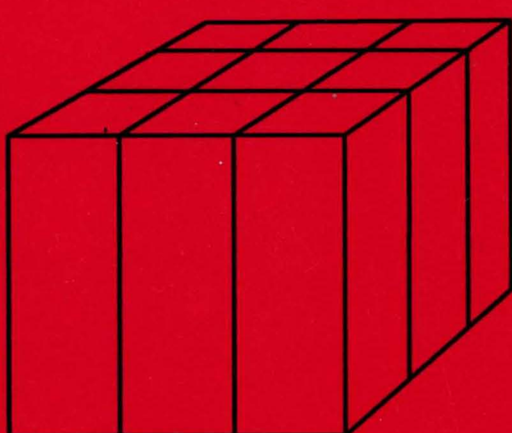
IBM

VSE/Advanced Functions

Data Management Concepts



VSE/Advanced Functions

Data Management Concepts

Program Number: 5666-301

Order Number: GC33-6192-1
File No.: S370/4300-30

Second Edition (March 1985)

This edition is a major revision of GC33-6192-0, an edition which IBM has not made generally available; it applies to Version 2, Release 1 of IBM Virtual System Extended/Advanced Functions (VSE/AF), 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.

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

Publications are not stocked at the addresses given below; requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed either to:

IBM Corporation
Dept. 6R1
180 Kost Road
Mechanicsburg, PA 17055, USA

or to:

IBM Deutschland GmbH
Dept. 3248
Schoenaicher Strasse 220
D-7030 Boeblingen, Federal Republic of Germany

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

PREFACE

This book introduces the basic concepts of data management and label processing as they appear in the VSE system. Knowledge of these concepts is needed mainly by the application programmer , may he program in assembler and use the IBM-supplied assembler macros, or in one of the other supported programming languages such as COBOL, PL/I, RPG II, or VS FORTRAN.

For new terms not explained in the text, the glossary at the end of this book may be helpful. Terms which are defined in the text and abbreviations are to be found in the index.

The book is divided into two parts and two appendixes.

- Part 1 describes subjects related to Data Management in general.
- Part 2 contains information about the Labels used under VSE/Advanced Functions.
- The appendix contains some information about ISAM and DAM files.

The following manuals are referred to in the text:

- VSE/Advanced Functions Application Programming: Macro User's Guide, SC33-6196
- VSE/Advanced Functions Application Programming: Macro Reference, SC33-6197
- VSE/Advanced Functions System Control Statements, SC33-6198
- VSE/Advanced Functions System Utilities, SC33-6100
- VSE/POWER Installation and Operations Guide, SH12-5329
- Using the VSE/VSAM Space Management for SAM Feature, SC24-5192
- IBM Disk Storage Management Guide, Background Reference Information, GA26-1675
- Device Support Facilities User's Guide and Reference , GC35-0033
- VSE/Data Interfile Transfer, Testing, and Operations (DITTO) Program Reference and Operations Manual, SH19-6073
- Introduction to IBM Direct Access Storage Devices, SR20-4738

Detailed information on label processing under VSAM can be found in VSE/VSAM Programmer's Reference, SC24-5145.

VSE/Advanced Functions, Planning and Installation, SC33-6193, lists the other system books.

More titles are listed in the IBM System/370, 30XX and 4300 Processors Bibliography, GC20-0001. Terminology is defined in IBM Vocabulary for Data Processing, Telecommunications, and Office Systems, GC20-1699.

CONTENTS

FIGURES

SUMMARY OF AMENDMENTS

Version 1 Release 3

The manual has been updated by functions and computing services new in Release 3 of VSE/Advanced Functions.

Version 1 Release 3.5

The book has been rewritten to reflect the present conditions in the VSE system and to avoid duplication of information in the VSE system library.

Version 2 Release 1

The book was supplemented by information about label processing which can be found in the second part of this manual. The former label layout sections have been implemented in the VSE/Advanced Functions Application Programming: Macro User's Guide.

PART 1: DATA MANAGEMENT CONCEPTS

The following sections describe some terms of data management, like volume, file or record. The various access methods which can be used to retrieve and store data are also discussed.

WHAT IS DATA MANAGEMENT?

This chapter gives an overview of what data management is and how it is done.

The term data management collectively describes those control program functions that provide access to data, enforce data storage conventions and regulate the use of input/output devices.

A typical application nowadays is designed to build a stock of data, keep it up to date, and extract information out of it. The output may thus take the form of a simple answer or of an analytical report.

This means that the data structures for a planned application must be developed with care as nothing is more important for the ease, reliability, size, and speed of the program as a well adapted and clearly structured set of data files.

VSE/Advanced Functions offers job control statements and a choice of Assembler macros at various levels for the definition and handling of data.

Besides the support given to data management by VSE/Advanced Functions and VSE/VSAM, the following data base programs run under VSE: DOS/VS DL/I and SQL/DS.

Reading Input Data

A program reads data from

- magnetic tapes
- disks
- diskettes
- punch cards

or the data is entered directly from a

- terminal (display device) with a supporting program such as VSE/Interactive Computing and Control Facility (ICCF)
- console
- magnetic or optical character reader

Most usually the data is entered and stored on disk or diskette before it is read from there for further processing.

The reading of data takes more time than its processing because of the mechanical movements in the I/O devices, as opposed to electronic reactions.

For this reason, units of data are often read into alternating buffers so that processing and reading can overlap and waiting times are reduced.

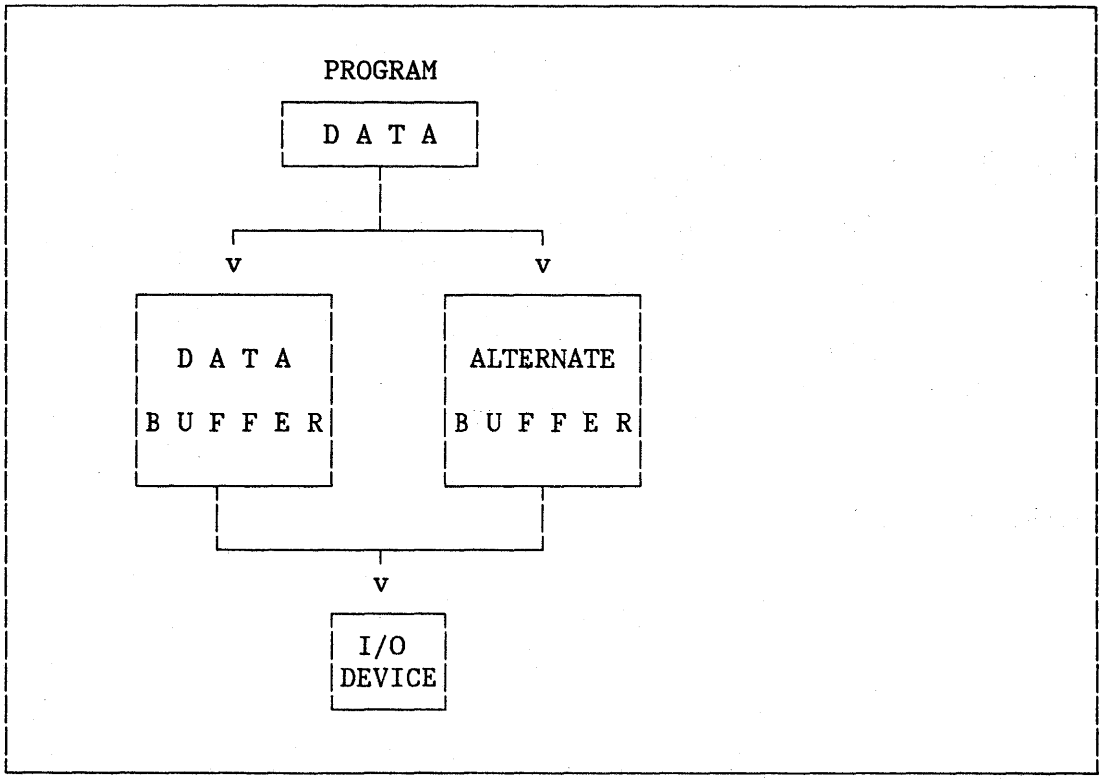


Figure 1. Data Buffering to Reduce the Processing Time

Storing Data

The program stores the data read or modified, on a storage device such as disk, diskette, or tape according to certain conventions. We can process data in a program only by using such conventions.

Fields in Records

Data, for this purpose, is organized as lists of similar items called records. Each record can consist of one or more fields, each with a certain predefined content. The classic example is a personnel file with a record for each employee and in each record one field for name, one for address, one for birth date, etc.

NAME1	ADDRESS1	DATE1	NAME2	.....

<——— ONE RECORD ———>

A file consists of one or more related records.

File Definition in Application Programs

The fields are the same for each record in a file, and the rules for their contents are part of the application program using the data. For example, a personnel file has a field for name consisting of characters and a field for birth date consisting of six numerals (YYMMDD). The program scans the data offered to it to see if it conforms to these rules. The data itself is stored and accessed by data management macros.

Data Types: Character or Numeral

The data in each field is declared in one of two ways:

- as alphanumeric characters such as address: "Maplestreet 12"
- as numerals for calculation such as salary: "5462.20"

As all data in the system is stored as binary digits, numerals are stored and calculated with their binary value, while alphanumeric characters are stored mostly in EBCDIC code (some tape files can use ASCII code instead); cards are punched in column binary code.

Retrieving Data out of Storage

The application program finds the specified data in the specified file. The data found may have to be restructured for output according to a request or to the standard form of output of the program.

Writing Output Data

Output data can go to the same storage medium as mentioned above under "Reading Input Data", except that most optical or magnetic character readers are used only for input, and a printer is used only for output.

The same time lag that occurs when reading data occurs in writing data to an output device, because of the movements involved in writing. Therefore the same technique of writing data from alternating output buffers is used, so that writing and preparing the output can overlap.

Summary

Conventions for Storing Data:

Files have records

Records have fields

Fields of a file are described in the application program

Fields contain characters or numerals

What is Data Management:

To read and write data:	Using buffers
To store and retrieve data:	Restructuring its format as needed

HOW TO DO DATA MANAGEMENT

VSE/Advanced Functions and VSE/VSAM offer some Assembler macros to choose your data structures and access methods if you program in an Assembler-based language. If you program in any of the other programming languages supported by VSE, such as PL/I or COBOL, the language offers you equivalent choices of data definition and handling instructions which deal with the same formats as the Assembler macros.

This book presents the basic formats and data handling choices you have under VSE/Advanced Functions. It does not describe the use of the macros in detail. This is done in the guide and reference books for the VSE/Advanced Functions Assembler macros. See VSE/Advanced Functions Application Programming: Macro Reference and VSE/Advanced Functions Application Programming: Macro User's Guide .

Summary

How to Do Data Management:

Read this book for a VSE base and program with

-- VSE I/O macros or

-- VSE/VSAM and AMS or

-- Any VSE-supported programming language

DEVICES, VOLUMES, AND FILES

This chapter discusses the physical aspects of data handling devices which we traditionally call input/output devices or I/O devices.

Only the device characteristics relevant to the programmer are described. The internal division of files into records and the record organization are discussed in the next chapters.

I/O DEVICES

I/O devices are:

Serial Devices:
- card readers and card punches
- printers
- magnetic tape drives
- optical and magnetic character readers
- terminals with a supporting program such as ICCF
- consoles

Disk Devices:
- CKD devices
- FBA devices

Diskette Devices

Device Characteristics

The following considerations influence the choice of the storage device for a given application:

- capacity
- type of access
- access time
- data transfer rate

Capacity

The capacity of a storage device is the amount of data that can be stored on it. Part of this capacity may be used by the device itself for error checking and for synchronizing the transfer.

The capacity of a storage device is expressed in bytes.

Data Management

Type of Access

SEQUENTIAL ACCESS: Sequential access to the data, that is, reading the records one after the other as they are stored, is possible on any device.

DIRECT ACCESS: Direct access, that is, to find a specific record for processing it alone, is slow or impossible on serial devices but fast and easy on disk devices, where data is found by going directly to its physical address.

Access Time

On disk devices, access time is the time it takes to place the read-write head on a particular track, plus the time to reach the beginning of the desired record on that track. Once the beginning of the record has been reached, transfer of data begins.

Transfer Rate

The number of bytes transferred within a second between external and processor storage is called the transfer rate. This is larger for a disk compared with a card reader, for example.

Device Addressing

Devices are either attached to the system with a channel which means they are continually under the control of the system, or they are connected over a link, for example a telephone connection.

Physical Addresses

At installation time, I/O devices are attached to a channel, either one or several devices on one channel. Each device has a hexadecimal three-digit address, for example 160, where the first digit shows the number of the channel, the second digit represents the number of the control unit, and the third digit determines the unit number. This is the physical address of the device.

```
Device Address

                   1 6 0
                   | | |
Channel number ----┘ | |
Control unit number--┘ |
Unit number -----------┘
```

Symbolic Addresses

The symbolic device address is the address which is specified in the application program. It is the address of the symbolic I/O device which is used by the programmer to process the data.

A symbolic address is assigned to a physical address via the job control statement ASSGN.

Symbolic device address defined in the program		Physical device address (for example a printer device)
SYSLST	// ASSGN SYSLST,00E	00E

The symbolic names have the form SYSxxx in the VSE system. The xxx is either a symbolic letter combination like RDR for 'reader', or a number like 001 or 002.

It is possible to have several symbolic addresses assigned to the same physical device. Disk devices can be used at the same time by several programs.

The symbolic device address technique allows the user to write his programs as if all devices were available. When the program is running and a device type called for is not available, the symbolic device name can then be assigned to some other device.

Summary

I/O Devices:

Measured by:	Capacity
	Access time
	Transfer rate
Classified by:	Access type
Addressed by:	Physical address, such as 160
	Symbolic address, such as SYS001

VOLUMES

The term volume has been defined as the mounted unit for tape, disk, and diskette devices. Such a volume is either a disk pack, a diskette, or a magnetic tape reel.

Each volume is identified by a volume label (optional for tape volumes). For a description of the label handling please see "Labels Overview" on page 57.

Disk Volumes

A disk pack consists of disks, one placed above the other. Each disk has two writing surfaces except the top and bottom disks that usually have only one.

```
 ||<-- Access Mechanism
 ||                   ┌────┐      Disks
 ||       ============|    |============
 ||         ==        |    |
 ||─────────┘  <-- Access Arms with
 ||─────────┐      Read/Write Heads
 ||         ==        |    |
 ||       ============|    |============
 ||         ==        |    |
 ||─────────┘         |    |<-- Spindle
 ||─────────┐         |    |
 ||         ==        |    |
 ||       ============|    |============
 ||         ==        |    |
 ||─────────┘         |    |
 ||─────────┐         |    |
 ||         ==        |    |
 ||       ============|    |============
 ||                   |    |
```

Figure 2. Disk Pack and Access Mechanism

Tracks

Each disk in a pack contains several concentric tracks, each track having the same data capacity for a given device type. On the outermost tracks, the data is more spread out with more space between the bits, whereas on the innermost tracks, data is more tightly packed with less space between the bits.

Cylinders

Tracks on all surfaces located above each other are pictured by the programmer as a cylinder containing as many tracks as there are writing surfaces in the pack. See Figure 3.

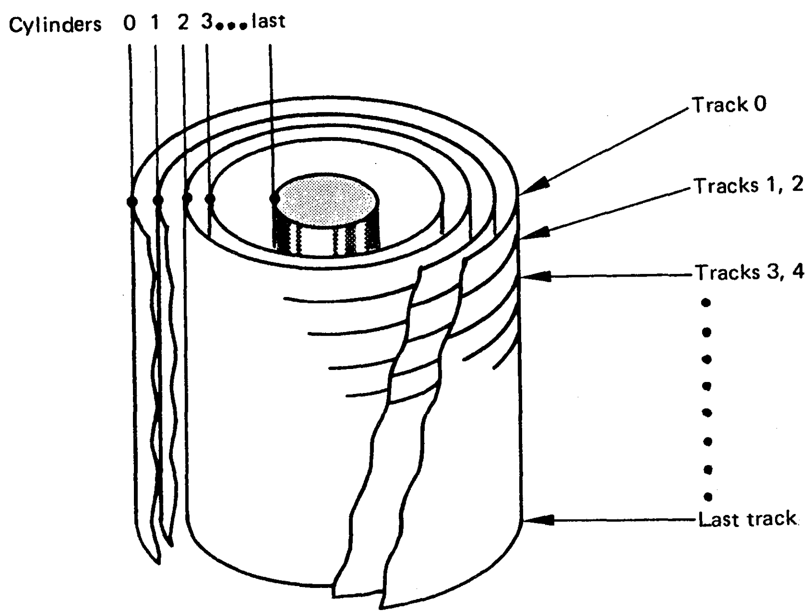


Figure 3. Cylinders in a Disk Pack

Each surface is accessed either by one read-write head that can be moved from one cylinder to another, or by a set of fixed read-write heads, one for each track.

