label in the same manner as those in the format-1 label.

User-Standard Label

When user-standard labels are to be written for a file, DTFSD, DTFDA, or DTFPH LABADDR=Name must be specified.

Whenever LABADDR=Name is specified, at least one UHL label and one UTL label will be written.

For a _SAM_ file, IOCS writes user-standard header labels after it writes the standard file labels. In a multivolume file, IOCS writes user-standard header labels in each volume.

For a _DAM_ file, IOCS writes user-standard header labels after it writes the standard file labels of a single-volume file. In a multivolume file, IOCS writes all labels when the file is initially opened. At that time, IOCS writes the standard file labels on the first volume, and then writes the user-standard header labels on the first volume. Next, IOCS writes the standard file labels in the second volume and writes user-standard header labels on that volume. Label processing progresses in this manner through all on-line volumes, before any data records are processed.

IOCS writes user-standard trailer labels on an end-of-volume or end-of-file condition. IOCS indicates the status of the file through the low-order byte in register 0. The indication is O, V, or F; meaning open, end-of-volume, or end-of file, respectively.

The input file (such as a card reader) that contains the user's information for writing user-standard labels must be opened before the file on which the UHL labels are to be written. To do this, the input file must be specified before the file to be labeled in the same OPEN instruction, or a prior separate OPEN instruction must be issued.

The user must build each user-standard label. To provide for this, IOCS branches to the user's label routine. The same routine (specified by LABADDR=Name) is used for building both user-standard header labels (UHL) and user-standard trailer labels (UTL). IOCS supplies a code in the low-order byte of register 0 to indicate which type of label should be built:

UHL - Code O (letter O)

UTL - Code F for end-of-file condition

Code V for end-of-volume condition

You must establish an 80-byte area to build your labels, and you must load the address of that area in register 0.

When building the label, you must include UHL or UTL in the first 3 bytes of the 80-byte data area followed by a digit 1-8 in the fourth byte. You may include whatever information you need in the remaining 76 bytes.

After building a label, you return to IOCS by issuing a LBRET instruction. IOCS then writes the label on the volume.

You control the building and writing of successive user-standard labels by the operand in the LBRET instruction. If another label is to be written, specify operand 2 and IOCS again branches to your label routine. When you have built you last user label, issue the LBRET macro with the operand 1. IOCS writes the last label.

A maximum of eight user-standard header and eight user-standard trailer labels may be written. After eight labels, IOCS terminates the label writing, regardless of the LBRET macro instruction.

Each user-label set (header or header and trailer) is terminated by an end-of-file record, which is a data record with a data length of 0.

DISKETTE FILES: INPUT FILE

VOL1 Label

The VOL1 label is on track 0, record 7.

Whenever a logical file is to be processed, IOCS reads and checks the VOL1 label against the Volume Serial Number that you supply in an EXTENT statement. For a multivolume file, IOCS performs this check for each EXTENT. If an error is detected, a message is issued to the operator. The operator may mount the correct volume and continue processing, or he may terminate the job.

If you omit the Volume Serial Number, IOCS assumes that the correct volume is mounted and does not check the VOL1 label.

For a multivolume file, only one extent is processed at a time. IOCS automatically feeds between volumes of a multivolume file.

The VTOC on a diskette is always on track 0, records 8-26.

HDR1 Label

You must supply one DLBL statement for the logical file to be processed, and one EXTENT statement for each volume on which the file is contained. One exception exists to this: the EXTENT statement may be omitted if the file is on a single volume and the DTF DEVADDR entry is included.

If you omit the Symbolic Unit field on the EXTENT statement of a _single volume_ file, or on all EXTENT statements of a _multi-volume_ file, IOCS uses the Symbolic Unit specified in the DTF. If you also omit Symbolic Unit in the DTF, you get an error message. All symbolic unit fields provided on the EXTENT statements must be identical.

IOCS locates the HDR1 label of the file to be processed by searching the VTOC for the HDR1 label that contains the File Identification that you specify in the DLBL. The File Identification (field D4) was written in the label when the file was created. Thus, you must specify the same identification now as you did when the file was written as an output. See "Section Label Fields" (Figure 32 on page 136 and Figure 33 on page 136) for the details about each field of the label.

If you omit the File Identification, IOCS searches for the label in the VTOC, using the DTF name that you specify in the DLBL Filename field.

Label fields D8, D10, and D23 define the area (extent) of the diskette where data records are located. These fields contain the lower limit (starting address), the upper limit (ending address), and the end-of-data address (address of the last record in the file +1). Files with multiple extents on a single volume are not supported on diskettes. IOCS ignores any starting and ending addresses you supply on the EXTENT statement.

For multivolume diskette input files using DTFDU, the EXTENT statements and the multivolume indicator are used in conjunction by the OPEN transients to determine when end of file has occurred. If three extents were provided by you, the multivolume indicator combinations shown in Figure 17 could occur.

Multivolume.indicator	Action taken by OPEN transients
b, anything	Process first volume and issue warning message.
L, anything	No volumes are processed, permanent-error message issued.
C, b	Process first volume; the b indicates that no further volume checking is to be done.
C, x	Process first volume, and issue permanent-error message because the file was not found.
C, L, anything	Process through the "L" and issue warning message.
C,C,C	Process through the number of extents. No message is issued.
C,C,L	Process through the "L." No message is issued.

Figure 17. Multivolume Indicator Combinations (3 Extents)

In summary, for DTFDU the number of diskettes can be less than the number of extents provided.

For all other supported DTF's, processing continues until the number of extents is exhausted. Regardless of the DTF type for system files, processing continues until all extents are exhausted.

DISKETTE FILES: OUTPUT FILE

VOL1 Label

The VOL1 label is on track 0, record 7.

IOCS will update the accessibility indicator (field D4) to an "S"whenever a secured file is created on the volume.

Whenever a logical file is to be processed, IOCS reads and checks the VOL1 label against the Volume Serial Number that you supply in an EXTENT statement. For a multivolume file, IOCS performs this check for each EXTENT. If an error is detected, a message is issued to the operator. The operator may mount the correct volume and continue processing, or he may terminate the job.

If you omit the Volume Serial Number, IOCS assumes that the correct volume is mounted, and does not check the VOL1 label.

For a multivolume file, only one extent is written at a time. IOCS automatically feeds from the one volume of a multivolume file to the next.

The VTOC on a diskette is always on track 0, records 8-26.

HDR1 Label

You must supply one DLBL statement for the logical file, and one EXTENT statement for each volume on which the file is to be written.

If you omit the Symbolic Unit field on the EXTENT statement of a _single volume_ file, or on all EXTENT statements of a _multi-volume_ file, IOCS uses the Symbolic Unit specified in the DTF. If you also omit Symbolic Unit in the DTF, you get an error message. All symbolic unit fields provided on the EXTENT statements must be identical.

The extent limits for the file are determined by IOCS from available space on the diskette, and any extent limits provided by you on the EXTENT statement are ignored.

The name of the output file to be created is the same as the File Identification you specify in the DLBL statement. If you omit the File Identification, the name will be the same as the DTF name that you specify in the DLBL Filename field.

If the name of the output file to be created is equal to that of an unexpired or write-protected (field D14) file already present on the volume, you will get an error message and the job will be canceled. You will not be allowed to request that the duplicate file (unexpired or write-protected) be deleted.

If the duplicate file is expired and not write-protected, or if a duplicate file is not being created, IOCS will allocate space for the file starting at the track following the end of the last unexpired or write-protected file on the volume, and ending at the end of the volume (track 73, record 26). If expired and non-write-protected files are overlapped by this allocation, their labels are deleted from the VTOC by writing delete records in their location in the VTOC.

If there is not at least one track of space available on the volume, you will get an error message and the job will be terminated.

IOCS created the HDR1 label based on the information supplied by you in the DLBL or in a DTF entry, or information generated by the system. See "Section Label Fields" (Figure 32 on page 136 and Figure 33 on page 136) for the details about each field of the label.

If you omit some specifications IOCS defaults to predetermined values. See "Section Label Fields" (Figure 33 on page 137).

After writing the label, IOCS makes the area of the diskettes available for writing the data records.

At CLOSE time IOCS reads and rewrites the HDR1 label in order to update certain fields. They are:

End-of-Data (D23) This field will be set up as the address of the record following the last record in the file.

End-of-Extent (D10) This field will be updated to be the address of the last record in the data set.

Multivolume Indicator (D16) This field will be set up to indicate if this is a multivolume file; a blank indicates a single volume file; a C indicates all but the last volume of a multivolume file; and an L indicates the last volume of a multivolume file.

LABEL PROCESSING FOR ISAM FILES

This section summarizes DASD label processing performed for indexed sequential files. Processing performed for format-1 and format-2 labels is described under _Load(Create, Extend) Function_, _Add Function_ and _Retrieve Function._ The ADD and RETRVE (retrieve) functions can be combined into one operation by specifying the DTFIS entry IOROUT=ADDRTR. ISAM is not supported for FBA devices, the 3330-11, or the 3350 except when operated in 3330-1 compatibility mode.

ISAM FILES, LOAD (CREATE, EXTENT)

FUNCTION

VOL1 Label

The standard volume label (VOL1) must be on cylinder 0, track 0, record 3 (except for 3350 operated in 3330-1 compatibility mode). If it is not, the job is canceled.

The VOL1 label contains a permanent Volume Serial Number.

IOCS neither rewrites nor alters the VOL1 label in any way.

Whenever a logical file is to be processed, IOCS reads and checks the VOL1 label against the Volume Serial Number that you supply in an EXTENT or XTENT statement. For a single volume (requiring a minimum of two extents), or for a multivolume file, IOCS performs this check for each EXTENT. If an error is detected, a message is issued to the operator. The operator may mount the correct volume and continue processing, or terminate the job.

If you use EXTENT and omit the Volume Serial Number, IOCS checks against the number of the previous EXTENT. If there was no previous EXTENT, IOCS assumes that the correct volume is mounted and does not check the VOL1 label.

For a multivolume file, all extents (and therefore all volumes) are opened before any data records are written. Thus, all volumes that will contain the file must be on-line and ready at the same time. Each different symbolic unit must be assigned to a separate physical device.

IOCS determines the location of the VTOC from the Data File Directory field of the VOL1 label.

If any additional volume labels (VOL2-VOL8) follow the VOL1 label, IOCS ignores them.

Format-1 Label

You must supply one DLBL statement for the logical file, and one EXTENT statement for each separate area (extent) that the file will occupy on the volume.

EXTENT statements define the areas (extents) of the volume where the data records (prime data area), cylinder and master indexes, and independent overflow records are to be written.
An EXTENT statement provides the starting address (called relative track) and the number of tracks which indirectly give the ending address. An EXTENT statement so contains a code for the type of records to be written, and indicates the sequence in which this statement should be inserted into the input stream. The EXTENT statements you supply for a LOAD function must also be supplied for any subsequent ADD or RETRVE (retrieve) functions. (See <u>Add Function</u> and <u>Retrieve Function,</u> in this chapter, for details on additional requirements for these functions.)

The prime data area (data records) and the cylinder index area are required, and you must supply an EXTENT statement for each. If you want a master index and/or an independent overflow area, you must also supply an EXTENT statement for each desired area.

The prime data area for a logical file must be one continuous area on any one volume. It cannot be scattered over separate areas of a single volume. The prime data area can, however, extend to one or more volumes, in which case a separate EXTENT statement is required for each volume.

The prime data area on any pack must start on track 0 of any cylinder, with the exception of track 0 of cylinder 0, which is reserved for labels and system use. Therefore, the prime data area must start on some cylinder other than cylinder 0 and is never written on cylinder 0, track 0 of any pack.

For a multivolume file, the prime data area of the <u>first</u> pack may start on any cylinder (except 0) and must extend through the last track on the pack. On all packs after the first, the prime data area must start on cylinder 1, track 0 so that IOCS considers the prime data area as one continuous area. On all succeeding packs (except the last) the prime data area must extend through the last track on the pack. On the last pack, it may end at the end of any cylinder. Thus, in a multipack file, all packs, except the first and the last, are completely allotted to the prime data area from cylinder 1, track 0 through the last track on the last cylinder.

For a multivolume file, the VTOC for the first volume must precede the prime data area. On the last volume, the VTOC may be on cylinder 0 or it may follow the prime data area. On all other volumes, the VTOC must be on cylinder 0.

Because the prime data area must be considered as one continuous area in a multivolume file, the master/cylinder index and independent overflow area must be located before the prime data area on the first volume or after the prime data area on the last volume.

During the load operation ISAM builds a separate _track index_ for each cylinder used by the file. Track indexes are considered a part of the prime data area and, as such, do not require separate EXTENT statements. Each track index starts on the first track (0) of the cylinder that it is indexing.
It can occupy a full track, more than one track, or part of a track and share that partially used track with prime data records (shared track).

Also within the prime data area certain tracks may be reserved, if desired, for overflow records that will occur when records are added to the file in later operations. These tracks, called a _cylinder overflow_ area, must be reserved during the _load_ operation by specifying the DTFIS entry CYLOFL. Because this is part of the prime data area, no separate EXTENT statements are required.

The master index and the cylinder index are separate indexes and require two separate EXTENT statements. However, when they are written on the volume, IOCS combines them into one index area and writes the address of that combined area in the format-1 label. Therefore, for these indexes, you must specify (in the EXTENT statements) two areas that are adjacent to each other.

SAM builds the master (if used) and cylinder indexes during the _load_ operation. These indexes must be separate from the prime data area and wholly contained on one volume. They can be on the same volume with the prime data or on a separate volume. They even can be on a different type of device from the prime data area.

You must specify the location of the _cylinder_ index by an EXTENT statement. It must immediately follow the master index on a volume, and it may be located on one or more successive cylinders. You must also specify an Extent Type of 4 and an Extent Sequence Number of 1 in the EXTENT statement. If you use EXTENT and omit the Extent Type, IOCS assumes the code for a data area. This index contains one entry for each cylinder occupied by the data file.

If you plan to use a _master index_ for a file, you must specify this option with the DTFIS entry MSTIND and you must specify its location by EXTENT. It must immediately precede the cylinder index on a volume, and it may be located on one or more successive cylinders. You must also specify an Extent Type of 4 and an Extent Sequence Number of 0 in the EXTENT statement. If you use EXTENT and omit the Extent Type, IOCS writes in the code for a data area. This index contains an entry for each track of the cylinder index.

ISAM OPEN first validates the EXTENT statement specifications:

1. All prime data extents must be continuous.
2. The master and cylinder index extents must be continuous and on the same unit.
3. No extents must overlap.
4. The valid extent types are:

   1 - Prime Data
   2 - Independent Overflow
   4 - Master Index

4 - Cylinder Index

5. The Extent Sequence Number must be in a specified order:

0	Master Index
1	Cylinder Index
2 through n	Prime Data
n+1	Independent Overflow

OR

0	Master Index
1	Cylinder Index
2	Independent Overflow
3 through n	Prime Data

If a master index is not used, Extent Sequence Number begins with 1.

IOCS OPEN routines locate the VTOC by reading its address in the VOL1 label.

IOCS checks the limits of an EXTENT statement against the limits of each Extent field of each label already written in the VTOC. If the new extents overlap any previously written extents, IOCS checks the expiration date of the old (being overlapped) file to ensure that the data records are no longer active.

If the expiration date has passed, IOCS deletes the old label(s), which in effect removes the expired file from the volume.

If the expiration date has not passed, a message is issued to the operator. The operator can terminate the job or delete the unexpired file and continue processing.

IOCS reads the format-4 label (first record in VTOC) to determine where to write the format-1 and format-2 labels and then writes the labels. In a multivolume file, IOCS writes the format-2 label only in the volume that contains the cylinder index.

For a multivolume file, all extents (and therefore all volumes) are opened before any data records are written. Thus, all volumes that will contain the file must be on-line and ready at the same time. For each volume, IOCS checks the extents specified in the extent statements for that volume (for example, checks that the data extents are continuous). IOCS also checks the standard VOL1 label and then goes to the VTOC to check the file label(s). Then, the next volume is opened. After all the volumes have been opened, the file is ready for processing.

If you use EXTENT and omit the Symbolic Unit field, IOCS uses the Symbolic Unit of the preceding EXTENT. The first EXTENT must contain the Symbolic Unit. If you use EXTENT for a multivolume file, you must supply, for each volume, at least one EXTENT statement containing the Symbolic Unit. All extents on one physical unit must have the same symbolic unit number.

For label fields K1 and D1-D21 of the format-1 label, IOCS writes the information supplied by you in DLBL or a DTFIS entry, or generated by the system. See "Section: Label Fields" (Figure 20 on page 105 and Figure 21 on page 107) for the details about each field of the label.

Specify in the DLBL statement a File Type if ISC when using LOAD to _create_ a file and a File Type of _ISE_ when using LOAD to _extend_ the file. If you use DLBL and omit this field (D11 of the format-1 label), IOCS writes the code for a SAM file in this field. If an ISC is specified in DLBL for a non-load function, the system cancels the job.

The LOAD function is specified by the DTFIS entry IOROUT. The functions of originally loading a file of presorted records and of _extending_ the file by adding new presorted records are the same. Both are considered a LOAD operation.

If you use DLBL and omit some specifications, IOCS writes predetermined default values. See "Section: Label Fields" (Figure 21 on page 107)

Label fields D22-D25, D26-D29, and D30-D33 of the format-1 label define three areas (extents) of the volume, IOCS writes these fields from the EXTENT statements you supply.

The Extent Type 1 (prime data), 2 (independent overflow area), and 4 (master or cylinder index) in the EXTENT is converted to a hex 01, 02, and 04 respectively by IOCS in fields D22, D26, and D30 of the format-1 label.
If you use EXTENT and omit the type, IOCS writes hex 01, the designation for a prime data area.

Extent data is written in the format-1 label in the same order that EXTENT statements are supplied. This order is specified by the Extent Sequence Number, fields D23, D27, and D31. First Extent (D22-D25) is for master index (if specified) and cylinder index. Additional Extent (D26-D29) is for the prime data area, and Additional Extent (D30-D33) is for the independent overflow area (if specified). Prime data area and independent overflow area may be reversed.

During the load operation, ISAM uses the Extent Sequence Number and the Symbolic Unit (SYSnnn, specified in EXTENT) to determine on what volume the extent area is located. The EXTENT statements must be entered in ascending order by Extent Sequence Number, with none missing.

During a <u>LOAD</u> <u>Extend</u> function, IOCS checks the Extent Upper Limit (fields D25 and D29 in the format-1 label) against the upper limit specified by EXTENT. If the specified limit of either the cylinder index or the prime data area is beyond the upper limit in the label, IOCS changes the label and makes the new area available for records. If the prime data area is extended onto a new volume, IOCS writes the lower and upper limits of the next extent specified by EXTENT. Under any other condition, the job is canceled if the limits do not agree.

After writing the label(s), IOCS makes the areas of the volume available for writing the data records, index(es), and independent overflow records. In a multivolume file, IOCS makes all the volumes available at the same time.

IOCS always writes a format-2 label and writes the address of that label in the Pointer Field (D34) of the format-1 label.

Format-2 Label

A format-2 label is required and maintained by ISAM. This label is used to carry updated information from one use (function) of the file to the next and to retain many fields of the DTFIS table.

No separate EXTENT statements are required for the format-2 label. ISAM OPEN/CLOSE routines write the label by using the information that you supply in DTF specifications or that is calculated during the processing of data records. Generally, job control statement information is not used. The OPEN routine, however, uses EXTENT specifications in four fields during LOAD Create function.

Some of the fields in the format-2 label are written when the file is opened, whereas other fields are written when the file is closed.

IOCS always writes a format-2 label in the VTOC after it has written the format-1 label. It writes the address of the format-2 label in the Pointer Field (D34) of the format-1 label.

The RECSIZE written in the format-2 label is used for any following ADD or RETRVE functions.

If a file occupies two or more volumes, IOCS writes the format-2 label only on the volume containing the cylinder index. This volume may or may not contain data records. The format-2 label is not repeated on the additional volumes (as the format-1 label is).

If a load file is not closed, such as during an abnormal end of job, the format-2 label associated with that file is not completely updated with the information that is in the DTF. <u>Caution</u>: Further processing of this file may give unpredictable results.

The statistics provided in several fields of the format-2 label can be used to determine whether you should reorganize the file:

D12 - Tag Deletion Count: The number of records you identify (tag) for deletion (not processed by IOCS).

D13 - Non-First Overflow Reference Count: The number of times a READ instruction causes a search of the overflow area(s) for a record that is the second or higher in an overflow chain.

D16 - Prime Record Count: The number of logical records written in the organized file in the prime data area(s). IOCS accumulates this count during a LOAD operation.

D27 - Number of Independent Overflow Tracks: The number of tracks still available in the independent overflow area.

D28 - Overflow Record Count: The number of records written in all the overflow areas for the file (cylinder overflow areas and/or independent overflow area).

D29 - Cylinder Overflow Area Count: The number of cylinder overflow areas that have been filled.

See "Section: Label Fields" (Figure 24 on page 120) for the details about each field of the format-2 label.

ISAM FILES, ADD FUNCTION

VOL1 Label

The standard volume label (VOL1) must be on cylinder 0, track 0, record 3. If it is not, the job is canceled.

The VOL1 label contains a permanent Volume Serial Number.

IOCS neither rewrites nor alters the VOL1 label in any way.

Whenever a logical file is to be processed, IOCS reads and checks the VOL1 label against the Volume Serial Number that you supply in an EXTENT statement.
For a single volume (requiring a minimum of two extents), or for a multivolume file, IOCS performs this check for each EXTENT. If an error is detected, a message is issued to the operator. The operator may mount the correct volume and continue processing, or terminate the job.

If you use EXTENT and omit the Volume Serial Number, IOCS checks against the number of the previous EXTENT. If there was no previous EXTENT, IOCS assumes that the correct volume is mounted and does not check the VOL1 label.

For a multivolume file, all extents (and therefore all volumes) are opened before any data records are written. Thus, all volumes that will contain the file must be on-line and ready at the same time.

IOCS determines the location of the VTOC from the Data File Directory field of the VOL1 label.

If any additional volume labels (VOL2-VOL8) follow the VOL1 label, IOCS ignores them.

Format-1 Label

You must supply one DLBL statement for the logical file, and one EXTENT statement for each separate area (extent) that the file occupies on the volume.

EXTENT statements define the areas (extents) of the volume where the data records (prime data area), cylinder and master indexes, and independent overflow records are written. An EXTENT statement provides the starting address (called relative track) and the number of tracks which indirectly give the ending address. An EXTENT statement also contains a code for the type of records to be written, and indicates the sequence in which this statement should be inserted into the input stream.

For an <u>ADD</u> function, you must supply the same EXTENT statements that you supplied for the LOAD function. You must supply an EXTENT for the prime data area (data records) and the cylinder index. If you specified a master index, you must supply an EXTENT for this area. If you specified an independent overflow area during the load operation, you must supply an EXTENT for this area. If an independent overflow area has not been established, you can specify one during the add function by supplying an EXTENT statement for this area.

The prime data area for a logical file is one continuous area on any one volume. It is not scattered over separate areas of a single volume. If, however, the prime data area extends to one or more other volumes, you must supply a separate EXTENT statement for each volume.

The prime data area on any volume starts on track 0 of any cylinder with the exception of track 0 of cylinder 0, which is reserved for labels and system use. Therefore, the prime data area starts on some cylinder other than cylinder 0 and is never written on cylinder 0, track 0 of any volume.

