LY24-5213-1
File No. S370-30

Program Product

VSE/VSAM Backup/Restore Feature Logic

Program Number 5746-AM2

Release 2

IBM

Second Edition (July 1981)

This edition, LY24-5213-1, is a major revision of LY24-5213-0. It applies to Release 2 of the Virtual Storage Extended/Virtual Storage Access Method (VSE/VSAM) Backup/Restore Feature, which is part of Program Product 5746-AM2, and to all subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are periodically made to the information contained herein. Before using this publication in connection with the operation of IBM systems, consult the latest edition of *IBM System/370 and 4300 Processors Bibliography,* GC20-0001, for the editions that are applicable and current.

A change to the text or an illustration is indicated by a vertical bar to the left of the change.

Summary of Amendments

For a list of changes, see page iii.

Summary of Amendments for VSE/VSAM Backup/Restore Feature Logic

Summary of Amendments for LY24-5213-1

VSE/VSAM Backup/Restore Release 2

VSE/VSAM Backup/Restore Release 2 lets you perform the following actions:

- Backup and restore empty objects
- Restore objects to a DASD volume of a different device type than the backup volume. You can move objects in the following ways:
  - From one CKD device to another CKD device
  - From one FBA device to another FBA device
  - From a CKD device to an FBA device
  - From an FBA device to a CKD device.
- Change the allocation size for the data component of an object at restoration (new DATARECORDS parameter).
- Change the index CI size at restoration (new INDEXCISIZE parameter).

A message-to-module cross-reference has been added to this manual, indicating which Backup/Restore modules could have issued each message.

Preface

This logic manual provides detailed information about the VSE/VSAM Backup/Restore Feature. It is intended for persons involved in program maintenance and for system programmers who are altering the program design. It is not required for effective operation of the product.

This manual contains information supplementing that contained in the following volumes:

- *VSE/VSAM VSAM Logic, Volume 1: Catalog Management, Open/Close, DADSM, ISAM Interface Program, and Control Block Manipulation,* LY24-5191.
- *VSE/VSAM VSAM Logic, Volume 2: Record Management,* LY24-5192.
- *VSE/VSAM Access Method Services Logic,* LY24-5195.

This manual refers to these books when appropriate; information in them is not duplicated here.

Organization of this Publication

This manual's structure differs from that of the conventional logic manual. Chapters 1, 2, and 7 should be read completely; chapters 3 through 6 are for reference.

- "Chapter 1: Format of the Backup File" describes the records and control information present on a backup file or volume.
- "Chapter 2: General Concepts" describes processing internals. Topics include control area processing, buffer handling, and the use that BACKUP and RESTORE make of control blocks. A summary of the major operations of the BACKUP and RESTORE commands is also included.
- "Chapter 3: Control Block Structure" summarizes the use of the major control blocks used by this Feature. The control block fields are *not* documented; refer to program listings for this information.
- "Chapter 4: Module Structure" shows the module-to-module flow for BACKUP and RESTORE. It also lists all executable and non-executable modules and their functions.
- "Chapter 5: Phase Structure" lists BACKUP/RESTORE phases, their functions, and the modules in each. The phase-to-link book structure is also shown.
- "Chapter 6: Macro Directory" lists the macros used by BACKUP and RESTORE and their functions.
- "Chapter 7: Diagnostic Aids" lists dump points, trace tables, abort codes and a message cross-reference table. It describes how to find some of the control blocks, how to determine which module was in control at the time of failure, which condition codes were issued, and which modules can issue each message.

Prerequisite Publications

You should be familiar with the following manuals before using this publication:

- *Using the VSE/VSAM Backup/Restore Feature,* SC24-5216
- *Using VSE/VSAM Commands and Macros,* SC24-5144
- *VSE/VSAM Programmer's Reference,* SC24-5145

Contents

Chapter 1: Format of the Backup File

The BACKUP command creates a labeled or unlabeled tape file, depending on whether or not STDLABEL was specified.

The backup file is a single-volume or multi-volume file consisting of several smaller subfiles that are separated by tape marks and do not contain their own sets of labels. The tape marks allow skipping individual files during restoration without reading and bypassing the individual blocks of the files to be skipped. Instead, Forward Space File commands, which free the tape channel for the duration of the skip operation, are used to skip from tape mark to tape mark. Because of the interspersed tape marks, labeled backup files cannot share a tape volume with other labeled files. The backup file, whether labeled or unlabeled, always starts at the beginning of a tape volume. Figure 1-1 illustrates the physical layout of the backup file.

The VOL1, HDR1, EOV1, and EOF1 labels are present only if the STDLABEL parameter for the BACKUP command was specified (that is, the backup file is labeled).

Directory

Each volume of the backup file contains a directory that contains two time stamps, some general information concerning the backup file, and a list of all objects included in the backup file.

The directory consists of one or more fixed-size blocks that are subdivided into a header, called the *directory block header*, and a set of *directory entries.* The last directory block may only be partially filled with directory entries. The number of objects of the backup file is identical to the number of directory entries unless the creation of the backup file was prematurely terminated, in which case there may be more directory entries than objects on the backup file. The premature end of the backup file is determined from the EOT (end-of-tape) record on the last backup volume, assuming that an EOT record was written. (An EOT is written if the BACKUP was prematurely terminated by an error other than a tape I/O error and was not canceled.)

The number of directory entries determines the number of directory blocks since each directory block has a fixed size of 1680 bytes on tape. The directory is preceded and followed by one tape mark.

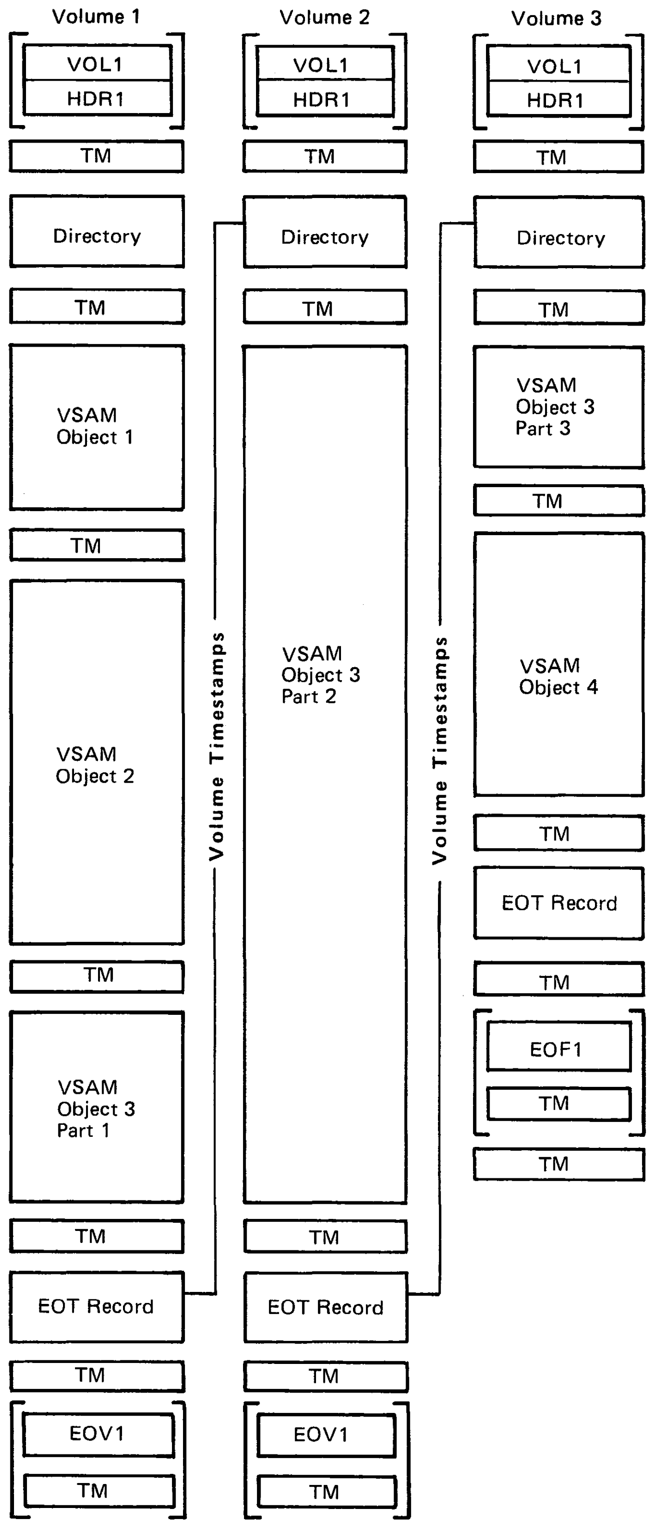


Figure 1-1. Format of the Backup File

The layout of the directory is shown in Figure 1-2.

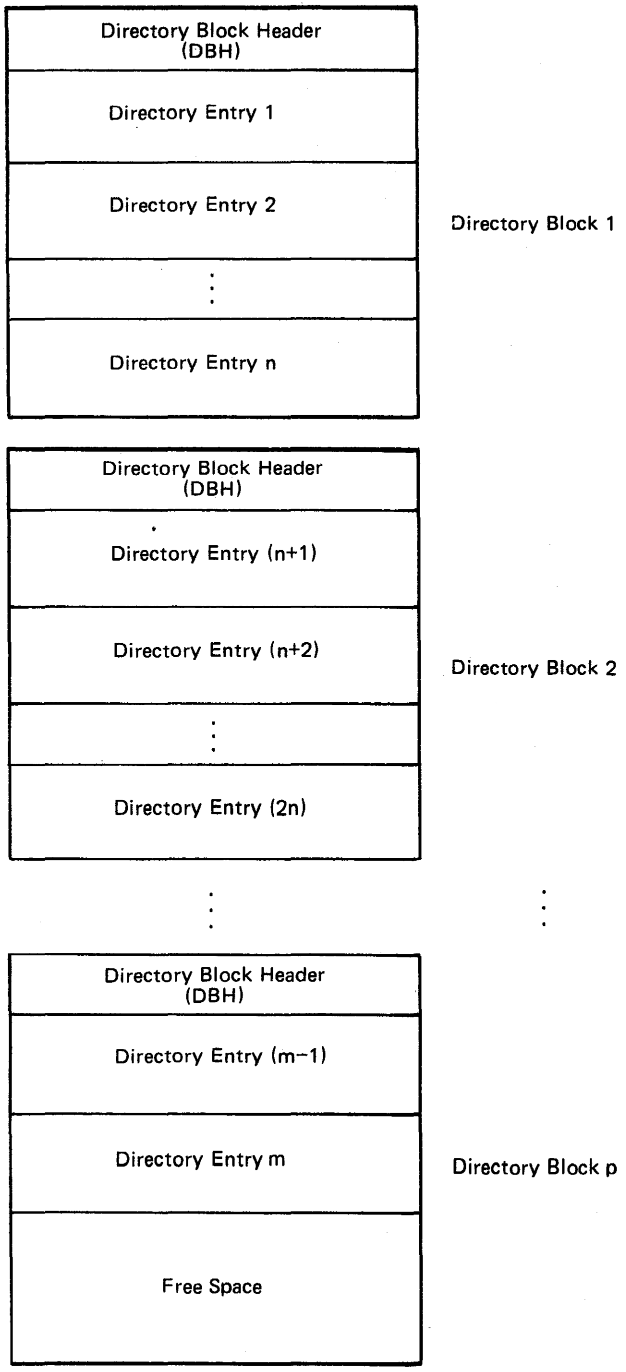


Figure 1-2. Layout of the Directory

Directory Block Header

Each directory block of the backup file starts with a 48-byte directory block header (DBH). The primary purpose of the directory block header is to control the space of the directory block to which it belongs. In addition, the first directory block contains information pertaining to the whole backup file and two time stamps:

- The time stamp indicating when the backup file was created (backup file creation time stamp), and
- The time stamp indicating when the particular backup volume was created (backup volume creation time stamp).

The volume creation time stamp of a backup volume other than the first is identical to the volume termination time stamp (the time when volume backup was completed) contained in the EOT record of the preceding backup volume.

The backup file creation time stamp is used when random mounting is performed in order to verify that the newly mounted volume belongs to the backup file being processed.

The backup volume creation time stamp is used when an object crosses backup volumes in order to verify that the newly mounted backup volume is the exact successor of the previously mounted volume and was not tampered with.

The format of the directory block header is as follows:

Offset	**Length**	**Contents**
0	4	CL4 'DBHb̸' identifies this block as a directory block.
4	4	First directory block: volume sequence number of backup volume. Subsequent directory blocks: binary zeros.
8	6	First directory block: creation date of backup file (mmddyy or ddmmyy). Subsequent directory blocks: binary zeros.
14	4	First directory block: backup file creation time of day in time units (TUs). Subsequent directory blocks: binary zeros.
18	6	First directory block: creation date of backup volume (mmddyy or ddmmyy). Subsequent directory blocks: binary zeros.
24	4	First directory block: backup volume creation time of day in time units (TUs). Subsequent directory blocks: binary zeros.
28	2	First directory block: number of dummy blocks provided for read ahead on RESTORE. Subsequent directory blocks: binary zeros.
30	2	Reserved (binary zeros).
32	4	First directory block: total number of directory blocks for directory. Subsequent directory blocks: binary zeros.

36	4	First directory block: total number of entries in directory. Subsequent directory blocks: binary zeros.
40	4	Number of this directory block (1 for first directory block, 2 for second directory block, etc.).
44	2	Offset of free space in this directory block plus 8. (The increment of 8 is caused by the fact that directory blocks in virtual storage are preceded by 4-byte forward and backward chain pointers.)
46	2	Length of remaining free space in this directory block.