Disk Data Formats

There are two types of disk data formats depending on the device type: Count-key-data (CKD) format or fixed-block-architecture (FBA) format.

COUNT KEY DATA FORMAT (CKD)

For count-key-data devices, a track address and, by convention, a track descriptor record are written at the beginning of each track.

The track address written at the beginning of the track is called the home address (HA). The descriptor record is record zero (R0).

The home address and the record zero are written at manufacture and can be rewritten using the Device Support Facility program. For a description of this program see the manual Device Support Facilities User's Guide and Reference.

Track Format

Each CKD track has the same format:

- Track home address
- Track descriptor record
- Data records

The home address contains:

- The track address as CCHH (representing cylinder and head number).
- A flag byte which states the condition of the track (for example alternate or defective track).

The track descriptor record which is always the first record on a track contains:

- A count area.
- The data area.

The descriptor record is used for alternate and defective track handling.

Data records contain a:

- Count area (to find the data area within the record)
- Key area (optional, may be used by the programmer to identify the record)
- Data area (containing the actual data of the records)

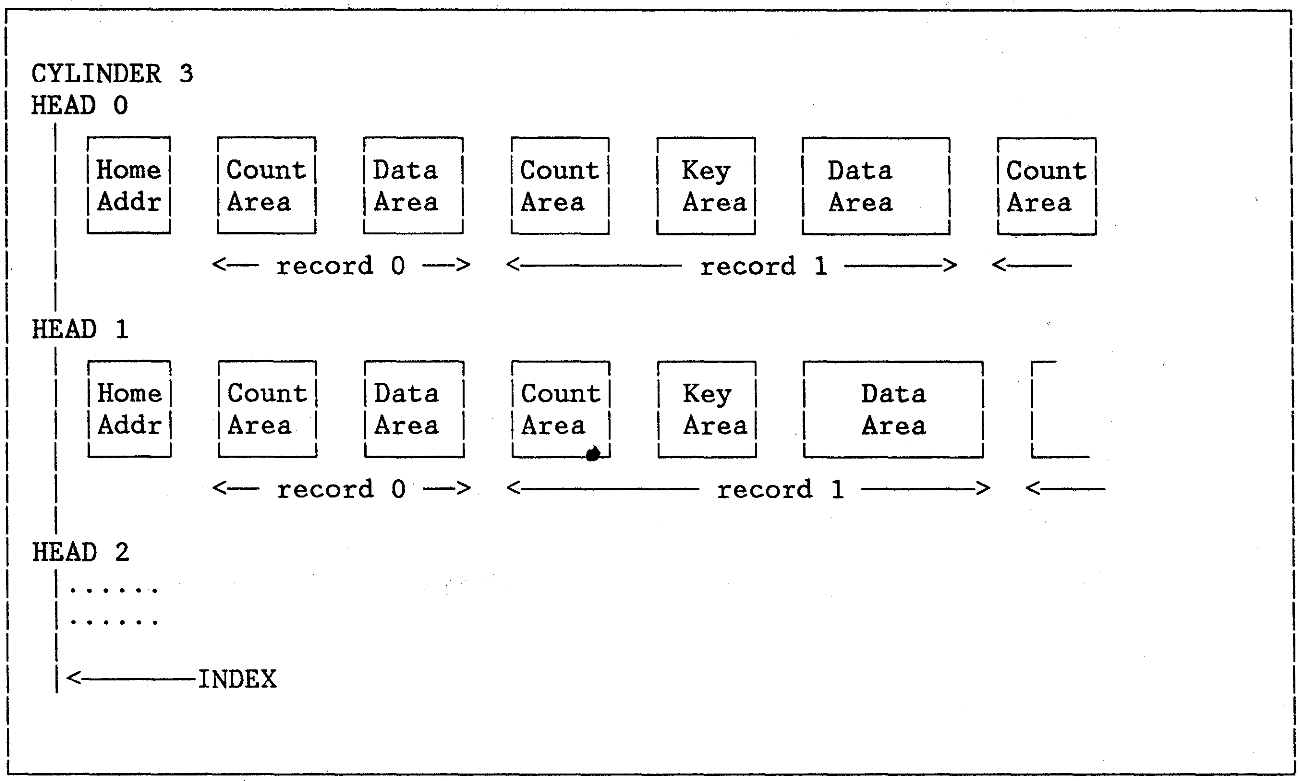


Figure 4. Track and Record Formats for CKD Devices

Figure 4 shows three tracks (head0, head1, head2) on cylinder 3 and the format of the records residing on them. The beginning of the home address is always behind the hardware index marker (hardware positioning).

FIXED BLOCK ARCHITECTURE FORMAT (FBA)

For fixed-block-architecture device types (3310, 3370-1, 3370-2), tracks and records are formatted at the time of manufacture. There is no track home address as on CKD devices.

Track Format

Each track contains several blocks. The blocks contain:

- An ID area (containing the address of the block itself or of the alternate block if it is defective).
- A data area (fixed number of bytes).

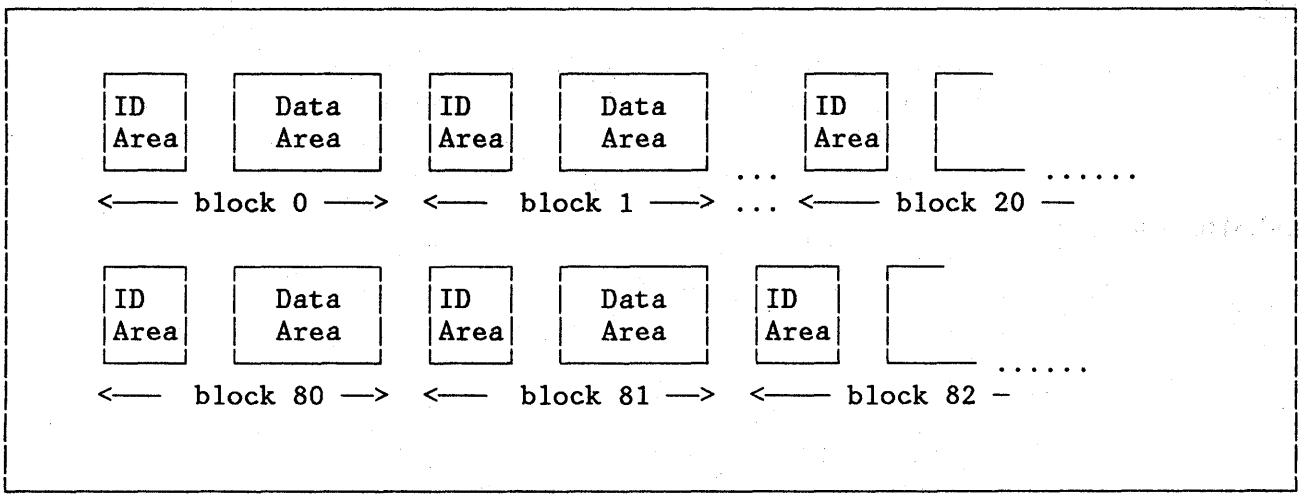


Figure 5. Track and Record Formats for FBA Devices

Alternate Tracks

Not all CKD tracks or FBA blocks are available for primary use; some are reserved as alternate tracks or blocks to contain data in place of a defective primary track or block.

Disk Volume Capacity

A volume has to accommodate some non-data areas: count fields and gaps. Therefore the net data capacity varies with the number of records stored.

Disk Volume Initialization

A disk volume must be initialized by the user with an IBM-standard volume label located on cylinder 0, track 0, record 3 on a CKD device, or in block 1 on a FBA device, using the Device Support Facility (DSF) program. For more details see the Device Support Facilities User's Guide and Reference.

This volume label contains:

Label-id VOL1
Volume serial number
Address of the VTOC

VTOC

A VTOC (volume table of contents) is a reserved area on each disk volume which contains file label information of all files residing on the volume.

Diskette Volumes

The diskette supported by VSE/Advanced Functions has a volume with a single writing surface and the following organization:

- Track 0 is used for the VTOC.
- The other tracks are used for data and as alternate tracks, if one of the primary tracks becomes defective.
- Each track is divided into 26 sectors of 128 bytes. Therefore, a block of data should also have 128 bytes. If it is shorter, the rest of the 128 bytes of the sector is wasted.

Diskette Initialization

A diskette volume is initialized by the factory with an IBM-standard volume label on track 0, sector 7. The label contains among other fields:

VOL1
Volume serial number
Accessibility indicator
Standard label level indicator

Tape Volumes

Records stored on magnetic tape are separated by interblock gaps (IBGs). These spaces between the transfer blocks are needed to let the tape accelerate before reading or writing and then come to a halt.

On an 8809 tape device, it is possible through an internal blocking algorithm, to read and write without any stops. This method is called streaming mode processing. The device can also be used in the non-streaming mode.

A magnetic tape reel can have either 7 or 9 tracks. This has little effect for the user; but the system is informed of it at IPL time when the device type code (with T7 or T9) is associated with the physical address.

A tape volume can be read forward or backward, but a T7 volume cannot always be read backward.

Data Management

Tape Initialization

Tape volumes are initialized with an IBM supplied utility program called INTTP. See VSE/Advanced Functions System Utilities.

In IBM-standard label reading provision is made to accept additional volume labels (VOL2 to VOL8) but writing such labels is not supported. Tape volumes may also be left unlabeled.

A tape volume can be initialized by the user with an IBM-standard label or an ANSI-standard label written as the first record of the reel. If user-standard labels are to follow, this first one is required. A standard volume label for tape contains:

VOL1 to VOL8
volume serial number
other control information

The tape volume can also be left unlabeled or have non-standard labels. Label handling routines check the volume label and, if no standard volume label is present on an output tape, a message is issued. The operator can then enter the volume serial number, so that the volume serial number can be written.

Summary

Volumes:

Identified by :	Label with serial number, such as 666666		
Types of Volumes:	Disk: CKD	--	with cylinders and tracks
	FBA	--	with blocks
	Diskette	--	with tracks and sectors
	Tape	--	7 or 9 tracks

FILES

The term file is defined as a set of related records. A file is set up in a uniform and strictly defined structure for use in one or several applications. Finding the most practical organization for each file is one of the most important tasks of program development.

Files and Volume Relationship

Multi-Volume File

One file may be written over any number of volumes. For sequential processing, the volumes of such a multi-volume file can be mounted and processed one after the other. For direct-access processing, however, all volumes of a multi-volume file must be mounted.

Multi-File Volume

A volume can also accommodate several files. It is then called a multi-file volume.

These relationships are illustrated in Figure 6 on page 20.

```
A. Single File/Single Volume
 ┌─────────┐
 │  Volume │
 └─────────┘
 |<------->|
    File

B. Multi-Volume File
 ┌──────────┐ ┌──────────┐ ┌──────────┐
 │ Volume 1 │ │ Volume 2 │ │ Volume 3 │
 └──────────┘ └──────────┘ └──────────┘
 |<---------------------------------->|
               File

C. Multi-File Volume
 ┌────────────────────────────────────────
 │        Volume
 └────────────────────────────────────────
 |<------>|<------>|<------>|<------>|
   File 1   File 2   File 3   File 4

D. Multi-Extent File on one Volume
 ┌──────────────────────────────────────────
 │        Volume
 └──────────────────────────────────────────
 |<------->|          |<------>|<------->|
   Extent1              Extent2   Extent3

   The Extents combined to one File
 ┌─────────────────────────────────┐
 │ Extent1 │ Extent2 │  Extent3    │
 └─────────────────────────────────┘
 |<--------- File ---------------->|
```

Figure 6. File and Volume Patterns

EXTENTS ON A VOLUME

Disk Extents

A continuous piece of a file is called an extent. A file can reside on one or more disk extents.

Multi-Extent File

Files consisting of more than one extent are called multi-extent files. The extents for a file can exist on one or more volumes as shown in Figure 6 on page 20.

You define the place on disk and the length of an extent via the EXTENT Job Control statement. For multi-extent files multiple extent statements are required.

Split-Cylinder

A special case is a split-cylinder extent. This means that an extent can occupy, for example, tracks 0 to 8 only, on several cylinders.

This organization may save access time if related files can thus be placed in this way because for read/write operations the movement of the read/write mechanism can be decreased.
Changing tracks (read/write heads) then is only an electronical action and therefore faster than the mechanical movement. You can define the split-cylinder via the EXTENT statement. Please refer to the manual VSE/Advanced Functions System Control Statements for details.

Diskette Extents

On a diskette volume, each file can have only one extent. Only the overflow of a file goes to the next volume as a new extent.

File Labels

Files have to be labeled when they are stored on disk, diskette, or tape devices.

In case of disk devices these labels are used to identify and address each extent on the volume. The labels of all files residing on the volume are stored in the VTOC (Volume Table Of Contents). The location for a given file extent on a disk volume is found by using the extent address from the appropriate file label in the VTOC. The VTOC itself is addressed by the volume label.

```
                  DISK VOLUME
┌──────────────────────────────────────────────┐
│                                              │
│ VOLUME LABEL                                 │
│      │             VTOC                      │
│      │       ┌──────────────┐                │
│      └──────>│ FILE LABELS  │                │
│              └──────────────┘                │
│                       │                      │
│                       └──> DATA EXTENTS      │
│                                              │
└──────────────────────────────────────────────┘
```

For a more detailed description of labels refer to "Labels Overview" on page 57.

File Specification

The space for the file is defined via the DLBL and EXTENT job control statements. With these two statements information like the following is submitted:

- File name (for which the space is defined)
- File id (for which the space is defined)
- Logical device address (SYSXXX specifying the device)
- Volume serial number (specifying the volume)
- Sequence number (number of the extent in a multi-extent file)
- Extent begin address (defining the file's position on the volume)
- Extent length (specifying the file's length)

The characteristics of the file have to be defined in the application program. For example, programs written in assembler language contain a DTF statement (Define The File) to define the characteristics of the used file.

The connection between the application program's file definition (DTF, for example) and the specified extent has to be established. Figure 7 on page 23 shows:

[1] How to connect the logical to the physical device address (ASSGN).

[2] How to establish the connection between the application program's file name and the name of the file on the volume (DLBL).

[3] How to define the characteristics and the logical device address for the file in the program (DTF).

[4] How to define the volume and the position on the disk for the file (EXTENT).

```
   ┌───────────────────────────┬─────────────────────────────┬──────────────────────────┐
   │          PROGRAM          │    JOB CONTROL LANGUAGE     │     PHYSICAL DEVICE      │
   ├───────────────────────────┼─────────────────────────────┼──────────────────────────┤
   │ NAME:PROGFIL              │                  ┌─────────>>│ ADDRESS: 160             │
   │                           │ // JOB FILE      │          │                          │
   │      START                │   ..             │          │                          │
(1)│      .                    │ // ASSGN SYS010,160         │                          │
   │      .                    │   ..            │           │                          │
   │      .              ┌─────┼─────────────────┘           │  ┌───────────────────┐   │
   │      .              │     │   ..                        │  │                   │   │
(2)│      .              │     │ // DLBL FILE,'FILEID'       │  │ File with         │   │
(3)│FILE  DTFXX DEVADDR=SYS010│   ..       │     │           │  │ file-id:          │   │
   │ └───────────────────┼─────┼────────────┘     └───────>>│  │ FILEID            │   │
   │      .              │     │   ..                        │  │ ┌─────────────┐   │   │
(4)│      .              │     │ // EXTENT SYS010,VOLSER,..  │  │ │  D A T A    │   │   │
   │      .              │     │   ..         │      │       │  │ └─────────────┘   │   │
   │      .              └─────┼──────────────┘      │       │  │ on                │   │
   │      .                    │   ..                │       │  │ volume with       │   │
   │      OPEN FILE            │ // EXEC PROGFIL     │       │  │ serial-number:    │   │
   │      .                    │ /&                  └─────>>│  │ VOLSER            │   │
   │      END                  │                             │  │                   │   │
   │                           │                             │  └───────────────────┘   │
   └───────────────────────────┴─────────────────────────────┴──────────────────────────┘
```

Figure 7. Locating the Disk Data

Specifying VSE/VSAM Files

Management for VSE/VSAM files differs from the file-volume scheme described above: VSE/VSAM manages one disk area which may spread over one or more volumes to accommodate the files under its administration. Individual VSAM files in this area are placed, found, and maintained through a VSAM catalog. Therefore an extent statement is not required for VSAM files.