For a multivolume file, the prime data area of the first volume starts on any cylinder (except 0) and extends through the last track on the volume. On all volumes after the first, the prime data starts on cylinder 1, track 0 so that IOCS considers the prime data area as one continuous area. On all succeeding volumes (except the last), the prime data area extends through the last track on the volume. On the last volume, it ends at the end of any cylinder. Thus, in a multivolume file all volumes, except the first and last, are completely allotted to the prime data area from cylinder 1, track 0 through the last track in the last cylinder.

For a multivolume file the VTOC for the first volume precedes the prime data area. On the last volume, the VTOC is on cylinder 0 or it follows the prime data area. On all other volumes, the VTOC is on cylinder 0.

Because the prime data area is considered as one continuous area in a multivolume file, the master/cylinder index and independent overflow areas are located before the prime data on the first volume or after the prime data area on the last volume.

During the load operation, ISAM has built a separate track index for each cylinder used by the file. Track indexes are considered a part of the prime data area and, as such, do not require separate EXTENT statements. Each track index is located on the first track (0) of the cylinder that it is indexing. It can occupy a full track, more than one track, or part of a track and share that track with prime data records (shared track).

Also, within the prime data area, certain tracks may have been reserved for overflow records that occur when records are added to the file. These tracks, called a cylinder overflow area, are reserved during the load operation and used during an add operation. If you use cylinder overflow areas, you must specify the DTFIS entry CYLOFL during both the load and add functions. Because the cylinder overflow areas are a part of the prime data area, no separate EXTENT statements are required.

The master index and the cylinder index are separate indexes and require two separate EXTENT statements. However, when they were written on the volume, IOCS combined them into one index area and wrote the address of that combined area in the format-1 label. Therefore, for these indexes, you must specify (in the EXTENT statements) two adjacent areas.

ISAM has built the master (if used) and cylinder indexes during the load operation. These indexes are separate from the prime data area and wholly contained on one volume. They can be on the same volume with the prime data or on a separate volume. They can also be on a different type of device than the prime data area.

You must specify the location of the cylinder index by EXTENT. It immediately follows the master index on a volume, and it may be located on one or more successive cylinders. You must also specify an Extent Type of 4 and an Extent Sequence Number of 1 in the EXTENT statement. If you use EXTENT and omit the Extent Type, 1 is assumed.

If you specified a master index during the load operation, you must again specify this option for the add function with the DTFIS entry MSTIND. You must specify its location by EXTENT. The master index immediately precedes the cylinder index on a volume, and it may be located on one or more successive cylinders. You must also specify an Extent Type of 4 and an Extent Sequence Number of 0 in the EXTENT statement. If you use EXTENT and omit the Extent Type, this field is not checked.

An independent overflow area may be specified for storing overflow records that occur when records are added to the file. If you plan to use an independent overflow area, you must supply an EXTENT to specify its location on the volume. The independent overflow area may be on the same volume with the prime data area, or on a separate volume, but it must be wholly contained on one volume. It must be on the same type of device as that containing the prime data area. You can specify this area during a load or add operation, but it is used during the add operation.

If you specify both an independent overflow area and cylinder overflow area (by DTFIS entry CYLOFL), ISAM places overflow records first in the cylinder overflow area within the prime data area. When any cylinder overflow area becomes filled, ISAM writes the additional overflow records from that cylinder in the independent overflow area.

ISAM OPEN first validates the EXTENT statement specifications:

1. The master and cylinder index extents must be continuous and on the same unit.

2. No extents must overlap.

3. The valid extent types are:

   1 - Prime Data
   2 - Independent Overflow
   4 - Master Index
   4 - Cylinder Index

4. The Extent Sequence Number must be in a specified order:

0	Master Index
1	Cylinder Index
2 through n	Prime Data
n+1	Independent Overflow

OR

0	Master Index
1	Cylinder Index
2	Independent Overflow
3 through n	Prime Data

IOCS checks the limits of an EXTENT against the limits of each Extent field of each label already written in the VTOC. If the new extents overlap any previously written extents, IOCS checks the expiration data of the old (being overlapped) file to ensure that the data records are no longer active.

If the expiration data has passed, IOCS deletes the old label(s), which in effect, removes the expired file from the volume.

If an expiration date has not passed, a message is given to the operator. The operator can terminate the job or delete the unexpired file and continue processing.

If you use EXTENT and omit the Symbolic Unit field, IOCS uses the Symbolic Unit of the preceding EXTENT. The first EXTENT must contain the Symbolic Unit. If you use EXTENT for a <u>multivolume file,</u> you must supply, for each volume, at least one EXTENT statement containing the Symbolic Unit.

For a multivolume file, all extents (and therefore all volumes) are opened before any data records are added. Thus, all volumes that contain the file must be on-line and ready at the same time.

IOCS locates the format-1 label of the file to be processed by first reading the address of the VTOC in the VOL1 label and then searching the VTOC for the format-1 label that contains the File Identification you specify in DLBL. If an independent overflow area is specified during the add function on an existing volume, IOCS updates the format-1 label.

If the independent overflow area is specified on a new volume, IOCS writes a format-1 label for that volume.

The File Identification (field K1) was written in the Key area of the label record when the file was created. Thus, you must specify the same identification now as you did when the file was written during the load operation.

If you use DLBL and omit the File Identification, IOCS searches for the label in the VTOC by using the DTF name that you specify in the DLBL Filename field.

For label fields, D1-D21, IOCS OPEN routines check the appropriate fields against the corresponding information supplied by you in DLBL or in a DTF specification. Some fields provide information that is required during the processing of data, and other fields are ignored. See "Section: Label Fields" (Figure 20 on page 105 and Figure 21 on page 107) for the details about each field of the label.

You must specify in the DLBL statement a File Type of ISE for the ADD function. If you use DLBL and omit this field (D11 of the format-1 label), IOCS assumes the code for an SAM file in this field.

The ADD function is used to insert new records into an organized file and is specified by the DTFIS entry IOROUT.

If you use DLBL and omit some specifications, IOCS assumes predetermined default values (see Figure 21 on page 107)

Label fields D22-D25, D26-D29, and D30-D33 of the format-1 label define three areas (extents) of the volume, IOCS wrote these fields during the load operation from the EXTENT statements you supplied.

The Extent Type 1 (prime data), 2 (independent overflow area), and 4 (master or cylinder index) in the EXTENT is converted to a hex 01, 02, and 04, respectively by IOCS in fields D22, D26, and D30 of the format-1 label. If you use EXTENT and omit the type, this field is not checked.

The extent information was written in the format-1 label in the same order that the EXTENT statements were supplied during the load operation. The EXTENT statements must be supplied in this same order for the retrieve function. This order is specified by the Extent Sequence Number, fields D23, D27, and D31. The First Extent (D22-D25) is for the master index (if specified) and cylinder index. Additional Extent (D26-D29) is for the prime data area, and Additional Extent (D30-D33) is for the independent overflow area if specified. The prime data area and independent overflow area may be reversed.

During an ADD operation, ISAM uses the Extent Sequence Number in conjunction with the Symbolic Unit (SYSnnn, specified in EXTENT) to determine on what volume the extent area is located. The EXTENT statements must be entered in ascending order by Extent Sequence Number, with none missing.

During an ADD function, IOCS checks the Extent Upper Limit (fields D25 and D29 in the format-1 label) against the upper limit specified by EXTENT. If the specified limit of the independent overflow area is beyond the upper limit in the label, IOCS changes the label and makes the new area available for records. If an independent overflow area is established at this time, IOCS writes the lower and upper limit of the new extent specified by EXTENT. Under any other condition, the job is canceled if the limits do not agree.

After checking the label(s), IOCS makes the areas of the volume available for processing. In a multivolume file, IOCS makes all volumes available at the same time.

IOCS locates the format-2 label by reading the address of that label in the Pointer Field (D34) of the format-1 label.

Format-2 Label

A format-2 label is required and maintained by ISAM. This label is used to carry updated information from one use (function) of the file to the next and to retain many fields of the DTFIS table.

No separate EXTENT statements are required for the format-2 label. ISAM OPEN/CLOSE routines wrote the label during the load operation by using the information that you supplied in the DTF specification or that was calculated during the processing of data records.

Some of the fields in the format-2 label are written when the file is opened, whereas other fields are written when the file is closed.

IOCS always writes a format-2 label in the VTOC after it has written the format-1 label. It writes the address of the format-2 label in the Pointer Field (D34) of the format-1 label.

If a file occupies two or more volumes, IOCS writes the format-2 label only on the volume containing the cylinder index. This volume may or may not contain data records. The format-2 label is not repeated on the additional volumes (as the format-1 label is).

The RECSIZE written in the format-2 label by the LOAD is used by the LOAD during the ADD operation, _not_ the RECSIZE in the ADD DTFIS.

Several fields of the format-2 label can be used to determine the status of overflow areas:

D9 - Highest "R" on Overflow Tracks: The number of the last record on each track of the cylinder and/or independent overflow area.

D25 - Last Independent Overflow Record Address: The address of the last record written in the independent overflow area.

D27 - Number of Independent Overflow Tracks: The number of tracks still available in the independent overflow area.

D28 - Overflow Record Count: A count of the records written in the cylinder overflow areas and/or independent overflow.

D29 - Cylinder Overflow Area Count: A count of the cylinder overflow areas that have been filled. The statistics provided in several fields of the format-2 label can be used to determine whether you should reorganize the file:

D12 - Tag Deletion Count: The number of records _you_ identify (tag) for deletion (not processed by IOCS).

D13 - Non-First Overflow Reference Count: The number of times a READ instruction causes a search of the overflow area(s) for a record that is the second or higher in an overflow chain.

D16 - Prime Record Count: The number of logical records written in the organized file in the prime data area(s). IOCS accumulates this count during a LOAD operation.

D27 - Number of Independent Overflow Tracks: The number of tracks still available in the independent overflow area.

D28 - Overflow Record Count: The number of records written in all the overflow areas for the file (cylinder overflow areas and/or independent overflow area).

D29 - Cylinder Overflow Area Count: The number of cylinder overflow areas that have been filled.

See "Section: Label Fields" (Figure 24 on page 120) for the details about each field of the format-2 label.

ISAM FILES, RETRIEVE FUNCTION

VOL1 Label

The standard volume label (VOL1) must be on cylinder 0, track 0, record 3. If it is not, the job is canceled.

The VOL1 label contains a permanent Volume Serial Number.

Whenever a logical file is to be processed, IOCS reads and checks the VOL1 label against the Volume Serial Number you supply in an EXTENT statement. For a single volume (requiring a minimum of two extents), or for a multivolume file, IOCS performs this check for each EXTENT. If an error is detected, a message is issued to the operator. The operator may mount the correct volume and continue processing, or terminate the job.

If you use EXTENT and omit the Volume Serial Number, IOCS checks against the number of the previous EXTENT. If there was no previous EXTENT, IOCS assumes that the correct volume is mounted and does not check the VOL1 label.

For a multivolume file, all extents (and therefore all volumes) are opened before any data records are read or updated. Thus, all volumes containing the file must be on-line and ready at the same time.

IOCS determines the location of the VTOC from the Data File Directory field of the VOL1 label.

If any additional volume labels (VOL2-VOL8) follow the VOL1 label, IOCS ignores them.

Format-1 Label

You must supply one DLBL statement for the logical file, and one EXTENT statement for each separate area (extent) that the file occupies on the volume(s).

EXTENT statements define the areas (extents) of the volume where the data records (prime data area), cylinder and master indexes, and independent overflow records are written. An EXTENT statement provides the starting address (called relative track) and the number of tracks which indirectly give the ending address. An EXTENT statement also contains a code for the type of records to be written, and indicates the sequence in which this statement should be inserted into the input stream.

For a retrieve (RETRVE) function you must supply the same EXTENT statements that you supplied for the LOAD or ADD function. You must supply an EXTENT for the prime data area (data records) and the cylinder index. If you specified a master index during the load operation, you must supply an EXTENT for this area. If you specified an independent overflow area during the load or add operation, you must supply an EXTENT for this area.

The prime data area for a logical file is one continuous area on any one volume. It is not scattered over separate areas on a single volume. If the prime data area, however, extends to one or more other volumes, you supply a separate EXTENT statement for each volume.

The prime data area on any volume starts on track 0 of any cylinder, with the exception of track 0 of cylinder 0, which is reserved for labels and system use. Therefore, the prime data area starts on some cylinder other than cylinder 0 and is never written on cylinder 0, track 0 of any volume.

For a multivolume file, the prime data area of the _first_ volume starts on any cylinder (except 0) and extends through the _last_ track on the volume. On all volumes after the first, the prime data starts on cylinder 1, track 0 so that IOCS considers the prime data area as one continuous area. On all succeeding volumes (except the last) the prime data area extends through the last track on the volume. On the last volume, it ends at the end of any cylinder. Thus, in a multivolume file all volumes, except the first and last, are completely allotted to the prime data area from cylinder 1, track 0 through the last track in the last cylinder.

For a multivolume file, the VTOC for the first volume precedes the prime data area. On the last volume, the VTOC is on cylinder 0 or it follows the prime data area. On all other volumes, the VTOC is on cylinder 0.

Because the prime data area is considered as one continuous area in a multivolume file, the master/cylinder index and independent overflow areas are located before the prime data on the first volume or after the prime data area on the last volume.

During the load operation, ISAM has built a separate _track index_ for each cylinder used by the file. Track indexes are considered a part of the prime data area and, as such, do not require separate EXTENT statements. Each track index starts on the first track (0) of the cylinder that it is indexing. It can occupy a full track, more than one track, or part of a track and share that track with prime data records (shared track).

Also, within the prime data area, certain tracks may have been reserved for overflow records that occur when records are added to the file. These tracks, called a _cylinder overflow_ area, are reserved during the load operation by specifying the DTFIS entry CYLOFL, and are used during an add operation.
Because this is part of the prime data area, no separate EXTENT statements were required during the load operation. For the retrieve function, the DTFIS entry is not required.

The master index and the cylinder index are separate indexes and require two separate EXTENT statements. However, when they were written on the volume, IOCS combined them into one index area and wrote the address of that combined area in the format-1 label. Therefore, for these indexes, you must specify (in the EXTENT statements) two adjacent areas.

ISAM has built the master (if used) and cylinder indexes during the load operation. These indexes are separate from the prime data area and wholly contained on one volume. They can be on the same volume with the prime data or on a separate volume. They can also be on a different type of device than the prime data area.

You must specify the location of the _cylinder index_ by EXTENT. It immediately follows the master index on a volume, and it may be located on one or more successive cylinders. You must also specify an Extent Type of 4 and an Extent Sequence Number of 1 in the EXTENT statement. If you use EXTENT and omit the Extent Type, this field is not checked.

If you specified a _master index_ during the load operation, you must again specify this option for the _retrieve_ function with the DTFIS entry MSTIND. You must specify its location by EXTENT. The master index immediately precedes the cylinder index on a volume, and it may be located on one or more successive cylinders. You must also specify an Extent Type of 4 and an Extent Sequence Number of 0 in the EXTENT statement. If you use EXTENT and omit the Extent Type, this field is not checked.

An _independent overflow_ area may have been specified for storing overflow records that occurred when records were added to the file. If you specified an independent overflow area during the load or add function, you must again supply an EXTENT statement to specify its location during the retrieve function. The independent overflow area may be on the same volume with the prime data area, or on a separate volume, but it must be wholly contained on one volume. It must be on the same type of device as that containing the prime data area.

If you specified both an independent overflow area and cylinder overflow areas (by DTFIS entry CYLOFL during the load operation), ISAM placed overflow records first in the cylinder overflow area within the prime data area. When any cylinder overflow area became filled, ISAM wrote the additional overflow records from that cylinder in the independent overflow area.

If you use EXTENT and omit the Symbolic Unit field, IOCS uses the Symbolic Unit of the preceding EXTENT. The first EXTENT must contain the Symbolic Unit. If you use EXTENT for a _multivolume file_, you must supply, for each volume, at least one EXTENT statement containing the Symbolic Unit.

For a multivolume file, all extents (and therefore all volumes) are opened before any data records are retrieved. Thus, all volumes that contain the file must be on-line and ready at the same time.

IOCS locates the format-1 label of the file to be processed by first reading the address of the VTOC in the VOL1 label and then searching the VTOC for the format-1 label that contains the File Identification that you specify in DLBL. The File Identification (field K1) was written in the key area of the label record when the file was created. Thus, you must specify the same identification now as you did when the file was written.

If you use DLBL and omit the File Identification, IOCS searches for the label in the VTOC by using the DTF name that you specify in the DLBL Filename field.

IOCS does not check label fields D1-D21 against the corresponding information supplied by you in DLBL or in a DTF specification as it did during the load or add operation.

You must specify in the DLBL statement a File Type of ISE for the retrieve (RETRVE) function. If you use DLBL and omit this field (D11 of format-1 label), IOCS assumes the code for an SAM file in this field.

Use the DTFIS entry IOROUT to specify the retrieve (RETRVE) function, which retrieves records from a file for either random or sequential processing and/or updating. You must also specify the DTFIS entry TYPEFLE for a retrieve function to designate whether the type of processing to be performed is random, sequential, or both.

If you use DLBL and omit some specification, IOCS assumes predetermined default values (see Figure 21 on page 107).

Label fields D22-D25, D26-D29, and D30-D33 of the format-1 label define three areas (extents) of the volume. IOCS wrote these fields from the EXTENT statements you supplied during the load operation.

The Extent Type 1 (prime data), 2 (independent overflow area), and 4 (master or cylinder index) in the EXTENT is converted to a hex 01, 02, and 04, respectively, by IOCS in fields D22, D26, and D30 of the format-1 label. If you use EXTENT and omit the type, this field is not checked.

The extent information was written in the format-1 label in the same order that the EXTENT statements were supplied during the load operation. The EXTENT statements must be supplied in this same order for the retrieve operation. This order is specified by the Extent Sequence Number, fields D23, D27, and D31. The First Extent (D22-D25) is for the master index (if specified) and cylinder index. Additional Extent (D26-D29) is for the prime data area, and Additional Extent (D30-D33) is for the independent overflow area (if specified). The prime data area and independent overflow area may be reversed.

During a <u>retrieve</u> operation, ISAM uses the Extent Sequence Number in conjunction with the Symbolic Unit (SYSnnn, specified in EXTENT) to determine on what volume the extent area is located. The EXTENT statements must be entered in ascending order by Extent Sequence Number, with none missing.

IOCS makes available (for processing) the areas you specify in EXTENT (the data records, index(es), and independent overflow records) without checking against the limits in the label (Extent Sequence Number and Symbolic Unit are checked). In a multivolume file, IOCS makes all the volumes available at the same time.

IOCS locates the format-2 label by reading the address of that label in the Pointer Field (D34) of the format-1 label.

Format-2 Label

A format-2 label is required and maintained by ISAM. This label is used to carry updated information from one use (function) of the file to the next and to retain many fields of the DTFIS table.

No separate EXTENT statements are required for the format-2 label. ISAM OPEN/CLOSE routines wrote the label during the load operation by using the information you supplied in DTF specifications or that were calculated during the processing of data records.

Some of the fields in the format-2 label are written when the file is opened, whereas other fields are written when the file is closed.

IOCS always writes a format-2 label in the VTOC after it has written the format-1 label. It writes the address of the format-2 label in the Pointer Field (D34) of the format-1 label.

If a file occupies two or more volumes, IOCS writes the format-2 label only on the volume containing the cylinder index. This volume may or may not contain data records. The format-2 label is not repeated on the additional volumes (as the format-1 label is).

The RECSIZE written in the format-2 label by the LOAD is used during the retrieve operation, <u>not</u> the RECSIZE in the RETRVE DTFIS.

The statistics provided in several fields of the format-2 label can be used to determine whether you should reorganize the file:

D12 - Tag Deletion Count: The number of records _you_ identify tag for deletion (not processed by IOCS).

D13 - Non-First Overflow Reference Count: The number of times a READ instruction causes a search of the overflow area(s) for a record that is the second or higher in an overflow chain.

D16 - Prime Record Count: The number of logical records written in the organized file in the prime data area(s). IOCS accumulates this count during a LOAD operation.

D27 - Number of Independent Overflow Tracks: The number of tracks still available in the independent overflow area.

D28 - Overflow Record Count: The number of records written in all the overflow areas for the file cylinder overflow areas and/or independent overflow area.

D29 - Cylinder Overflow Area Count: The number of cylinder overflow areas that have been filled.

See "Section: Label Fields" (Figure 24 on page 120) for the details about each field of the format-2 label.

LABEL FIELDS FOR SAM AND DAM FILES ON DASD AND DISKETTE DEVICES

This section describes all the DASD and diskette labels supported by VSE:

Volume Label 1 (VOL1)) - DASD and Diskette

Format-1 label

Format-2 label

Format-3 label

Format-4 label

HDR1 label - Diskette

User-Standard Header Label (UHL1-UHL8)

User-Standard Trailer Label (UTL1-UTL8).

Each label is illustrated, and each field of each label is described in detail. The individual fields in the illustrations are numbered to relate to the corresponding descriptions. The label fields located in the key area of a DASD record are numbered K1-Kn. The fields of a diskette record, or in the data area of the DASD record are numbered D1-Dn.

The descriptions of the label fields include the:

- _Displacement_ in hex notation.
- _Field_ _Number_ - Kn or Dn
- _Length_ of the field in bytes (hex notation).
- _Content_ of each field, together with the name of the field.

An additional table shows for each field:

- _Source_ _of_ _Information_ for checking or writing this field.
- _Purpose_ of the field.
- _Processing_ performed on input/output.

- DLBL/EXTENT Default for the format-1 and format-3 labels only.

Throughout this section, the requirements and specifications relating to the 3330 apply also to the 3333 and the 3350 in 3330-1 compatibility mode. The requirements and specifications given for the 3340 apply also to the 3344.

LABEL FIELDS FOR DASD

Volume Label on Disk (VOL1)

The volume label has a 4-byte key area and an 80-byte data area. Both the key area and the first four bytes of the data area always contain the characters "VOL1" for the first volume label. Additional volume labels are ignored by VSE.

The displacement is in hex notation, counting from the beginning of the label.(after the key fields). The fields are identified by the numbers K1,K2 (key fields) and D1 to D13 (data fields).

Figure 18 shows the format of volume labels on disk.