Directory Entries

The directory block header of each directory block is immediately followed by directory entries.

In general, all directory blocks except the last are completely filled with directory entries. However, this is not a necessity. The free space offset and the free space length in the directory block header completely control the space utilization of the corresponding directory block and must be used in order to determine where directory entries are in a directory block. Do not assume that a directory block is completely filled with data.

Each object of the backup file has an entry in the directory. The directory entry gives the name of the object and contains, for those objects that reside or start on earlier volumes of the backup file than the volume containing the directory in question, additional information about the object:

- The type of object,
- The relational level of the object,
- The starting volume sequence number of the object,
- The starting volume serial number of the object (labeled tapes only), and,
- The number of volumes occupied by the object, if the particular backup volume does not contain any part of the object.

The directory entries are used by RESTORE to determine if a specified object is on the backup file and to allow efficient selective restoration of objects with random volume mounting.

The format of directory entries (58 bytes each) is as follows:

Offset	Length	Contents
0	44	Name of object, left-adjusted and padded with blanks.
44	1	Object type (decimal): 0 - Object type has not yet been established. 4 - Invalid object. The directory entry was reserved during initial creation of the directory for an object which later proved not to be a KSDS, ESDS, RRDS, SAM ESDS in CI-format, an AIX, or a path. 8 - Erroneous object (an object that could not be backed up successfully). 12 - Skipped object. During backup, this object was skipped due to an error condition for the base cluster or the path entry cluster (upon which the object is based) or because the object's base or path entry cluster was skipped. 16 - The object is a KSDS. 20 - The object is an ESDS. 24 - The object is a RRDS. 28 - The object is an AIX. 32 - The object is a path. 36 - The object is a SAM ESDS in control interval format.
45	1	Relational level of object on the backup file. Level numbers are used to express if the represented object is a dependent object (alternate index or path) of the preceding object of the backup file. A level number of 1 indicates that the object is not a dependent object of any other object of the backup file. A level number of 2 or 3 indicates that the object is a dependent object of the preceding object. A KSDS, ESDS, RRDS, or SAM ESDS always has the relational level 1. An AIX has the relational level 1 if its base cluster is not a member of the backup file. It has the level number 2 if its base cluster was also specified for backup. A path has the relational level 2 if it is immediately based on a cluster, or if its path entry AIX has been specified for backup without its base cluster. A path has the relational level 3 if directory entries are present for both its path entry AIX and the base cluster for the path entry AIX.
46	2	Volume count (number of volumes occupied by the object, if known).
48	4	Starting volume sequence number of the object. A volume sequence number of zero indicates that the object resides on this or a later volume of the backup file.
52	6	Starting volume serial number of the object. (Only if labeled backup file and if the object starts on an earlier backup volume; binary zeros otherwise.)

End-of-Tape (EOT) Record

Each volume of the backup file is terminated with an EOT record preceded and followed by a tape mark. For a labeled backup file, the trailing tape mark is followed by an EOV1 or EOF1 label. On the last volume of the backup file, an additional tape mark follows either the trailing tape mark (for an unlabeled backup file) or the EOF1/tape mark combination (for a labeled backup file). See Figure 1-1.

The presence of an EOT record indicates that processing of the mounted backup volume is complete.

The EOT record contains an identifier, an indication whether or not this is the last volume of the backup file, and the volume termination time stamp of the mounted backup volume. The volume termination time stamp is used on RESTORE when sequential tape mounting is performed. It must be identical to the volume creation time stamp contained in the first directory block header of the next sequential backup volume.

The format of the EOT record is as follows:

Offset	Length	Contents
0	4	CL4 'EOTb̸' identifies this block as an EOT record.
4	1	Type of EOT record: C'F' - End of backup file (last volume of the backup file). C'V' - End of backup volume (not the last volume of the backup file).
5	1	Reserved (binary zeros).
6	6	Backup volume termination date (mmddyy or ddmmyy).
12	4	Termination time of day for backup volume in time units (TUs).
16	8	Reserved (binary zeros).

Representation of Objects

Each part of an object on the backup file is preceded and followed by tape marks and starts with a header record.

The tape marks allow you to skip objects whose restoration is not desired by means of Forward Space File commands; you do not have to read and bypass the individual blocks of the skipped data sets. Thus, the tape channel can be freed for the duration of the skip operation.

The *header record* describes which object or which part of the object follows.

The first or only part of an object whose backup could be successfully started is preceded by an *Object Header* (OHD) which basically contains the name and the catalog information for the object.

The second or any later part of an object starts with a *Continuation Header* (CHD) which indicates that the subsequent data blocks (until the next tape mark) belong to an object that started on an earlier backup volume.

An object that was recognized as invalid, for which an error occurred before its backup could be started, or whose backup was skipped, is represented by an *Error Object Header* followed by no data at all. An Error Object Header is a special form of an Object Header and allows RESTORE to recognize invalid, skipped, or erroneous objects before any restoration for them is attempted. Note that objects for which an error occurred in the midst of the backup process are preceded by a regular Object Header and not by an Error Object Header. The premature termination of their backup is recognized by the unexpected encounter of dummy records (see "Dummy Records" below) which are not followed by an EOT record.

As mentioned before, an invalid, skipped, or early-recognized erroneous object is represented by an Error Object Header (which is preceded and followed by a tape mark). In the same way, a path object or empty object (which does not include any data) is simply represented by an Object Header (preceded and followed by a tape mark) that names the path and contains the pertinent catalog information for the path.

Parts for objects with data start with an Object Header (first part) or a Continuation Header (second or later part). The header is followed by data blocks containing the actual data of the object backed up. The data blocks in turn are followed by dummy records. The dummy records, which are "short blocks," are added to each object part of a data object (KSDS, ESDS, RRDS, SAM ESDS, or AIX) to facilitate buffering and read-ahead during restoration. If they were not provided, no read-ahead of tape blocks could be done during restoration, because otherwise, at the end of a tape volume, the tape could run off the tape reel.

Figures 1-3 through 1-5 summarize the representation of the individual object types on the backup file.

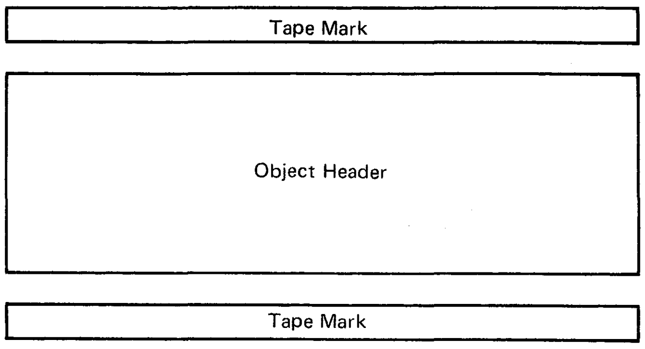


Figure 1-3. Representation of a Path or Empty Object

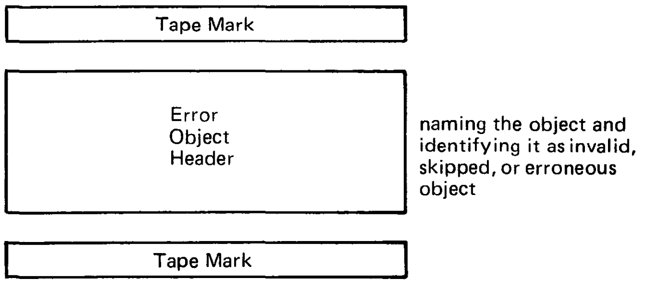

Tape Mark
Error Object Header
naming the object and identifying it as invalid, skipped, or erroneous object
Tape Mark

Figure 1-4. Representation of an Invalid, Skipped, or Early-Recognized Erroneous Object

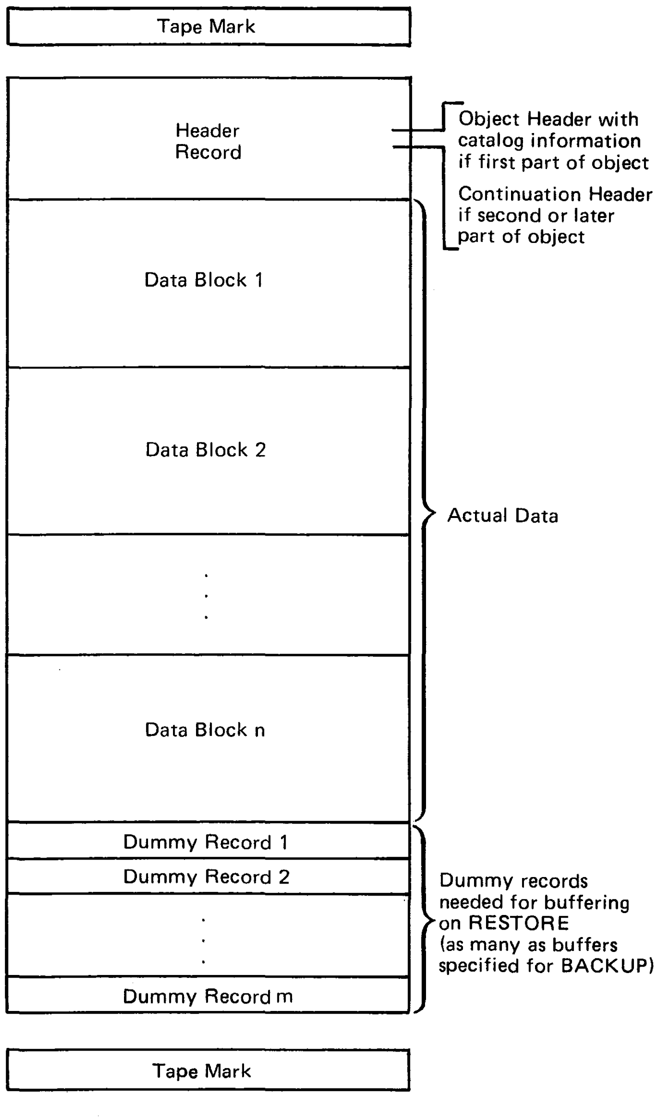

Tape Mark
Header Record
Object Header with catalog information if first part of object
Continuation Header if second or later part of object
Data Block 1
Data Block 2
Actual Data
Data Block n
Dummy Record 1
Dummy Record 2
Dummy Record m
Dummy records needed for buffering on RESTORE (as many as buffers specified for BACKUP)
Tape Mark

Figure 1-5. Representation of a Part of a Data Object

Object Header

The first part of each object of the backup file that is not invalid, that has not been skipped, or that has not been recognized as erroneous before its backup, is preceded by an object header.

The purpose of the object header is to identify the object and to provide the information necessary to redefine the object in the VSAM catalog when the object is restored.

As shown in Figure 1-6, the object header is logically broken into three parts:

- object header control portion
- dictionaries
- catalog information area

The individual items are described in the subsequent sections. Physically, the object header is subdivided into one or more fixed tape blocks of 1280 bytes each. The last tape block is padded with binary zeros if necessary. The physical mapping is transparent to the logical layout of the object header.

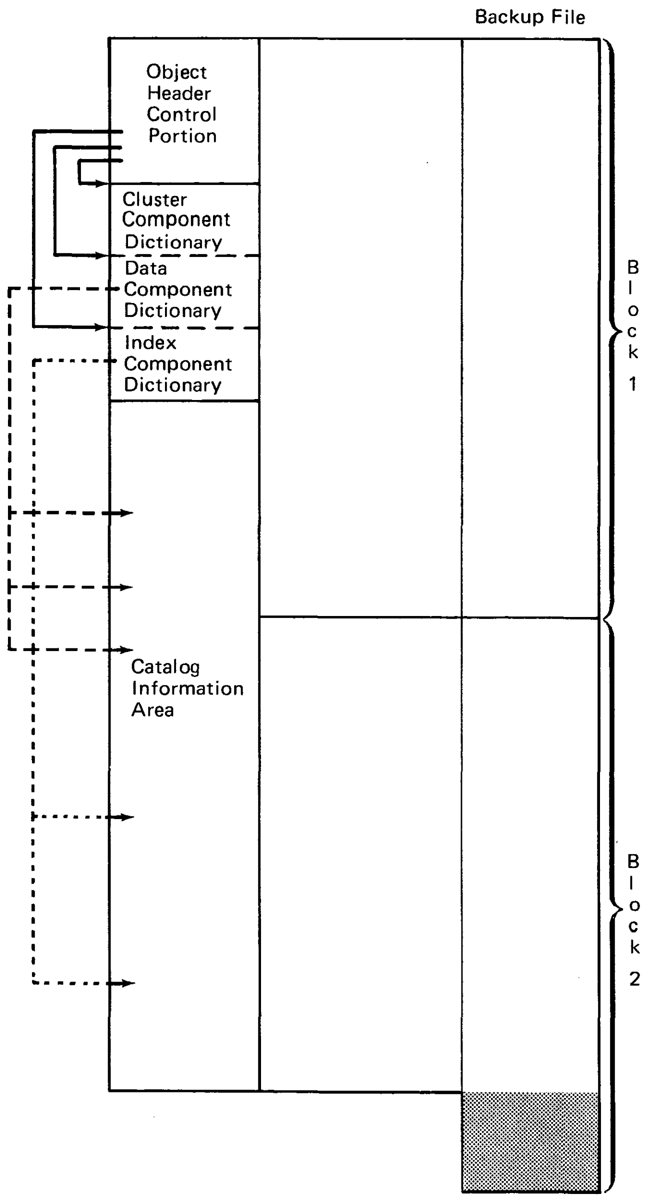

Backup File
Object Header Control Portion
Cluster Component Dictionary
Data Component Dictionary
Index Component Dictionary
Catalog Information Area
Block 1
Block 2

Figure 1-6. Object Header

Object Header Control Portion

The Object Header Control Portion contains:

- Information about the physical mapping of this particular object header (block size, number of physical blocks on tape, actual length of the object header).
- The type of the object and the offset to the name of the object within the catalog information area of the object header.
- Control information about the other parts of the object header.
- The buffer size that was used for backup (and which must be used for the restoration as well).
- The basic physical data set characteristics that prevailed when the backup was performed and which must be preserved on restoration.
- The data set high-used RBA as it was when the backup operation was performed.
- The data set statistics that applied when the object was backed up and which must be transported on the backup file because they cannot be recreated during restoration without the information saved in the Object Header Control Portion.