Summary

Files:

Located on:	One or several volumes
Developed:	From logical requirements into physical organization
Identified by:	Labels
Consisting of:	Records

RECORDS, BLOCKS, AND CONTROL INTERVALS

This chapter describes the use of terms like record, block, and control interval within the VSE system and then proceeds to a discussion of record formats available for each device type, including the control information needed.

Records

A record is a collection of related fields of information. The user defines the data type and the length for each field to hold the largest item of data that may be encountered. The sum of all field lengths in a record is the length of the record.

A distinction is made between a logical record, which is a collection of related data fields, and a stored record, which contains the same data plus some control information fields. Figure 8 and Figure 9 show the difference between logical records and stored records.

Logical Record

```
 ┌────────────┬─────────────┬────────────┐
 │    NAME    │   ADDRESS   │    DATE    │
 └────────────┴─────────────┴────────────┘
 <- field 1->|<- field 2 ->|<- field 3->

 <──────────── LOGICAL RECORD ──────────>
```

Figure 8. Logical Record

Stored Record

```
 ┌────────┐ ┌────────────┬─────────────┬────────────┐
 │   RL   │ │    NAME    │   ADDRESS   │    DATE    │
 └────────┘ └────────────┴─────────────┴────────────┘
           <────────── logical record ──────────>

 <──────────────── STORED RECORD ────────────────>
```

Figure 9. Stored Record

In Figure 9 the control information which has been added to the logical record is the record length (RL).

Blocks

Blocks of Records

To save time and space in processing, records can be grouped into blocks to get larger transfer units. The number of records in one block is called blocking factor. Figure 10 illustrates the blocking of records.

```
|BL|RL|NAME|ADDRESS|DATE|RL|NAME|ADDRESS|DATE|

   <- stored record -->|<- stored record -->

 <------------- BLOCK ---------------->
          with BLOCKING FACTOR 2

BL stands for block length
```

Figure 10. Block, Blocking Factor

Blocks of Storage

The space on a FBA disk is divided into equal blocks of 512 bytes. FBA blocks are addressed by relative block numbers. Each FBA block may contain one or more records (see Figure 11).

```
|BL|RL|NAME|ADDRESS|DATE|RL|NAME|ADDRESS|DATE|   |
|<------ block (blocking factor 2) ------->       |
|                                                 |
|                                                 |
|                                                 |

 <------------- 1 FBA BLOCK ---------------->
```

Figure 11. FBA Block

VSAM (Virtual Storage Access Method) also uses the fixed block architecture.

Control Intervals (CI)

A file on a FBA disk is an ordered set of units of data transfer. This unit of data transfer is called a control interval (CI). A CI may contain one or more FBA blocks.

The CI can contain unused space at the end. A description of each data block stored in the CI is added at the end of each control interval (RDFs). RDF stands for record definition field. The CIDF (Control Interval Definition Field) is the last entry in each CI.

Figure 12 shows one control interval which contains:

- Two FBA blocks
- Two data blocks
- Control information for each data block (RDFs)
- Control information for the CI (CIDF)

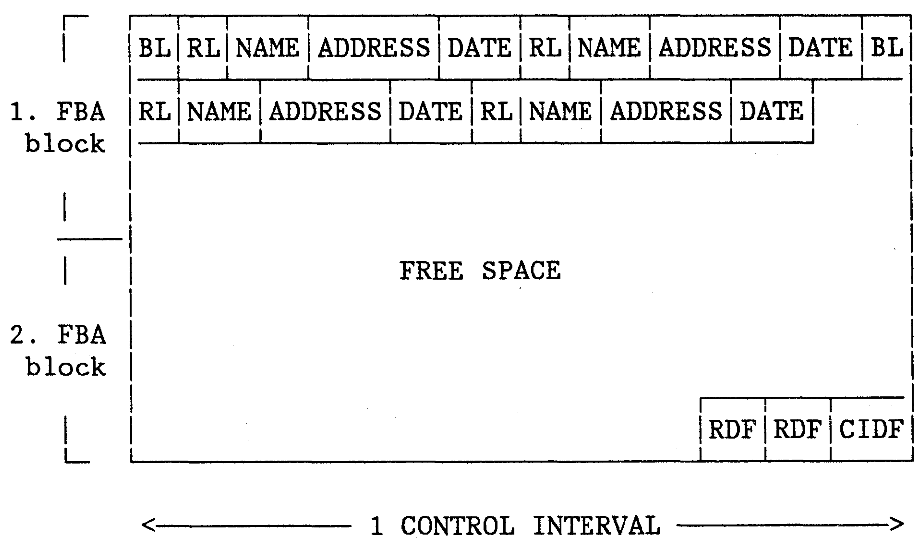


Figure 12. Control Interval

```
Summary

Records  --  Blocks  --  FBA Blocks -- Control Intervals

   Field + Field + Field ...      =  Record

   Record + Record + Record ...   =  Block

   Rec. or Block + Rec. or Block ... + Free Space + ci = CI

ci = control information, here RDFs and CIDF
```

RECORD FORMATS

Record formats acceptable in VSE can:

- have fixed or variable length
- be blocked or unblocked
- allow records to continue across block boundaries or not (spanning)
- be totally undefined

Through combinations of these types and depending on the device, you can have up to seven different options for the record format:

- Fixed length, blocked or unblocked
- Variable length, unspanned, blocked or unblocked
- Variable length, spanned, blocked or unblocked
- Undefined

Fixed-Length Records, Blocked or Unblocked

In the fixed-length record format, the records have an unchanging length. A list of names, for example, can be declared as a fixed length record file. Such a file is determined by fixed length records, with one record for each name. All data does not necessarily have the same length. This means that names shorter than the specified length are padded with blanks and longer names are truncated to make them fit into the name field of the record.

Punched card records have a fixed length as the card takes 80 bytes (or 96 bytes for some devices).

Fixed-length records are easy to process and to control. They may be blocked or unblocked. The number of records in a block (blocking factor) is constant, with a possible exception in the last block.

Variable-Length Records, Blocked or Unblocked

A record can be of variable length for two reasons:

- It contains one or more variable-length fields.
- It contains a variable number of fields.

Block and Record Descriptors

Figure 13 on page 30 shows that variable-length records may be blocked or unblocked. They always contain a control field showing their length (RL). Each block of variable records contains a control field for the complete block length (BL). Note, that the unblocked format is considered blocked also, but with a blocking factor of 1.

```
Unblocked Record (blocking factor 1)

    | BL | RL | Record |

              |<------>|
                 Data
         |<----------->|
         Record Length
    |<---------------->|
         Block Length


Blocked Records (blocking factor 2)

    | BL | RL | Record | RL |     Record     |

                       |<----------------->|
                          Record Length

    |<------------------------------------->|
                 Block Length

BL = Block Length Field
RL = Record Length Field
Note: The value in BL and RL includes the length of control fields BL and
RL, respectively.
```

Figure 13. Variable-Length Records, Blocked and Unblocked

The fields indicated as BL and RL are called descriptors, each is 4 bytes long. The I/O areas must be large enough to accommodate the maximum block length expected, including the control fields.

If an application program writes variable length records, it must specify the length of each record.

Spanning Records across Block Boundaries

A record may be split so that one part of it is in one block and the remainder in another block, and reassembled again for processing. This technique is called spanning of records and must be declared as a special format. It is done automatically if necessary, but not for ASCII coded records.

Spanned records may be useful when a file is to be moved between device types allowing different block sizes. The maximum block size of the receiving device may be smaller than one record. Every record has to be cut into segments. Another example when spanned records might be useful is in text processing applications where very long strings of text must be written.

The programmer does not have to be aware of the maximum data capacity of the I/O areas. Data management divides the records into segments that never exceed the size of the output area.

Undefined Records

Any record not conforming to the rules of fixed-length, variable-length, or spanned record formats as described above, is considered an undefined record. Data management permits the processing of such records but does not support it. Therefore, if you want to block or deblock such records, an application program has to be written to do these functions.

Programs writing undefined records must communicate the size of each record to data management. Problem programs reading such records are informed of their length by data management routines.

Summary

Record Formats:

Fixed Length:	Blocked	
	Unblocked	
Variable Length:	Unspanned:	Blocked
		Unblocked
	Spanned:	Blocked
		Unblocked
Undefined:	Unblocked	

HOW RECORD STRUCTURES DEPEND ON I/O DEVICES

How data must be presented to the various I/O devices depends on the device type.

Records on CKD Devices

There are three areas in a CKD record (or block in blocked format):

count area
key area (optional)
data area

Count Area

This area is recorded automatically and maintained by the system programs. It consists of:

- flag byte
- ID field
- key length
- data length
- check bytes

The ID field contains the address of the record or block in the format CCHHR for cylinder, track (or head), and record. The record number is the sequential position of the record on the track.

Key length and data length specify the length of the key area and the data area.

Optional Key Area

This area consists of:

- key field
- check bytes

The key field is the ID of the record such as a part number or an employee number. For a block, the ID is the highest or lowest key of the record group. The length of a key can vary from zero to a maximum of 255 bytes.

Data Area

This area contains:

- data
- check bytes

The data area of a block contains one or more records.

If spanned records are processed, the data area may contain all or part of a record and only the first segment can have a key. Disk records do not span volumes as tape records can.

Figure 14 on page 33 shows the structure of a record on a CKD disk device.

Record with a Key Area

<——Count Area——>					<–Key Ar.–>		<–Data Ar.–>	
Flag Byte	Address cchhr	Key Length	Data Length	Check Bytes	Key* Field	Check Bytes	Data*	Check Bytes

Record without a Key Area

<——Count Area——>					<—Data Area—>	
Flag Byte	Address cchhr	Key Length (0)	Data Length	Check Bytes	Data*	Check Bytes

*) These fields have variable length.

Figure 14. Structure of a CKD Record

CKD records of fixed and variable length format, both blocked and unblocked, are shown in Figure 15 on page 34. The variable-length records always have record and block descriptors to show the varying lengths.

```
A. Fixed-Length Unblocked CKD Records
  ┌───────────┬──────────────┐
  │Count Area │     Data     │
  └───────────┴──────────────┘
              |<------------>|
                  Record
  |<------------------------>|
       Physical  Block

B. Fixed-Length Blocked CKD Records (blocking factor 3)
  ┌───────────┬──────┬──────┬──────┐
  │Count Area │ Data │ Data │ Data │
  └───────────┴──────┴──────┴──────┘
                            |<---->|
                             Record
  |<------------------------------>|
       Physical  Block

C. Variable-Length Unblocked CKD Records
  ┌───────────┬────┬────┬──────────┐
  │Count Area │ BL │ RL │ Data     │
  └───────────┴────┴────┴──────────┘
                   |<------------->|
                         Record
  |<------------------------------>|
           Physical Block

D. Variable-Length Blocked CKD Records (blocking factor 3)
  ┌───────────┬────┬────┬──────┬────┬──────┬────┬──────┐
  │Count Area │ BL │ RL │ Data │ RL │ Data │ RL │ Data │
  └───────────┴────┴────┴──────┴────┴──────┴────┴──────┘
                                         |<---------->|
                                             Record
  |<-------------------------------------------------->|
                  Physical Block

 BL = Block Length Field does not include the length of the count area
 RL = Record Length Field
```

Figure 15. Records of Fixed and Variable Length, without Key Area on CKD

Figure 16 shows an example of spanned records on a CKD device.

```
Unblocked Spanned CKD Records

|Count Area|BL|SL|Data|Count Area|BL|SL|Data|Count Area|BL|SL|Data|

<--------------------->
   Physical Block
                <---->              <---->              <---->
                  |_____________________|____________________|
                                 Spanned Record

Blocked Spanned CKD Records

|Count Area|BL|SL|Data|SL|Data|SL|Data|Count Ar.|BL|SL|Data|SL|Data|

<------------------------------------>                      <------->
          Physical Block                                     Segment
                                 <---->              <---->
                                   |____________________|
                                        Spanned Record

BL = Block Length Field
SL = Segment Descriptor Field shows the length of the
     segment and whether it is the first, the only, the last, or a
     middle segment of the record.
```

Figure 16. Spanned Records on CKD Device

Undefined records have the same format as fixed-length unblocked records: they have no descriptors.

Records on FBA Devices

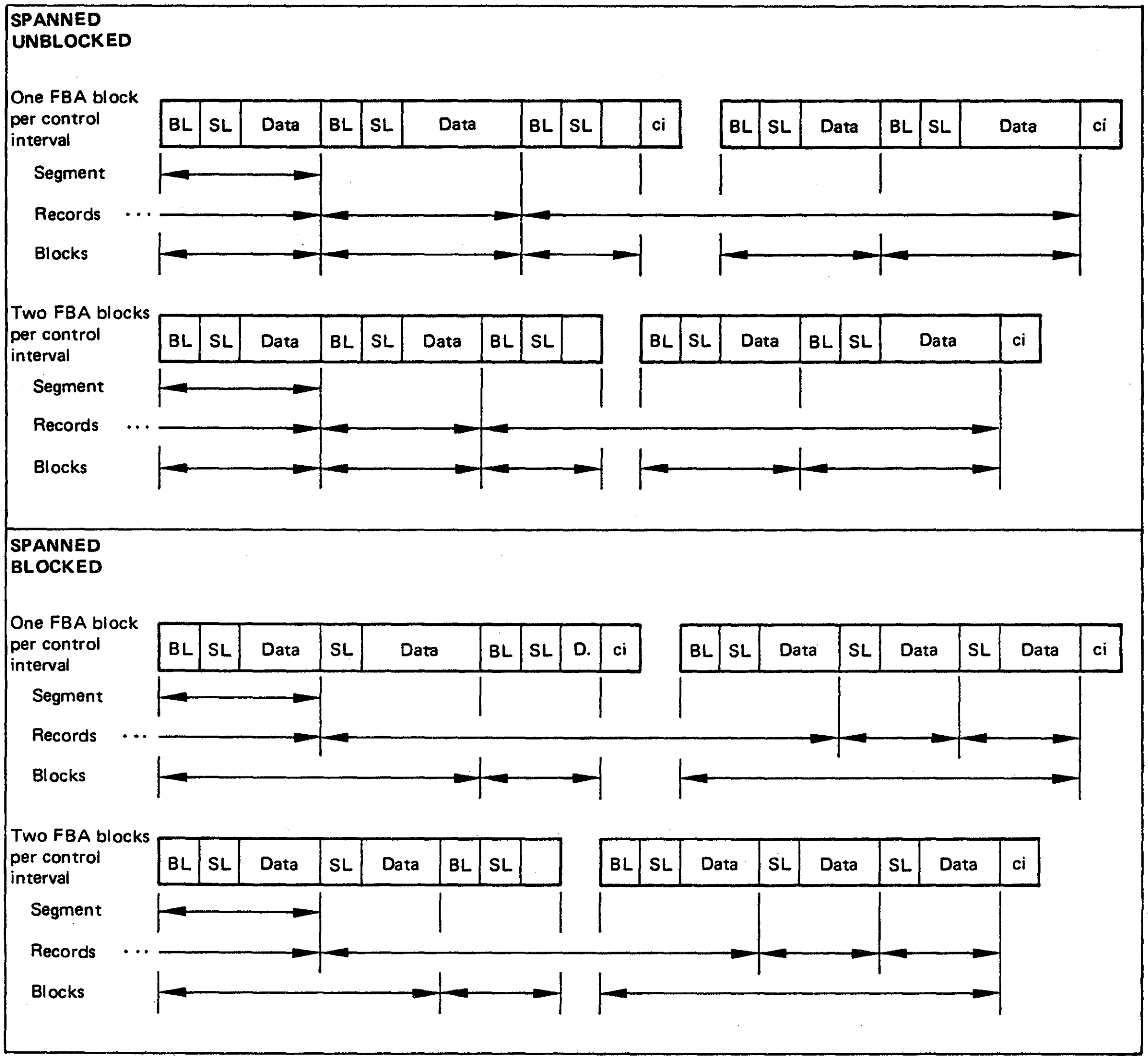
Data resides on FBA devices in FBA blocks addressed by block number. A file on an FBA device is a set of transfer units called CIs (control intervals). Recording the data on an FBA device, data management writes a control interval (CI) over a full number of FBA storage blocks. The CI contains control information consisting of a CIDF (Control Interval Definition Field) and RDFs (Record Definition Fields) to describe the records or blocks of records in the CI. Figure 17 shows FBA record structures for the various formats. The record or block of records can be smaller or bigger than a block of FBA storage and the control interval can have one or more blocks of FBA storage. Therefore, the field "Data" in the figure may represent in case of:

- Fixed-length records:
  - one or several records
- Variable-length records:
  - one or several |BL|RL|data| units
  - one or several |BL|RL|data|RL|data|RL|data| ... units

```
┌──────────────────────────────┬────────────┬──────────────┐
│             Data             │ Free Space │ RDF(s) and CIDF│
└──────────────────────────────┴────────────┴──────────────┘
|<──────────────────────────────>|
Record(s) or Logical Block(s)
|<────────────────────────────────────────────────────────>|
          FBA Block(s)  =  Control Interval
```

Figure 17. FBA or VSAM Control Interval

Figure 18 shows the various possibilities of organizing spanned records on FBA devices.