Displ.	Field	Length	Content
0	K1	3	Identifier: VOL. IOCS checks whether a VOLUME Label is present on the Volume.
3	K2	1	Volume Label No. VSE supports only VOL1.
0	D1	3	Identifier: VOL. Checked by IOCS
3	D2	1	Volume Label No. only VOL1 supported
4	D3	6	Volume serial number provides a unique identification for the volume. It is generally assigned when the volume is first received in the installation. The source of information is the EXTENT statement. IOCS checks the Serial No. given in the EXTENT statement against this field. If no Serial No. Operand is specified in the EXTENT Statement IOCS assumes the correct volume mounted.
A	D4	1	Security byte used by OLTEP
B	D5	5	VTOC address. Contains the address of the Format-4 label. This address is written at initialisation time.
10	D6	5	Blank
15	D7	4	CI-size for FBA, blanks for CKD
19	D8	4	Number of blocks per CI for FBA, blanks for CKD
1D	D9	4	Number of labels per CI for FBA, blanks for CKD
21	D10	4	Blank
25	D11	E	Owner code for LVTOC listing
33	D12	1D	Blank

Figure 18. Disk Volume Label (VOL1 Label)

Field	
K1,D1	Source of Information : System Purpose: Identifies the standard volume label. This field is written in the first three positions of both the key and data areas of the volume label record. Processing: On both input and out,IOCS checks this field to verify that a standard volume label is present on the volume.The volume label should be written previously, before a logical file of data records is written on the volume.
K2,D2	Source of Information : System Purpose: Indicates the sequence of this label within the volume label (VOL) group. DOS/VSE supports Volume Label only, but provision is made for additional standard volume labels if required in other systems. This field is written in the fourth position of both the key and data areas of the volume label record. Processing: This field is processed in conjunction with the label identifier to completely identify the volume label.
D3	Source of Information : EXTENT Purpose: Provides a unique identification for a volume. The number is generally assigned when the volume is first received in th stallation. This number is also used as the File Serial Number in the format-1 label of each logical file written on the volume. This provides a unique identification of the volume/file relationship. If a multivolumelogical file is written, the Volume Serial Number of the first volume becomes the File Serial Number in the format-1 label on all volumes. Processing: On both input and out,IOCS checks this field against the number supplied by the user in the Volume Serial Number field of EXTENT. If EXTENT is used and no operand is specified, IOCS assumes the correct volume is mounted and does not check this field.
D4	Source of Information : - Purpose: Provides a code to indicate that additional identification is required before a volume can be considered the correct one for processing. DOS/VSE does not use this field, but provision is made for additional sec in other system. For example,OS/VS allows an operator response of a predetermined 'password' to futher authorize a volume for processing. Processing: On both input and out,IOCS ignores this field.
D5	Source of Information : - Purpose: Provides the starting address of the Volume Table of Contents (VTOC). This address is written along with the Volume Serial Number when the volume is initialized. Processing: On both input and out,IOCS refers to this field to find out where standard labels are located on this volume.
D6	Source of Information : - Purpose: Reserved for future use. Should contain blanks.

Figure 19 (Part 1 of 2). Standard Volume Label 1 Fields (DASD)

Field	
D7	Source of Information : - Purpose: Identifies the fixed-length control interval size by which the VTOC for FBA devices is subdivided.
D8	Source of Information : - Purpose: Indicates the number of physical blocks per control interval.
D9	Source of Information : - Purpose: Indicates the number of fixed-length slots in each control interval which may contain labels.
D10	Source of Information : - Purpose: Reserved for futher use. Should contain blanks.
D11	Source of Information : - Purpose: Identifies the owner or assignee to whom this volume belongs, such as a customer, installation, department, or system. This can be of value for controlling the allocation of volumes in a large installation. This field is printed on SYSLST when the LISTVTOC program is executed. Processing: On both input and out,IOCS ignores this field.
D12	Source of Information : - Purpose: Reserved for future use. Should contain blanks.

Figure 19 (Part 2 of 2). Standard Volume Label 1 Fields (DASD)

IBM-Standard File Labels on Disk

Types: Traditionally, four types of IBM-standard file labels are counted:

- Format-1, the normal disk file label for the first 3 extents
- Format-3, a file continuation label for the next 13 extents
- Format-2, used with ISAM only.
- Format-4, the VTOC file label, written at initialization of the device

Size: An IBM-standard file label is 140 bytes long and consists of a 44-byte key area and a 96-byte data area.

The VTOC: All IBM-standard file labels on a volume are in the VTOC, a directory of all files on the volume. The VTOC itself is a file also and has its own file label, the VTOC label. The VTOC is located where you specify it when you initialize your volume. The address of the VTOC label (format-4) is saved in the volume label.

Several Volumes: For several volumes of one file, the file label is repeated in the VTOC of each volume. The file label on each volume describes the portion of the file on that volume and its extents.

Figure 20 on page 105 to Figure 24 on page 120 show IBM-standard label formats for disk files, that is, the first IBM-standard file label, the continuation label, the ISAM label and the VTOC label.

Displ.	Field	Length	Content
0	K1	2C	File-ID: 1-35 bytes if generation number (Gnnn) and version number (Vnn) are specified, else 1 to 44. Source of Information: DLBL or IOCS VSAM catalog routines, AMS DEFINE command. Processing: The File-ID may be specified in the DLBL-File-ID field, if this specification is omited, IOCS uses the DTF-name specified in the DLBL-filename-field (stored in the key area of the label record) to search (on input) in the VTOC key areas for the file entry. Under VSAM a data space name (VSAM catalog routine) or the name of an index or data is the contents (AMS DEFINE Stmt. or generated by VSAM).
2C	D1	1	Format ID: 1. Written (on output) and checked by IOCS to distinguish this label from the other types (Format 2-5).
2D	D2	6	Volume serial no.: numeric identification for the first volume of the file. Written by IOCSon output.
33	D3	2	Volume sequence number within the file to identify the volume in an multivolume file. Written (on output) and checked by IOCS.
35	D4	3	Creation date: yyddd. By IOCS from SYSCOM (on output), checked against label record (DLBL) on input. The actual year may be calculated by adding yy to 1900.
38	D5	3	Expiration date indicates when the data record is considered inactive. (Same format as creation date.) Source: DLBL, IOCS, AMS or System (creation date + 7 by default).
3B	D6	1	Number of extents of the multi extent file on this volume.
3C	D7	1	Used by OS/VS
3D	D8	1	Reserved
3E	D9	D	System code: indicates the Programming System which has written the file. IBMDOSVS is the code written by IOCS if DLBL is used.
4B	D10	3	Date of last access: yyddd; not used by VSE
4E	D10A	2	Reserved
50	D10B	2	Number of blocks per CI for FBA, blanks for CKD
52	D11	2	File type: hex 0008 for VSAM hex 2000 for DAM hex 4000 for SAM (default, field in DLBL omited) hex 8000 for ISAM Checked against type of DTFon input. Written from DLBL by IOCS on output.

Figure 20 (Part 1 of 2). IBM-Standard Disk File Label (Format-1)

Displ.	Field	Length	Content
54	D12	1	Record Format: Used by OS/VS. IOCS writes 0
55	D13	1	Flags for optional areas used for ISAM file: Bit 2: Master index Bit 3: Independent overflow area Bit 4: Cylinder overflow area From DTF and EXTENT
56	D14	2	Byte length of ISAM blocks, from DTF
58	D15	2	Record length of ISAM files. From DTF
5A	D16	1	Key length of ISAM blocks. From DTF
5B	D17	2	Key field location in ISAM block. From DTF
5D	D18	1	Flags: Bit 0: Last volume (SAM only) Bit 3: File security. From DLBL
5E	D19	1	Original space request was: Bit 1: in blocks 4: for continuous extent 5: for maximum continuous extent 6: not under specified minimum
5F	D19A	3	Used by OS/VS. IOCS writes blanks
62	D20	5	Used by OS/VS. IOCS writes zeros
67	D21	2	Start of next record to end-of-data distance (negative displacement)
69	D22	1	Type of extent: Categorie of records (from EXTENT) (default) 01: (prime) data area or data space extent 02: independent overflow area extent 04: master/cylinder index area extent 40: extent for user-standard labels 80: split cylinder extent (SAM)
6A	D23	1	Sequence number of extent in the file. From EXTENT or IOCS
6B	D24	4	Extent lower limit (cchh for CKD, bbbb for FBA)
6F	D25	4	Extent upper limit (cchh for CKD, bbbb for FBA)

The fields D22 - D25 are now repeated twice as D26 - D33 to describe the next two extents still allowed on this label. The Format-1 label can reflect 3 extents of a multiextent-file, additional extents are documented in an corresponding Format-3 label.

87	D34	5	Address of next label for the file on this volume. Written and used by IOCS

Figure 20 (Part 2 of 2). IBM-Standard Disk File Label (Format-1)

Field	
K1	DLBL/EXTENT Default : DTF Filename Source of Information: : DLBL/IOCS IOCS (VSAM Catalog Routines)/ Access Method Services DEFINE command Purpose: File-ID permits you to identify your logical file by an application-oriented unique name. A file-ID can be composed by linking together key words in a form compatible with the OS/VS file structure (for example, DOS.SYSSLB.FILE.VOLUME.3). Generation number identifies the various editions of a file, such as a grandfather-father-son relationship. Thus, it can be used to ensure that the desired edition of the file is selected for processing, if several editions are maintained. The editions should be numbered in sequence. Version number provides a more detailed identification of the editions of a file. For example, generation could specify a month (1-12), and version could specify a particular week (1-5) of the month. File-ID, generation, and/or version occupy the key area of the label record. IOCS uses this to identify the label of a file specified for processing. Processing: You can specify the file to be processed in the corresponding field of DLBL. If you use DLBL and omit this field, IOCS uses the DTF name specified in the DLBL Filename field. On input, IOCS searches the VTOC key areas for this identification specified (in File Identification or Filename) in the key area of the label record. Under VSAM, this field will normally contain a data space name generated by VSAM catalog routines. However, if this data space contains the data or the index of a Unique file, this field contains the name of the data or index specified in the DEFINE statement or generated by VSAM (if a name was not specified).
D1	DLBL/EXTENT Default : 1 Output only Source of Information : DLBL for Input DLBL/IOCS for Output Purpose: Distinguishes this type of label (format-1) from other types (formats 2, 3, 4, and 5) Processing: IOCS checks, or writes, the type of label specified by DLBL. If you use DLBL, IOCS ignores this field on input; on output, IOCS writes 1.
D2	DLBL/EXTENT Default : Volume Serial Number of first volume of the file. Output only. Source of Information : DLBL for Input DLBL/IOCS for Output Purpose: Provides a numeric (or code) identification for this logical file. It contains the Volume Serial Number from the VOL label, and this uniquely identifies the volume/file relationship. In a multivolume file, the format-1 label on each volume contains the Volume Serial Number of the first volume. Processing: IOCS checks, or writes, the file serial number specified by DLBL. If you use DLBL, IOCS ignores this field on input; on output, IOCS writes the Volume Serial Number of the first or only volume.

Figure 21 (Part 1 of 11). Format-1 Label Fields

Field	
D3	DLBL/EXTENT Default : 01 Source of Information : DLBL/IOCS Purpose: Identifies the order of the volumes of data records in a multivolume logical file. Processing: In a multivolume file you need to specify in DLBL the number of the first volume only. When you use DLBL, IOCS supplies 01 for the first volume. IOCS increases the number by 1 for each additional volume. IOCS checks, or writes, the number specified or updated. In DLBL specify a 4-digit EBCDIC number, which is converted to a 2-byte binary number in the label.
D4	DLBL/EXTENT Default : Today's date Output only Source of Information : DLBL for Input System for Output Purpose: Provides the date that the file was originally created. This can be used at a later time to determine how old the records are. Or, it can be used (in conjunction with, or in place of, generation number) to ensure that the desired edition of the file is selected for processing. Processing: On input, IOCS checks this date against that supplied by DLBL. Specify YYDDD (year and day of the year), which is converted to a 3-byte discontinuous binary number (ydd) in the label. If you use DLBL, the creation date in the label is not checked. On output, IOCS writes the system date that is available in the communication region of the Supervisor. You do not supply a creation date for an output file, in either DLBL.

Figure 21 (Part 2 of 11). Format-1 Label Fields

Field	
D5	DLBL/EXTENT Default : Creation Date Plus 7 Output only Source of Information : DLBL for Input DLBL/IOCS/ System for Output (Action P/W) DLBL/Access Method Services (VSAM) for Output (Action W) Purpose: Indicates the date that the data records may be considered inactive. At that time, the label of the old file may be deleted from the VTOC, which, in effect, deletes the entire file and makes the extent(s) available for new data. Processing: On input, IOCS checks this date against that supplied by DLBL. If you use DLBL, this field is not checked. On output, IOCS first determines if the extent(s) specified for the new output file overlaps an existing file. If so, IOCS then checks the expiration date of the existing file by comparing this field in the old file label to the system date in the communications region of the Supervisor. If the old file has expired, IOCS writes the label(s) for the new file in the VTOC. This label includes the new expiration date supplied by DLBL or DLBL. The extent(s) is then available for the data records of the new output file. If the old file has not expired, a message is given to the operator, who determines whether to overwrite the old data. The expiration date of a VSAM data space is for information only. A data space can only be deleted, whether it has expired or not, by an Access Method Services DELETE statement. In DLBL, specify yyddd (year and day of the year). In the DLBL Expiration Date field, you may specify either the date the file will expire, or a retention period for the file. For expiration date, specify yy/ddd (year/day of the year). The day may have 1-3 digits. For a retention period, specify d-dddd (1-4 digits, 0-9999). If you omit this field in DLBL, IOCS adds a 7-day retention period to the system date in the communication region of the Supervisor and writes the resulting date. In each case, the expiration date (after calculation, if necessary) written in the label is a 3-byte discontinuous binary number.
D6	DLBL/EXTENT Default : - Source of Information : IOCS for Output Purpose: Provides a control of the number of separate areas (extents) established for this file, as represented by the Extent fields written in the format-1 label (3 fields) and the format-3 label (13 fields). In a multivolume file, the count is accumulated separately for each volume. Processing: On input, IOCS ignores this field. On output, IOCS writes the accumulated count in this field, or gives a message to the operator and cancels the job if the count exceeds the allowable number. The maximum allowable count is: 3 - for an ISAM file. (Because the master and cylinder indexes are combined into one area, a maximum of 3 areas are set up from 4 EXTENT statements.) 15 - for a DAM file with user-standard labels. (Because IOCS sets up one extra Extent field for the user-standard label track, 16 areas are set up from a maximum of 15 EXTENT statements.) 16 - for a DAM file without user-standard labels, and for a VSAM data space. SAM files may have any number of extents.

Figure 21 (Part 3 of 11). Format-1 Label Fields

Field	
D7	DLBL/EXTENT Default : Blank. Output only Source of Information : IOCS for Output Purpose: Used by OS/VS for partitioned data sets. VSE does not use this field. Processing: On output, IOCS writes a blank.
D8	DLBL/EXTENT Default : Blank. Output only Source of Information : IOCS for Output Purpose: Reserved for future use. IOCS writes a blank.
D9	DLBL/EXTENT Default : DOS/360 Version 3. Output only Source of Information : DLBL/IOCS for Output Purpose: Indicates the Programming System under which this file is written. This can be of value when an installation uses more than one programming system. Processing: On input, IOCS ignores this field. On output, IOCS writes the information supplied in DLBL. If you use DLBL, IOCS writes: IBMDOSVS.
D10	DLBL/EXTENT Default : - Source of Information : - Purpose: Indicates the date of last access of this data set.
D10A	DLBL/EXTENT Default : Blanks. Output only Source of Information : IOCS for Output Purpose: Reserved for future use. IOCS writes blanks.
D10B	DLBL/EXTENT Default : DLBL/DTF Source of Information : - Purpose: Indicates the number of physical blocks per control interval for the FBA device file.
D11	DLBL/EXTENT Default : X'4000' Source of Information : DTF for Input DLBL/DLBL for Output Purpose: Verifies the type of organization used for this file. Processing: On input, IOCS checks this field against the type of DTF (DTFSD, DTFDA, or DTFIS) that you specify. For an output file, IOCS converts the code specified in DLBL Type of File, and writes this field: If you omit this field in DLBL, IOCS writes X'4000' in the label field.

Figure 21 (Part 4 of 11). Format-1 Label Fields

Field	
D12	DLBL/EXTENT Default : - Source of Information : IOCS for Output Purpose: Used by OS/VS to define the type of records: fixed length, blocked, truncated, etc. DOS/VSE does not use this field. Processing: On output, IOCS writes a binary zero.
D13	Source of Information : DTF/ EXTENT for Output Purpose: Indicates which optional areas are built for an ISAM file. This field is provided for use by VSE. Processing: IOCS checks, or writes, the appropriate code from the DTF specifications and extent information that you supply.
D14	Source of Information : DTF/DLBL for Output Purpose: Tells the length of the blocks of logical records (and therefore, the length of a physical record). Processing: On input, IOCS refers to this field to determine the length of the blocks of records previously written in the file. On output, IOCS writes the block length from the DTF specification that you supply.
D15	Source of Information : DTF for Output Purpose: Tells the length of each logical record. Processing: On input, IOCS refers to this field to determine the length of the logical records previously written in this file. On output, IOCS writes the record length from the DTF specification that you supply.
D16	Source of Information : DTF for Output Purpose: Tells the length of the key area for each record (unblocked records) or block of records. Processing: On input, IOCS refers to this field to determine the length of the key area used in this file. On output, IOCS writes the key length from the DTF specification that you supply.
D17	Source of Information : DTF for Output Purpose: Tells the location of the key field within the logical records, when blocked records are written in the file. Processing: On input, IOCS refers to this field to determine where the key field is located within each record. On output, IOCS writes the location of the high-order position of the key field from the DTF specification that you supply.

Figure 21 (Part 5 of 11). Format-1 Label Fields

Field	
D18	Source of Information : IOCS / DLBL for Output Purpose: Indicates that this is the last volume of a multivolume file that has been closed. Processing: On input, IOCS ignores this field. When an output file is closed, IOCS writes 1 in this field of the label on the last (or only) volume of the file. For all other volumes, IOCS writes 0 in this field on an end-of-volume condition. Purpose: Invokes data set security to prevent problem programs from accidentally accessing a data secured file. Processing: On input, this field is checked for the data security indicator. If it is ON, a message is issued to the operator stating that a data secured file is being accessed. On output, if DSF is specified in the DLBL statement, bit 3 is set to 1. Bit 3 is set ON for all VSAM format 1 labels.
D19	DLBL/EXTENT Default : - Source of Information : - Purpose: Indicates the type of request that was issued for the initial allocation. Bit 0,1 01 = Original request was in blocks 2,3 (Reserved, binary zeros.) 4 1 = Original request was for a contiguous extent. 5 1 = Original request was for the maximum contiguous extent. 6 1 = Original request was for the five or less largest extents that are larger than or equal to a specified minimum. 7 (Reserved, binary zeros.)
D19A	DLBL/EXTENT Default : - Source of Information : IOCS for Output Purpose: Used by OS/VS to indicate the amount of storage requested at the end of each extent. VSE does not use this field. Processing: On output, IOCS writes blanks.
D20	DLBL/EXTENT Default : - Source of Information : IOCS for Output Purpose: Used by OS/VS to point to the last record of a sequential or partition-organization file. VSE does not use this field. Processing: On output, IOCS writes binary zeros.

Figure 21 (Part 6 of 11). Format-1 Label Fields

Field	
D21	DLBL/EXTENT Default : - Source of Information: : - Purpose: Indicates the starting position of the next sequential record relative to the End-of-Date Pointer if it is used, and contains a binary value to be used as a negative displacement.
D22 D26 D30	DLBL/EXTENT Default : X'01' Output only Source of Information : EXTENT for Input EXTENT/ IOCS for Output Purpose: Defines the category of records (data, overflow, index, or user-standard labels) for which this area is reserved. The area is specified by the Extent Lower Limit and Extent Upper Limit fields. Processing: IOCS checks, or writes, this label field with the extent type specified by EXTENT: If you use EXTENT and omit the type, this field is not checked on input; on output, IOCS writes X'01'. If you include user-standard labels, IOCS establishes an area for them. You do not include EXTENT for this area. IOCS uses the First Extent field (D22-D25) to define this extent, codes it Extent Type X'40' (blank), and numbers it Extent Sequence 0. If less than 3 extents are required for a file, IOCS writes X'00' in the Extent Type field of the unused Additional Extent fields (D26 and D30).
D23 D27 D30	DLBL/EXTENT Default : D23-0 D27-1 D31-2 SD/DA files Output only Source of Information : EXTENT for Input EXTENT/ IOCS for Output Purpose: Determines the proper order of the extent areas in a multiextent file. For a SAM, or DAM file, separate extents may be located on the same or different volumes. For an ISAM file, multivolumes may be used for the data records (prime data area), but on any one volume, the data records must be contained within one extent. ISAM indexes and the ISAM independent overflow area are separate extents, on the same volume as the prime data area or on a different volume than the prime data area. Processing: For a SAM or DAM file, or VSAM data space, IOCS checks, or writes, this label field with the sequence number supplied by EXTENT. You may specify any sequence numbers you choose, but the numbers must be in ascending order. If you include user-standard labels, IOCS establishes an extent area and assigns it sequence number 0 (see Extent Type, field D22). If you use EXTENT for a SAM or DAM file, or for the creation of a VSAM data space, and omit the extent sequence number, this field is not checked on input. On output IOCS writes Extent Squence 0 in the First Extent Field, and adds 1 for each subsequent Extent field used. Extent sequence 0 represents the first EXTENT card or, if they are used, it represents the area for user-standard labels. In the latter case, the first EXTENT card becomes Extent Sequence 1. For an ISAM file, IOCS processes this field the same way it processes the Lower and Upper Limit fields, D24 and D25. You must include EXTENT for each area and specify Extent Sequence Number. Extent information must be supplied in a specified order.

Figure 21 (Part 7 of 11). Format-1 Label Fields

Field	
D24 D28 D32	DLBL/EXTENT Default : - Source of Information : EXTENT Purpose: Defines the beginning of a disk area allocated to this file. Processing: For a SAM input file, IOCS checks this field and the Upper Limit field (D25/D29/D33) against the starting and ending addresses supplied by EXTENT. IOCS makes the area specified by EXTENT available for processing if it is equal to, or falls within, the limits defined by these label fields. If not, a message is issued to the operator. If you omit EXTENT* for a SAM input file, IOCS does no checking and makes available the area defined by the label. For a DAM input file, IOCS makes the area defined by EXTENT available. It does not check this field against the EXTENT specifications. For a SAM, or DAM output file, IOCS writes, in this field the starting address (lower limit) supplied by EXTENT.

Figure 21 (Part 8 of 11). Format-1 Label Fields

Field	
D24 D28 D32	For an ISAM file, processing of this field varies with the type of operation performed: • LOAD Create: IOCS writes the starting address (lower limit) supplied by EXTENT. • LOAD Extent: IOCS checks this field against the lower limit supplied by EXTENT. If the limits are not the same, the job is canceled. If the prime data area is extended onto a new volume, IOCS writes the lower limit of the new extent specified by EXTENT. • RETRVE: This field determines the lower limit of the extent. If you use EXTENT, you need specify only Operation, Symbolic Unit, and Volume Serial Number. • ADD or ADDRTR: IOCS checks this field against the lower limit supplied by EXTENT. If the limits are not the same, the job is canceled. If an independent overflow area is established at this time, IOCS writes the lower limit of the new extent specified by EXTENT. If you include EXTENT for both a master index area and a cylinder index area in an ISAM file, IOCS combines the two areas into one extent and uses the lower limit of the master index for this field. For the creation of a VSAM data space, IOCS writes the starting address (lower limit) supplied by the EXTENT statement. In EXTENT, specify a Relative Track number (n-nnnnn). This is the sequential number of the track relative to cylinder 0, track 0: For 2311, Relative Track = 10 x cylinder number + track number. For 2314 or 2319, Relative Track = 20 x cylinder number + track number. For 3330, Relative Track = 19 x cylinder number + track number For 3340, Relative Track = 12 x cylinder number + track number. For 3350, Relative Track = 30 x cylinder number + track number. In EXTENT, for 2311/2314/2319 specify: 000CCC0HH where CCC = Cylinder number (000-199) HH = Head (or track) number (00-09) for 2311; (00-19) for 2314 or 2319. IOCS converts the specification to CCHH for the label field. Because cylinder 0, track 0 on each volume must be reserved for labels and system use, never specify a lower limit of all zeros.