The layout of the Object Header Control Portion (112 bytes) is shown below.

Offset	Length	Contents
0	4	CL4 'OHDb̸' identifies this block as an object header.
4	1	Type of object being described by this object header: C'C' - object header for a cluster (KSDS, ESDS, RRDS, or SAM ESDS). C'G' - object header for an alternate index. C'R' - object header for a path. Other type codes are used to differentiate an error object header (the object header for an erroneous, invalid, or skipped object) from a regular object header. These error type codes are described under the heading "Error Object Header" below.
5	1	Object header flags indicating special conditions for the object: Bit 0 = 1: The passwords for the object were suppressed during backup because the specified password was not the master password; the backup file does not contain the passwords for the object. Bit 0 = 0: The passwords were not suppressed and are contained on the backup file (assuming passwords existed). Bits 1 through 7 are reserved and set to zero.
6	2	Release indicator; set to zero.
8	4	Actual (used) length of Object Header. Padded bytes in the final Object Header block are not included.
12	4	Size of Object Header blocks (1280 bytes).
16	4	Number of blocks for this Object Header.
20	4	Offset, relative to the beginning of the Object Header, of the 44-character name of the object represented by the Object Header.
24	4	Offset of the first dictionary for the object (the dictionary containing pointers to the catalog information of the C-type, G-type, or R-type catalog record that is included in the catalog information area of the Object Header).
28	4	Offset of the catalog work area (in the catalog information area) for the component pertaining to the first dictionary.
32	4	Offset of the second dictionary (the data component dictionary) for the object if the object has a data component; otherwise zero.
36	4	Offset of the catalog work area containing the data component catalog information; zero if the object has no data component.
40	4	Offset of the third dictionary (the index component dictionary) for the object. This field is zero if the object does not have an index component.
44	4	Offset of the catalog work area containing the index component catalog information for the object; zero if the object does not have an index component.
48	4	Buffer size used for backup.
52	4	VSAM physical record size for the data component of the object at backup.
56	4	Data control interval size of the object at backup.
60	4	Data control area size of the object at backup (set to zero for a SAM ESDS).
64	4	Index control interval size of the object at backup.
68	4	Data set high-used RBA of the object at backup.
72	4	Number of logical records of the object at backup.
76	4	Number of deleted records before backup.
80	4	Number of inserted records before backup.
84	4	Number of updates before backup.
88	4	Number of record retrievals before backup.
92	4	Reserved (must be zero).
96	4	Number of control interval splits before backup.
100	4	Number of control area splits before backup.
104	4	Number of EXCPs for the data component before backup.
108	4	Number of EXCPs for the index component before backup.

Fields that are not applicable to an object are initialized to zero. All offsets are relative to the beginning of the Object Header.

Dictionaries

Up to three dictionaries are provided in the Object Header (see Figure 1-6). The Object Header Control Portion specifies where these dictionaries are located in the Object Header.

The purpose of the dictionaries is to identify the individual pieces of catalog information in the catalog information area of the Object Header.

The first dictionary refers to the catalog information for the C-type cluster catalog record of a KSDS, an ESDS, an RRDS, or a SAM ESDS; to the catalog information for the G-type record of an alternate index; or to the catalog information for the R-type record of a path.

The second dictionary refers to the catalog information for the data component of the object, whereas the third dictionary applies to the index component catalog information. These dictionaries are only present if the object has data and index components.

The entities identified by dictionary entries are those retrieved by field or combination names through catalog Locate operations during backup. The same entities and field/combination names are used during restoration in order to redefine the object and its components in the VSAM catalog.

For each entity of catalog information for a component, the component dictionary has a "dictionary entry" of the following format:

Offset	Length	Contents
0	4	Length of catalog information.
4	4	Offset of catalog information relative to the beginning of the component's catalog work area pointed to by the Object Header Control Portion.

Each dictionary has the same set of dictionary entries. If the corresponding catalog information does not exist or is not applicable to the component, both the length and the offset fields of the dictionary entry are zero. The order of dictionary entries in a dictionary is fixed and is in the order of the catalog field and combination names listed below:

Dictionary Entry Number	Field/Combination Name
0	ENTYPE
1	ENTNAME
2	DSATTR
3	OWNERID
4	DSETCRDT
5	DSETEXDT
6	BUFSIZE
7	LRECL
8	SPACPARM
9	PASSWALL
10	LOKEYV
11	HIKEYV
12	VOLSER
13	AMDSBCAT
14	EXCPEXIT
15	RGATTR
16	Name of base cluster or path entry cluster
17	Master password of base cluster or path entry cluster

For the last two dictionary entries, no catalog field name or combination name exists.

The catalog information represented by the dictionary entries is the one located under the associated catalog field or combination name.

Catalog Information Area

The catalog information area (see Figure 1-6) contains the catalog information for all components of the object as it was retrieved by means of catalog Locate operations during backup and as it is used during restoration for the definition of the object in the VSAM catalog.

The catalog information for a component is stored consecutively and corresponds to the contents of the "catalog work area" provided for and filled by the appropriate catalog Locate operation for the component. The information includes both the work area length provided to Locate and the required length returned by Locate. For an alternate index or a path, the information is augmented by the name and the master password of the base cluster or the path entry cluster.

For all objects except paths, the space allocation parameters retrieved via Locate are converted to device-independent units (RECORDS). In order to do this conversion, constants such as physical record size, blocks per track, and tracks per control area are retrieved for the data component. Because these constants are only required for conversion of allocation units at backup, they are not saved as part of the catalog information area in the backup file.

Figure 1-7 shows the interaction of Object Header Control Portion, dictionary, and catalog information area.

Error Object Header

The Error Object Header constitutes a special form of an Object Header.

Because an Error Object Header represents either an invalid object, an object whose backup was skipped, or an object that was early recognized as erroneous (because it represents an object that was never restored), it is not necessary to carry the catalog information for such an object or any information that would normally be needed for restoration.

The Error Object Header merely indicates that an attempt was made to back up such an object.

The format of an Error Object Header is described below. Some fields have the same meaning as for the regular Object Header described above.

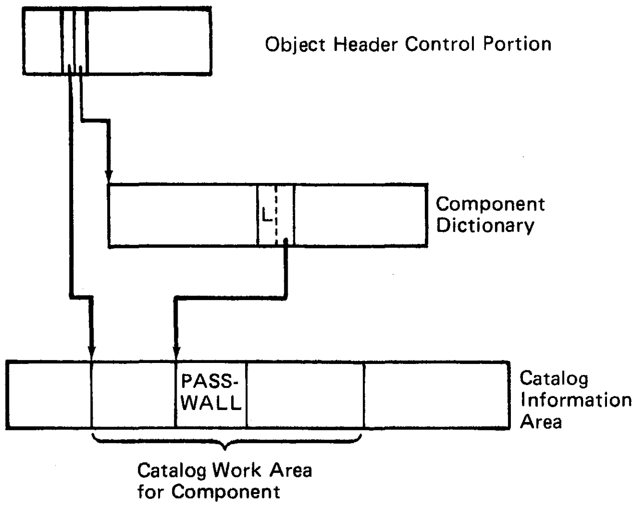

Object Header Control Portion
Component Dictionary
L
PASS-WALL
Catalog Information Area
Catalog Work Area for Component

Figure 1-7. Interaction of Object Header Control Portion, Dictionary, and Catalog Information Area

Offset	Length	Contents
0	4	CL4 'OHDƀ' identification as Object Header.
4	1	Type of object being described: X'FF' - Object Header for an invalid object. X'FE' - Object Header for an erroneous object. X'FD' - Object Header for an object whose backup was skipped.
5	1	Reserved (binary zeros).
6	2	Release indicator; set to zero.
8	4	Actual (used) length of Error Object Header
12	4	Block size of Error Object Header (1280 bytes).
16	4	Number of blocks for this Error Object Header.
20	4	Offset, relative to the beginning of the Error Object Header, of the 44-character name of the invalid, erroneous, or skipped object within the Error Object Header
24	88	Reserved (binary zeros).
112	44	Name of invalid, erroneous, or skipped object (left-adjusted and padded with blanks as necessary).

Continuation Header

The continuation header precedes the second or any later part of an object that spans backup volumes. The continuation header indicates that the subsequent data blocks until the next tape mark belong to an object that started on an earlier backup volume.

The continuation header allows non-consecutive mounting of backup volumes on RESTORE and allows the user to mount any volume other than the first one as initial volume during restoration. If continuation headers were not provided, the first data block of an object that is continued on the mounted backup volume could be mistaken for an Object Header. Note that the data blocks of an object contain user data (which may be anything) and do not have a special identification as data blocks.

The format of the continuation header (24 bytes) is as follows:

Offset	Length	Contents
0	4	CL4 'CHDƀ' identification as a continuation header.
4	20	Reserved (binary zeros).

Data Blocks of an Object

For data sets (KSDS, ESDS, RRDS, SAM ESDS, AIX), the Object Header is followed by *data blocks*, that is, blocks that contain the data of the object that was backed up.

With VSE/VSAM Backup/Restore, the emphasis is placed on fast transfer of VSAM data sets (data objects) to the backup file and back to disk storage, taking into account that the restoration is normally onto the same medium as the data set was backed up from and that the basic structural data set characteristics (physical record size, control interval size, and control area size) are preserved.

In contrast with the Access Method Services EXPORT/IMPORT facility, BACKUP/RESTORE transfers the physical records of a control area (which is, as the basic allocation unit, a physically consecutive disk-storage area) in *physical sequential order* from disk to the backup file (with the BACKUP command) and back (with the RESTORE command). Control intervals are not recognized, either during the transfer or on the backup file. Physical records, however, are recognized in the transfer process. In other words, the backup function basically creates a *physical image copy* of each control area on the backup file.

Because of the physical-sequential retrieval during the backup process, it is not necessary to step through the individual index entries of a sequence-set record. Because of spanned records, however, it is not possible to reconstruct the logical sequence of the control intervals of a KSDS from the image copy of the control areas alone. Therefore, the sequence-set record of each control area is also copied onto the backup file and reinstated by the restoration operation, thereby modifying the base and horizontal relative byte addresses, the only location-dependent variables in a sequence-set record.

The data blocks of an object on the backup file contain the user data as well as the sequence-set records of a KSDS. All data blocks of an object have the same fixed size. The size is equal to the buffer size recorded in the Object Header Control Portion for the object. The size is determined from the user's BLOCKSIZE specification on the BACKUP command and is always chosen so that:

- It is an integral multiple of the physical record size of the data component of the object; and
- It is not smaller than the index control interval size of the index component of the object.

Data component data and sequence-set control intervals are not mixed in the same data block. A sequence-set record on the backup file occupies a whole data block, the remainder of which is padded with zeros.

The last data block of a control area is partially padded with zeros if the control area size is not an integral multiple of the block size (buffer size). SAM entry-sequenced data sets form an exception because they do not have control areas. For them, the whole data component is consecutively stored so that all data blocks (except the last) are completely filled with data.

Each data block with data from the data component of the object consists of an integral number of physical records of the data component.

In contrast with the physical-sequential processing of the physical records of a control area, the individual control areas as a whole are processed in logical sequence, that is, the sequence is determined by the horizontal relative-byte addresses of the sequence-set records for a KSDS. Because control areas are, in general, a cylinder in size, the transition from one control area to another is not a frequent operation. Therefore, for the backup procedure it is not necessary to replace the logical retrieval of control areas with a physical retrieval. In addition, logically sequential control areas are also normally stored in physical sequence, because control area splits, which would disturb the physical sequence, occur less often than control interval splits.

The ability to reorganize control areas as a whole during restoration would be lost if control areas were not backed up in their logical sequence. After the restoration, the physical and logical sequence of the control areas coincide, thus preventing arm movements on subsequent sequential processing.

Figures 1-8 and 1-9 summarize the mapping of data objects onto the backup file.