BL = Block Length Field
SL = Segment Descriptor Field shows the length of the segment and whether it is the first, the only, the last, or a middle segment of the record.
ci = control information, that is, RDFs and CIDF

Figure 18. Spanned Records on FBA Device

Records on Diskettes

Records or blocks can be read from diskette only in fixed-length (up to 128 bytes), unblocked format. Each track has a fixed format with 26 sectors.

Although records cannot be blocked on the diskette, it is possible to chain records together in the I/O area and have them processed in groups of 2, 13, or 26 records to optimize I/O operation.

Control information consists of:

- index marker for a track (IM)
- sector identifiers for each sector (ID)

The index marker shows the beginning of a track. The sector ID gives the address of each of the 26 sectors on a track. The sector ID includes bytes which are used to verify the validity of reading and writing.

Figure 19 shows the form of a diskette sector.

```
...Gap|ID|Gap|Data|Gap|IM|Gap|ID|Gap|Data|Gap|ID|Gap|Data|...

          ------->|<--------
          Track Beginning
```

Figure 19. Diskette Sector Format

Records on Magnetic Tapes

All standard record formats are acceptable for magnetic tape. Spanned records may span volumes.

Records or blocks of an ASCII coded tape file may be preceded by a block prefix which can be up to 99 bytes long. All ASCII records are processed in EBCDIC, and the system takes care of the conversion and translation.

Summary

Record Formats on Disk -- Diskette -- Tape:

Disk Records:

Formats: all (Records span extents, but not volumes.)

A CKD record or block contains: count area
(key area)
data area

An FBA or VSAM CI contains: blocks of one or more
records free space
RDFs defining these blocks
a CIDF defining the CI

Diskette Records:

Format: fixed-length unblocked

A record contains: record id
data

A track contains: index marker
26 sectors

Tape Records:

Formats: all

Records may span volumes

ASCII code files may have block prefixes

Records for Printers

Printed output may be of any unblocked format. The maximum size of a record is defined by the maximum length of a print line.

Carriage control for the printer may be specified via a control character in the data records. See VSE/Advanced Functions Application Programming: Macro User's Guide for a detailed description.

For the 3800 Printing Subsystem, a character arrangement table can be selected by specifying a table reference number preceding the control character for carriage control.

Loading Print Control Buffers

For most printers, print control buffers have to be loaded. Some need a forms control buffer (FCB), others an universal character set buffer (UCB), and several both.

The forms control buffer describes the format of the printed page. It determines, for example, where the first printed line is placed on the page, how many lines per inch have to be printed, and where the end of form is defined.

A UCB, a universal character set buffer, describes which print train is being mounted. This is not done by control information in the records, but since it influences the output of data, we discuss it here shortly. The buffers of a printer can be loaded as follows:

1. Automatically during Initial Program Load. The $$BUFLDR program is executed as part of IPL if a PRT1, 3800, or 1403U printer is attached to the system.

2. Dynamically by issuing the LFCB or LUCB attention command. See VSE/Advanced Functions System Control Statements.

3. Dynamically by issuing the LFCB macro in an application program. See VSE/Advanced Functions Application Programming: Macro Reference.

4. As a separate job step by executing the SYSBUFLD program. See VSE/Advanced Functions System Control Statements.

5. Under POWER via the $$ LST statement. See VSE/POWER Installation and Operations Guide.

Records on Card Devices

Card records for input consist of fixed-length, unblocked records and have up to 80 or 96 characters depending on the devices used.

Card output may be any unblocked format. When variable-length records are punched, the length fields for each record (BL and RL) are not punched.

The user can specify by a control character in the data records which stacker is to be used for output. See VSE/Advanced Functions Application Programming: Macro User's Guide for a detailed description.

Summary

Records on Cards or Printers

Card Records:	Format: input : fixed-length unblocked output: any unblocked
	Length: 80 or 96 bytes
	Control Character for Stacker
Printer Records:	Format: any unblocked
	Length: line length
	Control Character for Carriage Control
	Table Reference Number (for 3800)
	Print-Control Buffers FCB and UCB

Records for Optical and Magnetic Character Readers

The following devices are discussed here:

1270 MICR
1275 OCR
1287/1288 OCRs
3881 OMR
3886 OCR

With a 1270 and 1275, only undefined records can be read. A record type can be chosen by the field selector switches on the reader.

For 1287 and 1288, fixed-length records, blocked or unblocked, and undefined record formats can be used. Each field to be read can be treated as fixed-length or undefined.

From a 3881, records can be read only in fixed-length, unblocked format. They can be up to 900 bytes long. The first six bytes of the record are used for record descriptor information, the remainder for data.

For the 3886, fixed-length blocked or unblocked format is used.

Records on Consoles

Records may be entered or displayed in fixed-length or undefined format. The blocksize must not exceed 256 characters.

HOW RECORDS ARE ORGANIZED FOR PROCESSING

This chapter describes the different types of data processing an application may require and the methods of data organization and retrieval available for such processing.

SEQUENTIAL PROCESSING OR DIRECT ACCESS

The efficiency of sequential processing depends on the percentage of records handled in one run and on the block size used.

If the file is frequently used or the transactions for single records can be saved up over a long period of time, a high percentage of records can be handled in one run and sequential processing is indicated. For sparse activity or if immediate processing is required, direct-access processing is indicated.

If the number of blocks is smaller than the number of affected records multiplied by 1.7, sequential processing is faster.

SERIAL DEVICES AND DISK DEVICES

The first devices developed were serial devices only, like card devices or magnetic tape devices.

As these devices are not very efficient for direct-access processing, it made a great change when disk devices were developed.

On disk devices, the reading mechanism can go directly to the track wanted and find the right record. Therefore we now have two types of devices, disk devices and serial devices (all others).

FORMS OF DATA ORGANIZATION

The records in every file are sequenced; either as they were entered, or according to one of their fields (key field), or they are numbered and ordered by a key external to their data content.

Records on disk devices have a cylinder, track, and record address: they are ordered either by a relative record number or located via an index which leads to their address.

Thus, we have the following organization and retrieval methods:

- entry sequence
- key fields
- keys

and with disk addresses only:

- relative record numbers
- indexes

These techniques can be combined. In VSE/VSAM, for example, they are all combined to give the user an optimal choice of organization.

Organization without Keys for Sequential Access

A file can have all its records in a certain sequence as, for example, the lines of a print file.

This file is mostly accessed sequentially for reading or printing, but it can also be accessed with the intention to find a certain record. The method used is a sequential search, which, however, is slow, since statistically half the file has to be scanned.

Organization with Keys for Sequential Access

If the sequential file has records with a key field, for example, a number for each record, we can sort the records in ascending order of their keys. It is faster to find a specific record if its key is known.

The key can also be one of the data fields, like the names of a personnel list, that can be ordered alphabetically. Figure 20 on page 45 shows how records can be sorted by different key fields.

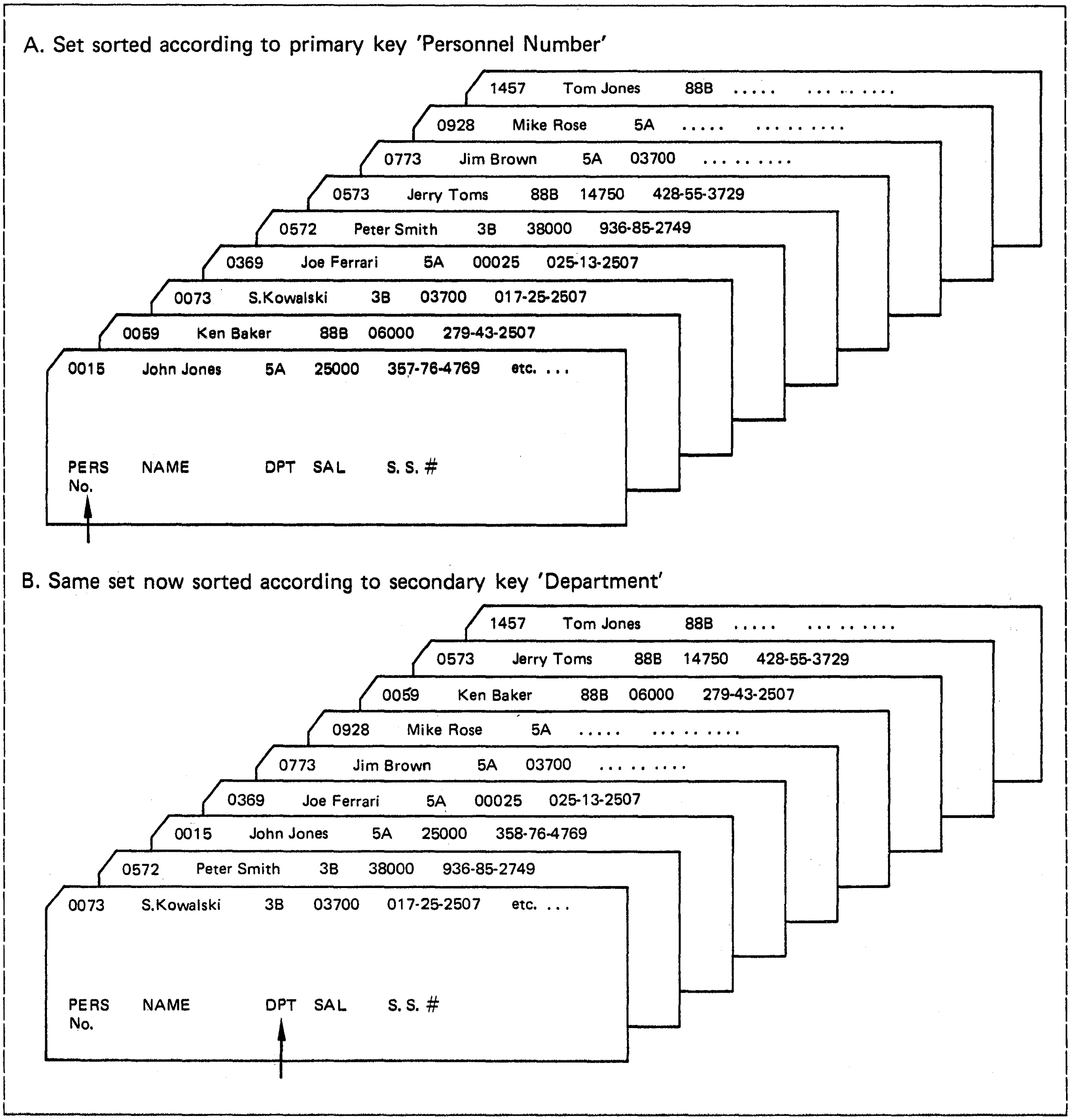


Figure 20. Records Sorted by Different Key Fields

If the keys are not unique, for example, if there are several persons with the same name, a secondary key field can help to make direct-access processing faster. The method to use such a key organization is comparable to the usage of a telephone directory. It is called a binary search because at each point the program decides if the key wanted is smaller, equal, or greater than the key under examination, and few records have to be examined.

Organization by Relative Record Number for Direct-Access Processing

For direct organization by relative record number, all records of a file must have the same length. Each record is assigned a unique number. To find the start address of a specific record within the file, this number is multiplied by the record length to find the relative byte address of the record within the file.

Organization by Indexes and Catalog for Direct-Access Processing

An index is an ordered list of keys matched to physical addresses. It is like an address book for records.

The records need not be ordered in key sequence. Only the index is ordered in key sequence and a binary search is done on the index to find the address of the record which is randomly placed on the disk pack volume. A catalog is an index to address and identify data files.

Alternate Indexes

As the records can have several fields, they can have several indexes, each using a different field as its key. For example, the personnel file can be ordered in one index by names, in the next index by addresses, and in the next by date of birth. Usually, one index is called the primary index and the others alternates or alternatives. This structure gives the user different aspects or logical views of the same physical file. He can use the first index for a quick listing of all employees whose names begin with a K, use the next index for a list of all employees from a certain region, and use the third index for a list of all employees over 40 or for all employees whose birthday is tomorrow.

Index Levels and Catalog

Indexes can also be constructed in several levels, the higher levels listing larger groups of records (logically) or of blocks (physically). In VSAM, all files are moreover listed and described in a master catalog.

Summary

Record Organization

Processing Types:	Sequential or Direct-Access
Device Types:	Serial or Disk
Organization Methods:	Entry Sequence Key Fields Keys With disk addressing (track or FBA block): Relative Record Number Indexes Index Hierarchy Catalog

HOW TO CHOOSE THE RIGHT DATA MANAGEMENT SUPPORT

This chapter shows the new and the old access methods and how they are supported in programming languages.

VSE used to offer a choice of access methods with special advantages in each. These access methods are still supported under VSE/Advanced Functions. They are SAM, DAM, and ISAM.

For new applications, however, the question is which level of data management service is required:

- SAM (Sequential Access Method), for simple sequential applications
- SAM in VSAM-managed space, for sequential applications with more flexibility
- VSAM (Virtual Storage Access Method), a full data management program with all modern facilities

SAM (SEQUENTIAL ACCESS METHOD)

The method can be used for files on all supported I/O devices.

SAM stores and retrieves the records of a file sequentially.

SAM can update a file in place.

SAM can process records or blocks.

One or two I/O areas are used, so that overlapping I/O and system processing is possible.

SAM is only efficient if most or all records are to be processed in sequential order. For details on SAM macros, see VSE/Advanced Functions Application Programming: Macro Reference and VSE/Advanced Functions Application Programming: Macro User's Guide.

SAM IN VSAM-MANAGED SPACE

This type of file is called a SAM ESDS (entry sequenced data set). It can be defined and accessed with SAM macros and with VSAM macros.

The SAM feature also serves to convert SAM files into VSAM files.

For details on this support, see Using the VSE/VSAM Space Management for SAM Feature.

VSE/VSAM (VIRTUAL STORAGE ACCESS METHOD)

VSE/VSAM is a complete file management system. It offers automatic space allocation and more than one index. This makes the user less dependent on specific devices and physical file organization. Space and files of a VSAM system are centrally managed in a VSAM catalog.

VSAM includes a set of service programs, Access Method Services, which can:

- Define and maintain VSAM files
- Load records into a VSAM file
- Build one or more alternate indexes for a file
- Copy and print files
- Create a backup copy of a VSAM file
- Recover from damage to data
- Convert a SAM or ISAM file to the VSAM format

In VSAM, a user may choose from three types of data set (file) organization:

- Key-sequenced data organization (KSDS)
- Entry-sequenced data organization (ESDS)
- Relative-record data organization (RRDS)

all of which offer further processing options. The VSE/VSAM books describe the details.