Figure 21 (Part 9 of 11). Format-1 Label Fields

Field	
D25 D29 D33	DLBL/EXTENT Default : - Source of Information : EXTENT Purpose: Defines the end of a disk area allocated to this file. Processing: For a SAM input file, IOCS checks this field and the Lower Limit field (D24/ D28/ D32) against the addresses supplied by EXTENT. IOCS makes the area specified by EXTENT available for processing if it is equal to, or falls within, the limits defined by these label fields. If not, a message is issued to the operator. If you omit EXTENT for a SAM input file, IOCS does no checking and makes available the area defined by the label. For a SAM or DAM output file, IOCS writes, in this field, the ending address (upper limit) supplied by EXTENT. For an ISAM file, processing of this field varies with the type of operation performed: • LOAD Create: IOCS writes the ending address (upper limit) supplied by EXTENT. • LOAD Extent: IOCS checks this field against the upper limit specified by EXTENT. If the specified limit of either the cylinder index or the prime data area is beyond the upper limit in the label, IOCS changes the label and makes the new area available for records. If the prime data area is extended onto a new volume IOCS writes the upper limit of the new extent specified by EXTENT. Under any other condition, the job is canceled if the limits do not agree. • RETRVE: This field determines the upper limit of the extent. If you use EXTENT, you need specify only Operation, Symbolic Unit, and Volume Serial Number. • ADD or ADDRTR; IOCS checks this field against the upper limit specified by EXTENT. If the specified limit of the independent overflow area is beyond the upper limit in the label, IOCS changes the label and makes the new area available for records. If an independent overflow area is established at this time, IOCS writes the upper limit of the new extent specified by EXTENT. Under any other condition, the job is canceled if the limits do not agree. If you include EXTENT for both a master index area and a cylinder index area in an ISAM file, IOCS combines the two areas into one extent and uses the upper limit of the cylinder index for this field. For the creation of a VSAM data space, IOCS writes the ending address (upper limit) supplied by the EXTENT statement. In EXTENT, specify a Relative Track number (n-nnnnn) for the starting address, as described for label Fields D24/D28/D32 and the Number of Tracks (n-nnnnn). From these, IOCS computes the upper limit. In EXTENT, for disk specify: 000CCC0HH where: CCC = Cylinder number HH = Head (or track) number IOCS converts the specifications to CCHH for the label field.

Figure 21 (Part 10 of 11). Format-1 Label Fields

Field	
D34	DLBL/EXTENT Default : - Source of Information : IOCS for Output Purpose: Provides the address of the next label for this file on this pack, if required. For an ISAM file, it points to a format-2 label. For a SAM/DAM file, or VSAM data space, it points to a format-3 label if more than three extents are used on this volume. Processing: On input, IOCS refers to this field to find the address of the next label, if any. On output, whenever a format-2 or format-3 label is required for a file, IOCS finds a VTOC location for the label and writes its address in this Pointer field. IOCS always writes a format-2 label for an ISAM file. For a SAM/DAM file or VSAM data space, IOCS establishes a format-3 label if another EXTENT card is read after the format-1 label if filled. If a SAM/DAM file or VSAM data space does not require a format-3 label, IOCS writes binary zeros in this field.

Figure 21 (Part 11 of 11). Format-1 Label Fields

FORMAT-3 LABEL LAYOUT AND CONTENT

```
Displ. Field Length Content

  0      K1     4    Key code for continuation label(03030303)
                     Written by IOCS
  4      K2     1    Type of extent, from EXTENT:
                       01 = data extent (default)
                       80 = split cylinder extent
  5      K3     1    Extent sequence number (3 or more)
  6      K4     4    Extent lower limit (cchh for CKD, bbbb for FBA)
  A      K5     4    Extent upper limit (cchh for CKD, bbbb for FBA)
The fields K2 to K5 are repeated three times as K6 - K17, to
describe the extents 2, 3, and 4 of the key area.

 2C      D1     1    Continuation label code: EBCDIC 3, from IOCS
The fields K2 to K5 are now repeated nine more times as D2 - D37,
to describe the nine extents of the data area.

 87     D38     5    Address of next contin.label (cchhr or 0bbbb)
                     or zeros. From SAM IOCS only
```

Figure 22. IBM-Standard Disk File Continuation Label (Format-3)

Field	
K1	DLBL/EXTENT Default : - Source of Information: : IOCS REMARKS: Provides a code to distinguish this key from the keys (File Identification) of format-1 labels.
K2, K14, D2, D34	DLBL/EXTENT Default : X'01' Output only Source of Information: : EXTENT for Input EXTENT/IOCS for Output REMARKS: Like the extents in the format-1 label, the first byte (Extent Type) of each Extent field defines the category of records for which this area is reserved. IOCS checks against, or writes, the extent type specified by EXTENT: Type EXTENT Specifications Label Field Data area 1 X'01' Data area with split cylinder (SAM) 8 in EXTENT 128 in XTENT X'80' If you use EXTENT and omit the type, this field is not checked on input; on output, IOCS writes 01. IOCS writes 00 in the Extent Type fields of any unused Extents (2-13). Extent Types 02, 04, and 40, which may be written in a format-1 label, do not occur in a format-3 label. Types 02 and 04 apply only to ISAM files, which support three extents and the format-1 label only. Type 40 indicates user-standard labels, which precede the first data area extent for the file and therefore appear in the format-1 label.
K3, K15, D3, D35	DLBL/EXTENT Default : 3, 4, ... 15 Output only Source of Information: : EXTENT for Input EXTENT/IOCS for Output REMARKS: The second byte (Extent Sequence Number) of each Extent field in this label serves the same purpose as in the format-1 label. It determines the proper order of the extent areas in a multiextent file. IOCS checks against, or writes, the sequence number specified by EXTENT. If you use EXTENT and omit the extent sequence number, this field is not checked on input. On output, IOCS writes 0 for the first extent for the file (in the format-1 label), and adds 1 for each succeeding EXTENT. Thus, the first extent sequence number in the format-3 label is 3.
K4, K16, D4, D36	DLBL/EXTENT Default : - Source of Information: : EXTENT* for Input EXTENT for Output REMARKS: Bytes 3-6 (Lower Limit) of each Extent field define the beginning of a volume area allocated to this file. Your EXTENT specification and the processing of this field are the same as that described for the lower limit of a SAM/DAM extent in the format-1 label.

Figure 23 (Part 1 of 2). Format-3 Label Fields

Field	
K5, K17, D5, D37	DLBL/EXTENT Default : - Source of Information: : EXTENT* for Input EXTENT for Output REMARKS: Bytes 7-10 (Upper Limit) of each Extent field define the end of a volume area allocated to this file. Your EXTENT specification and the processing of this field are the same as that described for the upper limit of a SAM/DAM extent in the format-1 label.
D1	DLBL/EXTENT Default : 3 Output only Source of Information: : IOCS REMARKS: Distinguishes this type of label (format 3) from other types (formats 1, 2, 4, and 5). IOCS writes a format-3 label on any volume of the file that requires more than three extents, as indicated by a series of EXTENT statements and user-standard labels, if used (see format 1, Extent Type Field D22).
D38	Source of Information: : IOCS for Output REMARKS: Provides the address of another format-3 label, if required for a SAM file. DAM files support a maximum of 16 extents, which are defined by a format-1 label and one format-3 label. On input, IOCS refers to this field to find the address of the next label, if any. On output for a SAM file, if another EXTENT card is read after a format-3 label is filled, IOCS establishes an additional format-3 label. IOCS finds a VTOC location and writes its address in this Pointer field. If another format-3 label is not required, IOCS writes binary zeros in this field.

Figure 23 (Part 2 of 2). Format-3 Label Fields

FORMAT-2 LABEL LAYOUT AND CONTENT

Field			
K1 (00)	Name	:	Key Identification
	No. of Bytes:	:	1
	Content	:	X'02'
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	IOCS
	REMARKS: Provides a code to distinguish this key from the keys (File Identification) of format-1 labels		
K2 (01)	Name	:	Address of 2nd Level Master Index
	No. of Bytes:	:	7
	Content	:	Binary Zeros - Applies to OS/VS only.
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	IOCS
	REMARKS: Used by OS/VS to provide the address (MBBCCHH) of the first track of the second level of the master index.		
K3 (08)	Name	:	Last 2nd Level Master Index Entry
	No. of Bytes:	:	5
	Content	:	Binary Zeros - Applies to OS/VS only.
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	IOCS
	REMARKS: Used in OS/VS to provide the address (CCHHR) of the last entry in the second level of the master index.		
K4 (0D)	Name	:	Address of 3rd Level Master Ind.
	No. of Bytes:	:	7
	Content	:	Binary Zeros - Applies to OS/VS only.
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	IOCS
	REMARKS: Used by OS/VS to provide the address (MBBCCHH) of the first track of the third level of the master index.		

Figure 24 (Part 1 of 12). Format-2 Label Fields

Field			
K5 (14)	Name	:	Last 3rd Level Master Index Entry Address
	No. of Bytes:	:	5
	Content	:	Binary Zeros - Applies to OS/VS only.
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	IOCS
	REMARKS: Used by OS/VS to provide the address (CCHHR) of the last entry in the third level of the master index.		
K6 (19)	Name	:	(Reserved)
	No. of Bytes:	:	11
	Content	:	Binary Zeros
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	IOCS
	REMARKS	:	Reserved for future use.
K7 (24)	Name	:	Last Prime Track Address
	No. of Bytes:	:	8
	Content	:	DASD Address (CCHHR)
	Open/Close	:	-/-
	Functions	:	-
	Source of Information	:	-
	REMARKS: Indicates the address of the last prime track on the last prime cylinder.		
D1 (2C)	Name	:	Format Identifier
	No. of Bytes:	:	1
	Content	:	2 = Format 2 Numeric EBCDIC
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	IOCS
	REMARKS: Distinguishes this type of label (formats 2) from other types (formats 1, 3 4, and 5).		

Figure 24 (Part 2 of 12). Format-2 Label Fields

Field			
D2 (2D)	Name	:	Number of Index Levels
	No. of Bytes:	:	1
	Content	:	1 = Cylinder Index 2 = Cylinder Index and Master Index Binary
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	DTFIS
	REMARKS: A cylinder index is always required. Also, you may specify a master index (DTFIS MSTIND), if desired.		
D3 (2E)	Name	:	High Level Index Development Indicator
	No. of Bytes:	:	1
	Content	:	X'02'
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	IOCS
	REMARKS: Used by OS/VS to indicate that a master index is used and to tell the number of tracks reserved for it.		
D4 (2F)	Name	:	First Data Record in Cylinders
	No. of Bytes:	:	3
	Content	:	DASD address (HHR)
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	DTFIS + Calculation
	REMARKS: Provides the address of the first data record in each cylinder. This record follows the track index, which is written at the beginning of each cylinder. IOCS uses the record key length, specified by DTFIS KEYLEN, in the calculation of the length of the track index.		
D5 (32)	Name	:	Last Data Track in Cylinders
	No. of Bytes:	:	2
	Content	:	DASD Address (HH)
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	DTFIS + Calculation
D5 (32)	REMARKS: Indicates the last track that can be used, in each cylinder, for the organized file of data records. If this is other than 09 (19), a cylinder overflow area follows the organized file. IOCS determines this track from the size of the cylinder overflow area that you specify in DTFIS CYLOFL.		

Figure 24 (Part 3 of 12). Format-2 Label Fields

Field	
D6 (34)	Name : Number of tracks for Cylinder Overflow No. of Bytes: : 1 Content : Binary Zeros - Applies to OS/VS only Open/Close : X/- Functions : LOAD Create Source of Information : IOCS REMARKS: Used by OS/VS. Contains the number of tracks allocated to each cylinder overflow area.
D7 (35)	Name : Highest 'R' on High Level Index Tracks No. of Bytes: : 1 Content : Record Number Open/Close : X/- Functions : LOAD Create Source of Information : DTFIS + Calculation REMARKS: Provides the number of the last record on each track of the master and/or cylinder indexes. IOCS uses the record key length, specified by DTFIS KEYLEN, to determine how many index entries can be written on each track.
D8 (36)	Name : Highest 'R' on Prime Data Tracks No. of Bytes: : 1 Content : Record Number Open/Close : X/- Functions : LOAD Create Source of Information : DTFIS + Calculation REMARKS: Provides the number of the last data record, or block of records, on each full track of the organized file in the prime data area. IOCS uses the DTFIS specifications for record length, key length, and blocked records to calculate how many physical records can be written on each track. The number of the last data record in the first track of the file differs from the others if data records and track index entries share the same track (see Field D10).
D9 (37)	Name : Highest 'R' on Overflow Tracks No. of Bytes: : 1 Content : Record Number Open/Close : X/- Functions : LOAD Create Source of Information : DTFIS + Calculation REMARKS: Provides the number of the last record on each track of the cylinder and/or independent overflow area. IOCS uses the DTFIS specifications for record length and key length in the calculation of the number of records that can be written on an overflow track.

Figure 24 (Part 4 of 12). Format-2 Label Fields

Field			
D10 (38)	Name	:	'R' of Last Data Record on Shared Tracks
	No. of Bytes:	:	1
	Content	:	Record Number
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	DTFIS + Calculation
	REMARKS: If data records and track index entries are written on the same track (shared track), this field provides the number of the last data record on this track. IOCS uses the DTF specifications for record length, key length, and blocked records to determine how many physical data records can be written after the track index on the shared track.		
D11A (39)	Name	:	'R' of Last Date record on Unshared Track
	No. of Bytes:	:	1
	Content	:	Record Number
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	DTFIS + Calculation
	REMARKS: Indicates the record number of the last data record on an unshared track of the track index.		
D11B (3A)	Name	:	Highest 'R' on Independent Overflow Track
	No. of Bytes:	:	1
	Content	:	Record Number
	Open/Close	:	X/-
	Functions	:	-
	Source of Information	:	-
	REMARKS: Indicates the highest possible record number for independent overflow tracks with format F records.		

Figure 24 (Part 5 of 12). Format-2 Label Fields

Field	
D12 (3B)	Name : Tag Deletion Count No. of Bytes: : 2 Content : Number of records. Binary Open/Close : -/X Functions : RETRVE Source of Information : Count you accumulate in filenameT REMARKS: Provides a count of the number of records you identify (tag) for deletion. As you tag a record during a retrieve operation, you should add 1 to the counter addressed as filenameT. In subsequent retrieve operations, the count is read from the label back into filenameT, and additional tagged records can be added. The delete option is not supported by VSE. You must provide coding to test for records tagged for deletion. VSE passes this field between the format-2 label and the DTF. You can use this statistic (along with those in Fields D13, D16, D27, D28, and D29) to determine whether the file should be reorganized.
D13 (3D)	Name : Non-First Overflow Reference Count No. of Bytes: : 3 Content : Number of random references. Binary Open/Close : -/X Functions : RETRVE Source of Information : IOCS: Count in filenameR REMARKS: Provides a count of the number of times a READ instruction causes a search of the overflow area(s) for a record that is the second or higher in an overflow chain. IOCS accumulates this count in filenameR during a retrieve operation. In subsequent retrieve operations, IOCS reads the count from the label back into filenameR, and adds to it as required. You can use this statistic (along with those in Fields D12, D16, D27, D28, and D29) to determine whether a file should be reorganized.
D14 (40)	Name : Number of Bytes for Highest Level Index No. of Bytes: : 2 Content : Size of master index. Binary Open/Close : X/- Functions : LOAD Create Source of Information : DTFIS + Calculation REMARKS: Provides the size of the master index and thus indicates how many bytes of main storage are required for this index. IOCS calculates the size from the EXTENT limits and DTFIS KEYLEN specification.

Figure 24 (Part 6 of 12). Format-2 Label Fields

Field	
D15 (42)	Name : Number of Tracks for Highest Level Index No. of Bytes: : 1 Content : Tracks of master index. Binary Open/Close : X/- Functions : LOAD Create Source of Information : EXTENT REMARKS: Provides the number of tracks required for the master index. IOCS obtains this from the EXTENT limits for Extent Sequence 0.
D16 (43)	Name : Prime Record Count No. of Bytes: : 4 Content : Number of logical records. Binary Open/Close : X/- Functions : LOAD/ ADD Source of Information : IOCS: Count in filenameP+4 or filenameP REMARKS: Provides a count of the logical records written in the organized file in the prime data area(s). In a multi-volume file, this count is a total of the logical records on all volumes. During a LOAD operation, IOCS accumulates this count in the filename P+4. For an ADD operation, IOCS reads this count into filenameP and updates it to include the added records. You can make note of the count and use it during a retrieve operation to verify that all records are read. You can also use this statistic (along with those in Fields D12, D13, D27, D28, and D29) to determine whether a file should be reorganized.

Figure 24 (Part 7 of 12). Format-2 Label Fields

Field	
D17 (47)	Name : Status No. of Bytes: : 1 Content : Codes for filled area: Bit No. ON Meaning 2 File has been successfully closed 6 Last track full 7 Last block full Otherwise each bit is OFF (0) Open/Close : -/X Functions : LOAD/ ADD Source of Information : IOCS REMARKS: If bit 2 is OFF, the file is being used for an ADD or ADDRTR. If an OPEN is then issued to the file for ADD or ADDRTR when HOLD=YES, the problem program is canceled because another program is already using the file for ADD or ADDRTR. If an ADD or ADDRTR program terminates without issuing a CLOSE to the file, bit 2 remains OFF. Bit 2 should be set ON by issuing a CLOSE to that file in in any job in which ADD or ADDRTR is specified and HOLD does not equal YES. Bits 6 and 7 indicate that the organized file completely fills the prime data area. Bit 6 is ON when the last track that can be used for data records is filled and the end-of-file record is written on the last track of the area. When blocked records are specified, the last block may or may not be filled. If it is not, bit 7 is OFF and more logical records may be added to the last block. When the last block becomes full, bit 7 is turned ON. Thus, when both bits 6 and 7 are ON, any additional records for the file are written in an overflow area.
D18 (48)	Name : Address of Cylinder Index No. of Bytes: : 7 Content : DASD address (MBBCCHH) Open/Close : X/- Functions : LOAD Create Source of Information : EXTENT REMARKS: Provides the address of the first track of the cylinder index. IOCS obtains this address from the starting address you supply in the cylinder index (Extent Sequence 1) EXTENT statement.
D19 (4F)	Name : Address of Lowest Level Master Index No. of Bytes: : 7 Content : DASD address of index (MBBCCHH) Open/Close : X/- Functions : LOAD Create Source of Information : EXTENT
D19 (4F)	REMARKS: Provides the address of the first track of the master index. IOCS obtains this address from the starting address you supply in the master index (Extent Sequence 0) EXTENT statement. In VSE, this field, and Field D20, are identical whenever a master index is specified. The two fields are provided for use by OS/VS, which provides for three levels of master indexes. If a master index is not specified, this field contains binary zeros.

Figure 24 (Part 8 of 12). Format-2 Label Fields

Field			
D20 (56)	Name	:	Address of Highest Level Index
	No. of Bytes:	:	7
	Content	:	DASD address of master or cylinder index (MBBCCHH) M = Extent sequence number
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	EXTENT
	REMARKS: Provides the address of the first track of the master index, if specified (same as Field D19). If a master index is not used, this field contains the address of the cylinder index (same as Field D18). This field, and Field D19, are provided for use by OS/VS, which provides for three levels of master indexes.		
D21 (5D)	Name	:	Last Prime Data Record Address
	No. of Bytes:	:	8
	Content	:	DASD address (MBBCCHHR) M = Extent sequence number
	Open/Close	:	-/X
	Functions	:	LOAD/ ADD
	Source of Information	:	IOCS
	REMARKS: Provides the address of the last record (or block of records) written in the organized file in the prime data area. This address is first written during a LOAD operation and then updated, if necessary, during a LOAD Extend or ADD operation.		
D22 (65)	Name	:	Last Track Index Entry Address
	No. of Bytes:	:	5
	Content	:	DASD Address (CCHHR)
	Open/Close	:	-/X
	Functions	:	LOAD
	Source of Information	:	IOCS
	REMARKS: Provides the address of the last normal entry in the last track index currently written for the file. This address is first written during a LOAD Create operation, and then updated during a LOAD Extend Operation.		

Figure 24 (Part 9 of 12). Format-2 Label Fields

Field			
D23 (6A)	Name	:	Last Cylinder Entry Address
	No. of Bytes:	:	5
	Content	:	DASD Address (CCHHR)
	Open/Close	:	-/X
	Functions	:	LOAD
	Source of Information	:	IOCS
	REMARKS: Provides the address of the last entry written in the cylinder index. This address is written during a LOAD Create operation, and then updated during a LOAD Extend operation.		
D24 (6F)	Name	:	Last Master Index Entry
	No. of Bytes:	:	5
	Content	:	DASD Address (CCHHR)
	Open/Close	:	-/X
	Functions	:	LOAD
	Source of Information	:	IOCS
	REMARKS: Provides the address of the last entry written in the master index, if used. If a master index has not been specified, this field contains binary zeros. This address is first written during a LOAD Create operation, and then updated, if necessary, during a LOAD Extend operation.		
D25 (74)	Name	:	Last Independent Overflow Record Address
	No. of Bytes:	:	8
	Content	:	DASD Address (MBBCCHHR) M = Extent sequence number
	Open/Close	:	-/X
	Functions	:	LOAD Create/ ADD
	Source of Information	:	IOCS
	REMARKS: Provides the address of the last record written in the independent overflow area. This address is first written during a LOAD Create operation, when an end-of-file record is entered as the first record of the independent overflow area. The address is updated if records are transferred to the independent overflow area during an ADD operation.		
D26 (7C)	Name	:	Bytes Remaining on Overflow Track
	No. of Bytes:	:	2
	Content	:	Binary Zeros. Applies to OS/VS only.
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	IOCS
	REMARKS: Used by OS/VS to indicate the number of bytes that are still available in the last track in use at this time in the independent overflow area.		

Figure 24 (Part 10 of 12). Format-2 Label Fields

Field	
D27 (7E)	Name : Number of Independent Overflow Tracks No. of Bytes: : 2 Content : Number of unused tracks. Binary. Open/Close : -/X Functions : ADD Source of Information : IOCS: Count in filenameI. REMARKS: Provides the number of tracks that are still available in the independent overflow area. IOCS maintains this count in filenameI during an ADD operation. In subsequent ADD operations, IOCS reads the count from the label back into filenameI, and updates it as required. You can use this statistic (along with those in Fields D12, D13, D16, D28 and D29) to determine whether a file should be reorganized.
D28 (80)	Name : Overflow Record Count No. of Bytes: : 2 Content : Number of records. Binary Open/Close : -/X Functions : ADD Source of Information : IOCS: Count in filenameO. REMARKS: Provides a count of the records written in all the overflow areas for the file (cylinder overflow areas and/or independent overflow area). IOCS accumulates this count in filenameO during an ADD operation. In subsequent ADD operations, IOCS reads the count from the label back into filenameO and adds to it for additional overflow records. You can use this statistic (along with those in Fields D12, D13, D16, D27, and D29) to determine whether a file should be reorganized.
D29 (82)	Name : Cylinder Overflow Area Count No. of Bytes: : 2 Content : Number of overflow areas. Binary Open/Close : -/X Functions : ADD Source of Information : IOCS: Count in filenameA. REMARKS: Provides a count of the cylinder overflow areas that have been filled. IOCS accumulates this count in filenameA during an ADD operation. In subsequent ADD operations, IOCS reads the count from the label back into filenameA, and adds to it as required. You can use this statistic (along with those in Fields D12, D13, D16, D27, and D28) to determine whether a file should be reorganized.