Dummy Records

Each part of a data object (KSDS, ESDS, RRDS, SAM ESDS, or AIX) on the backup file is terminated by a set of dummy records. The dummy records are "short blocks" and are provided to facilitate buffering and read-ahead during restoration. Recognition of the

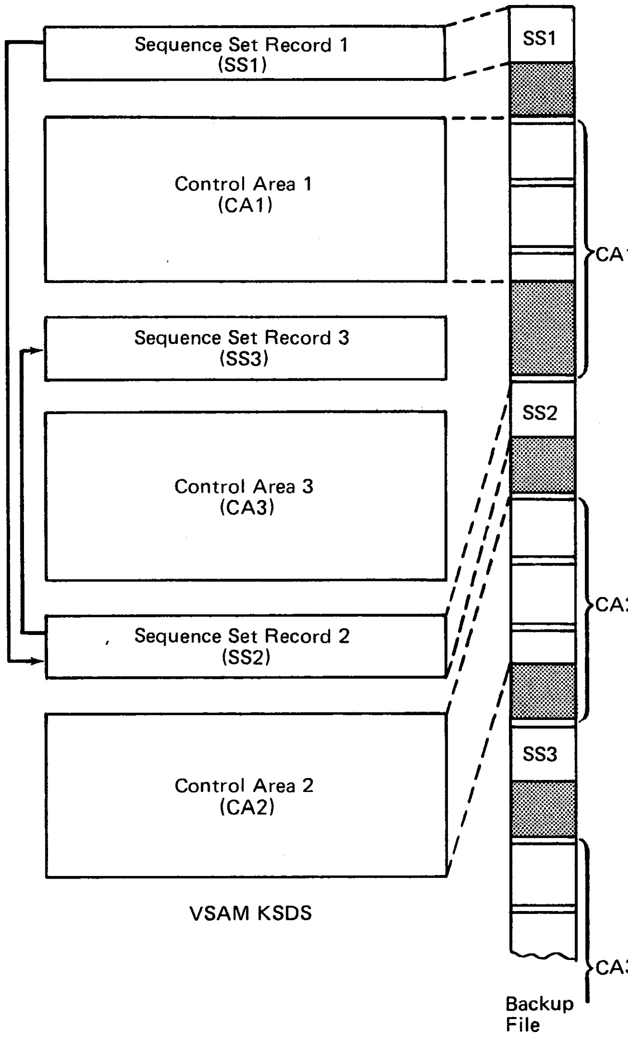


Figure 1-8. VSE/VSAM Backup/Restore Mapping

dummy records signals the end of the current part of the data set being restored and causes the mounting of the subsequent backup volume.

The number of dummy records is equal to the number of buffers specified (or defaulted to) on the BACKUP command. This number is recorded in the Directory Block Header of the first Directory Block on each backup volume.

The number of buffers that is allocated during restoration is never larger than the number of dummy records, and VSE/VSAM Backup/Restore never has more outstanding I/O requests for the backup file than there are buffers. Accordingly, each outstanding I/O request can be matched with a tape block so that the tape will not run off the tape reel.

The format of the dummy records (24 bytes each) is as follows:

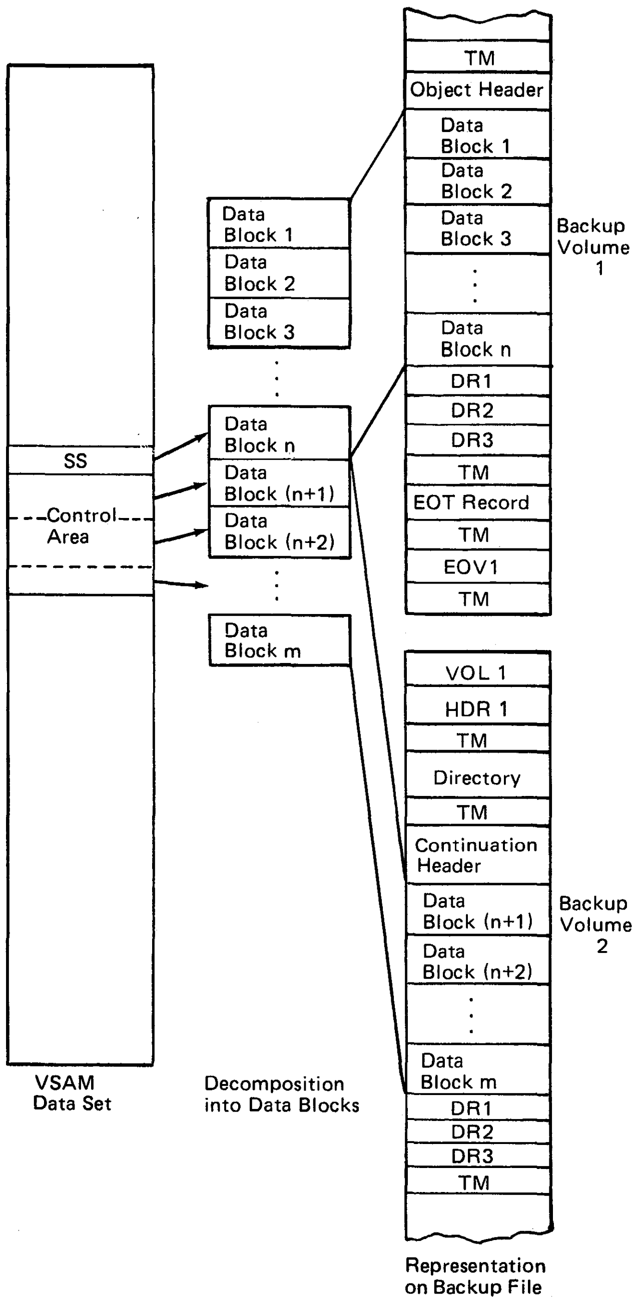


Figure 1-9. Transformation onto Backup File

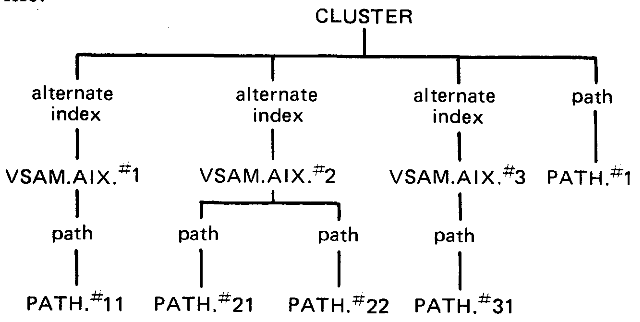
Offset	Length	Contents
0	4	CL4 'DRDb̸' identifies this block as dummy record
4	20	Reserved (binary zeros)

Sequence of Objects on the Backup File

The sequence of dependent objects on the backup file is important to ensure that all desired objects are actually restored and to avoid restoring objects twice.

If a cluster has alternate indexes and paths defined on top of it, the cluster is first on the backup file. It is followed by its first alternate index which, in turn, is followed by its paths. Then the second alternate index and its associated paths follow. Paths that are immediately defined over a cluster and not over an alternate index are treated in the same manner as alternate indexes with regard to their sequence on the backup file. They must follow the base cluster on which they are defined and may not be interspersed between an alternate index and its paths.

Assume that the cluster 'CLUSTER' has the following associations defined for it and recorded on the backup file:



For this cluster, the sequences below are valid:

CLUSTER		CLUSTER
VSAM.AIX.#1		PATH.#1
PATH.#11		VSAM.AIX.#1
VSAM.AIX.#2		PATH.#11
PATH.#21	or	VSAM.AIX.#3
PATH.#22		PATH.#31
VSAM.AIX.#3		VSAM.AIX.#2
PATH.#31		PATH.#21
PATH.#1		PATH.#22

On the other hand, the sequence:

CLUSTER
OTHER.OBJECT
VSAM.AIX.#2
PATH.#1
PATH.#21
PATH.#22
VSAM.AIX.#3
PATH.#31
VSAM.AIX.#1
PATH.#11

where OTHER.OBJECT is another object of the backup file that is not dependent on CLUSTER, is not valid because:

- An object not belonging to the associations of CLUSTER (OTHER.OBJECT) has been interspersed.
- PATH.#1 separates VSAM.AIX.#2 from its associations PATH.#21 and PATH.#22.

Chapter 2: General Concepts

This chapter discusses some basic general concepts of VSE/VSAM Backup/Restore.

Restoration with File Modifications

The following file modifications are permitted at restoration:

- Moving files to a space of a different use class;
- Moving files to a volume of a different device type;
- Changing the data component allocation size for a specific file;
- Changing the index control interval size for a specific file.

Specifying a new use class has no appreciable effect on the performance of the RESTORE command or on the file's internal structure. For any of the other file modifications, however, one or more of the following attributes of the cluster is likely to change:

- CA size
- Physical record size
- Index CI size
- Space allocation size

These file modifications can result in degraded performance during RESTORE execution, changed space allocation sizes due to the new device characteristics, and additional buffers for output to disk (described below).

Physical-Sequential Processing of Control Areas

VSE/VSAM Backup/Restore transfers the physical records of a control area in *physical* sequence from disk to the backup file and back. The unit of transfer is a buffer consisting of multiple physical records. The sequence-set records of a KSDS are also copied onto the backup file. They occupy a complete unit of transfer (the remainder of which may be padded with binary zeros) and precede the data blocks for their control area on the backup file.

The mapping of objects is described in detail in Chapter 1.

Buffers

The buffers used by BACKUP and RESTORE when no file modifications (described above) are made do not depend on the control interval size and are *common for tape and disk*. This means that the size of the *DASD unit of transfer is equal to the size of the tape block.* If not specified via the BLOCKSIZE parameter in the BACKUP command, the size of the buffer (which is equal to the amount of data transferred with a single disk or tape I/O operation) is determined by Backup/Restore from the DASD device characteristics (for example, either half a track or a track), the physical data set characteristics, and the minimum buffer size requirements for streaming. Rounding to an integral multiple of the physical record size of the VSAM object that is being backed up ensures that an integral number of physical records is read during a backup operation. During restoration, the same buffer size as was used for the corresponding backup is chosen. The user can influence the buffer size via the BLOCKSIZE parameter of the BACKUP command, but only if the specified BLOCKSIZE value is larger than the minimum assumed by VSE/VSAM Backup/Restore. The buffer size that is actually used does not necessarily coincide with the specified BLOCKSIZE value, because it is rounded to an integral number of physical records.

Using common buffers for tape and disk has the advantage that expensive *data movement can be avoided* and *no blocking or deblocking is necessary.* The data read from disk into a buffer is transferred onto the backup file (or vice versa) from the same buffer without any intermediate data movement. VSE/VSAM Backup/Restore uses its own specialized buffer and I/O management and avoids overhead by choosing a DASD unit of transfer equal to the tape unit of transfer.

When file modifications are specified during restoration, it is not possible to use common buffers for tape and disk because the data must be reblocked. When reblocking is required, RESTORE uses the common data buffers to handle *input* from the tape backup file. RESTORE allocates additional buffers to accommodate the new file characteristics for the *output* (to disk) file. RESTORE then moves the data from the input buffers to the output buffers as it reblocks the data.

Common Data Buffers

The number of data buffers allocated by VSE/VSAM Backup/Restore is controlled via the BUFFERS parameter. Their size is calculated from the BLOCKSIZE parameter of the BACKUP command or from defaults.

In order to reduce the path length of the basic backup or restoration cycle, the data buffers pointed to by the Buffer Definition Blocks (BDB) are chained together in a loop as shown in Figure 2-1.

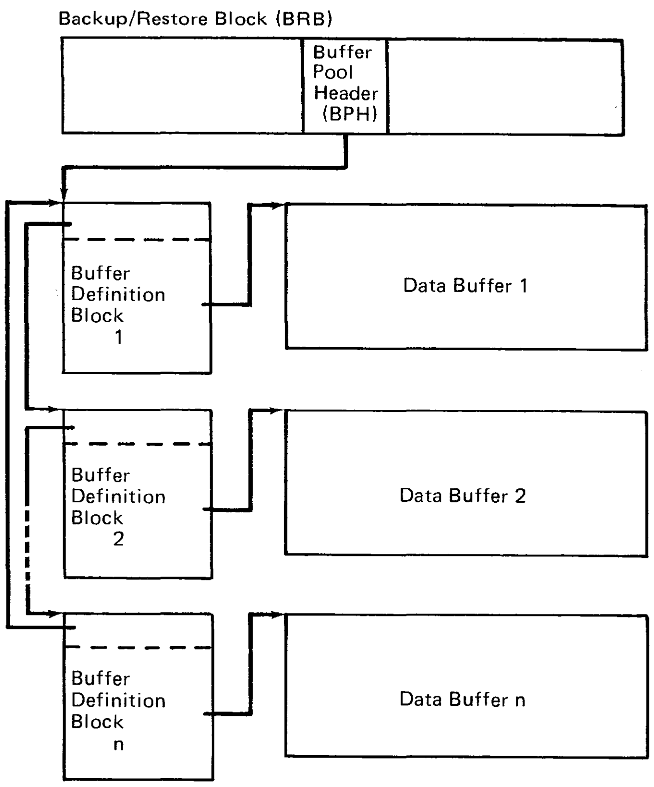

Backup/Restore Block (BRB)
Buffer Pool Header (BPH)
Buffer Definition Block 1
Data Buffer 1
Buffer Definition Block 2
Data Buffer 2
Buffer Definition Block n
Data Buffer n

Figure 2-1. Data Buffer Loop

Index Buffers

During backup, index control intervals of a KSDS are read to determine the logically next control area and are immediately written onto the backup file for reconstruction of the sequence set during restoration. Therefore, no special index buffers are needed or allocated during backup.

During restoration, however, the index of a KSDS must be reconstructed, requiring longer availability of index records or rereading of index records each time an index entry has to be made.

VSE/VSAM Backup/Restore reduces rereading of index control intervals by providing three special index buffers, each of index control interval size. These buffers help to minimize the disturbance of the regular restoration cycle at the end of a control area. They are an important factor in achieving streaming during restoration.

The first index buffer is reserved for sequence set control intervals. As soon as a sequence set control interval is read (into a data buffer) from the backup file, it is copied into the sequence set buffer for further processing, and backup file I/O is immediately rescheduled for the data buffer.

The second index buffer is reserved for second-level index control intervals. In this second-level index buffer, the index entries for the current second-level index control interval are constructed. In general, the second-level index buffer is not written before it has been completely filled with index entries. Format-write requirements for nonimbedded, non-keyrange KSDSs on CKD devices, however, may require an initial writing when the first sequence set control interval, represented by the second-level index record, is to be written.