LIOCS AND PIOCS

The access methods SAM, DAM, and ISAM, accessible through Assembler macros or higher level languages, are collected under the name of LIOCS (Logical Input Output Control System) routines. They provide the programmer only with a logical view of his I/O control.

There are, however, also some Assembler macros available which are almost machine instructions. They may be used if the programmer chooses to control the input/output process on a more physical level. These macros are grouped under the name of PIOCS (Physical Input Output Control System).

LIOCS PROGRAMS WITH LIMITED SUPPORT

The access methods DAM and ISAM are still supported but no longer recommended for new files, since the new methods are more advantageous. Note, that DAM and ISAM do not support FBA disk devices.

DAM (Direct Access Method)

DAM was developed to handle direct-access processing on CKD disk devices. It supports all unblocked record formats. The records can have fixed-length keys, but they do not have to be physically ordered by those keys. Otherwise they are only identified by their track address. DAM uses only one I/O area.

DAM does not:

- work on FBA DASDs
- handle overflow records
- locate synonym records
- delete records
- handle a file that is not entirely on-line

The user's program has to handle these functions.

Please see also Appendix B, "Job Control for DAM Files" on page 101.

ISAM (Indexed Sequential Access Method)

Compared with DAM ISAM is a more sophisticated access method to handle files on CKD devices. Records have always fixed-length keys and are listed in an index.

ISAM handles file overflow automatically.

ISAM does not:

- handle records of variable length
- handle a file that is not entirely on-line

Please see also Appendix A, "ISAM Files" on page 97.

FILE SIZE CONSIDERATIONS

The fact that an ISAM file or a DAM file must be entirely on-line whenever it is processed restricts the file size to the on-line capacity of the system.

Reorganization of a file is only possible if double the file size can be kept on-line or a tape can be used as intermediate storage. File size may vary considerably if the file has a high volatility, that is, if many records are added or deleted in the normal use of the program. VSAM space management offers good support for this case.

```
Summary

VSE Data Management Support Level


   Lowest level: SAM  ---  accessed with SAM macros
                      ---  uses: - data fields as keys
                                 - entry sequence

   Second level: Sam in VSAM-Managed Space
                      ---  accessed with SAM or VSAM macros

   Third level:  Full VSAM Support
                      ---  accessed with VSAM macros
                      ---  uses: -  entry sequence
                                 -  keys, relative numbers,
                                    indexes
                                 -  a central catalog to list
                                    files
                                 -  AMS to service files

   Limited Support:  DAM and ISAM

     LIOCS control:  SAM, DAM, and ISAM macros
     PIOCS control:  Low level Assembler macros
```

OVERVIEW OF ACCESS METHODS IN EACH LANGUAGE

An overview of the access methods supported in the various programming languages is given in Figure 21 on page 53.

Programming Language	Recommended A/M: SAM	SAM ESDS	VSAM	Supported A/M: DAM	ISAM
COBOL:	YES	YES	YES*	YES	YES
VS FORTRAN:	YES	YES	YES**	YES****	NO
RPG II:	YES	YES	YES	YES	YES
PL/I:	YES	YES	YES	YES***	YES
Assembler:	YES	YES	YES	YES	YES

A/M Stands for Access Method
*) COBOL 5746-CB1 only.
**) Only ESDS and RRDS
***) Either with keys or user-specified relative record numbers
****) The user specifies the relative record numbers in the file

Figure 21. Access Methods Supported in Programming Languages

Labels

PART 2: LABELS

This part of the book is a guide for application programmers who will use SAM, DAM, ISAM, or VSAM macros to write or process IBM-standard file labels or write PIOCS routines for user-standard or non-standard file labels.

To understand this part of the book, the reader should be familiar with computer systems and basic programming concepts and techniques.

Labels

LABELS OVERVIEW

The following sections describe the items:

- Volume and file labels
- Volume initialization
- File label specification
- Label processing
- Label area

VOLUME AND FILE LABELS

The following types of labels are supported under VSE/Advanced Functions:

- Volume labels (identifying the volume)

- File labels (identifying each file on the volume)

The following label formats can be used for:

Volumes:

- Standard Labels

Files:

- Standard Labels

- User-Standard Labels

- Non-Standard Labels

- Unlabeled

STANDARD LABELS: This type of label has a fixed length and contains specific fields of information. Four types of standard file labels are supported for disk volumes (Format-1,2,3,and 4). In case of tape labels two different formats are used, depending on the code (EBCDIC, or ASCII) in which the data is written.

USER-STANDARD LABELS: These labels are a variation of standard labels. That is, like standard labels they have a fixed length. However, they have only a partially fixed format. The two kinds of user-standard labels are:

- User header label UHLn
- User trailer label UTLn

NON-STANDARD LABELS (EBCDIC TAPES ONLY): Non-standard labels do not conform to the standard or user-standard specifications. They may be any length containing whatever you desire.

UNLABELED VOLUMES AND FILES (TAPES ONLY): Unlabeled files only require the proper positioning of the tape reel for reading/writing the tape-marks or data records. The exact format of each type of label and the contents of their fields is described in the VSE/Advanced Functions Application Programming: Macro Reference manual.

Figure 22 gives an overview of the various kinds of labels regarding their prefixes on diverse device types.

Type of Label	Prefixes at the Begin of each Label for: Disk	Diskette	Tape
Volume: Standard	VOL1	VOL1	VOL1
File: IBM-standard	file-name	HDR1	HDR1, EOF1, EOV1, HDR2, EOF2, EOV2
File: User-standard	UHLn, UTLn*	no	UHLn or UTLn*
File: Non-standard	no	no	EBCDIC only
Vol.and F.: Unlabeled	no	no	yes

VOL1 .... Volume label
HDR1,2... Header label
EOF1,2... End of file label
EOV1,2... End of volume label
UHL1 .... User header label
UTL1 .... User trailer label
*) SAM (and DAM) macros only

Figure 22. Various Kinds of Labels

On disk and diskette volumes, IBM-standard volume and file labels with fixed length format are required. New disk volumes must be identified (labeled) at the system environment that will use them. IBM-diskette volumes come already labeled from the factory.

On disk and tape, provision is made for additional volume labels (VOL2 to 8). They are not supported under VSE.

On disk and tape, user-standard file labels are supported as well as IBM-standard file labels. A user routine is necessary to handle them. VSAM (and ISAM) do not support them.

Tape volumes and files may be either labeled or unlabeled. If a labeled file is to be on an unlabeled volume, the routine opening the file asks the operator for a volume serial number. If he just presses ENTER, the system generates a volume label of its own.

Tape file labels may be IBM-standard, with or without user-standard labels in addition, or of non-standard type. If files with non-standard labels or unlabeled files were written on a volume with volume labels, those are destroyed. Therefore, standard labeled files and others must be kept on separate volumes.

VOLUME INITIALIZATION

When files of data records are to be stored on disk, diskette, or tape, that volume must be labeled for future identification.

A volume label (and, for disk volumes, the VTOC label) is written on the volume when the volume is initialized. Initialization itself differs depending on the device type:

Disks are initialized:

by the Device Support Facility program (DSF)

Diskettes are initialized:

in the factory or via VSE/DITTO

Tapes are initialized:

by utility, automatically or via VSE/DITTO

For disk initialization, please see Device Support Facilities User's Guide and Reference. The initialization includes writing of the VTOC label.

For diskette re-initialization or tape initialization, see VSE/Data Inter-file Transfer, Testing, and Operations (DITTO) Program Reference and Operations manual.

For tape initialization, see VSE/Advanced Functions, System Utilities.

For details on VTOC initialization, see "VTOC, Volume Table Of Contents" on page 77.

FILE LABEL SPECIFICATION

You supply the information for IBM-standard file labels to the system by the job control statements **OPTION, EXTENT, and DLBL or TLBL** and by operands in file definition macros like **DTFxx or ACB** in the application program. For additional file labels, the **DTFxx** macros in your program must point to your own label processing routine.

Please find a more detailed discussion in "How to Specify Label Information" on page 65.

IBM-standard file labels are stored in a label area by the job control program.

LABEL PROCESSING

Standard labels defined via job control statements (and stored in an area called label area) are written on the volume when the file is opened for output or they are checked when it is opened for input. Figure 23 on page 61 shows the different sources from which the label information is retrieved.

The label information is:

- Specified in job control statements and stored in the label area when the statements are processed.
- Contained in the program. The program defines the file.
- On the volume. The label information identifying the file is either written or checked using the information from the label area.

```
              LABEL
           INFORMATION
              from
           ┌────────┐
           │  DLBL  │
           │ EXTENT │────────┐  information
           │  TLBL  │        │  stored in
           └────────┘        V
                         LABEL AREA
                       ┌─────────────┐
                       │             │
                       │ containing  │
                       │label records│
                       │             │
             FILE      └─────────────┘
            defined           │
        in the PROGRAM        │              VOLUME
           ┌────────┐         │            ┌────────┐
           │ DTFxx  │         V            │        │
           │ ACB    │────────────────────> │ LABEL  │
           │ OPEN   │                      │        │
           └────────┘     check/write      └────────┘
                       label information
```

Figure 23. Label Information Relationship

LABEL AREA

The label area is used to hold the information defined via DLBL, EXTENT and TLBL job control statements.

You define this area at IPL time with the **DLA** command. If the DLA command is omitted at IPL, the system generates a default label area.

DLA Command (IPL)

The DLA command has the following operands:

```
DLA  NAME= ,UNIT= ,VOLID= ,DSF=
     ,CYL= ,NCYL= ,BLK=,NBLK=
```

Labels

Under CYL and NCYL (BLK and NBLK), you specify the beginning and the length of the label area.

The default size of the label area is as follows:

3340,3375	3 cylinders
3350	1 cylinder
Other CKD devices	2 cylinders
FBA devices	200 blocks

You can change the size and place of the label area by issuing a new DLA command during IPL.

The Label Sub-Areas

There are three areas where the label information can be stored depending on which job control option is being active:

- Temporary partition sub-area containing labels which follow the OPTION USRLABEL job control statement. These labels are deleted at end of job.

- Partition standard sub-area containing labels which are submitted after the OPTION PARSTD has been made active. These labels remain effective to all subsequent jobs in the current partition.

- System standard sub-area containing labels which are submitted after the OPTION STDLABEL has been made active. These labels remain effective to all subsequent jobs in all partitions.

AREA	OPTION	EFFECTIVE FOR	NOTES
Temporary partition sub-area	USRLABEL	the job	1
Partition standard sub-area	PARSTD	the partition	2
System standard sub-area	STDLABEL	the system	3

(1) The temporary labels are erased at end of job.
(2) This option directs the labels to the permanent label area for the corresponding partition.
(3) The system standard label area contains label information for all partitions.

Each option can be overwritten by an OPTION statement with the same options. This means a specification of OPTION PARSTD erases all labels specified previously under that option. And as OPTION USRLABEL is the default, also any new DLBL or EXTENT statement in a following job step erases all temporary labels of that partition.

To avoid this effect, existing PARSTD and STDLABEL labels can be saved by adding new ones in the form:

// OPTION PARSTD=ADD or // OPTION STDLABEL=ADD

followed by DLBL and EXTENT statements. As the search for a file label is always done in the sequence

USRLABEL ---> PARSTD ---> STDLABEL

USRLABEL labels override PARSTD labels and STDLABEL labels for one job, and PARSTD labels override STDLABEL labels in one partition. You can have different files for each partition with the same file-name (DTF-name) for all partitions. This is used for system input or output files without new DLBL and EXTENT statements each time a compilation or linkage editing function is performed.

Labels

HOW TO SPECIFY LABEL INFORMATION

The following sections describe how to specify disk, diskette, and tape file labels under the following headings:

- "DLBL and EXTENT Statements for Disk and Diskette"
- "TLBL Statements for Tape Files"
- "DTF and ACB Macros"
- "User-Standard Label Routine"
- "Non-Standard Label Routine"
- "Processing Unlabeled Files"
- "American National Standard Labels"

DLBL AND EXTENT STATEMENTS FOR DISK AND DISKETTE

You define new IBM-standard file labels or give information to check existing volume and IBM-standard file labels for disk via DLBL and EXTENT job control statements.

Note, that a detailed description of these statements can be found in the VSE/Advanced Functions System Control Statements manual.

The **DLBL statement** for disk has the following operands:

```
// DLBL filename,file-id,date,codes
       ,DSF,BUFSP= ,CAT=
       ,BLKSIZE= ,CISIZE=
       ,DISP= ,RECORDS= ,RECSIZE=
```

For diskette, the DLBL statement has only the following operands:

```
// DLBL filename,file-id,date,DU
```

The DLBL statement identifies the file by the same (file)name which is used for the DTFxx macro in the program. All other operands are optional, that is, default values are supplied.

Beside others the **EXTENT statement** has the following operands:

```
// EXTENT logical-unit
         ,volume-serial-number
         ,type
         ,sequence-number
         ,relative-track|block
         ,number-of-tracks|blocks
         ,split-cylinder-track
```

The EXTENT statement names the logical unit specified in the corresponding ASSGN statement and describes the extent for creating or checking IBM-standard file labels and for checking the "volume serial number" on the volume where the extent belongs.

Some EXTENT fields are optional, depending on the file definition macro support used. For a SAM input file on a single volume, the EXTENT statement may be omitted if the DTF operand DEVADDR=SYSxxx is specified. The DEVADDR operand can be used to define a logical unit if no EXTENT statement is provided. For example, single volume input files do not require an EXTENT statement. However, if one EXTENT statement is supplied, all extents created for the file must be specified if they are to be accessed.

"Relative track or block" identifies the beginning of the extent. For CKD devices it must be calculated for each device type using the cylinder-track address. The CKD devices have the following numbers of tracks per cylinder available for file extents:

Device	Number of Tracks per Cylinder
2311	10
2314, 2319	20
3330, 3333	19
3340	12
3350	30
3375	12
3380	15

The first available track is track 1.

Therefore, if the address is cylinder 15, track 7 on a 3350, the relative track is 15 x 30 + 7 = 457.

For diskette, the EXTENT statement has only the following operands:

```
// EXTENT logical-unit,volume-serial-number,type
```

as the extent limits are determined automatically from the space available on the diskette. Here, all EXTENT fields are optional.

Job Control Examples for Disk and Diskette Files

1. The first is an example of label specifications for a SAM file of two extents on disk which has to be updated.

```
// JOB UPDATE
// ASSGN SYS005,190
// ASSGN SYS007,191
// DLBL DISKIN,'SEQUENTIAL DISK FILE',1985/060,SD
// EXTENT SYS005,111111,1,0,1600,300
// EXTENT SYS007,222222,1,1,0031,450
// DLBL DISKOUT,'SEQUENTIAL DISK FILE',1985/060,SD
// EXTENT SYS005,111111,1,0,1600,300
// EXTENT SYS007,222222,1,1,0031,450
// EXEC UPDATE
/&
```

In the example above one extent is used for input and output.

2. The second is an example of job control for a diskette file of three extents, all on different volumes.

```
// JOB USE DISKETTE FILE
// ASSGN SYS004,060
// DLBL OUT,'DISKETTE',1985/060,DU
// EXTENT SYS004,987652,1
// EXTENT SYS004,987653,1
// EXTENT SYS004,987654,1
// EXEC USEDIK
/&
```

TLBL STATEMENTS FOR TAPE FILES

You create a new IBM-standard file label or give information to check existing volume and IBM-standard file labels for tape with a TLBL job control statement.

For EBCDIC:

```
// TLBL filename,file-id,date
        ,file-serial-number
        ,volume-sequence-number
        ,file-sequence-number
        ,generation-number
        ,version-number
        ,DISP=NEW|OLD|MOD
```

For ASCII:

```
// TLBL filename,file-id,date
        ,set-identifier
        ,file-section-number
        ,file-sequence-number
        ,generation-number
        ,version-number
        ,DISP=NEW|OLD|MOD
```

All TLBL operands other than filename are optional, that is, a default is supplied.