Figure 24 (Part 11 of 12). Format-2 Label Fields

Field			
D30 (84)	Name	:	Dummy Track Index Entry
	No. of Bytes:	:	3
	Content	:	DASD Address (HHR)
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	-
	REMARKS: Contains the address (HHR) of the dummy track index entry.		
D31 (87)	Name	:	Pointer
	No. of Bytes:	:	5
	Content	:	Binary Zeros. Applies to OS/VS only.
	Open/Close	:	X/-
	Functions	:	LOAD Create
	Source of Information	:	IOCS
	REMARKS: Used by OS/VS to provide the address (CCHHR) of a format-3 label if more than three extents are used on this volume. VSE does not support more than three extents for an ISAM file.		

Figure 24 (Part 12 of 12). Format-2 Label Fields

FORMAT-4 LABEL LAYOUT AND CONTENT

Every field in this label, except the VSAM indicators (D9A), is written by DSF at initialization time.

Disp.	Field	Length	Content
0	K1	2C	Key code for VTOC label: 44 times 04
2C	D1	1	VTOC label identifier: EBCDIC 4.
2D	D2	5	Used by OS/VS
32	D3	2	Number of available file label spaces in VTOC at initialization (tracks x cylinder minus 2)
34	D4	4	Address of next alternate track (cchh), for FBA: zeros. From DSF
38	D5	2	Number of alternate tracks left. For FBA zeros From DSF
3A	D6	1	Flags: Bit 0: always on Bit 3: Volume reserved for emulators Bit 5: VTOC being updated by VSAM
3B	D7	1	Extent count. Always 1. VTOC is 1 extent
3C	D8	2	Reserved
3E	D9	E	CKD device constants: (FBA: zeros)
3E		2	Number of cylinders
40		2	Tracks per cylinder
42		2	Track length
44		1	Overhead bytes for I*
45		1	Overhead bytes for L*
46		1	Overhead bytes for K*
47		1	Flag byte Bit 4: I or L value* has two bytes for 3350 Bit 7: A tolerance is added to each record except the last on a track
48		2	Tolerance** per device type
4A		1	Number of labels on VTOC track per device
4B		1	Reserved
4C	D9A	B	VSAM indicators, from VSAM catalog routines
4C		8	Time when last data space was added
54		1	Ownership byte: Bit 0: Owned by VSAM catalog
55		2	Number of first track of CKD catalog recovery area, for FBA zeros
57	D10A/B	9	Used by OS/VS
60	D10C	4	Number of first block of FBA catalog recovery area, for CKD zeros
64	D10D	5	Reserved
69	D11	1	Extent type: 01 for VTOC extent
6A	D12	1	Extent sequence number: 00 (VTOC has 1 extent)
6B	D13	4	Start address of VTOC (label).
6F	D14	4	End address of VTOC. Used by IOCS
73	D15	19	Zeros

*) I = for a record with key area
L = for a last record with key area on a track
K = for a key area

**)The tolerance is added to the length of a record if bit 7 in the flag byte is on.

Figure 25. VTOC Label (Format-4)

User-Standard File Labels on Disk

Figure 26 shows user-standard disk file labels (header and trailer).

Displ.	Field	Length	Content
0	K1	3	UHL or UTL
3	K2	1	Label sequence number: 1 to 8 for header labels 0 to 7 for trailer label
4	D1	3	Same as field K1
7	D2	1	Label sequence number: 1 to 8 for all
8	D3	4C	User's label information

Figure 26. User-Standard Disk File Labels (Header and Trailer)

Field	
K1,D1	Source of Information: : IOCS REMARKS: This field identifies the label as a user-standard header (UHL) or trailer (UTL) label. It is written in the first three positions of both the key and data areas of the label record. On input you can refer to this field to determine whether a header or trailer label is to be processed. On output, IOCS provides the information (UHL/UTL) for this field.
K2	Source of Information: : IOCS REMARKS: Indicates the sequence of this label within this header label set (UHL) or trailer label set (UTL). This field is written in the fourth position of the key area of the label record. User-standard header labels are numbered UHL1-UHL8. User-standard trailer labels are numbered UTL0-UTL7. This field is processed with the Label Identifier to completely identify the user-standard label.
D2	Source of Information: : IOCS REMARKS: Indicates the sequence of this label within this header label set (UHL) or trailer label set (UTL). This field is written in the fourth position of the data area of the label record. User-standard header and trailer labels are numbered UHL1-UHL8 and UTL1-UTL8. This field is processed with the Label Identifier to completely identify the user-standard label.
D3	Source of Information: : User REMARKS: Provides a means for you to label your SAM/DAM file with any information you need in addition to that supplied by the standard labels.

Figure 27. User-Standard Label Fields

User-standard labels may be included for SAM or DAM files. VSAM and ISAM do not support them.

User-standard labels are header labels located and processed before the data of the file, and trailer labels located before and processed after the data of the file.

These labels have a 4-byte key area and an 80-byte data area. Both the key area and the first four bytes of the data area contain UHLn or UTLn; the remaining 76 bytes of the data area contain user-chosen data. A maximum of eight header and eight trailer labels may be written to describe a file.

There is always one header and one trailer label more written than specified. This extra label has only a 4 byte key area and no data area.

An example of a file for which five header labels and four trailer labels were specified is shown in Figure 28 on page 134.

Label#	Key Area	Data Area
1	UHL1	UHL1 + 76 bytes of label data fields
2	UHL2	UHL2 "
3	UHL3	UHL3 "
4	UHL4	UHL4 "
5	UHL5	UHL5 "
6	UHL6	
7	UTL0	UTL1 + 76 bytes of label data fields
8	UTL1	UTL2 "
9	UTL2	UTL3 "
10	UTL3	UTL4 "
11	UTL4	

Here follow the data.

Figure 28. User-Standard Disk File Labels (5 UHLs and 4 UTLs Specified)

If only header labels are specified, one UTL0 label without data is written by the system. An example is shown in Figure 29 where only 3 header labels were specified.

Label#	Key Area (4 bytes)	Data Area (80 bytes)
1	UHL1	UHL1 + 76 bytes of label data fields
2	UHL2	UHL2 "
3	UHL3	UHL3 "
4	UHL4	
5	UTL0	

Here follow the data.

Figure 29. User-Standard Disk File Labels (3 UHLs Specified)

You can include definitions or descriptions of your file in addition to those provided by the standard labels. For example, you may want to identify end-of-volume as opposed to end-of-file conditions, or you may have subcategories that you want to define for your files, or you may want to maintain an audit trail in these labels without the system security standards.

LABEL FIELDS FOR DISKETTE

Volume Labels on Diskette

Figure 30 shows the format of a diskette volume label.

Displ.	Field	Length	Content
0	D1	3	Label ID: VOL
3	D2	1	Ignored by VSE
4	D3	6	Volume serial number from EXTENT
A	D4	1	Accessibility indicator: S or Blank. From DTF
B	D5,D6	1A	Reserved
25	D7	E	Name or code of volume owner
33	D8	1C	Reserved
4F	D9	1	Label standard level: W

Figure 30. Diskette Volume Label

Field	
D1	Source of Information: : System Purpose: Identifies the standard volume label. This field is written in the first three positions of the volume label record. Processing: On both input and output, IOCS checks this field to verify that a standard volume label is present on the volume.
D2	Source of Information: : System Purpose: Indicates the sequence of this label within a volume label set; must contain a '1'. Processing: This field is processed in conjunction with the label identifier, to completely identify the volume label.
D3	Source of Information: : EXTENT Purpose: Provides a unique identification for a diskette (volume); generally assigned when the diskette is first received in the installation. Processing: On both input and output, IOCS checks this field against the number supplied by the user in the Volume Serial Number field of EXTENT. If no operand is specified, IOCS assumes the correct volume is mounted and does not check this field.
D4	Source of Information: : - Purpose: Provides a code which indicates that additional qualification is needed before a volume can be processed. Processing: If the volume is secure, an operator message is written any time a file is to be read or written on this volume. The operator must then make the appropriate response. For more information see 'VSE/Advanced Functions Messages.'
D5	Source of Information: : - Purpose: Reserved for future use; should contain blanks.
D6	Source of Information: : - Purpose: Reserved for future use; should contain blanks.
D7	Purpose: This field specifies the owner of the volume. Processing: On both input and output, IOCS ignores this field.
D8	Source of Information: : - Purpose: Reserved for future use; should contain blanks.

Figure 31 (Part 1 of 2). Diskette Standard Volume Label Fields

Field	
D9	Source of Information: : System Purpose: Identifies the version of label standard to which the labels and data formats on this volume conform; must contain 'W'. Processing: IOCS checks this field on both input and output; if not a 'W', job is terminated with a message.

Figure 31 (Part 2 of 2). Diskette Standard Volume Label Fields

A diskette volume has one volume label of 80 bytes. It is located on track 0, sector 7 and begins by VOL.

IBM-Standard File Labels on Diskette

Figure 32 shows the format of the diskette file label.

Displ.	Field	Length	Content
0	D1	3	Label ID: HDR
3	D2	1	Label sequence number: 1
4	D3	1	Blank
5	D4	8	File-ID from DLBL or system
D	D5	9	Blanks
16	D6	5	Record length. From IOCS
1B	D7	1	Blank
1C	D8	5	Start address of extent: Track and sector. From IOCS
21	D9	1	Blank
22	D10	5	End address of extent: Track and sector. From IOCS
27	D11	1	Blank
28	D12	1	Bypass byte: B or blank: B = job ends on input
29	D13	1	Security byte: S or blank
2A	D14	1	Write protection byte: P or blank
2B	D15	1	Interchange level: blank= sector length 128, unblocked, unspanned, sequential non-blank= job ends on input
2C	D16	1	Volume byte: blank= file complete on this volume C= file continued on next volume L= file ends on this volume
2D	D17	2	Volume sequence number
2F	D18	6	Creation date: YYMMD
35	D19	D	Blanks
42	D20	6	Expiration date: Default= 7 days after output
48	D21	1	Verify byte: V or blank
49	D22	1	Blank
4A	D23	5	End of data address
4F	D24	1	Blank

Figure 32. Diskette File Label

Field	
D1	Source of Information: : System Purpose: Identifies the Header label; must contain 'HDR'. Processing: IOCS checks this field on input, writes it on output.
D2	Source of Information: : System Purpose: Indicates the sequence of this label within a header label set; must contain a '1'. Processing: IOCS checks this field on input, writes it on output.
D3	Source of Information: : - Purpose: Reserved for future use; should contain blanks.
D4	Source of Information: : DLBL/IOCS Purpose: File ID permits you to identify your logical file. Processing: You can specify the file to be processed in the corresponding field of the DLBL. If you omit this field, IOCS uses the DTF name specified in the DLBL Filename field. On input, IOCS searches the VTOC for this identification. On output, IOCS writes the identification specified (in File Identification or Filename) in the label record. If this name is the same as an unexpired or write-protected file on the diskette, the job is terminated.
D5	Source of Information: : - Purpose: Reserved for future use; should contain blanks.
D6	Source of Information: : IOCS Purpose: Contains length of the data records recorded in this file. Processing: For an input DTFPH file, IOCS uses this field to set up the length field in the Read CCW. On output, the IOCS sets up this field.
D7	Source of Information: : - Purpose: Reserved for future use; should contain blanks.
D8	Source of Information: : IOCS Purpose: Defines the beginning of the diskette area allocated to this file. Processing: For an input file, IOCS makes available the area defined by the label. For an output file, IOCS writes, in this field, the starting address (lower limit) of the file. This address will be the address of the first record of the first track following the last unexpired or write-protected file on the diskette. IOCS ignores any values specified on the EXTENT card for both input and output files.

Figure 33 (Part 1 of 4). Diskette HDR 1 Label Fields

Field	
D9	Source of Information: : - Purpose: Reserved for future use; should contain blanks.
D10	Source of Information: : IOCS Purpose: Defines the end of the diskette area allocated to this file. Processing: For an input file, IOCS makes available the area defined by the label. For an output file at OPEN time, IOCS writes, in this field, the address of the last record on the diskette (73026). At CLOSE time, IOCS updates this field to be the address of the last record in the file. IOCS ignores any values specified on the EXTENT card for both input and output files.
D11	Source of Information: : - Purpose: Reserved for future use; should contain blanks.
D12	Source of Information: : IOCS Purpose: Indicates whether or not a file is to be interchanged. Processing: IOCS terminates the job on input if this field is non-blank. For an output file, IOCS creates this field as a blank.
D13	Source of Information: : IOCS Purpose: Indicates whether or not additional qualifications must be supplied in order to access this file. Processing: For an input file, if this byte is an 'S', an operator message is written. The operator must reply 'YES' to access the file. For an output file, if the user specifies (in the DTF) the file to be created as a secure file, IOCS will create this field as an 'S'.
D14	Source of Information: : IOCS Purpose: Indicates whether or not a file may be overwritten. Processing: For input files, IOCS ignores this field. For output files, if the user indicates in the DTF that the file is to be write-protected, IOCS puts a P in this field. If a file is created write-protected, it cannot be overwritten.

Figure 33 (Part 2 of 4). Diskette HDR 1 Label Fields

Field	
D15	Source of Information: : IOCS Purpose: Identifies the following file attributes: • Physical Record Length = 128 bytes • Record Length - Fixed = 128 bytes • Record Attributes - unblocked/ unspanned • File Organization - Sequential Processing: On both input and output, IOCS assumes the above attributes if this field is a blank. IOCS will create this field as a blank on output. If this field is not blank on an input file, the job will be terminated.
D16	Source of Information: : IOCS Purpose: Indicates whether a file is complete on this volume, continued to another volume, or completed on this volume: ($u8$2)$u - file complete on this volume C - file continued to another volume L - file completed on this volume. Processing: On input, IOCS checks this field to ensure that this indicator is correct. The only correct values are: ($u8$2)$u, for a single volume file; C for all but the last volume of a multivolume file; and L for the last volume of a multivolume file. On output, IOCS will set this indicator to the proper value based on the type of file being created.
D17	Source of Information: : IOCS Purpose: Indicates the order of a volume in a multivolume file, relative to the first volume of that file. Processing: On input, if the DTFDU VOLSEQ parameter is specified, IOCS will check that the volume sequence numbers of a multivolume file are in consecutive, sequential, ascending order, starting with 1. On output IOCS will automatically create consecutive, sequential, ascending sequence numbers for a multivolume file (starting with 1).
D18	Source of Information: : IOCS/System Purpose: Indicates the date the file was created; the format is YYMMDD. Processing: On input, IOCS ignores this field. On output, IOCS creates this field equal to the current system date.
D19	Source of Information: : - Purpose: Reserved for future use; should contain blanks.

Figure 33 (Part 3 of 4). Diskette HDR 1 Label Fields

Field	
D20	Source of Information: : IOCS/ EXTENT Purpose: Indicates the date this file may be purged; the format is YYMMDD. Processing: On input, IOCS ignores this field. On output IOCS creates this field equal to the expiration date specified on the EXTENT card. If a retention period is specified, the expiration date is calculated from that. If no date is specified, IOCS creates this date equal to seven days from the current system date. When creating an output file, IOCS deletes expired and non-write-protected files which begin after the last unexpired or write-protected file on the volume.
D21	Source of Information: : IOCS Purpose: Indicates whether or not data has been subjected to a verification procedure. Processing: IOCS ignores this field on both input and output.
D22	Source of Information: : - Purpose: Reserved for future use; should contain blanks.
D23	Source of Information: : IOCS Purpose: Contains the address of the next higher consecutively numbered unused record; the format is CCHHR. On input, IOCS supplies this field as the actual end-of-data address. On output, IOCS creates this field as the actual end-of-data address.
D24	Source of Information: : - Purpose: Reserved for future use; should contain blanks.

Figure 33 (Part 4 of 4). Diskette HDR 1 Label Fields

The IBM-standard file label on diskette is 80 bytes long. The key area of 4 bytes always contains the characters HDR1. The 76 byte data area contains the start and end address of the file or of the extent of a file on this volume. As only one extent of each file is on a diskette, no continuation labels are needed.

All IBM-standard file labels for all files on a diskette volume are stored in the VTOC on track 0, sectors 8-26.

Only IBM-standard file labels are supported on diskettes.

LABEL PROCESSING FOR TAPE FILES

STANDARD LABELS, INPUT FILE

VOL1 Label

The standard volume label (VOL1) must be the first record on the reel when standard labels (FILABL=STD) are specified.

The VOL1 label can be written by the IBM-supplied utility program, Initialize Tape. It is generally written once, when the reel of tape is first received in an installation. At that time, a permanent volume serial number is assigned to the reel and written on it as part of the volume label. This provides a permanent identification of the reel, as long as it is used for files with standard labels. Following the VOL1 label, Initialize Tape writes a dummy HDR1 label and a tapemark. Either IBM or American National Standards Institute, Inc. labels may be specified in the Initialize Tape program.

Whenever the tape reel is positioned at the load point for processing the first or only volume of an input data file (or multifile), IOCS reads and checks the VOL1 label against the File Serial Number supplied by the user in TLBL. If an error is detected, a message is given to the operator. The operator may mount the correct volume, continue processing with the mounted volume (if your system does not have data protection), or terminate the job.

If TLBL is used and the File Serial Number is not specified, IOCS assumes that the correct volume is mounted and does not check the VOL1 label.

In a multivolume file, the VOL1 label of succeeding volumes after the first one processed is not checked (see "Input File, Multivolume File").

If any additional volume labels (VOL2—VOL8) follow a VOL1 label, IOCS bypasses them. Similarly, IOCS bypasses additional user volume labels (UVL1—UVL9) on an ASCII tape file.

HDR1 Label

IOCS identifies the appropriate file to be processed by reading the HDR1 label and comparing the File Serial Number, the Volume Sequence Number, and the File Sequence Number in the label, to those numbers supplied by TLBL. If the specified header label cannot be found, a message is issued to the operator. The operator must mount the correct volume, or terminate the job.

IOCS checks fields 3 and 7—10 (described in Figure 38 on page 162) against information supplied in TLBL. Fields 11—14 are ignored, unless READ=BACK has been specified.

If the TLBL minimum specification (File Name only) is given, IOCS assumes that the correct file is positioned for processing and does not check the HDR1 label.

In a multivolume file, the HDR1 label on each volume after the first one processed is checked against the TLBL information that has been updated by IOCS where necessary (see "Input File, Multivolume File").

If any additional HDR labels (HDR2—HDR8 for EBCDIC files or HDR2—HDR9 for ASCII files) follow an HDR1 label, IOCS bypasses them.

If any user-standard labels (UHL1—UHL8 for EBCDIC files or UHLa for ASCII files) follow the HDR label(s) and if DTFMT or DTFPH LABADDR=Name has been specified, IOCS branches to the user's label routine. If not, IOCS positions the tape at the first date record.

EOF1/EOV1 Label

IOCS reads an EOF1 or EOV1 trailer label after the tapemark that follows the last data record of a file or volume.

EOF1 indicates to IOCS that an end-of-file condition exists. EOV1 indicates to IOCS that an end-of-volume condition exists (see "Input File, Multivolume File").

For either label, IOCS checks the Block Count field only.

If any additional trailer labels (for EBCDIC files EOF2—EOF8 or EOV2—EOV8; for ASCII files EOF2—EOF9 or EOV2—EOV9) follow an EOF1 or EOV1 label, IOCS bypasses them.

If any user-standard trailer labels (UTL1—UTL8 for EBCDIC files or UTLa for ASCII files) follow the EOF or EOV label(s) and if DTFMT or DTFPH LABADDR=Name has been specified, IOCS branches to the user's label routine. If not, IOCS reads the tapemark that follows the last EOF label (see "Input File, Tapemarks").

If processing of an input file is terminated by a CLOSE or FEOV instruction before the end of the input data on the volume is reached, the EOF1 or EOV1 label is not read and checked. IOCS rewinds the tape as specified by DTF REWIND.

User-Standard Labels (UHL/UTL)

When user-standard labels (UHL/UTL) are to be checked and logical IOCS macros are used for the file, DTF LABADDR=Name must be specified. If it is not specified, IOCS bypasses all user-standard labels.

When physical IOCS macros are used for a file and DTFPH is specified, LABADDR=Name must be included if user-standard 'header labels (UHL) are to be checked. IOCS does not provide for user checking of user-standard trailer labels (UTL).

The input file (such as a card reader) that contains the user's information for checking user-standard labels must be opened ahead of the file with the UHL labels. This is done by specifying the label-information file ahead of the labeled file in the same OPEN instruction, or by issuing a separate OPEN instruction ahead.

IOCS identifies the user-standard labels by UHL or UTL in the first three bytes of the label.

IOCS reads each user-standard label, one at a time, into a label area used by IOCS for standard labels. IOCS supplies the address of this area in Register 1.

After a label is read in, IOCS branches to the user's label-checking routine. The same routine (specified by DTF LABADDR=Name) is used for checking both user-standard header (UHL) and user-standard trailer (UTL) labels. The user can identify the type of label by the UHL or UTL in the first three positions of the label itself.

After the user checks a label, he returns to IOCS by issuing a LBRET macro instruction. He controls the checking of any remaining user-standard labels by the operand in the LBRET instruction. A LBRET 2 instruction permits the checking of another label. A LBRET 1 instruction or a tapemark terminates label checking.

If the user, or a tapemark, does not terminate the label checking, IOCS reads in the next user-standard header label.

Multivolume File

When the volumes of a multivolume file are to be processed in sequence, starting with the first volume, no special instructions need be made by the user for the transition from one volume to the next. Logical IOCS recognizes the end-of-volume condition,and uses the existing CLOSE and OPEN routines to process, first the trailer labels, and then the header labels on the following volume.

When an EOV1 label is read or an FEOV macro is executed, IOCS checks trailer labels as described in the sections "Input File, EOF1/EOV1 Label" and "Input File, User-Standard Labels (UHL/UTL)." IOCS then prepares for checking the HDR1label on the next volume. IOCS increases by 1 the Volume Sequence Number in storage (read in from TLBL), and alsoupdates the active drive number if an ASSGN statement or command specified an alternative drive (ALT) for the file.

After all trailer labels have been checked, IOCS switchesto the alternate tape drive, if one has been specified by ASSGN. If an alternate tape drive has not been specified, a message is given to the operator and the system enters the wait state. The operator must mount the new volume and restart processing.

IOCS verifies that a VOL1 label is present on each volume, but does not check the Volume Serial Number on any volume after the first one processed.

The HDR1 label of each volume, after the first one processed, is checked against the TLBL information that has been updated by IOCS where necessary (for example: Volume Sequence Number).

IOCS provides for user checking of user-standard header labels on the new volume.

If physical IOCS macros are used for a file, an OPEN instruction must be issued for the new volume. This causes IOCS to check the HDR1 label and provide for user checking of user-standard labels, if any.

If the user wants to start the processing of a multivolume file with some volume other than the first, he should supply TLBL information as follows:

Field 4: File Serial Number should contain the volume serial number of the first volume of the set (not the volume being processed).