The third index buffer is reserved for all higher-than-second-level index operations. Index control intervals are read into this buffer and written out as required. As long as the data set does not have more than three index levels, VSE/VSAM Backup/Restore will not perform any index read operations. The current third-level index control interval is kept in the third index buffer and written only if filled or if format-write considerations on CKD devices require an initial writing. Note that third-level index operations are infrequent and higher-than-third-level operations are rare.

By providing the three index buffers, VSE/VSAM Backup/Restore minimizes index I/O operations.

The index buffers are controlled by Index Buffer Blocks (XBB), as shown in Figure 2-2.

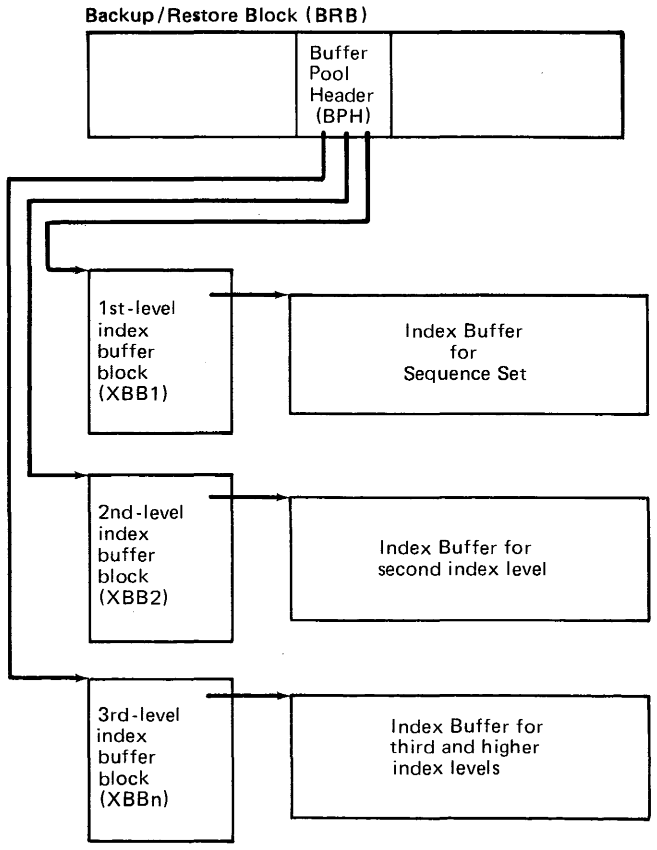

Backup/Restore Block (BRB)
Buffer Pool Header (BPH)
1st-level index buffer block (XBB1)
Index Buffer for Sequence Set
2nd-level index buffer block (XBB2)
Index Buffer for second index level
3rd-level index buffer block (XBBn)
Index Buffer for third and higher index levels

Figure 2-2. Index Buffer for RESTORE

Output Buffers for Restoration with File Modification

Restoration with file modification (described above) requires up to three additional buffers. These buffers are used only for output to disk; consequently, they have no associated tape channel programs. The preformat buffer is used for KSDS, ESDS, and RRDS to write "empty" control intervals to fill out control areas that are not full. For a KSDS, these empty CIs are used to restore the CA free space percentage to the file. An empty control interval for an RRDS is a control interval with empty record slots. For other files, an empty control interval consists of all zeros, except for a CIDF initialized with the length of the free space. No preformat buffer is used for a SAM ESDS.

The sequential write buffer is used for writing reblocked portions of the output file as they are encountered in ascending sequential order in the input. The size of the sequential write buffer is determined by rounding up the size of the common data buffer to an integral multiple of the new data CI size. This is done so that no more disk I/O operations are required (for data encountered sequentially) than would be required for a restoration without file modification.

The random write buffer is used only for a KSDS. It contains control intervals that must be inserted into the sequentially written data at a point prior to the current sequential position in the file.

These output buffers are shown in Figure 2-3.

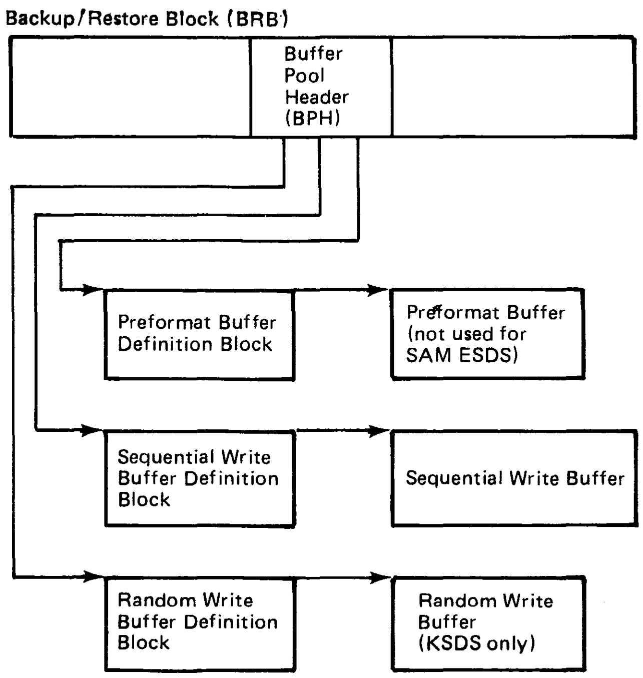

Backup/Restore Block (BRB)
Buffer Pool Header (BPH)
Preformat Buffer Definition Block
Preformat Buffer (not used for SAM ESDS)
Sequential Write Buffer Definition Block
Sequential Write Buffer
Random Write Buffer Definition Block
Random Write Buffer (KSDS only)

Figure 2-3. Output Data Buffers for RESTORE with File Modification

Channel Programs per Buffer

Each common data buffer has its own set of disk and tape channel programs to allow complete independence in the I/O scheduling of the individual buffers. In this way, several tape requests can be present in the channel queue at the same time, even if another tape request is still being executed. This allows, for example, the EXCP instruction for a second tape buffer to be issued before the I/O interrupt of the first tape buffer has occurred, and the SIO request for the second tape buffer can be issued immediately following the interrupt for the first I/O operation.

When file modifications are specified, each output data buffer has its own disk channel program; no tape channel programs are provided for these buffers.

Pregenerated Channel Programs for Backup/Restore

In order to reduce the path length between two successive SIOs for the backup file to a minimum, both the disk and tape channel programs for the individual buffers are not built dynamically for each EXCP instruction, but rather are "pregenerated" when Backup/Restore begins, (built only once before the general backup or restoration loop is entered). Only trivial modifications of the disk channel programs occur in the loop, such as the updating of the seek address. The tape channel programs are never changed.

Buffer Management Concepts

For a time-critical device in a multiprogramming environment, partition priorities play a role in buffer management. The following sections describe the effects of priorities on the buffer management for backup. Similar considerations also apply for restoration.

For the subsequent discussion, the following definitions are assumed:

- The *lowest-priority partition* in the system at any particular moment is the partition whose processing can be interrupted by all other partitions in the system, if the resources they are waiting for become available.
- The *highest-priority partition* in the system at any particular moment is the one that can interrupt any other partition if the resource it is waiting for becomes available.

- *Reinstruction* is the issuing of an SIO instruction before completion of the previous SIO in order to facilitate streaming.

Lowest-Priority Partition

Processing of the lowest-priority partition can be interrupted at any time by any other partition. However, if processing is interrupted, it is very likely that the point of reinstruction of the time-critical device will be missed, so that streaming may not be achieved. In addition, if the lowest-priority partition suspends its processing and waits for the completion of an I/O operation, the whole system remains in a wait state until either a higher-priority partition or the lowest-priority partition becomes ready again.

Therefore, the following must be true for VSE/VSAM Backup/Restore to operate effectively in the lowest-priority partition:

- The path between two successive EXCP instructions for a time-critical device must be as short as possible in order to reduce the likelihood of an interruption by a higher-priority partition.
- When the lowest-priority partition gets control, it must make optimum use of the time it gets by placing as many I/O requests as possible for the time-critical device into the channel queue. If it is able to put *n* I/O requests for the time-critical device into the channel queue during the execution of one I/O operation, the period that lasts until the next I/O request must be put into the channel queue will be *n* times the I/O time for the data transfer of one buffer of the time-critical device, instead of the single I/O operation time. Consequently, an interruption by a higher-priority partition may be sustained more easily without missing the point of reinstruction.
- The disk operation should be completed as fast as possible so that the time available for issuing the corresponding tape EXCP request for the buffer is as large as possible. If the time available for the scheduling of the tape request is small, the point of reinstruction is easily missed if control is lost to a higher-priority partition or to the Supervisor (for the handling of interrupts for other partitions).

VSE/VSAM Backup/Restore buffer management allows the user to specify the number of buffers and schedules as many tape I/O requests as possible in accordance with that number before a WAIT request is issued for the completion of a tape I/O operation. With one disk I/O operation, only one buffer is read, as described in the last bulleted item above.

The path length between two successive EXCP instructions is extremely short.

Figure 2-4 illustrates the effectiveness of this buffer management for four buffers.

Highest-Priority Partition

The buffering strategy described in the preceding section must be reevaluated for the highest-priority partition. Unlike the lowest-priority partition, the highest-priority partition obtains control whenever it needs it and does not wait for I/O completion or for the availability of a shared resource. If the highest-priority partition uses extensive buffering as described above, the speed of the slowest device (the tape device, in the case of a backup operation to tape) becomes the limiting factor, so that, eventually, all buffers for the slowest device become scheduled and can be refilled only one by one as they become available after the completion of the I/O operations scheduled for the slowest device.

Because the highest-priority partition automatically receives control when an I/O operation that it is being waited for is completed, it is generally not necessary to provide more buffers (for the highest-priority partition) than are absolutely necessary to meet the time-critical condition. However, an imbalance in the I/O usage by lower-priority partitions may require additional buffers to be used for the highest-priority partition.

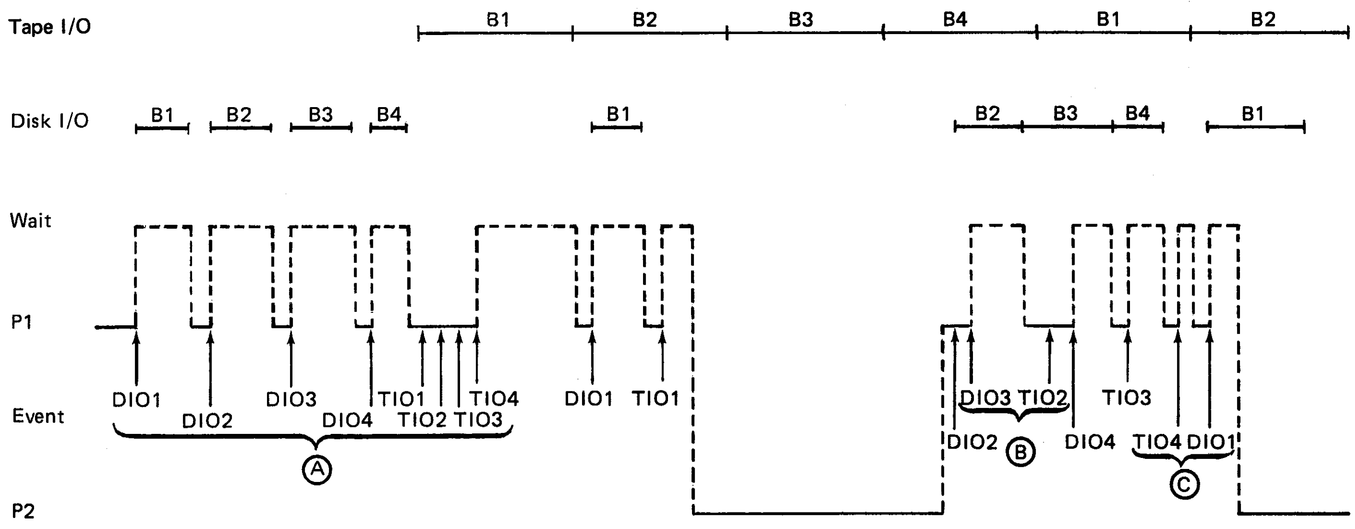
The buffer management for VSE/VSAM Backup/Restore allows the user to specify the number of common data buffers so that he can tune the space requirements for buffers in accordance with the priority of the partition in which he runs his VSE/VSAM Backup/Restore.

Restoration with file modification is not considered as performance-critical as normal restoration. Therefore, RESTORE does not consistently reinstruct time-critical devices in the required time. Buffer management is also more limited in that there is no flexibility in the number of special output data buffers when file modifications are required.

Locate Area

As described before, each volume of the backup file contains a directory listing all objects that will be contained on the backup file. The directory must be constructed before the first object is backed up. Generic names must be expanded to the set of entrynames they represent, and a determination must be made of which alternate indexes and paths must be backed up (automatically) because their base clusters or path entry AIXes are backed up.

In order to determine the set of objects for a generic name or to find the automatically backed up associa-



Legend:

Bn	=	buffer n
DIOn	=	disk I/O request for buffer n
TIOn	=	tape I/O request for buffer n
P1	=	the partition in which the VSAM backup operation is performed
P2	=	a second partition

Explanation:

A:	Initial filling of buffers with VSAM data and subsequent writing.
B:	If an empty buffer is available, a disk I/O request is issued before the tape I/O request for the preceding buffer.
C:	If no empty buffer is available, the tape I/O request for the preceding buffer is issued before the completion of a previous tape request.

Figure 2-4. VSE/VSAM Backup/Restore Buffer Management

tions of an object, it is necessary to retrieve at least the cluster (type C), alternate index (type G), or path (type R) catalog records of the objects being backed up before the first object is backed up. This catalog information is required later when the object header that precedes the object on the backup file is to be constructed.