If you use a labeled multi-volume file and you want to start the processing at a volume other than the first, you supply TLBL information as follows:

- 'file-serial-number' (EBCDIC), 'set-identifier' (ASCII)

  Contains the volume serial number of the first reel of the file.

- 'volume-sequence-number' (EBCDIC), 'file-section-number' (ASCII)

  Contains a four digit decimal number specifying the volume of a multi-volume file at which the processing should be started.

This will properly check the HDR1 label. However IOCS issues a message when it detects that the indicated volume is not the first one when it checks the VOL1 label (different volume serial numbers). You may bypass this condition and continue processing.

For a more detailed description of the various fields in the TLBL statement please refer to the VSE/Advanced Functions System Control Statements manual.

DTF AND ACB MACROS

DTF and ACB macros are used to define characteristics of files in your program.

Types of DTFxx Macros

Depending on the I/O macro support used, different file definition macros must be specified.

Recommended Level of Organization:	Definition Macro
SAM for disk	DTFSD, DTFDI*
SAM for diskette	DTFDU, DTFDI*
SAM for tape	DTFMT, DTFDI*
SAM in VSAM Managed Space for disk	DTFSD, DTFDI*
VSAM for disk	ACB**
PIOCS for disk, diskette, or tape	DTFPH***

Restricted Supported:	Definition Macro
DAM for disk	DTFDA
ISAM for disk	DTFIS

Figure 24. The Various DTFXX Macros

*) DTFDI gives device independence. The files may be assigned to unit record, tape, or disk devices at execution time.

**) VSAM does not use DTFxx macros. Label information is specified by a utility program, Access Method Services, and by job control statements. Labels describe VSAM data spaces; files are described in the VSAM catalog. A VSAM processing program is connected to a file through an ACB (Access Method Control Block) instead of a DTF.

***) Using PIOCS macros in your program to handle a file with IBM-standard labels, this file must be defined by a DTFPH macro.

Use of DTFxx Macros to Define Label Information

The following DTF operands handle the various types of labels.

Labels

Device	Class of Label	Operands		
Disk	IBM-standard:	—	—	XTNTXIT=name
	User-standard:	—	LABADDR=name	—
Diskette	IBM-standard:	—	—	—
Tape	IBM-standard:	FILABL=STD	—	—
	User-standard:	FILABL=STD	LABADDR=name	—
	Non-standard:	FILABL=NSTD	LABADDR=name	—
	Unlabeled:	FILABL=NO	—	EOFADDR=name

Figure 25. Labels Defined in Macros

USER-STANDARD LABEL ROUTINE

If you want to use user-standard file labels as well as the IBM-standard labels you must

- create your own label routine to write or check these user-standard labels. In this routine you build an 80-byte label with the first four bytes being UHL1 or UTL1. You load the symbolic address of this label into register 1 and then issue a LBRET macro to return control to IOCS.

- specify LABADDR in the DTFxx macro in your program to indicate to IOCS to branch to your label routine after processing the IBM-standard labels.

On disk, IOCS then establishes the first track of the first data area extent as user-standard label area, with extent sequence 0 and extent type X'40'. Please read "EXTENT Fields" on page 79 for more information. IOCS then writes your user-standard file label(s) on the volume.

On tape, the user standard labels follow the IBM standard labels.

The only operand of the LBRET macro is a number. You may define the following numbers: 1, 2, or 3.

- A LBRET 3 macro permits IOCS to update a label on the device and points to the next one (not for tape).

- A LBRET 2 macro permits reading the next label.

- A LBRET 1 macro terminates the processing of user-standard labels.

For more information on the user-standard label routine and the LBRET macro, see VSE/Advanced Functions Application Programming: Macro User's Guide, and VSE/Advanced Functions Application Programming: Macro Reference.

When PIOCS macros are used for a file and the DTFPH macro is specified with the LABADDR operand, only header labels are checked but no trailer labels.

NON-STANDARD LABEL ROUTINE

Non-standard labels are used for EBCDIC-code tape files only.

You supply the information for creating and checking non-standard header and trailer labels in a label routine in your application program. The address of this label routine in the program is specified in the DTFxx entry LABADDR=name.

In this label routine, you issue PIOCS macros to read or write the labels. You set up a command control block by issuing a CCB macro, and write a channel program consisting of CCWs.

You define your label read-in or read-out area. At the end of your routine, you return control to IOCS by issuing a LBRET 2 macro.

PROCESSING UNLABELED FILES

Only tape files may be unlabeled.

If DTFxx FILABL=NO is specified or the operand is omitted, IOCS assumes that a file does not contain labels, regardless of what is currently written on the tape. IOCS merely reads or writes tape marks or data.

Unlabeled Input Files

IOCS assumes the end of the input file when it reads the tape mark that follows the last data record. It branches to your end-of-file routine specified in DTFxx EOFADDR=name. In this routine, you check for an end-of-file or an end-of-volume condition, normally by requesting a reply from the operator.

End-of-file, should be handled by your program according to your end of data requirements.

On end-of-volume, your program must issue a FEOV macro. Then IOCS updates the active drive number if an ASSGN statement has specified an alternate drive (ALT) for the file and switches to the alternate drive. Else a message is issued to the operator.

If multiple files on the same volume are to be read in sequence, the DTFxx entry REWIND=NORWD must be specified for each file. In this case the tape is positioned correctly each time for the next file to be opened.

To position the tape for the first file on the reel, the programmer can include a CNTRL REW macro or a job control MTC REW statement or the operator can position the tape at the load point.

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 tape mark) may be read backward.

Unlabeled Output Files

FIRST RECORD: IOCS writes a tape mark as the first record (unless the user specified DTFxx TPMARK=NO) starting at the location where the tape is positioned. Thus if the tape has been rewound to the load point, IOCS warns the operator if it finds a volume label there before it writes the tape mark or data over any label(s) that is (are) already on the tape.

LAST RECORD: When a CLOSE macro is issued after all records for a file have been processed, IOCS writes two tape marks after the last block of data records. If the reflective marker at the end of the tape is found before the end of the output file or if a FEOV macro is issued, IOCS writes one tape mark.

MULTI-VOLUME FILE: If the next I/O macro after the reflective marker at the end of the tape is a CLOSE, an end-of-file condition exists. If, however, the next macro is a PUT or a FEOV (forced end-of-volume) macro, an end-of-volume condition exists. On end-of-volume, IOCS writes a tape mark and switches to the alternate drive. If no alternate drive has been specified, the operator is requested to mount a new volume. IOCS positions the new tape at the load point and writes a tape mark, unless DTFxx TPMARK=NO has been specified.

MULTI-FILE 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 stops behind the file just written.

AMERICAN NATIONAL STANDARD LABELS

VSE processes tape files written in the American National Standard Code for Information Interchange (ASCII), in addition to 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 titled American National Standard labels. ASCII files may be unlabeled or labeled with American National Standard standard or user-standard labels. Non-standard 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 differences between the ASCII tape volume label and the EBCDIC tape volume label fields are as follows:

Field Number	EBCDIC Name	ASCII Name	Hex Displacement EBCDIC	ASCII
4	Security byte	Accessibility	A	A
7	Reserved	Owner ID	1F	25
8	Owner ID	Reserved	29	33
9	Reserved	Standard byte	33	4F

Additional volume labels (VOL2-VOL8) are tolerated 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.

Labels

The default for the version number in the American National Standard file label is 00; the EBCDIC label version number defaults to 01.

EOV labels on an EBCDIC tape file are followed by one tape mark; on an ASCII tape file these labels are followed by two tape marks.

When an ASCII file is processed, IOCS translates the labels from ASCII into EBCDIC (on input) and from EBCDIC into ASCII (on output). There are two translate tables in the SVA for this purpose. Their address is stored in SYSCOM X'74 to 77'.

Tapes to be used for ASCII files may be initialized with American National Standard standard labels by the IBM-supplied program Initialize Tape. For more information see <u>VSE/Advanced Functions System Utilities</u>.

WHERE LABELS ARE PLACED

The following sections describe where the volume or file labels can reside on a:

- Disk Volume
- Diskette Volume
- Tape Volume

DISK VOLUME

Disk Volume Organization

Figure 26 on page 76 shows the relationship between the volume label, the VTOC, the labels stored in the VTOC, and the data extent on a disk volume. It pictures the way of how to find a certain extent on a disk volume, like:

```
Volume Label
     |
     └──────────> VTOC
                   |
                   └────> File Label
                               |
                               └────> Data Extent
```

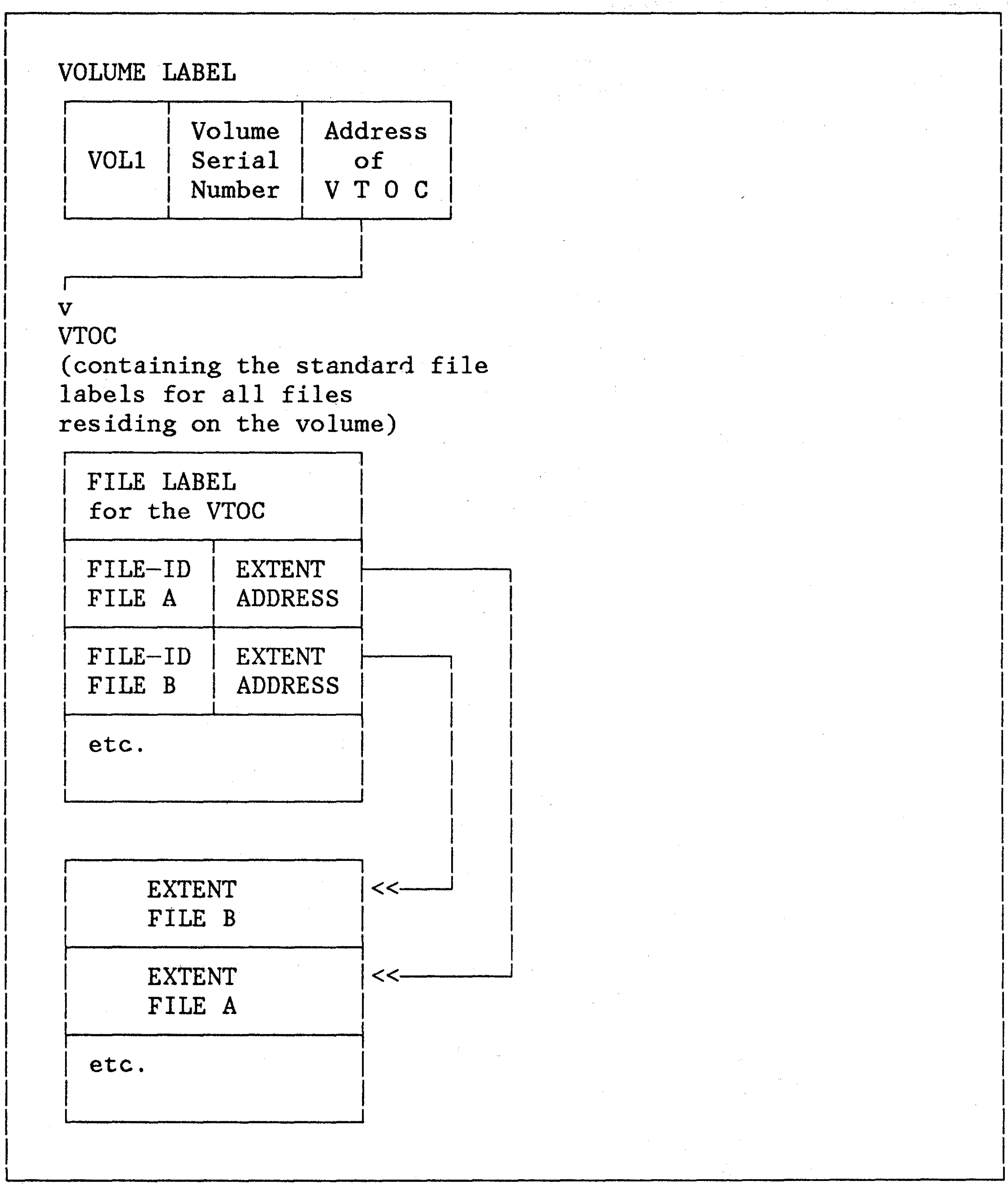


Figure 26. Disk Volume Organization

Place of Disk Volume Labels

On CKD volumes, the volume label always starts on cylinder 0, track 0, record 3. The first two records contains IPL records on a system volume and zero on other volumes.

On FBA volumes, the volume label is the first record in FBA block number one.

VTOC, Volume Table Of Contents

The IBM-standard file labels for all files on the volume are contained in an area called the Volume Table Of Contents (VTOC).

VTOC LOCATION: The VTOC may be placed anywhere on the volume, except in the system area on a SYSRES device or in the areas reserved for alternate use.

Location and length of the VTOC are specified at volume initialization. The address of the VTOC is saved in the volume label.

VTOC CREATION: The VTOC is created during the volume initialization with the Device Support Facility (DSF). The disk address of the VTOC is stored in the volume label.

The initialization program provides a 44-byte key field and a 96-byte data field for each label in the VTOC and fills them with binary zeros. The label information is written into these fields when each file is created. The initializing program also writes the label for the VTOC itself as the first record.

VTOC CONTENTS: The first label defines the VTOC itself. This label is called the format-4 label.

The second label in the VTOC is a label marked format-5, that is not used by VSE/Advanced Functions.

The following labels (format-1, format-3) in the VTOC define files or a VSAM data space on the volume. The labels are written in the order in which the corresponding EXTENT statements were processed and the files or VSAM data spaces were created. Each format-1 label can identify up to 3 separate areas (extents) on the volume for one file.

Whenever a file or a VSAM data space occupies more than 3 extents on the volume, an additional label, namely the format-3 label, is used.

The format-2 label is required for ISAM only.

VTOC FORMAT: Figure 27 on page 78 shows the general format of a disk VTOC with a VTOC label and all the IBM-standard file labels together.

VTOC-Label	not used	File-A Label	File-B Label	File-B addition. Label	File-C Label	....
Format-4	Format-5	Format-1	Format-1	Format-3	Format-1	

|<——— V T O C ———>|

Note: The format-2 label is used by ISAM

Figure 27. General VTOC Format for a Disk Volume

FORMAT-1 LABEL: Figure 28 gives an overview of a format-1 label. The data extents of a file are identified and located with the format-1 labels. The information about up to 3 extents can be held in one format-1 label. If a file has more than 3 extents, the use of an additional label is required. In this case a format-3 label is created by the system.

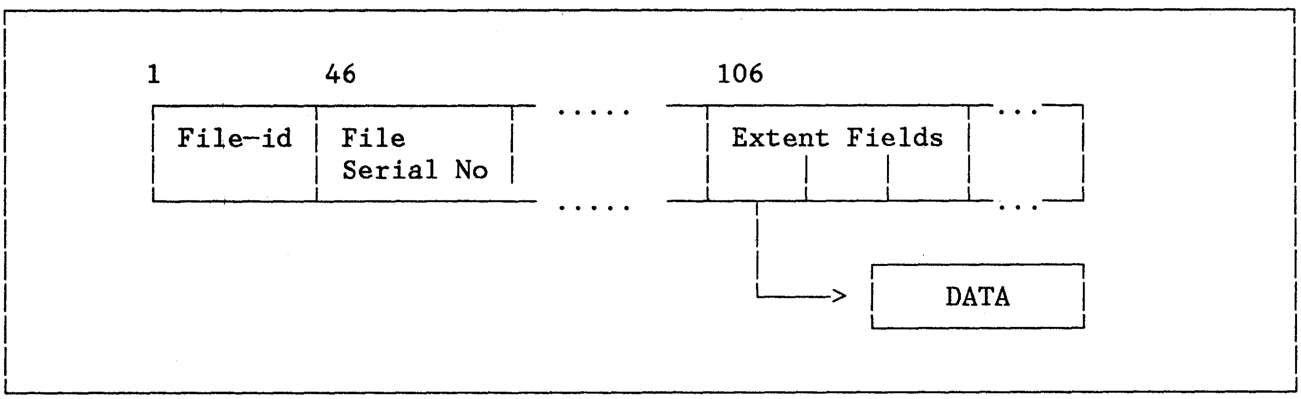


Figure 28. Format-1 Label Overview

EXTENT FIELDS contain the following information to define the extent:

- Extent type

  defines, for example:

  - that no extent is specified ... type X'00'
  - that this extent is used as a user standard label area ... type X'40'
  - that this data area is with split cylinder ... type X'80'

- Sequence number

  determines the order in a multi-extent-file.