Field 5: Volume Sequence Number should contain the sequence number of the volume that will be processed first in this run.

Field 6: File Sequence Number should contain the sequence number of the file to be processed, if this is a multifile multivolume set.

All other fields should contain the same information as when starting with the first volume of the set.

This will properly check the HDR1 label. IOCS checking of the VOL1 label will detect the discrepancy in the volume serial numbers and issue a message to the operator. The operator can bypass this condition and continue processing.

If a multivolume file is reopened after a CLOSE, IOCS expects that the volume available to OPEN is the same volume, on the same drive, as that in process when CLOSE was executed. If it is not, a message is issued to the operator. Also, the first record read on the reopen must be a file label.

When physical IOCS macros are used and DTFPH is specified for standard label processing, FEOV may not be issued for an input file.

Multifile Volume

TLBL must be submitted for each file to be processed.

IOCS locates the first or only file that is to be opened by verifying the Volume Serial Number in the VOL1 label and then searching the tape for the HDR1 label that contains the File Sequence Number specified in TLBL.

If two or more files are to be opened, all files may be opened without rewinding the tape provided they are specified in ascending sequence. For any file after the first one opened, IOCS merely searches the tape for the file with the specified file sequence number. IOCS does not check the VOL1 label again.

If the files to be opened are not specified in ascending sequence, the tape must be rewound before each file is opened.

If the tape is positioned beyond a specified file when OPEN for that file is executed, a message is issued to the operator. The operator may remount or reposition the tape, or terminate the job.

If the TLBL minimum specification (File Name only) is given for the file, either on input or when the file was originally written as an output file, the user must position the tape to read the desired file. For this, he can use the Job Control MTC FSF statement or command, and skip three tapemarks for each file to be bypassed.

Read Backward

For a read backward file (specified by DTF READ=BACK), the trailer label (EOF1) is read and checked by OPEN, and the header label (HDR1) is read and checked by CLOSE.

The trailer label should contain both the header (except HDR) and trailer (Block Count) information. If the file labels were originally written by IOCS with FILABL=STD specified, the trailer label will be complete.

IOCS checks only the File Identifier field (field 3), in the trailer label, against information supplied by the user in TLBL. If File-ID is not specified, no checking is performed.

The tape should be positioned so that the first record read, when OPEN is executed, is the tapemark immediately following the trailer labels. If the tape is not positioned this way, a message is issued to the operator and processing can be continued. The message will occur if the user begins reading backward in the middle of a file.

Reading backward is confined to one volume, and an end-of-file condition exists when IOCS reads a tapemark.

IOCS provides for user-checking of user-standard trailer and header labels.

If physical IOCS macros are used by the problem program to read records backward, IOCS does not check labels. The DTFPH definition must be omitted and the user must provide his own checking, if any.

Tapemarks

The tapemark that follows the set of standard volume and header labels for a file indicates, to IOCS, that the last header label has been checked. The tape is positioned for user reading of the first data record. If files on other volumes are to be opened, IOCS opens the next file specified. The header labels for that file are checked (or written).

The tapemark that follows the data records indicates, to IOCS, that the end of the input for the file or the volume has been reached. IOCS determines the EOF/EOV condition from the trailer label that follows the tapemark.

The tapemark that follows all trailer labels for a file or volume indicates to IOCS, that the last trailer label (EOF, EOV, or UTL) has been checked. If an EOF label has been read, IOCS branches to the user's end-of-file routine (specified by DTF EOFADDR=Name). If an EOV label has been read, IOCS provides for the processing of the next volume (including label checking).

STANDARD LABELS, OUTPUT FILE

VOL1 Label

A standard volume label (VOL1) should have been previously written as the first record on the volume, whenever standard file labels (FILABL=STD) are to be written.

The VOL1 label can be written by the IBM-supplied utility program, _Initialize Tape_, when the reel of tape is first received in the installation. At that time, a permanent volume serial number is assigned and written on the reel as part of the volume label. This provides a permanent identification of the reel as long as it is used for files with standard labels. Following the VOL label, Initialize Tape writes a dummy HDR1 label and a tapemark. Either EBCDIC or ASCII files can be initialized by this utility program.

For a 9-track dual density output tape, a comparison is made between the user specified density and the VOL1 density of the mounted tape. If a discrepancy is found, and if the tape is at load point, the volume label(s) are rewritten according to the user-specified density.

The volume on which an output file is written should be determined ahead of time if the user plans to include the File Serial Number field. This permits the volume serial number, already recorded in the VOL1 label, to be specified in the File Serial Number field.

Whenever the tape reel is positioned at the load point for writing the first or only volume of an output data file, IOCS reads and checks the VOL1 label against information supplied by the user. If an error is detected, a message is given to the operator. The operator may mount the correct volume, continue processing with the mounted volume, or terminate the job.

If TLBL is used and the File Serial Number is not specified, IOCS assumes that the correct volume is mounted and does not check the VOL1 label.

If the output tape is positioned at the load point and IOCS reads a record that is not a VOL1 label, a message is given to the operator. He can cancel the job, mount a different tape reel, or key in a six-digit Volume Serial Number. In the latter case, IOCSwrites theVOL1 label at the beginning of the tape and processing continues.

Because IOCS expects to read a record to check for a VOL1 label, the tape used for output must contain some type of record (a label, data record, or tapemark). If it does not, the entire reel of tape is passed through the tape unit.

In a multivolume file, the VOL1 label of succeeding volumes after the first one written is not checked (see "Output File, Multivolume File").

If any additional volume labels (VOL2—VOL8) follow a VOL1 label, IOCS bypasses them. Similarly, IOCS bypasses any user volume labels (UVL1—UVL9) on ASCII tape files.

HDR1/HDR2 Labels

If an output file is to be written on a tape reel that already contains standard file labels, IOCS first reads the old HDR1 label. It checks the expiration date to ensure that the data on the tape is no longer active.

If the expiration date has passed, IOCS backspaces the tape and writes the new HDR1 label immediately after the VOL label(s) and over the old HDR1 label.

If the expiration date has not passed, a message is given to the operator. The operator can ignore the expiration date and continue processing, mount a new volume, or terminate the job.

If an output file is to be written on a tape that does not contain standard file labels, IOCS assumes that the expiration date has passed. IOCS writes the new HDR1 label immediately after the VOL label(s).

If an output file is to be written on a multifile volume(s) with standard labels, only the expiration date of the first file to be overwritten is checked. IOCS assumes that all succeeding files have the same expiration date.

The HDR1 label is written from the information supplied by the user in TLBL, or generated by IOCS (see "Section: Label Fields for Tape").

If TLBL specifications are omitted, IOCS writes predetermined <u>default</u> values .

In a multivolume file, the HDR1 label on each volume after the first one processed is written with the TLBL information that has been updated by IOCS where necessary (see "Output File, Multivolume File").

In a multifile volume, the HDR1 label for each file after the first is written with information obtained partly from the preceding standard trailer label and partly from TLBL (see "Output File, Multifile Volume").

For EBCDIC IOCS does not write additional header labels (HDR2—HDR8) If the user wants to write any of these labels he can specify a label routine (DTF LABADDR=Name) and use physical IOCS macros (EXCP and WAIT).

For ASCII IOCS writes an additional header label (HDR2) which contains the record format, block length, record length and buffer offset.

If DTFMT or DTFPH LABADDR=Name is specified to indicate that user-standard header labels (UHL1—UHL8 for EBCDIC; UHLa for ASCII) are to be written after the HDRlabels(s), IOCS branches to the user's label routine. If not, IOCS writes a tapemark and positions the tape for writing the first data record.

EOF1/EOV1 and EOF2/EOV2 Labels

When IOCS CLOSE is executed, after all records for a file have been processed, it writes the last block of data records (if any), a tapemark, and an EOF1 trailer label.

If IOCS detects the reflective marker at the end of the tape before the end of the output file is reached (see "Output File, Multivolume File"), or if an FEOV macro is executed, IOCS writes a tapemark and an EOV1 trailer label.

The EOF1 or EOV1 trailer label is written with HDR1 information in all fields except Block Count. Block Count is written with count accumulated during processing of the data file.

For EBCDIC IOCS does not write additional EOF or EOV labels (EOF2—EOF8 or EOV2—EOV8) If the user wants to write any of these labels he can specify a label routine (DTF LABADDR=Name) and use physical IOCS macros (EXCP and WAIT).

For ASCII IOCS writes an additional trailer label (EOV2/EOF2) which contains the same information as the HDR2 label.

If DTFMT or DTFPH LABADDR=Name is specified to indicate that user-standard trailer labels (UTL1—UTL8 for EBCDIC; UTLa for ASCII) are to be written after the EOF/EOV trailer label(s), IOCS branches to the user's label routine. If not, IOCS writes one or two tapemarks as determined by an end-of-volume or end-of-file condition (see "Output File, Tapemarks"). Logical IOCS then rewinds the tape as specified by DTF REWIND.

User-Standard Labels (UHL/UTL)

When user-standard labels are to be written for an EBCDIC or ASCII file, DTFMT or DTFPH LABADDR=Name must be specified.

Whenever LABADDR=Name is specified, at least one UHL label and one UTL label must be written.

The input file (such as a card reader) that contains the user's information for writing user-standard labels must be opened ahead of the file on which the UHL labels are to be written. To do this, the input file must be specified ahead of the file to be labeled in the same OPEN instruction, or a separate OPEN instruction must be issued ahead.

The user must build each user-standard label. To provide for this, IOCS branches to the user's label routine. The same routine (specified by LABADDR=Name) is used for building both user-standard header labels (UHL) and user-standard trailer labels (UTL). IOCS supplies a code in the low-order byte of Register 0 to indicate which type of label should be built:

UHL —	Code O (letter O)
UTL —	Code F for end-of-file condition
	Code V for end-of-volume condition

The user must establish an 80-byte area within his problem program area of main storage for building his labels. He must load the address of the area he uses into Register 0 before returning control to IOCS.

When building the label for an EBCDIC file, the user must include UHL or UTL in the first 3 bytes and a digit 1—8 in the fourth byte. He may include whatever information he needs in the remaining 76 bytes.

Note: When user header and trailer labels are created for 7-track tapes, only unpacked data is valid in the 76-byte data portion of the label.

To comply with the standards for ASCII files, a user standard header and trailer label must contain UHL and UTL, respectively, in the first three bytes. Also, the fourth byte must be an ASCII character in the range 2/0 through 5/14, excluding 2/7 (quote). The remaining 76 bytes may be used as desired.

After building a label, the user returns to IOCS by issuing a LBRET instruction. IOCS moves the label to the standard label I/O area, if necessary, and then writes the label on the tape.

The user controls the building and writing of succeeding user-standard labels by the operand in the LBRET instruction. If another label is to be written, operand 2 is specified and IOCS again branches to the user's label routine. When the user has built his last user label, he issues the LBRET macro with the operand 1. IOCS writes the last label.

For EBCDIC files a maximum of 8 user-standard header and 8 user-standard trailer labels may be written. After 8 labels, IOCS terminates the label writing, regardless of the LBRET macro instruction. For ASCII files, theoretically there is no limit to the number of user standard labels (UHLa and UTLa). There is a physical limit since the physical tape may be reached before all user labels are written.

After the last user-standard header label (UHL), IOCS writes one tapemark. After the last user-standard trailer label (UTL), IOCS writes one or two tapemarks, as determined by an end-of-volume or end-of-file condition (see "Output File, Tapemarks").

Multivolume File

When a multivolume file is to be written, no special instructions need be made by the user for the transition from one volume to the next. Logical IOCS recognizes an end-of-volume condition and uses the existing CLOSE and OPEN routines to write, first the trailer label(s), and then the header label(s) on the following volume.

After IOCS detects the reflective marker at the end of tape, it determines the EOF/EOV condition by the next I/O instruction for this file in the problem program. If the instruction is CLOSE, an end-of-file (EOF) condition exists and IOCS writes a tapemark and an EOF1 label, followed by an EOF2 label (for ASCII only). If, however, the next instruction is a PUT, an end-of-volume (EOV) condition exists and IOCS writes a tapemark and an EOV1 label, followed by an EOV2 label (for ASCII) only. For an ASCII file, two tapemarks are written following the EOV2 label.

When an EOV condition exists or an FEOV (forced end-of-volume) macro is executed, IOCS permits the writing of user-standard trailer labels, if any, and then prepares for writing the HDR1 label on the next volume. IOCS increases by 1 the Volume Sequence Number in storage (read in from TLBL), and updates the active drive number if an ASSGN statement or command specified an alternate drive (ALT) for the file.

After all trailer labels have been written, IOCS writes one tapemark and switches to the alternate tape drive, if one has been specified by ASSGN. If an alternate tape drive has not been specified, a message is given to the operator and the system enters the wait state. The operator must mount the new volume and restart processing.

IOCS verifies that a VOL1 label is present on each volume, but does not check the Volume Serial Number on any volume after the first.

The HDR1 label of each volume after the first is written with the TLBL information that has been updated by IOCS where necessary (for example: Volume Sequence Number).

On <u>each</u> volume, the File Serial Number field of the HDR1 label is written with the Volume Serial Number of the <u>first</u> volume of the set. Thus on each volume after the first, the File Serial Number in the HDR1 label differs from the Volume Serial Number in the VOL1 label.

IOCS provides for user writing of user-standard header labels on the new volume.

If physical IOCS macros are used for the file, an OPEN instruction must be issued for a new volume. This causes IOCS to write the standard header label and provide for user writing of user-standard labels, if any.

Multifile Volume

TLBL must be submitted for each file to be written.

When two or more files are to be written in the same operation, the DTF entry REWIND=NORWD should be specified for each file. With this specification, the tape is located at the correct position for the OPEN routines to write the standard file header label for each additional file (after the first) on the reel.

To properly position the tape at the load point for the first file, the programmer can include a CNTRL REW macro instruction ahead of the OPEN instruction, or the operator can position the tape at the load point.

When the tape is at the load point for the first file, IOCS OPEN ensures that the correct volume has been mounted by checking the Volume Serial Number in the VOL1 label against the information supplied by TLBL. If the File Serial Number is not specified, IOCS assumes that the correct volume is mounted.

IOCS OPEN then checks the expiration date in the old HDR1 label (if any).

IOCS writes the HDR1 label for the first output file from the specifications supplied by the user in the TLBL or supplied by IOCS as default values.

For the HDR1 label of each file after the first, the OPEN routines obtain the file serial number (field 4), the volume sequence number (field 5), and the file sequence number (field 6) from the preceding EOF1 label. OPEN increases the file sequence number by 1 for the new file. The remaining fields of the header label are written with the information supplied by the user in TLBL or as default values.

If the tape is rewound or repositioned after a file is closed, it is the user's responsibility to properly position the tape for writing any additional file(s) at a later time. The tape must be positioned so that the file header label is written immediately after one tapemark following the last file currently on the tape. Thus, it must replace the second of the two tapemarks that normally follow the last file on the tape.

The tape can be advanced from the load point to the correct position by skipping three tapemarks for each file presently on the tape. A Job Control MTC FSF command or statement is used for this skipping and the DTF entry REWIND=NORWD must be included for the file. In this case, the VOL1 label is not checked, and the expiration date of the file to be overwritten (if any) is checked.

When the tape has been positioned and the file is opened, the OPEN routines obtain the information for writing the HDR1 label from the preceding standard trailer label and from TLBL. This is the same as described previously for multiple files that are written without rewinding the tape.

User-standard header and trailer labels may follow the standard header and trailer label for each file of a multifile volume.

Tapemarks

After all the header labels for a file are written, IOCS writes one tapemark. The tapemark follows the HDR label(s) if DTFMT or DTFPH LABADDR=Name is not specified for a file. If LABADDR is specified, IOCS writes the tapemark when the problem program issues a LBRET 1 instruction, which indicates that all the desired user-standard labels have been written, or when the maximum of eight UHL labels has been written. he tape is positioned for writing the first data record. If files on other volumes are to be opened, IOCS opens the next file specified and writes (or checks) the header label(s) for that file.

When the problem program issues either a CLOSE or FEOV macro instruction, or when IOCS detects the reflective marker at the end of the tape, IOCS writes a tapemark following the last block of data records.

IOCS writes two tapemarks after an EOF1 label, or after a set of EOF and UTL labels if LABADDR=Name is specified.

IOCS writes one tapemark after an EOV1 label, or after a set of EOV and UTL labels if LABADDR=Name is specified. For ASCII files IOCS writes two tapemarks.

In a multifile volume, one tapemark follows the end-of-file label(s) of each file except the last. Two tapemarks follow the end-of-file label(s) of the last file. If a file is added later to a multifile volume, the second tapemark is replaced by the HDR1 label of the additional file.

NONSTANDARD LABELS

The following discussion is concerned with nonstandard labels (DTFMT FILABL=NSTD). When nonstandard labels are specified, IOCS OPEN/CLOSE routines provide for user processing of nonstandard header labels at the beginning of the file, and nonstandard trailer labels at the end of the file. The functions performed by IOCS vary depending on input file vs output file and header label vs trailer label.

A user routine is required to supply the information for checking/creating nonstandard labels. This routine, the functions performed by IOCS OPEN/CLOSE routines, and the specific processing performed for each type of label (within each type of file) are summarized in this section.

Note: Nonstandard labels cannot be used on ASCII tape files.

IOCS Routines

IOCS OPEN and CLOSE routines provide for user processing of nonstandard labels. IOCS branches to the user's label routine if DTF LABADDR=Name is specified.

The OPEN/CLOSE routines are transient routines of the Supervisor. As such, they are stored on the system pack (SYSRES) and called into the transient area of main storage whenever an OPEN or CLOSE macro instruction is executed.

User Routine

The information for creating/checking nonstandard header and trailer labels is generally supplied by the user in a separate input file, such as a card reader. The information is stored by the user's problem program in a location that meets the requirements of his job.

Nonstandard labels must be read and checked, or built and written, by a routine supplied by the user. The symbolic address of the user's label routine must be specified in the DTF entry LABADDR=Name.

In his label routine, the user must issue physical IOCS macros instructions (EXCP and WAIT) to read or write the labels. He must set up a Command Control Block, by issuing a CCB macro instruction, and establish a CCW (Channel Command Word).

The user must define his own label read-in or read-out area.

In his label routine, the user performs whatever label reading and checking or building and writing he requires for his job.

At the end of his routine, the user returns to IOCS by issuing a LBRET 2 instruction.

NONSTANDARD LABELS, INPUT FILE

Header Label

If the input file was previously written using VSE (with FILABL=NSTD specified), the first record on the reel is the user's first nonstandard label. There is no volume label at the beginning of the reel.

Nonstandard header labels may, or may not, be followed by a tapemark. This choice, combined with the user's requirement to check the labels, or not, results in four possible conditions that can be encountered when an input file is opened:

- Label(s) followed by a tapemark are to be checked.
- Label(s) <u>not</u> followed by a tapemark are to be checked.
- Label(s) followed by a tapemark are <u>not</u> to be checked.
- Label(s) <u>not</u> followed by a tapemark are <u>not</u> to be checked.

For the <u>first two conditions,</u> DTF FILABL=NSTD and LABADDR=Name must be specified in the file definition. IOCS branches to the user's label routine when OPEN is executed.

For the third condition, DTF FILABL=NSTD must be specified. DTF LABADDR is omitted and IOCS skips all labels, passes the tapemark, and positions the tape at the first data record to be read.

For the fourth condition, DTF FILABL=NSTD and LABADDR=Name must be specified. IOCS branches to the user's label routine when OPEN is executed, and the user must read all labels even though checking is not desired. This positions the tape at the first data record. This is necessary because IOCS cannot distinguish labels from data records and because there is no tapemark to indicate the end of the labels. If this were not done, IOCS would search the tape for a tapemark, and thus pass the whole file until it reached the tapemark that follows the last data record.

When DTF LABADDR=Name is specified for checking labels, IOCS branches to the user's label routine only once. The problem program must perform all required reading and checking of header labels before returning to IOCS. The user can determine that the last label has been read by checking some identifying information he has provided in the last label, or by the tapemark, if any, that folllows the label(s).

After all header labels have been processed and the user has returned control to IOCS OPEN (by use of the LBRET 2 macro instruction), IOCS reads and checks the next record. If it is a tapemark, IOCS assumes that the following record is the first data record. If it is not a tapemark, IOCS backspaces the tape one record and assumes that this record is the first data record. Thus, the user should read all labels, before returning to IOCS so that the tape is properly positioned at the first data record.

If a file is reopened after a CLOSE, it is the user's responsibility to identify the first record read as a file label or a data record.

End-of-File/End-of-Volume Label

When DTF LABADDR=Name is specified for checking labels, IOCS branches to the user's label routine when it reads the tapemark that follows the last data record.

IOCS branches to the user's routine only once. The problem program must read and check all trailer labels before returning to IOCS.

From his trailer label, the user must determine if an end-of-file or an end-of-volume condition exists and indicate this to IOCS. For this he must load either EF (end-of-file) or EV (end-of-volume) in the two low-order bytes of Register 0.

After all trailer labels have been processed, the user returns control to IOCS by issuing a LBRET 2 macro instruction.

If an EF condition was indicated, IOCS branches to the user's end-of-file address (specified by DTF EOFADDR) when the problem program returns to IOCS at the end of the label routine. The user can perform whatever processing is required for the end of his data records, and he generally closes the logical file.

If processing of an input file is terminated by a CLOSE, or an FEOV (forced end-of-volume) instruction, before the tapemark at end of the input data is reached, IOCS does not branch to the user's label checking routine.

Multivolume File

When the problem program reads an end-of-volume label and specifies an EV condition to IOCS, or issues a forced end-of-volume instruction (FEOV), IOCS prepares for processing records from the next volume. IOCS updates the active drive number if an ASSGN statement or command specifies an alternate drive (ALT) for the file.

IOCS switches to the alternate drive, if one has been specified. If not, a message is given to the operator, and the system enters the wait state. The operator must mount the new volume and restart processing.

IOCS provides for user checking of header labels on the next volume, if LABADDR=Name is specified for the file.

When an alternate drive is assigned to a file, the number of the drive currently in use must be supplied to the Command Control Block (CCB) used by the EXCP macro for label reading. IOCS provides the hexadecimal value of this drive in the two low-order bytes of Register 1. The user's label routine should move this value to bytes 6 and 7 of the CCB.

If physical IOCS macros are used for a file, an OPEN instruction must be issued for a new volume. This causes IOCS to provide for user checking of nonstandard labels.

If a multivolume file is reopened after a CLOSE, IOCS expects that the volume available to OPEN is the same volume, on the same drive, as that in process when CLOSE was executed. If it is not, a message is issued to the operator.

Multifile Volume

If multiple files on the same volume are to be read in sequence, the DTF entry REWIND=NORWD should be specified for each file. With this specification, the tape is located at the correct position for the user to read his first header label or data record when each file (after the first on the reel) is opened.

To properly position the tape for the first file on the reel, the programmer can include a CNTRL REW macro instruction ahead of the OPEN instruction, or the operator can position the tape at the load point.

When the first file to be opened is not the first file on the reel, the tape can be advanced from the load point to the correct position by use of the Job Control MTC FSF statement or command. Either two or three tapemarks are skipped for each file to be passed. If TPMARK=NO was specified when the files were written, two tapemarks are skipped. If not, three tapemarks are skipped. The DTF entry REWIND=NORWD must be included for the file to be opened.

When any file is opened, IOCS branches to the user's label routine, if specified. The user can read and check header labels.