In order not to have to locate the catalog information for an object twice, VSE/VSAM Backup/Restore keeps the catalog information for the object in the *locate area* (see Figure 2-5). The locate area is an area in virtual storage consisting of multiple blocks that are chained together by forward and backward chain pointers.

The individual blocks of the locate area are allocated on an as-needed basis. If only one block is required, only one is allocated. VSE/VSAM Backup/Restore limits the total size for the locate area to 32K bytes. The size, however, can be arbitrarily changed by changing the field LCHMLS in the locate area control header (LCH) which is part of the Backup/Restore Block, the major control block for VSE/VSAM Backup/Restore.

If the locate area becomes full during directory construction, construction of the directory continues, but only the absolutely necessary catalog information is retrieved for the remaining objects to be backed up. Their catalog information must be located again when

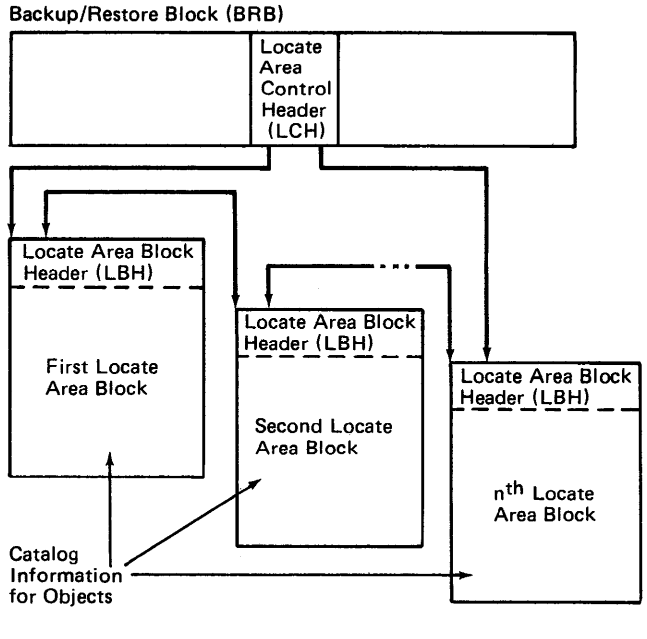

Backup/Restore Block (BRB)
Locate Area Control Header (LCH)
Locate Area Block Header (LBH)
First Locate Area Block
Locate Area Block Header (LBH)
Second Locate Area Block
Locate Area Block Header (LBH)
nth Locate Area Block
Catalog Information for Objects

Figure 2-5. Locate Area

space is available in the locate area or when the information is needed to construct the object header.

After all entries with catalog information in the locate area have been backed up, the locate area is reset to "empty" (marked as available but not freed), and the locate area is filled with catalog information for the next set of objects to be backed up. This process is repeated until all objects have been backed up.

Internal Directory Entries

As described in the previous section, catalog information for an object (directory entry) is retrieved when the directory is constructed and is kept in the locate area if space is available. Otherwise, the object's catalog information must be located again when locate area space becomes available.

In order to not have to reread the catalog high-keyrange record for an object when its catalog information is read to construct the object header, VSE/VSAM Backup/Restore keeps the control interval (CI) number of the low-keyrange record for the object in the *internal directory entry* for the object. The internal directory entries are extensions of the *external directory entries* that are recorded on the backup file. The internal directory entries are not written onto the backup file because they only contain information that is relevant for the backup operation for the object but is neither characteristic of the object nor relevant to the restoration of the object.

In virtual storage, the external and internal directory entries are allocated as shown in Figure 2-6.

The internal directory entry contains the control interval number of the C-type, G-type, or R-type catalog record for the object represented by the external directory entry. It also contains the address of the associated catalog information in the locate area, if present, and a pointer to the password to be used when locating the catalog information for the object.

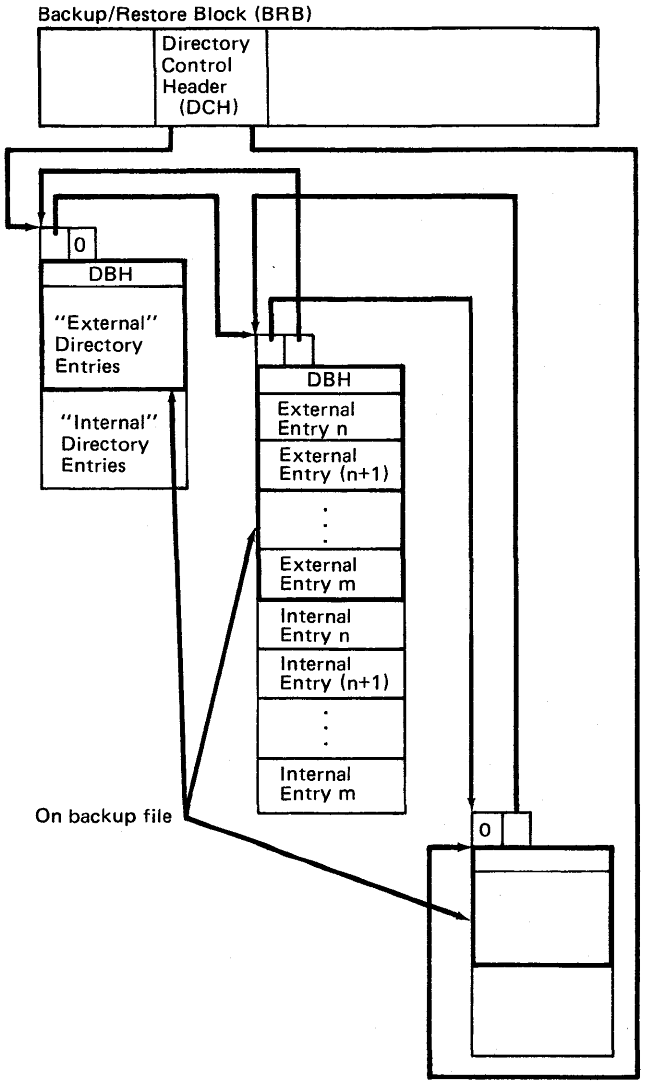


Figure 2-6. External and Internal Directory Entries

Volume List

At the end of BACKUP command execution, VSE/VSAM Backup/Restore prints the Backup Volume Cross Reference (BVCR) and the Backup Object Cross Reference (BOCR) listings. Both listings contain the volume sequence numbers and, for labeled backup files, the volume serial numbers of the individual backup volumes.

The volume sequence numbers are in ascending order, as assigned by VSE/VSAM Backup/Restore for reference purposes and in messages during restoration. The first backup volume has the volume sequence number one.

In order to print the volume serial numbers in the cross reference listings, VSE/VSAM Backup/Restore must gather the volume serial numbers as the individual backup volumes are mounted during backup and must keep them until the cross reference listings are printed.

VSE/VSAM Backup/Restore stores the volume serial numbers of the backup volumes into the *volume list* which consists of a set of virtual storage blocks, allocated as needed and chained by forward and backward chain pointers (see Figure 2-7). The volume serial numbers are stored in the sequence of the associated volume sequence numbers.

All blocks of the volume list have the same fixed length of 128 bytes. The size can be changed to any value by changing the field VLBNVLE (the number of entries in a volume list block) in the dummy section describing the layout of the volume list.

Restore Member List

The user does not have to specify the individual objects he wants to restore on the RESTORE command. He can use generic names where possible. Furthermore, some of the objects of the backup file are restored automati-

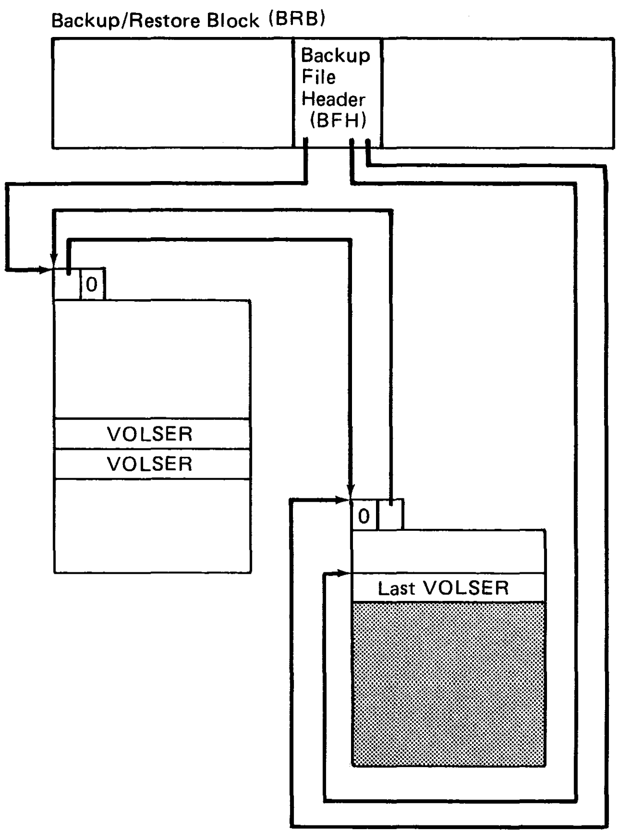


Figure 2-7. Volume List

cally without user specification. (Alternate indexes are restored along with their base cluster; paths are restored with their path entry cluster.) In addition, objects of the backup file can be excluded from restoration via the EXCLUDE parameter.

Therefore, the list of objects to be restored does not necessarily coincide with the list of objects specified in the command. Nor does it coincide with the list of entries in the directory. It is a subset of the directory entries.

Before any object of the backup file is restored, VSE/VSAM Backup/Restore constructs a list called the *restore member list* (or *restore list*), which contains one entry for each object that is actually restored (see Figure 2-8). The entries are ordered in the sequence the objects are restored.

The order in which the objects are restored depends on which volume is mounted first and is as follows:

- The objects of the initially mounted backup volume are restored first. They are first in the restore member list.
- Next are the objects of the backup volumes that follow (higher volume sequence numbers) the initially mounted backup volume. Their restoration sequence and sequence in the restore member list is the same as it is on the backup file.

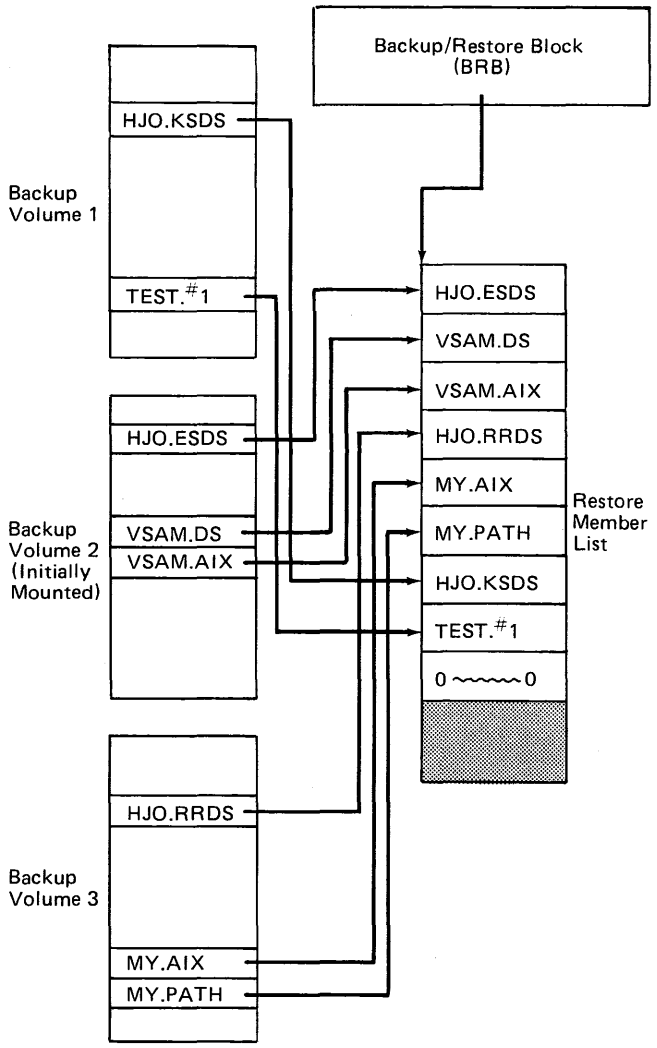


Figure 2-8. Restore Member List (RML)

- Last are the objects of the backup volumes that precede (lower volume sequence numbers) the initially mounted backup volume. Again their restoration sequence and sequence in the restore member list is the same as it is on the backup file.

One exception should be mentioned:

If an alternate index to be restored starts on the initially mounted (or a later) backup volume, but its base cluster starts on a backup volume that *precedes* the initially mounted backup volume, this alternate index is not restored before the base cluster is restored, and its entry in the restore member list follows the entry for the base cluster. The same exception applies to paths. Note that, in such a case, some of the backup volumes may have to be mounted twice.

The following general rules apply:

- Associations are always restored after the object they are based upon has been restored.
- The entries of associations in the restore member list always follow the restore member list entries for the objects the associations are based upon.

The restore member list is a consecutive list in virtual storage. The end of the list is indicated by an entry of zeros. The virtual storage allocated for the restore member list is chosen so that an entry for each object in directory plus a zero-entry would fit.

Each entry in the restore member list contains:

- A pointer to the associated directory entry that contains more information about the object.
- A pointer to the best-fit entry for the object in the object list of the RESTORE command. The best-fit entry is the one whose local modifications, like the VOLUMES specification, are to be applied to the object when it is defined in the catalog.
- A pointer to the entry of the object list of the RESTORE command whose password specification is to be used when an appropriate object with the same entryname is to be deleted from the catalog during restoration. In general, the password pointer is the same as the best-fit entry. For automatically restored associations, it may, however, be different (no best-fit entry).