- Extent start address

  determines the start address of the data extent.

- Extent end address

  determines the end address of the data extent.

Figure 29 shows the contents of one extent field.

106	107	108	112
Extent Type	Sequence No.	Start Address	End Address

Figure 29. Extent Field in a Format-1 Label

VTOC PLACEMENT: For **FBA devices** the VTOC can start on any block except blocks 0, 1, and system area on SYSRES. For each label on FBA, there is one 3-byte record definition field (RDF) and for the control interval (CI) one 4-byte control interval definition field (CIDF) to be counted, so that a calculation

CI-Size minus 4 divided by 143

will yield the number of labels that fit in a given control interval. This number is written in the volume label at displacement 1D.

The VTOC of an FBA volume can contain 3 to 999 file labels.

On FBA devices, the following blocks are available for VTOC placement:

3310: 2 to 126,000
3370: 2 to 557,984

Alternate blocks for FBA devices are assigned by the device support facility (DSF).

For **CKD devices**, the cylinder distribution is the following:

Device	VTOC on Cyls.	Altern. Track Cyls.
2311,2314,2319	0 - 199	200 - 202
3330 Mods. 1+2	0 - 403	404 - 410
3330 Mod. 11	0 - 807	808 - 814
3340 Mod. 35	0 - 347	348
3340 Mod. 70	0 - 695	696 - 697
3350	0 - 554	555 - 559
3375	0 - 945	946
3380	0 - 884	885

The limits for the IBM 3330 apply also to the 3333 and 3350 in 3330-1 mode.

The limits for the IBM 3330-11 also apply to the 3350 in 3330-11 mode.

The limits for the IBM 3340 also apply to the 3344.

Cylinder 0, track 0 of each volume contains the following records:

- 0: Alternate track assignment (track descriptor)
- 1 and 2: IPL records for SYSRES volumes, else zeros
- 3: Volume label(s)
- The remainder to end of track 0: VTOC, if this location was specified at initialization (and it is not a SYSRES), else not used.

For ISAM files, if the prime data area takes several volumes, 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.

Place of Disk User-Standard File Labels

User-standard labels on disk are written on the first track of the first extent allotted for data on CKD devices or in the first control interval (CI) for data on FBA devices. Therefore, the first extent on a CKD device must be a minimum of two tracks. Your data records then start with the second track in the extent, whether the labels require a full track or not.

Disk Volume Layout Examples

This section shows the arrangement of labels on volumes, cylinders, and tracks as IOCS or VSAM write them on output files and expect them on input files.

Each of the following illustrations shows:

- How the space on the volume is used (cylinder distribution).
- The track with volume labels and VTOC (cylinder 0, track 0).
- The extent fields (displacement X'69'-X'73') of each format-1 and format-4 label in the VTOC, showing the start and end address of the extent.

Figure 30 on page 82 shows one data file on one volume with standard VTOC location (cylinder 0, track 0, record 4 to end of track) on an IBM 3330 Model 1 or 2.

1. Cylinder Distribution

Cyl.0	Cyl.1-403	Cyl.404-410
Volume Label(s) and VTOC	Data	Alternate Tracks

2. Cylinder 0 Track 0

Records:

0	1 - 2	3	VTOC		
Track Descri-ptor	Zeros	Volume Label(s)	VTOC Label	Format-5 Label*	IBM Std File Label(s)

3. Extent Fields in File and VTOC Labels, Displacement X'69'-X'73'

	Extent Type	Seq. Number	Start Address Cyl.	Start Address Tr.	End Address Cyl.	End Address Tr.
VTOC Label	1	0	0	0	0	18
Std.File Label**	1	0	1	0	403	18

Notes:
*) The format-5 label is not used by VSE.
**)The file label here specifies a single extent for data records which fills cylinders 1 to 403.

Figure 30. Disk Volume Layout: One File, VTOC in Standard Place

Figure 31 on page 83 shows one file on one volume with a VTOC of one track on cylinder 100, track 1 on an IBM 3330 Model 1 or 2.

1. Cylinder Distribution

Cyl.0 Track 0	Cyl.0 Track 1 to Cyl.99 End	Cyl.100 Track 1	Cyl.100 Track 2 to Cyl.403 End	Cylinders 404 - 410
Volume Label(s)	Data	VTOC	Data	Alternate Tracks

2. Cylinder 0 Track 0

Records: 0	1 - 2	3	to End of Track
Track Descri-ptor	Zeros	Volume Label(s)	Unused

3. Cylinder 100 Track 1

Records: 0	VTOC		
Track Descri-ptor	VTOC Label	Format-5 Label*	IBM-Standard File Label(s)

4. Extent Fields: File and VTOC Labels, Displacement X'69'-X'73'

	Label Type	Seq. Number	Start Address Cyl.	Start Address Tr.	End Address Cyl.	End Address Tr.
VTOC Label	1	0	100	1	100	1
Standard File Labels**	1	0	0	1	99	18
	1	1	100	2	403	18

Notes:
*) The format-5 label is not used by VSE.
**)The file labels here specify two extents for data records.

Figure 31. Disk Volume Layout: One File, VTOC in Specified Place

Labels

Figure 32 shows a disk file which fills two volumes and partly a third one.

1. Cylinder Distribution

	Cyl.0	Cyl.1-403	Cyl.404-410
Volumes 1 and 2	Volume Label(s) and VTOC	Data	Alternate Tracks
Volume 3		1 - 100: Data	

2. Cylinder 0 Track 0 on Each Volume

Records: 0	1 - 2	3	VTOC		
Track Descri-ptor	Zeros	Volume Label(s)	VTOC Label	Format-5 Label*	IBM Std File Label(s)

3. Extent Fields: File and VTOC Labels, Displacement X'69'-X'73'

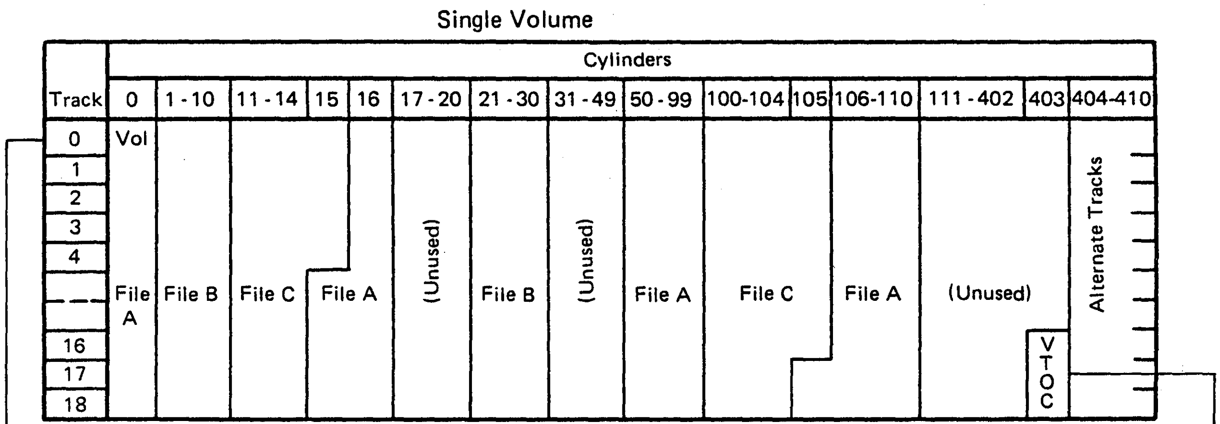
	Label Type	Seq. Number	Start Address Cyl.	Start Address Tr.	End Address Cyl.	End Address Tr.
VTOC Label: All Volumes	1	0	0	0	0	18
Std.File Labels:						
Volume 1 **	1	0	1	0	403	18
Volume 2	1	1	1	0	403	18
Volume 3	1	2	1	0	100	18

Notes:
*) The format-5 label is not used by VSE.
**)The file labels here specify a single extent for each volume.

Figure 32. Disk Volume Layout: Three-Volume File, VTOCs in Standard Place

Figure 33 shows a multi-file disk volume with multi-extent disk files on a 3330 Model 1 or 2.

1. Cylinder Distribution



2. Cylinder 0 Track 0

Records:	0	1 – 2	3	4 to End of Track
	Track Descriptor	Zeros	Volume Label	Unused

3. Cylinder 403 Track 16 (Beginning of VTOC)

Records:	0	1	2	3	4	5	6	7...
	Track Descriptor	VTOC Label	Format-5 Label*	File Label FileA	Contin. Label FileA	File Label FileB	File Label FileC	...

Figure 33 (Part 1 of 2). Disk Volume Layout: Multi-File Volume

4. Extent Fields: File and VTOC Labels, Displacement X'69'-X'73'

File		Label Type	Extent Seq. Number	Start Address Cyl.	Start Address Tr.	End Address Cyl.	End Address Tr.
A	File Labels :	1	0	0	1	0	18
		1	1	15	5	16	18
		1	2	50	0	99	18
A	Contin.Label:	1	3	105	17	110	18
B	File Labels :	1	0	1	0	10	18
		1	1	21	0	30	18
C	File Labels :	1	0	11	0	15	4
		1	1	100	0	105	16
—	VTOC Label :	1	0	403	16	403	18

Note:
*) The format-5 label is not used by VSE.

Figure 33 (Part 2 of 2). Disk Volume Layout: Multi-File Volume

Figure 34 on page 87 shows disk files with user-standard labels on an IBM 2311.

```
1. Cylinder Distribution

                Volume 1                                Volume 2
Cyl. |0-49|50-159|160-198|199|200-202|  |0-150|151-198|199|200-202|
     |File| File | File  | V | Alt.  |  |File | File  | V | Alt.  |
     | A  |  B   |  A    |TOC|  Tr.  |  | A   |  C    |TOC|  Tr.  |
     |____|______|_______|___|_______|  |_____|_______|___|_______|


2. Cylinder 0

     Track              Cylinder 0  (for both volumes)
                        ______________________________
       0               |Zeros|VOL1|                   |
       1               | File A UHL1-UHL8, UTL0-UTL7  |
       2               |       File A data            |
                       |            .                 |
                       |            .                 |
                       |            .                 |
       9               |       File A data            |
                       |______________________________|


3. First Cylinders of Extents:

              Cylinder 50                       Cylinder 151
Track   ______________________________     _______________________
  0    |File B UHL1-UHL8, UTL0-UTL7  |    |     File C data*      |
  1    |       File A data           |    |          .            |
       |            .                |    |          .            |
       |            .                |    |          .            |
  9    |       File A data           |    |     File C data       |
       |_____________________________|    |_______________________|


             Cylinder 160
Track   _____________________________
  0    |       File A data           |
  1    |            .                |
       |            .                |
       |            .                |
  9    |       File A data           |
       |_____________________________|

Note: *) File C has no user-standard labels specified.
```

Figure 34 (Part 1 of 2). Disk Volume Layout: Files with User-Standard Labels

4. VTOC on Cylinder 199

Volume 1

Track Descriptor	VTOC Label	Format-5 Label*	IBM-Standard File Labels		
			File A	File B	

Volume 2

Track Descriptor	VTOC Label	Format-5 Label*	IBM-Standard File Labels		
			File A	File C	

5. Extent Fields: File and VTOC Labels, Displacement 69 - 73

<u>Volume 1:</u>

	Label Type	Extent Seq. Number	Start Address Cyl.	Start Address Tr.	End Address Cyl.	End Address Tr.
A File Labels :	blank**	0	0	1	0	1
	1	1	0	2	49	9
	1	2	160	0	198	9
B File Labels :	blank**	0	50	0	50	0
	1	1	50	1	159	9
— VTOC Label :	1	0	199	0	199	9

<u>Volume 2:</u>

	Label Type	Extent Seq. Number	Start Address Cyl.	Start Address Tr.	End Address Cyl.	End Address Tr.
A File Labels :	blank**	0	0	1	0	1
	1	1	0	2	150	9
C File Label :	1	1	151	0	198	9
— VTOC Label :	1	0	199	0	199	9

Notes: *) The format-5 label is not used by VSE.
**) Label type for user-standard label area.

Figure 34 (Part 2 of 2). Disk Volume Layout: Files with User-Standard Labels

Figure 35 shows a multi-volume VSAM layout with the relationship between the catalog, labels, data space, and files.

```
1. Volume Layout
                          Volume 1
  ┌────┬──────────────┬────────────┬────────────┬────────────┐
  │VTOC│Data Space 1  │VSAM Catalog│Data Space 1│Data Space 2│
  └────┴──────────────┴────────────┴────────────┴────────────┘
        FileA FileB                 Free Space    File C*

                          Volume 2
  ┌────┬──────────────┬────────┬────────┬──────────────┐
  │VTOC│Data Space 3  │SAM File│DAM File│Data Space 3  │
  └────┴──────────────┴────────┴────────┴──────────────┘
           File D                          File E

2. VTOC Layout
                          Volume 1
  ┌─────┬──────────┬──────────────────────────────────────┐
  │VTOC │Format-5  │ IBM-Standard File Labels  for:       │
  │Label│Label**   │ Catalog │ Data Sp.1  │ Data Sp. 2    │
  │     │          │         │(2 Extents) │               │
  └─────┴──────────┴─────────┴────────────┴───────────────┘

                          Volume 2
  ┌─────┬──────────┬──────────────────────────────────────┐
  │VTOC │Format-5  │ IBM-Standard File Labels  for:       │
  │Label│Label**   │ Data Sp.3  │ DAM File │ SAM File     │
  │     │          │(2 Extents) │          │              │
  └─────┴──────────┴────────────┴──────────┴──────────────┘

Notes:  *) File C is a unique file.
       **) The format-5 label is not used by VSE.
```

Figure 35. Disk Volume Layout with VSAM Data Spaces

Labels

DISKETTE VOLUME

Place of Diskette Labels

The diskette volume label is placed on track 0, sector 7.

All IBM-standard diskette file labels are stored together in the VTOC on track 0, sectors 8 to 26. The VTOC is formatted at initialization by the factory. Label records are written into the formatted spaces when each file is created.

Diskette Volume Layout

The following layouts are possible:

One file on a single volume
One file on several volumes
Several files on one volume
Several files on several volumes

But it can be only one extent per file on one volume.

Files begin on track boundary. IOCS OPEN modules allocate space for the file on the track following the last unexpired or write-protected file on the diskette.

Track 0 of each volume contains the following:

```
Sectors 1 to  4 - Reserved
Sector 5        - Error Map
Sector 6        - Reserved
Sector 7        - Volume label
Sectors 8 to 26 - File labels
```

Files of data are written on tracks 1 - 73.

TAPE VOLUME

Place of Tape Labels

Tape volume labels are always the first record on the volume.

IBM-standard tape file labels are located immediately before and after the file, that is, a header label precedes each file and a trailer label follows each file.

User-standard labels on a tape always follow IBM-standard header and trailer labels. They are never written on a volume without IBM-standard labels.

Tape Volume Layout

On tape, each label is followed by an interblock gap. Each label and each tape mark constitutes such a block, as opposed to data blocks which may consist of any number of records.

A tape mark separates a set of labels from the data records. Tape marks also separate multiple files on a volume and show the end of records on a volume. Two tape marks follow the end of the last file on a volume.

IOCS writes these tape marks automatically unless you specified TPMARK=NO in the DTF.

Figure 36 on page 92 to Figure 38 on page 93 show various tape volume layouts with different file labels.

1. Single-Volume File

Minimum Label Set

Volume Label	File Label	T. M.	Data Records	T. M.	File Label	T. M.	T. M.

Maximum Label Set

Volume Label	File Label	User Labels	T. M.	Data Records	T. M.	File Label	User Labels	T. M.	T. M.

2. Multi-Volume File

First and Following Volumes

Volume Label	File Label	T. M.	Data Records	T. M.	File Label	T. M.

Last Volume

Volume Label	File Label	T. M.	Data Records	T. M.	File Label	T. M.	T. M.

3. Multi-File Volume

Vol. Label	File Label	T. M.	FileA Data	T. M.	File Label	T. M.	File Label	T. M.	FileB Data	T. M	File Label	T. M.	T. M.