Read Backward

The tape should be positioned so that the first record read, when OPEN is executed, is the tapemark immediately following the trailer labels.

When the file is opened, IOCS provides for user checking of the <u>trailer</u> label(s) in the same manner that header labels are checked on a forward read (see "Input File, Header Label").

IOCS assumes that the <u>end</u> of the input file has been reached when it reads a tapemark at the beginning of the tape (between the header label and the first data record).

IOCS CLOSE branches to the user's label routine (specified by DTF LABADDR=Name) where he can read and check the <u>'header</u> label(s).

If the tape does not contain a tapemark between the header label(s) and the first data record (TPMARK=NO was specified when the tape was written), the user must determine whether a record is a file label or a data record.

Tapemarks

IOCS does not expect a tapemark at the beginning of the volume. When OPEN is executed, IOCS branches immediately to the user's label routine (if LABADDR=Name is specified) so that the problem program can read and check the first record.

A tapemark may, or may not, follow the nonstandard labels depending on whether TPMARK=NO was specified when the file was written.

If nonstandard labels are <u>not</u> to be checked, IOCS can properly position the tape for reading the first data record, only if a tapemark exists between the labels and the data records. If a tapemark is not present, the user must read the labels in order to advance the tape to the proper position for reading the first data record. Thus, in this case, DTF LABADDR=Name must be specified even though labels are not to be checked.

The tapemark that follows all data records indicates to IOCS that the end of input from the file or volume has been reached.

NONSTANDARD LABELS, OUTPUT FILE

Header Label

Nonstandard header labels are written starting at the location where the tape is positioned. Thus if the tape has been rewound to the load point, the first nonstandard label is written over any label(s) that is already on the tape, such as a volume label.

IOCS does not check for the presence of a volume label or the expiration of a previously written standard or nonstandard file label.

Whenever DTF LABADDR=Name is specified, at least one header label must be written.

The input file (such as a card reader) that contains the user's information for writing nonstandard labels must be opened ahead of the file on which the header labels are to be written. To do this the input file must be specified ahead of the file to be labeled in the same OPEN instruction, or a separate OPEN instruction must be issued ahead.

The same routine (specified by DTF LABADDR=Name) is used for building and writing both nonstandard header and nonstandard trailer labels. When IOCS branches to this routine at OPEN time, it supplies the letter O in the low-order byte of Register 0 to indicate that a header label should be written.

IOCS branches to the user's label routine only once for header labels. Therefore, the problem program must build and write all the required header labels before returning to IOCS.

After all header labels have been written, the user returns control to IOCS OPEN by use of the LBRET 2 macro instruction.

IOCS writes a tapemark after the last header label, unless the user has specified DTF TPMARK=NO.

End-of-File/End-of-Volume Label

When IOCS CLOSE is executed after all records for a file have been processed, it writes the last block of data records (if any) and a tapemark, and then branches to the user's label routine.

If IOCS detects the reflective marker, at the end of the tape, before the end of the output file is reached (see "Output File, Multivolume File"), or if an FEOV macro is executed, IOCS writes a tapemark and branches to the user's label routine.

IOCS indicates (to the user) which type of trailer label should be written, by supplying a code in the low order byte of Register 0:

Code F — end-of-file label
Code V — end-of-volume label

The user should code his trailer label to indicate whether it is an end-of-file label or an end-of-volume label. This will be required by the user's label routine when the file is used later as input.

IOCS branches to the user's label routine only once on an end-of-file or end-of-volume condition. The problem program must build and write all the required trailer labels before returning to IOCS.

After all the trailer labels are written, the user returns control to IOCS by use of the LBRET 2 macro instruction.

IOCS writes one or two tapemarks as determined by an end-of-volume or end-of-file condition (see "Output File, Tapemarks").

Multivolume File

If IOCS detects the reflective marker at the end of the tape, it determines the end-of-file or end-of-volume condition by the next I/O instruction for this file in the problem program. If the instruction is a CLOSE, an end-of-file condition exists; however, if the next instruction is a PUT, an end-of-volume condition exists.

If the problem progam issues an FEOV (forced end-of-volume) macro instruction, an end-of-volume condition exists.

On any end-of-volume condition, IOCS writes a tapemark and branches to the user's label routine (if LABADDR=Name is specified), so that nonstandard trailer label(s) can be written.

After all trailer labels are written and the user has returned to IOCS by a LBRET 2 instruction, IOCS writes one tapemark and prepares for the next volume. IOCS updates the active drive number if an ASSGN statement or command specifies an alternate drive (ALT) for the file.

IOCS switches to the alternate drive, if one has been specified. If not, a message is given to the operator and the system enters the wait state. The operator must mount the new volume and restart processsing.

IOCS positions the new volume at the load point and branches again to the user's label routine, so that he can write the header label(s) on the new volume.

When an alternate drive is assigned to a file, the number of the drive currently in use must be supplied to the Command Control Block (CCB) used by the EXCP macro for label reading. IOCS provides the hexadecimal value of this drive in the two low-order bytes of Register 1. The user's label routine should move this value to bytes 6 and 7 of the CCB.

If physical IOCS macros are used for a file, an OPEN instruction must be issued for a new volume. This causes IOCS to provide for user writing of header labels.

Multifile Volume

Multiple files can be written on the same volume in the same operation without repositioning the tape, by specifying DTF REWIND=NORWD for each file. With this specification the tape is properly located for the user to write his nonstandard label for each additional file (after the first) on the reel.

To properly position the tape at the load point for the first file on the reel, the programmer can include a CNTRL REW macro instruction ahead of the OPEN instruction, or the operator can position the tape at the load point.

When any file is opened, IOCS branches to the user's label routine where he can write his nonstandard header label(s).

If the tape is rewound or repositioned after a file is closed, the user must properly position the tape to write any additional file(s) at a later time. The tape can be advanced from the load point to the correct position by skipping either two or three tapemarks for each file presently on the reel. If TPMARK=NO was specified for those files already written, two tapemarks are skipped. If not, three tapemarks are skipped. A Job Control MTC FSF statement or command is used for the skipping, and the DTF entry REWIND=NORWD must be included for the file.

Tapemarks

On an OPEN condition, when the tape is at the load point, IOCS immediately provides for user-writing of nonstandard header labels. IOCS does not write a tapemark ahead of the first header label.

IOCS writes a tapemark after the last nonstandard header label, unless DTF TPMARK=NO is specified.

On a CLOSE or end-of-volume condition, IOCS writes one tapemark after the last data record of the file or volume.

After the last trailer label is written for a file, IOCS writes two tapemarks.

After the last trailer label is written for an end-of-volume condition, IOCS writes one tapemark.

In a multifile volume, one tapemark follows the end-of-file label(s) of each file except the last. Two tapemarks follow the end-of-file label(s) of the last file.

PROCESSING OF UNLABELED TAPE FILES

The following discussion is concerned with unlabeled files (DTFMT FILABL=NO). Whenever the DTF entry FILABL=NO is specified, or the FILABL entry is omitted, IOCS assumes that a file does not contain labels, regardless of what is actually written on the tape. The functions performed by IOCS ahead of, and after, a file of data records consists merely of writing tapemarks and positioning the tape reel for reading or writing records. These functions are summarized in this section by type of file.

UNLABELED FILES, INPUT FILE

First Record

If the input file was previously written using VSE (with DTF FILABL=NO specified, or FILABL omitted), the first record for the file is either a tapemark or data record. Tapemarks are not written at the beginning of an unlabeled ASCII tape.

If a tapemark is present, IOCS assumes that the next record is the first data record of the logical file.

If IOCS does not detect a tapemark when it reads the first record from the tape, it backspaces the tape and assumes that the first record is a data record.

If the input file was previously written with labels, IOCS treats the label as a data record.

An unlabeled file may be opened anywhere in the midst of the file. Regardless of whether the file is opened at the first data record or somewhere in the middle of the file, no message is given to the operator (as it is with standard labels).

Last Record

IOCS assumes that the end of the input file has been reached when it reads the tapemark that follows the last data record. IOCS immediately branches to the user's end-of-file routine, specified by DTF EOFADDR=Name.

In his end-of-file routine, the user must determine if an end-of-file condition actually exists or if this is an end-of-volume condition.

On an end-of-file, the user performs whatever processing is required for the end of his data records, and he generally closes the logical file.

Multivolume File

If the user determines that an end-of-volume condition exists (instead of an end-of-file), he must indicate this to IOCS by issuing an FEOV macro instruction in his end-of-file routine.

Whenever an FEOV macro is executed, IOCS prepares for the next volume by updating the active drive number if an ASSGN statement or command specifies an alternate drive (ALT) for the file.

IOCS switches to the alternate tape drive if one has been specified. If not, a message is given to the operator and the system enters the wait state. The operator must mount the new volume and restart processing. IOCS then positions the new volume at the first data record.

If a multivolume file is reopened after a CLOSE, IOCS expects that the volume is on the same drive as that in use when CLOSE was executed. If it is not, a message is issued to the operator.

Multifile Volume

If multiple files on the same volume are to be read in sequence, the DTF entry REWIND=NORWD should be specified for each file. With this specification the tape is located at the correct position for the user to read his first record when each file (after the first on the reel) is opened.

To properly position the tape for the first file on the reel, the programmer can include a CNTRL REW macro instruction ahead of the OPEN instruction, or the operator can position the tape at the load point.

When the first file to be opened is not the first file on the reel, the tape can be advanced from the load point to the correct position by use of the Job Control MTC FSF command or statement. One tapemark is skipped for each file to be passed. If the reel contains a tapemark before the first file (TPMARK=NO was not specified when the file was written), that tapemark must also be skipped. The DTF entry REWIND=NORWD must be included for the file to be opened.

Read Backward

An unlabeled tape file can be read backward if it has not been written in the data conversion mode (7-track).

Because of special error-recovery procedures, unlabeled ASCII tapes (without any leading tapemark) may be read backward.

Tapemarks

IOCS expects the first record for a file to be either a tapemark or a data record. In either case, IOCS positions the tape so that the user can read the first data record. IOCS treats a label (if present) as a data record.

The tapemark that follows all data records indicates to IOCS that the end of input from a file or volume has been reached.

UNLABELED FILES, OUTPUT FILE

First Record

IOCS writes a tapemark as the first record, unless the user specified DTF TPMARK=NO.

The tapemark, or the first data record, is written starting at the location where the tape is positioned. Thus if the tape has been rewound to the load point, the tapemark or data is written over any label(s) that is already on the tape, such as a volume label.

If the tape is at load point, IOCS checks for the presence of a volume label.

Last Record

When IOCS CLOSE is executed after all records for a file have been processed, IOCS writes the last block of data records (if any) and two tapemarks.

If IOCS detects the reflective marker at the end of the tape before the end of the output file is reached (see "Output File, Multivolume File"), or if an FEOV macro is executed, IOCS writes one tapemark.

Multivolume File

If IOCS detects the reflective marker before a CLOSE is executed, it determines the end-of-file or end-of-volume condition by the next I/O instruction for this file in the problem program. If the instruction is a CLOSE, an end-of-file condition exists. If, however, the next instruction is a PUT, an end-of-volume condition exists.

If the problem program issues an FEOV (forced end-of-volume) macro instruction, an end-of-volume condition exists.

On any end-of-volume condition, IOCS writes one tapemark and prepares for the next volume. IOCS updates the active drive number if an ASSGN statement or command specifies an alternate drive (ALT) for the file.

IOCS switches to the alternate drive, if one has been specified. If not, a message is given to the operator and the system enters the wait state. The operator must mount the new volume and restart processing.

IOCS positions the new tape at the load point and writes a tapemark, unless DTF TPMARK=NO has been specified.

Multifile Volume

Multiple files can be written on the same volume in the same operation without repositioning the tape, by specifying DTF REWIND=NORWD for each file. With this specification, the tape is properly located for the user to write the first record for each file (after the first) on the reel.

To properly position the tape at the load point for the first file on the reel, the programmer can include a CNTRL REW macro instruction ahead of the OPEN instruction, or the operator can position the tape at the load point.

If the tape is rewound or repositioned after a file is closed, the user must properly position the tape to write any additional file(s) at a later time. The tape can be advanced from the load point to the correct position by skipping one tapemark for each file presently on the reel. If the reel contains a tapemark before the first file (TPMARK=NO was not specified for those files already written), that tapemark must also be skipped. A Job Control MTC FSF statement or command is used for this skipping, and the DTF entry REWIND=NORWD must be included for the file.

Tapemarks

IOCS writes a tapemark ahead of the first data record, unless DTF specifies TPMARK=NO.

IOCS writes two tapemarks after the last data record whenever the CLOSE macro is executed; if REWIND=NORWD is specified, the tape is then positioned between those two tapemarks.

IOCS writes one tapemark after the last data record for a volume when an FEOV (forced end-of-volume) macro is executed.

AMERICAN NATIONAL STANDARD LABELS

VSE processes tape files written in the American National Standard Code for Information Interchange (ASCII), in addition to processing tape files written in EBCDIC. ASCII is based on the specifications of the American National Standards Institute, Inc. and standard labels for ASCII files are referred to as American National Standard standard labels. ASCII files may be unlabeled or labeled with American National Standard standard or user-standard labels. Nonstandard labels are not permitted on ASCII files.

This section briefly summarizes the differences in specifications and processing of ASCII and EBCDIC standard labeled files. The American National Standard standard volume label and standard file 1 label are shown in Figure 36 on page 160 and Figure 40 on page 167, respectively. The fields are described in Figure 37 on page 160 and Figure 41 on page 167 respectively.

The differences between the American National Standard standard volume label and the IBM standard volume label fields are as follows:

Field No.	EBCDIC Name	ASCII Name	Bytes EBCDIC	Bytes ASCII
4	Volume Security	Accessibility	11	11
5	Data File Directory	(Reserved)	12-21	12-31
6	(Reserved)	(Reserved)	22-31	32-37
7	(Reserved)	Owner ID	32-41	38-51
8	Owner ID	(Reserved)	42-51	52-79
9	(Reserved)	Label Standard Level	52-80	80

Some fields in the American National Standard standard file 1 label have names different from the corresponding fields in the IBM standard file label. These differences are as follows:

Field No.	EBCDIC Name	ASCII Name	Bytes
4	File Serial No.	Set Identifier	
5	Volume Sequence No.	File Section Number	
11	File Security	Accessibility	

The optional standard volume labels VOL2——VOL8 are supported for EBCDIC files only. ASCII has the optional user volume labels (UVL1——UVL9) instead. VSE ignores these labels on input and does not create them on output.

EBCDIC files may have up to seven additional HDR, EOF, and EOV labels (HDR2-HDR8, EOF2-EOF8, EOV2-EOV8), whereas ASCII may have up to eight of each of these labels (HDR2-HDR9, EOF2-EOF9, EOV2-EOV9). By VSE, these additional labels are bypassed on input and not created on output, except the HDR2, EOF2, EOV2 labels which were created for ASCII output files. The user-standard header and trailer labels for each mode are:

EBCDIC --	UHL1-UHL8 and UTL1-UTL8.
ASCII --	UHLa and UTLa, where 'a' represents an ASCII character in the range 2/0 through 5/14, excluding 2/7 (quote).

The default for the version number in the American National Standard standard file label is 00; the IBM standard file label version number defaults to 01.

EOV labels on an EBCDIC tape file are followed by one tapemark; on an ASCII tape file these labels are followed by two tapemarks.

When an ASCII file is processed, IOCS translates the labels from ASCII into EBCDIC (on input) and from EBCDIC into ASCII (on output). Two translate tables are provided in the SVA for this purpose. The address of the ASCII-to-EBCDIC table is in the extension of each communication region in bytes 44-47. The address of the EBCDIC-to-ASCII table is 256 bytes higher than the address of the first table. The address of the communication region extension is found in bytes 136 - 139 of the communication region.

Tapes to be used for ASCII files may be initialized with American National Standard labels by the IBM-supplied program, Initialize Tape.

LABEL FIELDS FOR TAPE

Each label is illustrated, and each field of each label is described in detail. The individual fields in the illustrations are numbered D1-D to relate to the corresponding descriptions.

The descriptions of the label fields include the:

- Displacement in hex notation.
- Field Number - Kn or Dn
- Length of the field in bytes (hex notation).
- Content of each field, together with the name of the field.

An additional table shows for each field:

- Source of Information for checking or writing this field.
- Purpose of the field.
- Processing performed on input/output.
- TLBL Default The TLBL statement has only one required field, the 'name field'. All other fields are optional, and need be entered only if desired by the user. If any one of these fields is left blank for OUTPUT files, IOCS writes a certain default value in the corresponding output label field. If any one of these optional fields is left blank for INPUT files, no default value is assumed and no checking of the corresponding label field is performed.

Volume Labels on Tape

Figure 34 and Figure 36 on page 160 show volume labels for EBCDIC and ASCII tapes.

Displ.	Field	Length	Content
0	D1	3	Label ID: VOL
3	D2	1	Ignored by VSE
4	D3	6	Volume serial number
A	D4	1	Ignored by VSE
B	D5-D7	1E	Reserved
29	D8	A	Volume owner name or code
33	D9	1D	Reserved

Figure 34. Tape Volume Label for EBCDIC Code

Field	
D1	Source of Information : System Purpose: Identifies the standard volume label. Processing: On both input and output, IOCS checks this field to verify that a standard volume label is present on the tape when DTF FILABL = STD, or DTFPH, is specified for the first or only file on a tape reel (at the load point). The volume label should be written previously, before a logical file of data records is written on the tape.
D2	Source of Information : System Purpose: Indicates the sequence of this label within the volume label (VOL) group. VSE supports Volume Label 1 only, but provision is made for additional standard volume labels if required in other systems. Processing: This field is processed in conjunction with the label identifier (field 1) to completely identify the volume label.
D3	Source of Information : TLBL/System Purpose: Provides a unique identification for a reel (volume). The number is generally assigned when the reel is first received in the installation, and retained as long as the reel is used for files with standard labels. This number should also be used as the File Serial Number in the file HDR1 label of each logical file written on the volume. This provides a unique identification of the volume/file relationship. If a multivolume logical file is written, the Volume Serial Number of the first volume becomes the Volume Serial Number in the file HDR1 label on all volumes. Processing: On both input and output, IOCS checks this field against the number supplied by the user in the File Serial Number field of TLBL, for a singlevolume file. If TLBL is used and no operand is specified, IOCS assumes the correct volume is mounted and does not check this field. For a multivolume file, IOCS checks the Volume Serial Number of the first volume only. On succeeding volumes, the Volume Serial Number and the File Serial Number differ (as described in Purpose). On output if the tape does not contain a volume label, the operator may key in a 6-digit Volume Serial Number and IOCS then writes the volume label. In the TLBL control card, the Volume Serial Number may be between quotes, or without quotes. If between quotes, the Volume Serial Number is assumed alphabetic and the field in the label is assumed to contain trailing blanks. If without quotes, the Volume Serial Number is assumed numeric and the field in the label is assumed to contain leading zeros.
D4	Source of Information : - Purpose: Provides a code to indicate that additional identification is required before a volume can be considered the correct one for processing. VSE does not use this field, but provision is made for additional security in other systems. For example, OS/VS allows operator response of a predetermined 'password' to further authorize a volume for processing. Processing: On both input and output, IOCS ignores this field. Note: OLTEP and OLTSEP will access this byte to determine if the volume is security protected. If the byte contains other than HEX'F0', '40', or '00' on an EBCDIC VOL1 label (Tape or DASD), OLT(s), OLTSEP(s) will not allow this volume to be accesssed by ONLINE TESTS. If VOL1 label on Tape is an American National Standard Label, DOS/OLTSEP will not allow this volume to be accessed if the security byte is other than X'20', '30', or '00'.

Figure 35 (Part 1 of 2). Tape Standard Volume Label 1 Fields

Field	
D5	Source of Information : - Purpose: Used for Direct Access volumes only. Should contain blanks on tape volumes.
D6	Source of Information : - Purpose: Reserved for future use. Should contain blanks.
D7	Source of Information : - Purpose: Reserved for future use as required for American National Standard Institute, Inc. Should contain blanks.
D8	Source of Information : - Purpose: Reserved for the identification of the owner or assignee to whom this volume belongs, such as a customer, installation, department, or system. This can be a value for controlling the allocation of tape reels in a large installation. Processing: On both input and output, IOCS ignores this field.
D9	Source of Information : - Purpose: Reserved for future use. Should contain blanks.

Figure 35 (Part 2 of 2). Tape Standard Volume Label 1 Fields

Displ.	Field	Length	Content
0	D1	3	Label ID: VOL
3	D2	1	Ignored by VSE
4	D3	6	Volume serial number
A	D4	1	Accessibility
B	D5,D6	1A	Reserved
25	D7	E	Name or code of volume owner
33	D8	1C	Reserved
4F	D9	1	Standard byte: 1= file has ANSI standards blank= file does not have ANSI standard

Figure 36. Tape Volume Label for ASCII Code

Field	
D1	Source of Information : System Purpose: Identifies the standard volume label. Processing: On both input and output, IOCS checks this field to verify that a standard volume label is present on the tape when DTF FILABL = STD, or DTFPH, is specified for the first or only file on a tape reel (at the load point). The volume label should be written before a logical file of data records is written on the tape.
D2	Source of Information : System Purpose: Must be 1. Any other VOL labels will be ignored. Processing: This field is processed in conjunction with label identifier (field 1) to identify the volume label completely.
D3	Source of Information : TLBL Purpose: Provides a unique identification for a tape reel (volume). The number is generally assigned when the reel is first received in the installation, and retained as long as the reel is used for files with standard labels. This number should also be used as the File Serial Number in the file HDR1 label of each logical file written on the volume. This provides a unique identification of the volume/file relationship. If a multivolume logical file is written, the Volume Serial Number of the first becomes the File Serial Number of the file HDR1 label on all volumes. Processing: On both input and output, IOCS checks this field against the number supplied by the user in the File Serial Number field of TLBL, for a single-volume file. If TLBL is used and no operand is specified, IOCS assumes the correct volume is mounted and does not check this field. For a multivolume file, IOCS checks the Volume Serial Number of the first volume only. On succeeding volumes, the Volume Serial Number and the File Serial Number differ (as described in Purpose). On output if the tape does not contain a volume label, the operator may key in a 6-digit Volume Serial Number and IOCS then writes the volume label.
D4	Source of Information : - Purpose: Provides a code to indicate that additional identification is required before a volume can be considered the correct one for processing. Processing: On input, if this field is not x'40' IOCS calls phase $IJJTSEC for further checking (see Macro User's Guide). On output, IOCS writes space a space.
D5	Source of Information : - Purpose: Reserved for future use as required for American National Standards Institute, Inc. Should contain spaces.
D6	Source of Information : - Purpose: Reserved for future use as required for American National Standards Institute, Inc. Should contain spaces.

Figure 37 (Part 1 of 2). Tape Standard Volume Label 1 (ASCII Mode) Fields

Field	
D7	Source of Information : - Purpose: Provides for the identification of the owner or assignee to whom this volume belongs, such as a customer, installation, department, or system. This can be a value for controlling the allocation of tape reels in a large installation. Processing: On both input and output, IOCS ignores this field.
D8	Source of Information : - Purpose: Reserved for future use as required for American National Standards Institute, Inc. Should contain spaces.
D9	Source of Information : - Purpose: 1. This file observes the American National Standard Institute, Inc. standards. (Decimal 1) 2. This file does not necessarily observe the American National Standards Institute, Inc. standards but it follows an agreed format. (Space)

Figure 37 (Part 2 of 2). Tape Standard Volume Label 1 (ASCII Mode) Fields

The volume label for tapes is 80 bytes long and begins by VOL1 for the first volume label. Additional volume labels are ignored by VSE.