The format of the restore member list entry is illustrated in Figure 2-9.

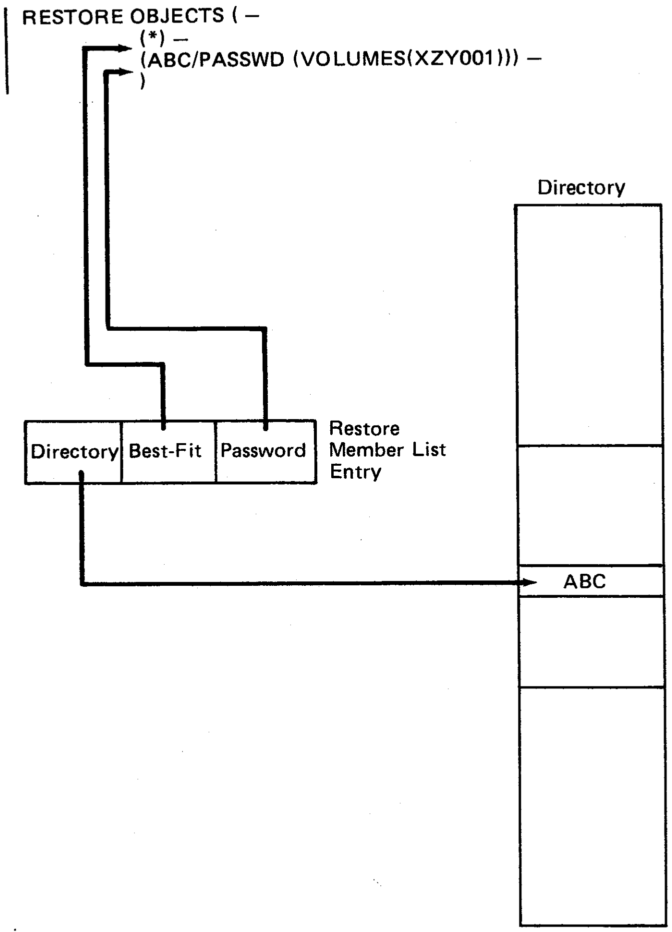


Figure 2-9. Restore Member List Entry

Index Information Blocks

VSE/VSAM Backup/Restore avoids time-consuming index-search operations in determining the location of and in reading higher-level index control intervals when an index entry has to be made.

During restoration, VSE/VSAM Backup/Restore provides an Index Information Block (XIB) for each potential index level. The index information block contains the relative byte address of the last index control interval of the appropriate index level so that the last index CI can be read immediately.

In addition, the index information block contains front-compression accumulators that allow simple calculation of the front-compression of an index entry from the front-compression of the section entries of the next lower level without performing an index decompression.

Essentially, the following rules apply for the calculation of front-compression:

- The front-compression of a regular index entry on level *n* is equal to the minimum of the front-compressions of the section entries of the index control interval of level *n*-1 represented by the index entry.
- The front-compression of a section entry of level *n* is equal to the minimum of the front-compressions of all index entries of the level *n* contained in the section in question.

These minimal values can be calculated easily as a by-product of the index construction on the next lower level. Accordingly, it is only necessary to determine the front-compressions on level one by decompression of the sequence set section entries and comparison with the high-key of the previous sequence set control interval. All higher-level front-compressions can be derived from the front-compressions on the sequence set level.

Because a VSAM data set is limited to 2^{32} bytes and the minimum control interval size is 512 bytes, there may be at most 2^{23} sequence set entries. Hence there will not be more than 23 index levels, provided at least two index entries fit into an index control interval. For the minimum index control interval size of 512 bytes, the key size should be not larger than 234 bytes. For larger index control interval sizes greater than 512, more than two index entries will fit.

The above considerations show that in nearly all cases the virtual storage required for the index information blocks will be less than 5K bytes.

In virtual storage, the index information blocks are allocated consecutively and can be indexed by means of the index level number. Sufficient space is allocated for the potential (in accordance with the key and index control interval size) maximum number of index levels plus one. The extra index information block is provided in order to allow the same index processing for all index levels, including the highest possible level.

The format of the index information blocks is shown in Figure 2-10.

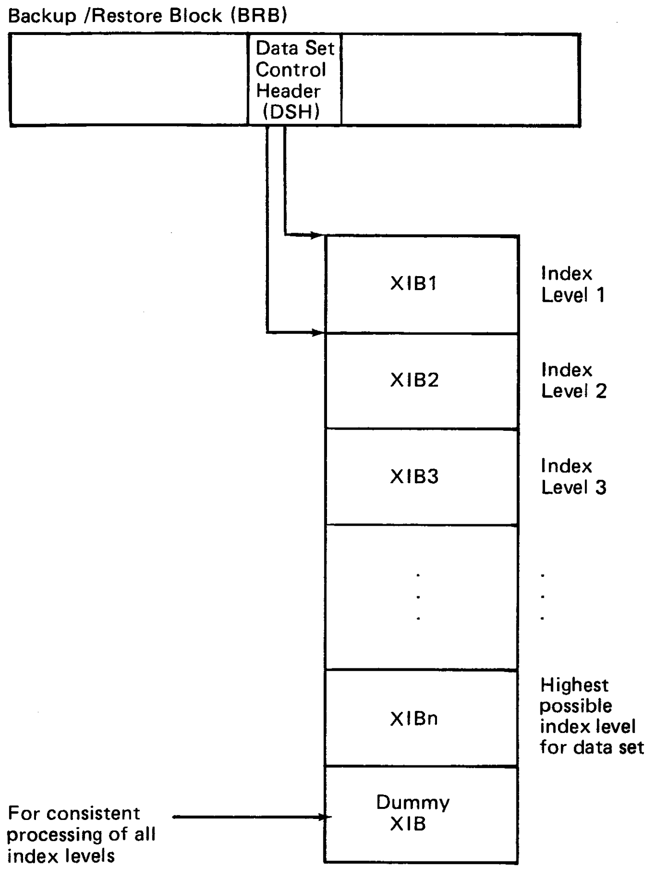

Backup /Restore Block (BRB)
Data Set Control Header (DSH)
XIB1
Index Level 1
XIB2
Index Level 2
XIB3
Index Level 3
XIBn
Highest possible index level for data set
Dummy XIB
For consistent processing of all index levels

Figure 2-10. Index Information Blocks

Backup and Restore Catalog Areas

Unlike the Access Method Services EXPORT and IMPORT commands, VSE/VSAM Backup/Restore does not acquire virtual storage each time a Catalog Parameter List (CTGPL), a Catalog Field Vector Table (CTGFV), or a Catalog Field Parameter List (CTGFL) is needed for catalog access.

The CTGPLs, CTGFVs, and CTGFLs required for catalog access are known to VSE/VSAM Backup/Restore in advance. They are pre-assembled and loaded (reentrant), when BACKUP or RESTORE command execution begins.

The catalog areas for BACKUP are contained in the Backup Catalog Area (BCA), and those for RESTORE are contained in the Restore Catalog Area (RCA), both of which are pointed to by the Backup/Restore Block.

Major Operations of the BACKUP Command

After the Access Method Services Executive transfers control to the BACKUP Functional Support Routine (FSR), the following basic operations are performed:

1. The Backup/Restore Block and the backup catalog area are loaded in a reentrant manner.
2. The correctness of the generic names in the BACKUP command is checked.
3. The directory is constructed:
   - Generic names are expanded to the set of entrynames they represent.
   - The associations of objects are automatically included.
   - Objects that are excluded from backup via the EXCLUDE parameter are not included in the directory.
4. In parallel with directory construction, the locate area is filled, as far as possible, with catalog information for the objects in the directory.
5. The backup file is opened and the directory is written onto the first backup volume.
6. The objects corresponding to the directory entries are backed up one by one. The backup process includes the following steps:
   a. It is ensured that the catalog information for the object to be backed up is contained in the locate area. If it is not, the locate area is refilled with the catalog information for the next set of objects.
   b. For a path, the object header is written onto the backup file.

   This is all that is done for a path. For non-path objects, steps c - g are also performed:

   c. The object is opened for input. If OPEN indicates the object is empty, only step e is performed.
   d. The buffer pool for the object's backup is constructed.
   e. The Object Header for the object is written onto the backup file.

   f. The object is copied onto the backup file.
   g. After the backup operation, the object is closed.

7. After all objects have been backed up, the Backup Volume Cross Reference Listing (BVCR) and the Backup Object Cross Reference Listing (BOCR) are printed.
8. The backup file is closed.
9. All allocated resources are released.
10. Control is transferred back to the Access Method Services Executive.

The BACKUP FSR invokes various subfunctions in order to perform the above actions.

Major Operations of the RESTORE Command

After the Access Method Services Executive transfers control to the RESTORE FSR, the following basic operations take place:

1. The Backup/Restore Block and the restore catalog area are loaded in a reentrant manner.
2. The correctness of the generic names in the RESTORE command is checked.
3. The backup file is opened and the directory is read.
4. The restore member list is created containing one entry for each object to be restored in the sequence the objects are restored. Restoration starts with the mounted volume and wraps around at the end of the backup file. Associations are never restored before the object they are based upon has been restored.

   Objects excluded from restoration via the EXCLUDE parameter of the RESTORE command are not in the restore member list.
5. The objects selected by the restore member list are restored one by one. The following steps are performed for each object:
   a. The backup file is searched for the object. The proper backup volume is mounted if it has not yet been mounted.
   b. The Object Header for the object is read.
   c. The object is defined in the VSAM catalog. An existing object with the same entryname is deleted before the definition. All local or global define modifications are applied.

   If the object is a path or an empty object, this is all that is done. For other objects, steps d - h are also performed.

   d. The object is opened for output.
   e. The buffer pool consisting of data buffers and, for a KSDS, three index buffers, is constructed.
   f. For a KSDS, the necessary number of index information blocks is provided.
   g. The object is restored. The index of a KSDS is reconstructed in the restoration process.
   h. The object is closed after it has been restored.
6. The backup file is closed and all allocated resources are released.
7. Control is transferred back to the Access Method Services Executive.

The RESTORE FSR invokes various subfunctions in order to perform the above actions.

Chapter 3: Control Block Structure

Figure 3-1 shows the basic control block structure for VSE/VSAM Backup/Restore. Most of the control blocks are discussed in previous sections and, therefore, are just summarized here.

Backup/Restore Block (BRB)

The Backup/Restore Block (BRB) is the major control block for VSE/VSAM Backup/Restore. It consists of seven sub-control blocks that control the resources used by VSE/VSAM Backup/Restore.

The sub-control blocks of the Backup/Restore Block are:

- Directory Control Header (DCH),
- Locate Area Control Header (LCH),
- VSAM Data Set Work Area (VDW),
- Data Set Control Header (DSH),
- Buffer Pool Header (BPH),
- Backup File Header (BFH), and
- Tape Command Parameter List (TCP).

Besides these sub-blocks, the Backup/Restore Block contains pointers to

- the Restore Member List (RML),
- the Backup Catalog Area (BCA), and
- the Restore Catalog Area (RCA).

In addition, the Backup/Restore Block contains work areas and a register save area pool for registers saved by the subfunctions invoked by the BACKUP FSR or the RESTORE FSR.

The Backup/Restore Block is always pointed to by register 13 and starts with a standard 72-byte save area for use by functions invoked by VSE/VSAM Backup/Restore (such as VSAM Open, Close, or Record Management).

The individual control blocks within the BRB are briefly described below.

Directory Control Header (DCH): A sub-block of the BRB controlling the virtual storage version of the directory. It contains directory block and entry pointers and counts.

Locate Area Control Header (LCH): A sub-block of the BRB controlling the Locate Area. It contains locate area block pointers and usage information.

VSAM Data Set Work Area (VDW): A sub-block of the BRB containing an ACB and related password and data set name areas used for opening an object to be backed up or restored. In addition, it contains the necessary call information to OPEN and CLOSE in order to provide reentrancy.

Data Set Control Header (DSH): A sub-block of the BRB containing the data set characteristics and additional object-related control information necessary for the backup or restoration of an object.

The DSH has three sub-blocks called Component Definition Blocks (CDB) describing the characteristics of the individual components of a VSAM data set. The CDBs are:

- the Data Component Definition Block (DCDB),
- the Sequence Set Component Definition Block (SSCDB), and
- the High-Level Index Component Definition Block (HXCDB).

VSE/VSAM Backup/Restore has different CDBs for the sequence set and the high-level index set in order to support mixed-architecture indexes.

The DSH also points to the index information blocks used for the reconstruction of the index during restoration.

The structure of the DSH is illustrated in Figure 3-2.

Buffer Pool Header (BPH): A sub-block of the BRB controlling buffer usage by VSE/VSAM Backup/Restore. It contains user-specified buffer options, buffer pool characteristics, and pointers to the first Buffer Definition Block (BDB) and Index Buffer Blocks (XBB).

Backup File Header (BFH): A sub-block of the BRB controlling the backup file. It contains the backup file and backup volume creation times, the volume sequence and volume serial numbers of the current backup volume, and pointers to the volume list for labeled backup files.

Tape Command Parameter List (TCP): A sub-block of the BRB containing a CCB, channel programs, and data areas for special tape (backup file) requests such as writing an EOT record or continuation header.