T.M stands for tape mark

Figure 36. Tape Volumes with IBM- and User-Standard Labels

Single–File Volume

Non–Standard Header Labels	T. M.*	File A Data Records	T. M.	Non–Standard Trailer Labels	T. M.	T. M.

Multi–File Volume

Non–St. Header Labels	T. M*	FileA Data Rec.s	T. M.	Non–St. Trailer Labels	T. M.	Non–St. Header Labels	T. M.	FileB Data Rec.s	T. M.	Non–St. Trailer Labels	T. M.	T. M.

T.M stands for tape mark
Note:
*) No tape mark is written here on output, when TPMARK=NO
was specified.

Figure 37. Tape Volumes with Non-Standard Labels

Single–File Volume

T. M.*	File A Data Records	T. M.	T. M.

Multi–File Volume

T. M.*	File A Data Records	T. M.	T. M.*	File B Data Records	T. M.	T. M.

T.M stands for tape mark
Note:
*) No tape mark is written here on output, when TPMARK=NO
was specified.

Figure 38. Tape Volume with Unlabeled Files

HOW TO DISPLAY THE LABEL INFORMATION

VTOC LISTINGS

Disk VTOC listings can be obtained by replying CANCELV or DSPLYV to certain LIOCS messages or with the LVTOC program. For more details on the LVTOC program refer to the VSE/Advanced Functions System Utilities manual.

You enter the following job to get a listing of all labels in the VTOC with the content of selected fields.

```
// JOB jobname
// ASSGN SYS004,cuu*
// ASSGN SYS005,SYSLST
// EXEC LVTOC
```

*)Here you enter the address of the volume whose VTOC you want to display.

LABEL AREA DISPLAY

A listing of the label area contents is printed if you enter:

```
// JOB jobname
// EXEC LSERV
```

APPENDIX A. ISAM FILES

ISAM is not supported for: FBA devices, 3330-11, 3350, 3375 or the 3380, except when they are operated in 3330-1 compatibility mode.

SPECIFICATION RULES FOR ISAM

You supply one DLBL statement for the file, and one EXTENT statement for each extent that the file will occupy on the volume. An EXTENT statement provides the starting address (called relative track) and the number of tracks which indirectly gives the ending address.

The prime data extent and the cylinder index extent are required. The master index and the independent overflow area are optional and, if chosen, need an EXTENT statement each.

The prime data area for a file must be in only one extent per volume.

One extent has to be allocated for the master index and one as for the cylinder index. Both extents must reside on the same volume.

The extent sequence number must be in this order:

```
0:      Master index
1:      Cylinder index
2 etc.: Prime data
last:   Independent overflow

or

0:      Master index
1:      Cylinder index
2:      Independent Overflow
3 etc.: Prime Data
```

LABEL FORMATS FOR ISAM

In addition to the normal IBM-standard label, the system writes an extra file label for an ISAM file. This label was traditionally called format-2 label. It cannot be specified by the user but is generated internally by the system.

If a file occupies two or more volumes, ISAM only writes this ISAM label on the volume containing the cylinder index.

Appendix

The statistics in several fields of the ISAM label can be used to determine whether you should reorganize the file:

- D12 - Tag deletion count: The number of records you tag for deletion (not processed by ISAM)
- D13 - Non-first overflow reference count: The number of times a READ macro causes a search of the overflow area(s) for a record that is the second or higher one in an overflow chain
- D16 - Prime record count: The number of logical records written in the organized file in the prime data area(s). ISAM accumulates this count during a LOAD operation
- D27 - Number of independent overflow tracks: Number of tracks still available in the independent overflow area
- D28 - Overflow record count: Number of records written in all the overflow areas for the file
- D29 - Cylinder overflow area count: Number of cylinder overflow areas that have been filled

Volume Label

The volume label must be on cylinder 0, track 0, record 3 (except for 3350 operated in 3330-1 compatibility mode).

Prime Data Area

The prime data area on any volume must start on track 0 of any cylinder except cylinder 0.

Cylinder Overflow Area

Within the prime data area, certain tracks may be reserved for overflow records. These tracks are called cylinder overflow area and must be reserved by specifying the CYLOFL operand in the DTFIS macro.

Track Index

ISAM builds a separate track index for each cylinder used by the file. Track indexes are considered a part of the prime data area. Each track index starts on track 0 of the cylinder that is indexed. 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.

Master and Cylinder Indexes

The master index and the cylinder index are separate from the prime data area and from each other. However, ISAM builds them on one volume into one index area and the address of that combined area is in the IBM-standard label. Therefore, the areas for these indexes must be specified in the EXTENT statements side by side. 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.

The cylinder index must immediately follow the master index and they must both be located on one or more successive cylinders.

Multi-Volume Files

For a multi-volume 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 a multi-volume file, the prime data area of the first volume may start on any cylinder (except 0) and must extend through the last track on the volume. On all succeeding volumes (except the last), the prime data area must start on cylinder 1, track 0 and extend through the last track on the volume, so that IOCS considers the prime data area as one continuous area. On all succeeding volumes (except the last) the prime data area must extend On the last volume, it may end at the end of any cylinder. Thus all volumes, except the first and the last, are completely allotted to the prime data area from cylinder 1, track 0 up to the last track on the last cylinder.

For a multi-volume 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.

The master and cylinder index and the 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.

Appendix

APPENDIX B. JOB CONTROL FOR DAM FILES

The following is an example of job control for a DAM file.

```
// JOB USE A DIRECT ACCESS FILE
// ASSGN SYS004,191
// ASSGN SYS005,192
// ASSGN SYS006,193
// DLBL DISK,'DA FILE.LOAD.ADD OR PROCESS',90/001,DA
// EXTENT SYS004,111111,1,0,1700,99
// EXTENT SYS005,123456,1,1,0010,1990
// EXTENT SYS006,234567,1,2,0010,1990
// EXEC USEDAM
```

GLOSSARY

This glossary lists terms used in this book. For a more complete list of data processing terms, the reader is referred to the IBM Data Processing Glossary, GC20-1699.

The glossary includes definitions developed by the American National Standards Institute (ANSI). This material is reproduced from the American National Directory for Information Processing, copyright 1977 by the Computer and Business Equipment Manufacturers Association, copies of which may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018.

access method. A technique for moving data between virtual storage and I/O devices. VSE/Advanced Functions offers several programs to support such techniques: LIOCS (SAM, DAM, and ISAM), PIOCS, and the more comprehensive program VSAM which includes a complete space and file management.

access method services (AMS). A program that defines VSAM files and allocates space for them, reorganizes them and modifies their attributes in the catalog, ease data portability between operating systems, creates backup copies, and lists records and catalog entries on a printer.

address. See device address, CKD address, FBA address, storage address.

alphanumeric. The alphabetic characters A through Z, the digits 0 through 9, and the special characters #, $, and @.

alternate block (FBA) or track (CKD). A block or track that is reserved for use in case of failure of a primary block or track.

alternate (or alternative) index. An index that lists records of a file in an order different from the primary index.

American Standard Code for Information Interchange (ASCII). A code translating a set of 128 letters, numbers, special characters, and graphic signs into seven-digit binary numbers. In VSE/Advanced Functions it is used only for some interfaces. The high-order bit in the VSE system 8-bit environment is zero.

AMS. See access method services.

application program. A program to do the user's own work.

ASCII. See American Standard Code for Information Interchange.

Assembler. DOS/VSE Assembler language, a source language whose symbolic statements are translated one to one into machine instructions.

to attach. To connect an I/O device to the system by a channel. This device is then channel attached as opposed to link-connected, which is said of a device connected by a telephone line.

auxiliary storage. See storage.

bit. Binary digit; the smallest unit of information in physical storage.

to block. To combine two or more records into one unit of transfer to save transmission time and external storage space; opposite: to deblock.

Glossary

block of storage, FBA or VSAM. An addressable fixed-length unit of storage on an FBA disk or under VSAM management.

block of records. Several records placed together, to be moved as a unit.

blocking factor. The number of records combined into a block of records.

buffer. An area of storage used to hold a block of data while it is waiting to be processed or written on an I/O device.

byte. Eight bits, the smallest addressable unit of storage or data in the system.

card punch. An output device that punches holes into a card to represent data.

card reader. An input device that senses hole patterns in a punch card and translates them into machine processable form.

catalog. A central file directory of VSAM space.

CCW. See channel command word.

channel. A small independent processor that handles the transfer of data between storage and locally attached I/O devices.

channel command word (CCW). A command to the channel to execute a specific part of an I/O operation.

channel program. One or more channel command words (CCWs) that control a sequence of I/O operations. Execution of the sequence is started by one Start I/O (SIO) machine instruction.

CI. See control interval.

CKD address. A record location on a CKD disk identified by cylinder, track, and record number.

CKD device. See count-key-data device.

console. An input/output device on a computer reserved for communication between the operator or the maintenance engineer and the computer.

control interval (CI). A unit of information moved to or from an FBA device, or by VSAM to or from a disk device.

count-key-data (CKD) device. A disk device organized by cylinder, track, and record addresses rather than by fixed-length blocks as the FBA devices.

cylinder. The set of all tracks on a disk pack with the same distance from the axis.

data base. A set of data files stored under one central administration and service system.

to deblock. To read the records from a block one by one.

to delete a record. To mark the address of a record as unused space, so that it will be overwritten on the next occasion.

device. See I/O device.

device address. A physical address (cuu) or symbolic name (SYSRDR) for an I/O device.

device independence. The ability to request I/O operations without regard for the different I/O devices. See also "symbolic device name".

Device Support Facilities. A program to initialize, inspect, reformat, and analyze a disk volume.

direct-access processing. Processing of records in an order given by the program. The records are retrieved by their individual key.

directory. See index.

diskette. A small flexible magnetic disk in a jacket.

disk pack. A removable assembly of magnetic disks.

display device. An output device that shows data on a screen.

EBCDIC code. See Extended Binary-Coded Decimal Interchange Code.

entry-sequenced data set (ESDS). A VSAM file whose records are stored in the order in which they are inserted without regard to keys, and whose relative byte address can not change; contrast with KSDS.

Extended binary-coded decimal interchange code (EBCDIC). A code to represent upper and lower case characters, numbers, special and graphic signs by a string of eight binary bits.

extent. A space on a disk or diskette device occupied by or reserved for a particular file. This space can be either continuous or consist of reserved tracks on several cylinders. See split-cylinder extent.

external storage. See storage.

FBA address. A record location on an FBA disk identified by FBA block and record number.

FBA block. See block of storage, FBA.

FBA device. See Fixed Block Architecture device.

file. A set of related records treated as a unit.

Fixed Block Architecture (FBA) device. A disk device where storage is preformatted in blocks of fixed size. These blocks are addressed by block number from the beginning of the file. The alternative organization is CKD.

hardware. Machinery in data processing; contrast with software.

header label. An internal label immediately preceding the first record of a file.

hex (hexadecimal). Belonging to a numeral system that uses sixteen different number signs (including 0).

index. An ordered list of key items together with their respective data locations.

initial program load (IPL). The initialization procedure that prepares an operating system to start work.

input/output control system (IOCS). A group of macros delivered by IBM which handle the I/O processing.

I/O device. A piece of equipment by which data may go into the system or be received from the system or both.

interface. A connection between two components. A mutually agreed procedure of handing over data from one component to the other.

IOCS. See input/output control system.

IPL. See initial program load.

key. One or more characters in a record used to place and find the record in the file.

key field. A portion of the data of a record used to find the record and give it its place in a sequence. If the keys of one key field are not unique, a secondary key field may be specified.

key-sequenced data set (KSDS). A VSAM file whose records are ordered by a key; contrast with entry-sequenced data set (ESDS).

label. An identification record for a disk, diskette, or tape volume or file.

label area. An area on auxiliary storage that stores label information read from job control statements or macros.

link-connected. Devices connected to a processor by a link. A link may be a telephone line or a microwave beam.

LIOCS. See logical IOCS.

logical IOCS (LIOCS). A set of macros for creation, retrieval, and modification of files.

macro. A group of assembler instructions to be called by one comprehensive name, optionally with a built-in possibility of minor modifications. A macro takes the place of a small routine.

magnetic ink character recognition (MICR). Recognition of characters printed with ink containing magnetic particles; contrast with optical character recognition (OCR).

message. A short piece of information generated by a program for the user of that program.

MICR. See magnetic ink character recognition.

multi-file volume. A volume containing more than one file.

multi-volume file. A file that uses more than one volume.

OCR. See optical character recognition.

off-line. The operation of a functional unit without the continual control of a computer.

on-line. Used for devices under control of the system, that is, connected to the system by a channel or a link.

operand. Information entered with a command to control the execution of the command processor in a specifically required way.

operating system. Software that controls the execution of application programs and may do scheduling, debugging, accounting, data management, and other services.

optical character recognition (OCR). Recognition of printed characters by light-sensitive devices; contrast with magnetic ink character recognition (MICR).

overflow. Data which does not fit into the space given for it in the file.

to overlap. To perform an operation at the same time that another operation is being performed.

PIOCS (physical IOCS). Assembler macros for input/output control almost on the level of machine instructions.

primary block (FBA) or track (CKD). Block or track used for normal data storage, as opposed to alternate block or track used to replace a defective one.

primary index. An index ordered by the contents of the main key field of data in a file; contrast with alternate index.

printer. A device that writes output data on paper in a visually readable form.

procedure. A set of job control statements to start one or more programs with one EXEC PROC= statement.

processor storage. See storage.

program. A set of instructions that is started by an EXEC statement.

to punch. To transfer information to cards marking them with hole patterns which represent data.

real storage. See storage.

record. A collection of related data fields which forms a unit within a file.

relative record data set (RRDS). A VSAM file whose records are loaded into fixed-length slots and addressed by the numbers of the slots beginning from the start of the file.

routine. A sequence of program instructions that form a functional unit; often used repeatedly.

RRDS. See relative record data set.

sector. A block of 128 bytes of storage on a diskette.

segment. A part of a spanned record that is contained in one block.

sequential processing. The processing of records in the order in which they are stored.

serial device. Collective name for all non-disk devices like printers, tapes, card devices, or terminals.

software. Data processing programs and procedures; contrast with hardware.

spanned record. A record spanned over block boundary or broken up into segments.

split-cylinder extent. An extent which consists of specified tracks, all the same numbers, on several cylinders, for example tracks 7 to 15 on cylinders 1 to 5.

to spool. To write input or output temporarily on tape or disk to compensate for a difference in speed between program processing and I/O operation.

standard label. A label format predefined for automatic processing by IBM supplied programs.

storage. A device or part of a device that can retain data. In VSE/Advanced Functions, the following types or names of storage are used:

auxiliary storage
external storage
internal storage
processor storage
real storage
secondary storage
virtual storage

storage address. A byte address in real or virtual storage.

storage device. An I/O device for data storage. One device may accommodate several removable volumes.

streaming mode. A way to write from or to tape without stopping between records.

symbolic device name. An application program refers to an I/O device by a symbolic name, in VSE it is SYSxxx. Before the program is executed, job control can be used to assign a specific device address (three hexadecimal digits) to that symbolic name.

terminal. An input/output device consisting of a keyboard and a screen or printer by which a user communicates with a data processing system.

track. That portion of a moving storage device such as disk, diskette, or tape, that is accessible to a given reading head position.

trailer label. A label following the data records of a file on a storage device.

unit record device. A card device. By extension, all devices with naturally fixed-length records.

user label. A label to be processed by user-written routines.

utility (program). A service program in general support of the operating system.

virtual storage. A virtual extension of real processor storage presenting to the user the image of a much larger continuous address space for processing. This is done by a programming algorithm called paging.

volume. A reel of magnetic tape, a diskette, or a disk pack.

volume label. A label by which the system can find a volume.

VTOC (volume table of contents). A pre-defined index of the extents on a disk volume.

INDEX

Index

Index

Index

VSE/Advanced Functions
Data Management Concepts
Order No. GC33-6192-1

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).

GC33-6192-1

Reader's Comment Form

Cut or Fold Along Line

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 6R1BP
180 Kost Road
Mechanicsburg, PA 17055

Fold And Tape | Please Do Not Staple | Fold And Tape

IBM®

GC33-6192-1