IBM-Standard File Labels on Tape

Figure 38 and Figure 40 on page 167 show IBM-standard file labels for tapes.

Displ.	Field	Length	Content
0	D1	3	Label ID: HDR, EOF, or EOV
3	D2	1	Label sequence number: 1
4	D3	11	File-ID from TLBL
15	D4	6	Volume serial number of the volume where the file begins
1B	D5	4	Volume sequence number within the file
1F	D6	4	File sequence number on the volume
23	D7	4	Version number of the file
27	D8	2	Sub-version number
29	D9	6	Creation date: cyyddd c indicates the century, blank=19, 0=20, 1=21
2F	D10	6	Expiration date: cyyddd
35	D11	1	Ignored by VSE
36	D12	6	Number of blocks; used in trailer labels only
3C	D13	D	System code: IBMDOSVS
49	D14	7	Reserved

Figure 38. IBM-Standard Tape File Label for EBCDIC Code

Field	
D1	TLBL Default (Output) : - Source of Information : System Purpose: Identifies the type of standard file label. HDR signifies a standard header label at the beginning of a logical data file, and EOF signifies a standard trailer label at the end of a logical data file. EOV is a standard trailer label that signifies the end of records on one reel of tape, with additional records on one reel of tape, with additional records for the same logical file on another reel (volume). Processing: On input, IOCS OPEN/CLOSE routines search for HDR to locate the beginning of a file, and check EOF/EOv to determine the end-of file vs end-of-volume condition. On output, IOCS OPEN/CLOSE writes the appropiate identification. The user never specifies this identification.
D2	TLBL Default (Output) : - Source of Information : System Purpose: Indicates the sequence of this label within a label group (HDR, EOF, EOV). VSE supports File Label 1 only, but provision is made for additional standard file labels in other systems. For example, OS/VS uses both HDR1, EOF1, EOV1 and HDR2, EOF2, EOV2 standard file labels. Processing: This field is processed in conjunction with the label identifier (field 1) to completely identify the type of standard label.
D3	TLBL Default (Output) : DTF Filename Source of Information : TLBL Purpose: Permits the user to identify his logical file by an application-oriented unique name. Processing: IOCS OPEN/CLOSE check against, or write, the name specified by the user, but do not use this field to select the proper file for processing. If TLBL is used and no operand is specified for an output file, IOCS writes the DTF filename.
D4	TLBL Default (Output) : Volume Serial Number of 1st file Source of Information : TLBL Purpose: Provides a numeric (or code) identification for the logical file. In a multivolume file, this field contains the same number in the header label on each volume. This field should contain the Volume Serial Number from the VOL label of the first or only volume of the file. If it does, this uniquely identifies the volume/file relationship. If it does not, an error message is issued to the operator when the volume label is checked (see Figure 35 on page 158, field 3). Processing: On input, IOCS OPEN/CLOSE uses this field in conjunction with label fields 5 and 6 to identify the file specified for processing. The file is specified, by the user by these same three fields in TLBL. On output, IOCS OPEN/CLOSE write the file serial number specified by the user. If TLBL is used and no operand is specified for an output file, IOCS writes the volume serial number of the first (or only) reel of the file.

Figure 39 (Part 1 of 4). Tape Standard File Label 1 Fields

Field	
D5	TLBL Default (Output) : 0001 Source of Information : TLBL/System Purpose: Identifies the order of the columns of data records in a multivolume logical file or in a multivolume multifile set. In a logical file the Volume Sequence Number should be 0001. Processing: In a multivolume file or ina multivolume multifile set, the user need specify (in TLBL) only the number of the first volume to be processed. IOCS increases this number by 1 for each succeeding volume after the first. On input, IOCS uses this label field, in conjunction with fields 4 and 6, to identify the file and volume specified (by TLBL). On output, IOCS writes the volume sequence number as specified by the user or updated by IOCS. If TLBL is used and no operand is specified for an output file, IOCS writes 0001.
D6	TLBL Default (Output) : 0001 Source of Information : TLBL/System Purpose: Identifies the order of the logical files on a multifile volume or in a multifile multivolume set. In a single-file volume, the File Sequence Number should be 0001. Processing: On input, IOCS OPEN uses this field in conjunction with label fields 4 and 5 to identify the file specified for processing. The file is specified, by the user, by these same three fields in TLBL. On output, when a multifile volume(s) is to be written starting at the load point, the user need specify the File Sequence Number of the first file only. If TLBL is used and no operand is specified for the output file, IOCS writes 0001. IOCS increases the number by 1 for each succeeding file. IOCS OPEN/CLOSE writes the appropiate number, as specified or updated.
D7	TLBL Default (Output) : blanks Source of Information : TLBL Purpose: Identifies the various editions of a file, such as a grandfather-father-son relationship. Thus it can be used to ensure that the desired edition of the file is selected for processing, if several editions are maintained in the library for history reference. The editions should be numbered in sequence. Processing: IOCS checks against, or writes, the number supplied by the user. If TLBLis used and no operand is specified for an output file, IOCS writes blanks.
D8	TLBL Default (Output) : blanks Source of Information : TLBL Purpose: Provides a more detailed identification of the editions of a file. For example, field 7 could specify a month (1-12), and this field could specify the activity for a particular week (1-5) of the month. Processing: IOCS checks against, or writes, the number supplied by the user. If TLBL is used and no operand is specified for an output file, IOCS writes blanks.

Figure 39 (Part 2 of 4). Tape Standard File Label 1 Fields

Field	
D9	TLBL Default (Output) : - Source of Information : TLBL for Input System for Output Purpose: Provides the date that the file was originally created. This can be used at a later time to determine how old the records are. Or, it can be used (in conjunction with or in place of generation number) to ensure that the desired edition of the file is selected for processing. Processing: On input, IOCS OPEN checks this date against the data supplied by the user. If TLBL is used, the data is supplied in the Date field. If it is omitted, the creation date in the label is not checked. The format of the date to be entered in TLBL is yy/ddd (year/day of the year). The day may have 1-3 characters. On output, IOCS writes the date that is available in the communication region of the Supervisor. The user does not supply a creation date for an output file.
D10	TLBL Default (Output) : Creation Date Source of Information : TLBL or SYSTEM for OUTPUT Purpose: Indicates the date that the records may be considered inactive. At that time the old file may be deleted by overwriting it with a current edition of the same logical data, or another file. Processing: If TLBL is used, this field is not checked on input. On output, IOCS OPEN compares this field in the old header label to today's date in the communication region to determine if the old label has expired. If so, IOCS overwrites the old label and data records. If not, a message is given to the operator, who then determines whether to to overwrite the old data. In a multifile volume(s) processed sequentially, IOCS checks the expiration date in the old header of only the first file processed. All succeeding files are considered to have expired on the same date. IOCS OPEN/CLOSE writes the expiration date supplied by the user for the new output file. If TLBL is used, the Date field can specify either the date that the file will expire, or a retention period for the file. If the expiration date is specified, the format is yy/ddd (year/day of the year). You may enter 1-3 characters for ddd. The retention period is specified as 0-9999. If this field is omitted, a 0-day retention period is assumed, and IOCS writes the date avaible in the communication region.
D11	TLBL Default (Output) : - Source of Information : System for Output Purpose: Provides a code to indicate that additional identification is required before a file can be considered the correct one for processing. VSE does not use this field, but provision is made for additional protection in other systems. For example, OS allows operator response of a predetermined 'password' to futher authorize a file for processing. Processing: On input IOCS ignores this field. On output IOCS OPEN/CLOSE writes the code supplied by the user. If none is specified, IOCS writes a character zero.

Figure 39 (Part 3 of 4). Tape Standard File Label 1 Fields

Field	
D12	TLBL Default (Output) : - Source of Information : System for Output Purpose: Provides the number of physical records (blocks) written in a file when it was created. This can be used (as a 'hash' total) to verify that all records have been read, when the file is processed later as an input file. The number of records is the total of all physical records between the header and trailer labels of a logical file, excluding tapemarks and checkpoint records. This field is used in trailer labels only. In header labels it contains character zeros. Processing: For a read forward input file, IOCS OPEN sets a counter at zero. During processing, IOCS routines accumulate a count of the blocks read from the tape. At the end of the file or volume, IOCS checks the accumulated count against that in the block count field of the trailer label. If an input file is read backwards, IOCS stores the block count read from the trailer label on OPEN, and decrements the count during processing. At the end of the file (header label), IOCS CLOSE checks the decremented count against the zero in the block count field of the header label. A read backwards file must be contained within one volume. If the accumulated block count does not agree with the count in the trailer label (or header label on read backwards), a message is given to the operator who may ignore the error or terminate the job. On output, IOCS OPEN writes character zeros in this field of the header label. During processing, IOCS routines accumulate the block count and write it in the trailer label at the end of the file or volume. In a multivolume file each EOV/EOF trailer label contains the number of blocks written on that volume only.
D13	TLBL Default (Output) : - Source of Information : System for Output Purpose: Provides a code to indicate the IBM Programming System under which this file is written. This can be of value when an installation uses more than one programming system. Processing: On input IOCS ignores this field. On output IOCS OPEN/CLOSE writes the code supplied by the user. If none is supplied by the user, IOCS writes IBMDOSVSbbbbb in this field.
D14	TLBL Default (Output) : - Source of Information : - Purpose: This field is reserved for future use as required for American National Standards Institute, Inc. Processing: On input IOCS ignores this field. On output, IOCS writes blanks.

Figure 39 (Part 4 of 4). Tape Standard File Label 1 Fields

Displ.	Field	Length	Content
0	D1	3	Label ID: HDR, EOF, or EOV
3	D2	1	Label sequence number: 1
4	D3	11	File-ID from TLBL
15	D4	6	Volume serial number of first volume of the fil
1B	D5	4	Volume sequence number within the file
1F	D6	4	File sequence number within volume(s)
23	D7	4	Version number of the file
27	D8	2	Sub-version number
29	D9	6	Creation date: cyyddd c indicates the century, X'40'=19, X'F0'=20, X'F1'=21
2F	D10	6	Expiration date: cyyddd
35	D11	1	Accessibility byte
36	D12	6	Number of blocks written; only in trailer label
3C	D13	D	System code: IBMZLB followed by two blanks
49	D14	7	Reserved

Figure 40. IBM-Standard Tape File Label for ASCII Code

Field	
D1	TLBL Default (Output) : - Source of Information : System Purpose: Identifies the type of standard file label. HDR signifies a standard header label at the beginning of the logical data file, and EOF signifies a standard trailer label at the end of a logical data file. EOVis a standard trailer label that signifies the end of a record on one reel of tape, with additional records for the same logical file on another tape reel (volume). Processing: On input, IOCS OPEN/CLOSE routines search for HDR to locate the beginning of a file, and check EOF/EOV to determine the end-of-file versus end-of-volume condition. On output, IOCS OPEN/CLOSE writes the appropiate identification. The user never specifies this identification.
D2	TLBL Default (Output) : - Source of Information : System Purpose: Indicates the sequence of this label within a label group (HDR, EOF, EOV). VSE supports File Label 1 and 2 for ASCII, and ignores subsequent numbers. Processing: This field is processed in conjunction with the label identifier (field 1) to identify the type of standard label completely.
D3	TLBL Default (Output) : DTF Filename Source of Information : TLBL Purpose: Permits the user to identify his logical file by a unique, application-oriented name. Processing: IOCS OPEN/CLOSE check against, or write, the name specified by the user, but do not use this field to select the proper file for processing. If TLBL is used and no operand is specified for an output file, IOCS writes the DTF filename.

Figure 41 (Part 1 of 5). Tape Standard File Label 1 (ASCII Mode) Fields

Field	
D4	TLBL Default (Output) : Volume Serial Number of 1st file Source of Information : TLBL Purpose: Provides a numeric (or code) identification for the logical file. In a multivolume file, this field contains the same number in the header label on each volume. This field should contain the Volume Serial Number from the first file (or only file). If it does, this uniquely identifies the volume/file relationship. If it does not, an error message is issued to the operator when the volume label is checked (see Figure 37 on page 160, field 3). Processing: On input, IOCS OPEN uses this field in conjunction with fields 5 and 6 to identify the file specified for processing. The file is specified, by the user, by these three fields in TLBL. On output, IOCS OPEN/CLOSE writes the set identifier specified by the user. If TLBL is used and no operand is specified for an output file, IOCS writes the volume serial number of the first (or only) reel of the file.
D5	TLBL Default (Output) : 0001 Source of Information : TLBL/System Purpose: Identifies the order of the volume of data records in a multivolume logical file or in a multivolume multifile set. In a singlevolume file, the File Section Number should be 0001. Processing: In a multivolume file or in a multivolume multifile set, the user need specify (in TLBL) only the number of the first volume to be processed. IOCS increases this number by 1 for each succeeding volume after the first. On input, IOCS uses this label field, in conjunction with fields 4 and 6, to identify the file and volume specified (by TLBL). On output, IOCS writes the volume sequence number as specified by the user or updated by IOCS. If TLBL is used and no operand is specified for an output file, IOCS writes 0001.
D6	TLBL Default (Output) : 0001 Source of Information : TLBL/System Purpose: Identifies the order of the logical files on a multifile volume or in a multifile multivolume set. In a singlefile volume, the File Sequence Number should be 0001. Processing: On input, IOCS OPEN uses this field in conjunction with label fields 4 and 5 to identify the file specified for processing. The file is specified, by the user, by these same three fields in TLBL. On output when a multifile volume(s) is to be written starting at the load point, the user must specify the File Sequence Number of the first file. If TLBL is used and no operand is specified for the output file, IOCS writes 0001. IOCS increases the number by 1 for each succeeding file. IOCS OPEN/CLOSE writes the appropiate number, as specified or updated.

Figure 41 (Part 2 of 5). Tape Standard File Label 1 (ASCII Mode) Fields

Field	
D7	TLBL Default (Output) : blanks Source of Information : TLBL Purpose: Identifies the various editions of a file, such as a grandfather-father-son relationship. Thus it can be used to ensure that the desired edition of the file is selected for processing, if several editions are maintained in the library for history reference. This edition should be numbered in sequence. Processing: IOCS checks against, or writes, the number supplied by the user. If TLBL is used and no operand is specified for an output file, IOCS writes blanks.
D8	TLBL Default (Output) : blanks Source of Information : TLBL Purpose: Provides a more detailed identification of the editions of a file. For example, field 7 could specify the month (1-12), and this field could specify the activity for a particular week (1-5) of the month. Processing: IOCS chechs against, or writes, the number supplied by the user. If TLBL is used and no operand is specified for an output file, IOCS writes blanks.
D9	TLBL Default (Output) : - Source of Information : TLBL for Input System for Output Purpose: Provides the date that the file was originally created. This can be used at a later time to determine how old the records are. Or, it can be used (in conjunction with or in place of generation number) to ensure that the desired edition of the file is selected for processing. Processing: On input, IOCS OPEN checks this date against the date supplied by the user. If TLBL is used, the date is supplied in the Date field. If it is omitted, the creation date in the label is not checked. The format of the date to be entered in TLBL is yy/ddd (year/ day of the year). The day may have 1-3 characters. On output, IOCS writes the date that is available in the communication region of the Supervisor. The user does not supply a creation date for an output file.

Figure 41 (Part 3 of 5). Tape Standard File Label 1 (ASCII Mode) Fields

Field	
D10	TLBL Default (Output) : Creation Date Source of Information : SYSTEM or TLBL for Output Purpose: Indicates the date that the records may be considered inactive. At that time the old file may be deleted by overwriting it with a current edition of the same logical data, or another file. Processing: If TLBL is used, this field is not checked on input. On output, IOCS OPEN compares this field in the old header label to today's date in the communication region to determine whether the old has expired. If so, IOCS overwrites the old label and data records with the new label and data records. If not, a message is given to the operator, who then determines whether to overwrite the old data. In a multifile volume(s) processed sequentially, IOCS checks the expiration date in the old header of only the first file processed. All succeeding files are considered to have expired on the same date. IOCS OPEN/CLOSE writes the expiration date supplied by the user for the new output file. If TLBL is used, the Date field can specify either the date that the file will expire, or a retention period for the file. If the expiration date is specified, the format is yy/ddd (year/day of the year). The day may have 1-3 characters. If a retention period is specified, 0-9999 days (1-4 characters) may be entered. If this field is omitted, a 0-day retention period is assumed, and IOCS writes the date available in the communication region of the Supervisor.
D11	TLBL Default (Output) : - Source of Information : - Purpose: Provides a code to indicate that additional identification is required before a file can be considered the correct one for processing. Processing: On input, if this field is not X'40' IOCS calls phase $IJJTSEC for further checking (see Macro User's Guide). On output IOCS writes a space.
D12	TLBL Default (Output) : - Source of Information : System Purpose: Provides the number of physical records (blocks) written in a file when it was created. This can be used (as a 'hash' total) to verify that all records have been processed, when the file is processed later as an input file. The number of records is the total of all physical records between the header and trailer labels of a logical file, excluding tapemarks. This field is used in trailer labels only. In header labels it contains character zeros. Processing: For a read forward input file, IOCS OPEN sets a counter to zero. During processing, IOCS routines accumulate a counter of the blocks read from the tape. At the end of the file or volume, IOCS checks the accumulated count against that in the block count field of the trailer label. If an input file is read backwards, IOCS stores the block count read from the trailer label on OPEN, and decrements the count during processing. At the end of the file (header label), IOCS CLOSE checks the decremented count against the zero in the block count field of the header label. A Read backwards file must be contained within one volume. If the accumulated block count does not agree with the count in the trailer label (or header label on read backwards), a message is given to the operator who may ignore the error or terminate the job. On output, IOCS OPEN writes character zeros in this field of the header label. During processing, IOCS routines accumulate the block count and write it in the trailer label at the end of the file or volume. In a multivolume file each EOV/EOF trailer label contains the number of blocks written on that volume only.

Figure 41 (Part 4 of 5). Tape Standard File Label 1 (ASCII Mode) Fields

Field	
D13	TLBL Default (Output) : - Source of Information : System for Output Purpose: Provides a code to indicate the IBM Programming System under which this file is written. This can be of value when an installation uses more than one programming system. Processing: On input IOCS ignores this field. On output IOCS OPEN/CLOSE writes the code supplied by the user. If none is supplied by the user, IOCS writes IBMDOSVSbbbbb in this field.
D14	TLBL Default (Output) : - Source of Information : - Purpose: This field is reserved for future use as required for American National Standards Institute, Inc. Processing: On input IOCS ignores this field. On output, IOCS writes spaces.

Figure 41 (Part 5 of 5). Tape Standard File Label 1 (ASCII Mode) Fields

IBM-standard file labels are 80 bytes long. Each file has a header and a trailer label which have the same format, for reading the tape forward or backward. The first four characters of each label identify the particular label:

```
header label  -- HDR1, HDR2 trailer label -- EOF1,EOF2 at the end of
                          a file
                 EOV1,EOV2 at the end of
                          a volume but not of
                          the file HDR2, EOF2 and EOV2 for ASCII only.
```

Additional labels (HDR3 to 8) are ignored by VSE.

User-Standard File Labels on Tape

Figure 42 shows user-standard file label format for tapes.

Displ.	Field	Length	Content
0	D1	3	Label ID: UHL or UTL
3	D2	1	Label sequence number: 1 to 8
4	D3	4C	User's label information

Figure 42. User-Standard Tape File Label

Field	
D1	Source of Information : System Purpose: Identifies the standard volume label. Processing: On both input and output, IOCS checks this field to verify that a standard volume label is present on the tape when DTF FILABL = NSTD or DTFPH, is specified for the first or only file on a tape reel (at the load point). The volume label should be written previously, before a logical file of data records is written on the tape.
D2	Source of Information : System Purpose: Indicates the sequence of this label within the user header (UHL) or the user trailer label (UTL) group. Processing: This field is processed in conjunction with the label identifier (field 1) to completely identify the user header label or the user trailer label.
D3	Source of Information : User Purpose: Provides a means for you to label your SAM/DAM file with any information you need in addition to that supplied by the standard labels.

Figure 43. Tape User-Standard Label Fields

User-standard labels are header labels located and processed before the data of the file, and trailer labels located and processed after the file. Header and trailer labels are identified by:

User header labels UHLn User trailer labels UTLn

n may be 1 to 8.

User-standard file labels are 80 bytes long. The first four bytes contain UHLn or UTLn and the remaining 76 bytes contain user data.

You can include definitions or descriptions of the file in addition to that in the IBM-standard labels. For example, you may have a unique numbering system for file identification or you may have subcategories that you want to define for the files, or you may want to maintain an audit trail in these labels.

Non-Standard File Labels on Tape

Non-standard labels are only supported on EBCDIC code tape labels. They may have any length, do not have a specified identification in the first four characters, and do not have a fixed format. They may contain whatever information the user desires, and in any arrangement. They are completely the responsibility of the user. He should, however, use some of the features found in standard labels. For example: header labels must be distinguished from trailer labels, end-of-file trailer labels must be distinguished from end-of-volume trailer labels, and some name or number must identify the file to which the label belongs.

When files with non-standard labels or unlabeled files are written on a volume, the volume label is destroyed. Therefore, these files can only be written on volumes that are not expected to be used again for files with standard labels.

LABEL RECORDS IN THE LABEL AREA

When the system reads the DLBL or TLBL and EXTENT statements, it first stores the label information in the label area. The format of the label records in this area is not quite the same as the actual labels on the device. It is shown in *VSE/Advanced Functions Diagnosis Reference: Initial Program Load and Job Control*.

APPENDIX D: MASTER INDEX FOR VSE/ADVANCED FUNCTIONS LIOCS

This Master Index contains references to the VSE/Advanced Functions Logical IOCS manuals. The number(s) after each entry is the key to the manual(s) in which the information is found. The keys correspond to the following manuals.

1. VSE/Advanced Functions LIOCS Volume 1: General Information and Imperative Macros , LY33-9116.
2. VSE/Advanced Functions LIOCS Volume 2: SAM , LY33-9117.
3. VSE/Advanced Functions LIOCS Volume 3: DAM and ISAM , LY33-9118.
4. VSE/Advanced Functions LIOCS Volume 4: SAM FOR DASD , LY33-9119.

INDEX

VSE/Advanced Functions Diagnosis Reference LIOCS Volume 1
General Information and Imperative Macros
Order No. LY33-9116-0

READER'S
COMMENT
FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*
Possible topics for comments are:

Clarity   Accuracy   Completeness   Organization   Coding   Retrieval   Legibility

If you wish a reply, give your name and mailing address:

____

____

____

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

What is your occupation? ____________________

Number of latest Newsletter associated with this publication: ____________

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

LY33-9116-0

Reader's Comment Form

Fold And Tape **Please Do Not Staple** **Fold And Tape**

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1
180 Kost Road
Mechanicsburg, PA 17055

Fold And Tape **Please Do Not Staple** **Fold And Tape**

IBM®

Cut or Fold Along Line

VSE/Advacned Functions Diagnosis Reference LIOCS Volume 1
(File No. S370/4300-30) Printed in U.S.A. LY33-9116-0

LY33-9116-0