Additional control blocks used by Backup/Restore are described below:

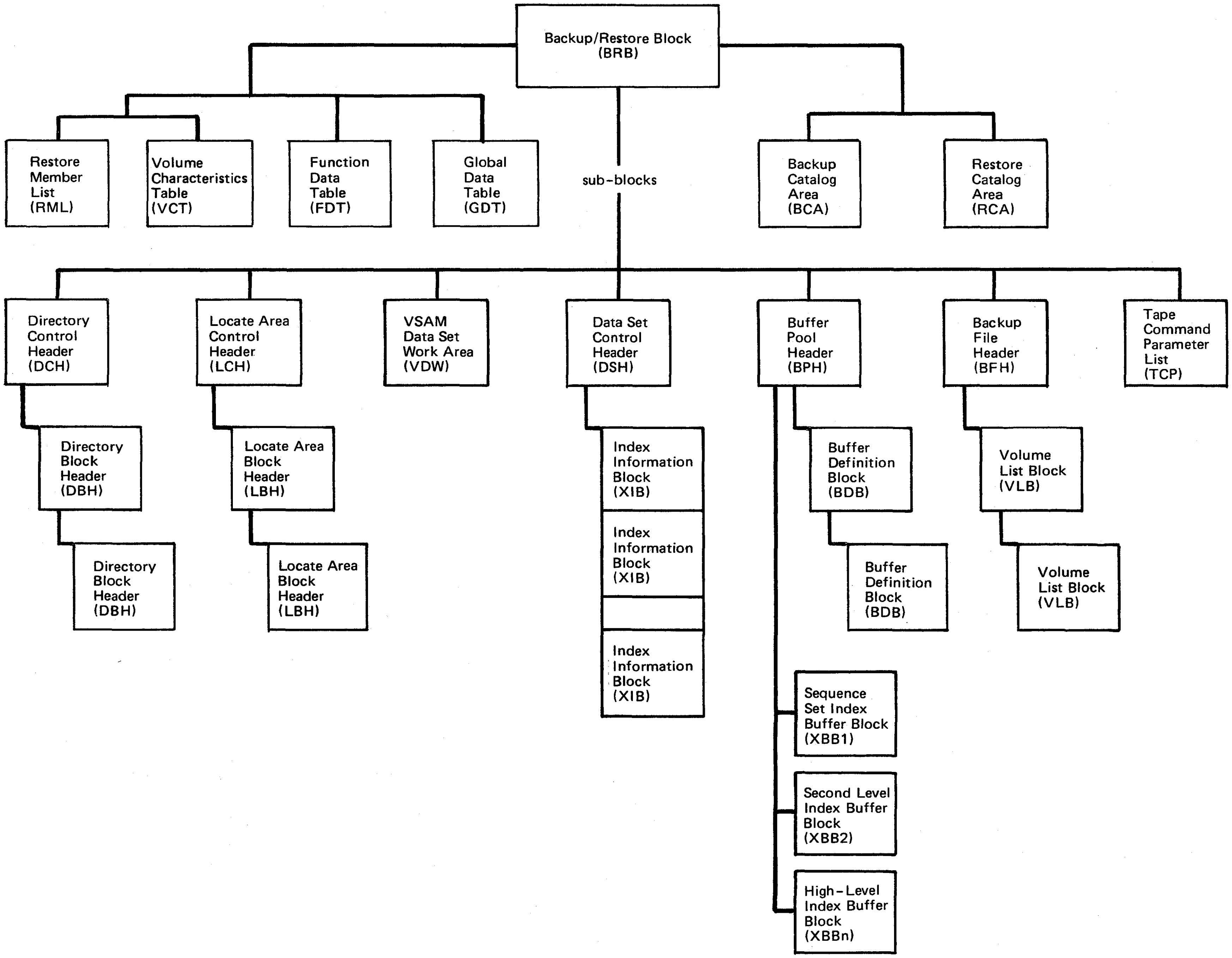

Backup/Restore Block (BRB)
Restore Member List (RML)
Volume Characteristics Table (VCT)
Function Data Table (FDT)
Global Data Table (GDT)
sub-blocks
Backup Catalog Area (BCA)
Restore Catalog Area (RCA)
Directory Control Header (DCH)
Directory Block Header (DBH)
Directory Block Header (DBH)
Locate Area Control Header (LCH)
Locate Area Block Header (LBH)
Locate Area Block Header (LBH)
VSAM Data Set Work Area (VDW)
Data Set Control Header (DSH)
Index Information Block (XIB)
Index Information Block (XIB)
Index Information Block (XIB)
Buffer Pool Header (BPH)
Buffer Definition Block (BDB)
Buffer Definition Block (BDB)
Sequence Set Index Buffer Block (XBB1)
Second Level Index Buffer Block (XBB2)
High-Level Index Buffer Block (XBBn)
Backup File Header (BFH)
Volume List Block (VLB)
Volume List Block (VLB)
Tape Command Parameter List (TCP)

Figure 3-1. Basic Control Block Structure

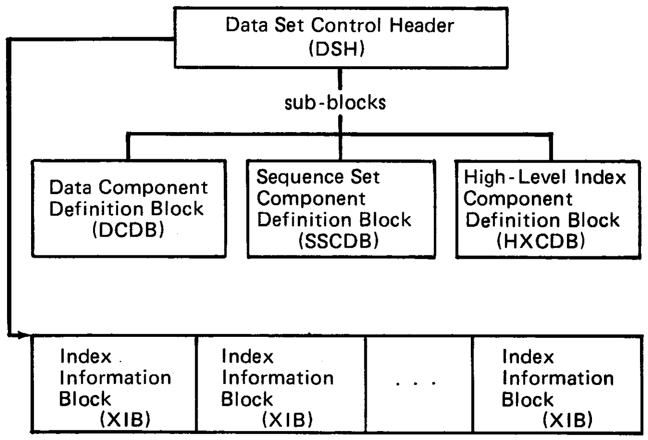

Data Set Control Header (DSH)
sub-blocks
Data Component Definition Block (DCDB)
Sequence Set Component Definition Block (SSCDB)
High-Level Index Component Definition Block (HXCDB)
Index Information Block (XIB)
Index Information Block (XIB)
. . .
Index Information Block (XIB)

Figure 3-2. Structure of the Data Set Control Header

Directory Block Header (DBH)
The header preceding each directory block and controlling the space utilization of the directory block.

Locate Area Block Header (LBH)
The header preceding each locate area block and controlling the space utilization of the locate area block.

Index Information Block (XIB)
A control block used to keep positioning and front-compression information for a particular index level.

Buffer Definition Block (BDB)
A control block controlling an individual data buffer in contrast to the total buffer pool. Besides pointers to the associated buffer and to the next buffer definition block in the "buffer loop," it contains IORBs, seek count fields, define-extent and locate parameter lists, and pointers to the disk and tape channel programs for the buffer.

Index Buffer Block (XBB)
A control block controlling an individual index buffer for index restoration. It contains pointers to the associated index buffer and its pregenerated disk channel programs. In addition, it contains an IORB and work areas for the channel programs.

Volume List Block (VLB)
A block of the volume list that contains the volume serial number of labeled backup volumes during backup.

Restore Member List (RML)
The expanded list of objects to be restored by the execution of a RESTORE command. The entries are in the same order as the corresponding objects are restored.

Volume Characteristics Table (VCT)
A chain of blocks containing an entry for each disk volume for which Backup has done a locate-by-volume-serial-number to find tracks-per-cylinder for conversion of allocation units. The use of this table lets Backup avoid repeated locates for the same volume.

Backup Catalog Area (BCA)
A control block containing all the fields, work areas, Catalog Parameter Lists, and Catalog Field Parameter Lists required for catalog access during backup.

Restore Catalog Area (RCA)
A control block containing all the fields, work areas, Catalog Parameter Lists, Catalog Field Vector Tables, and Catalog Field Parameter Lists required for catalog access during restoration.

Function Data Table (FDT)
A parameter list constructed by the Access Method Services Reader/Interpreter and passed by the Access Method Services Executive to the BACKUP or RESTORE FSR. It contains the internal representation of the parameters specified by the user on the BACKUP or RESTORE command.

Global Data Table (GDT)
A parameter list passed by the Access Method Services Executive to the function support routine and containing pointers to the Access Method Services service functions (such as UPRINT) and to the inter-module and intra-module trace tables.

Chapter 4: Module Structure

VSE/VSAM Backup/Restore is divided into a set of small, self-contained subfunctions with only minimal, well-defined interaction with surrounding functions. Maintainability is enhanced by this strict structuring because each function can be understood by itself.

Each function occupies one module.

Flow of Control

The functions (modules) of VSE/VSAM Backup/Restore always return control to the calling function so that the flow of control can be represented by a tree structure. Following is the flow of control for the BACKUP or RESTORE commands.

```
Access Method Services executive
    BACKUP
    BACKUP FSR (IDCBPFSR)
        message handler (IDCBPMSH)
        command analyzer (IDCBPCMA)
            message handler (IDCBPMSH)
        directory build (IDCBPDYB)
            open VSAM catalog (IDCBPOVC)
                obtain object name (IDCBPOON)
                    convert RBA (IDCBPCRB)
                        IKQEDX
                        IKQEOV
            scan exclusion list (IDCBPSXL)
            locate VSAM object (IDCBPLVO)
                scan exclusion list (IDCBPSXL)
                build locate entry (IDCBPBLE)
                    convert allocation units (IDCBPCAU)
                    add locate entry (IDCBPALE)
                add directory entry (IDCBPADE)
                search directory (IDCBPSRD)
                move directory entry (IDCBPMDE)
                obtain object name (IDCBPOON)
                locate VSAM object (IDCBPLVO)
                message handler (IDCBPMSH)
            message handler (IDCBPMSH)
        backup open (IDCBPBPO)
        secure locate entry (IDCBPSLE)
            reset locate area (IDCBPRSL)
            locate VSAM object (IDCBPLVO)
        VSAM open (IDCBPVOP)
            build RPSTAB (IDCBPBDR)
        build backup buffers (IDCBPBBF)
        write object header (IDCBPWOH)
            backup EOV (IDCBPBPV)
            message handler (IDCBPMSH)
        backup data set (IDCBPBDS)
            next backup volume (IDCBPNBV)
                backup EOV (IDCBPBPV)
            convert RBA (IDCBPCRB)
                IKQEDX
                IKQEOV
            data disk read (IDCBPDDR)
            data disk wait (IDCBPDDW)
        VSAM close (IDCBPVCL)
        backup close (IDCBPBPC)
            message handler (IDCBPMSH)
            print XREF (IDCBPPXL)
                directory sort (IDCBPDYS)
                message handler (IDCBPMSH)
        remove buffers (IDCBPRVB)
        remove locate area (IDCBPRVL)
        remove directory (IDCBPRVD)
```

```
RESTORE
RESTORE FSR (IDCRTFSR)
        message handler (IDCBPMSH)
        command analyzer (IDCBPCMA)
                message handler (IDCBPMSH)
        restore open (IDCRTRTO)
        build restore list (IDCRTBRL)
                scan exclusion list (IDCBPSXL)
        mount specific (IDCRTMTS)
                restore open (IDCRTRTO)
                operator (IDCRTOPI)
        read object header (IDCRTROH)
                mount next (IDCRTMTN)
                        operator (IDCRTOPI)
                mount later (IDCRTMTL)
                        restore open (IDCRTRTO)
                        mount specific (IDCRTMTS)
                                restore open (IDCRTRTO)
                                operator (IDCRTOPI)
                        operator (IDCRTOPI)
        define object (IDCRTDFO)
                build FVT (IDCRTBFV)
                delete VSAM object (IDCRTDVO)
                        message handler (IDCBPMSH)
                message handler (IDCBPMSH)
        VSAM open (IDCBPVOP)
                build RPSTAB (IDCBPBDR)
        build restore buffers (IDCRTBBR)
        build XIB (IDCRTBDX)
        restore data set (IDCRTRDS) or remap data set (IDCRTMDS)
```

Call IDCRTRDS for a basic restoration or IDCRTMDS if file modifications (restoration to volume of different device type or DATARECORDS or INDEXCISIZE specified) are required. These two paths are described on the following pages. After one of these two paths is completed, control returns to the main line for VSAM close processing.

```
        VSAM close (IDCBPVCL)
        delete VSAM object (IDCRTDVO)
                message handler (IDCBPMSH)
        remove XIB (IDCRTRVX)
        remove buffers (IDCBPRVB)
        restore close (IDCRTRTC)
```

Basic Restoration

```
restore data set (IDCRTRDS)
    get extent (IDCRTGEX)
        IKQNEX
    restore EOV (IDCRTREV)
        mount next (IDCRTMTN)
            operator (IDCRTOPI)
    convert RBA (IDCBPCRB)
        IKQEDX
        IKQEOV
    data disk write (IDCRTDWR)
    disk write wait (IDCRTDWW)
    add control area (IDCRTACA)
        get next index record (IDCRTGNX)
            get extent (IDCRTGEX)
                IKQNEX
        write index (IDCRTWRX)
            convert RBA (IDCBPCRB)
                IKQEDX
                IKQEOV
        read index (IDCRTRDX)
            convert RBA (IDCBPCRB)
                IKQEDX
                IKQEOV
        get extent (IDCRTGEX)
            IKQNEX
        write SEOF (IDCRTWRS)
            convert RBA (IDCBPCRB)
                IKQEDX
                IKQEOV
            data disk write (IDCRTDWR)
            data write wait (IDCRTDWW)
            write index (IDCRTWRX)
                convert RBA (IDCBPCRB)
                    IKQEDX
                    IKQEOV
    close index (IDCRTCLX)
        write SEOF (IDCRTWRS)
            convert RBA (IDCBPCRB)
                IKQEDX
                IKQEOV
            data disk write (IDCRTDWR)
            data write wait (IDCRTDWW)
            write index (IDCRTWRX)
                convert RBA (IDCBPCRB)
                    IKQEDX
                    IKQEOV
        write index (IDCRTWRX)
            convert RBA (IDCBPCRB)
                IKQEDX
                IKQEOV
    write SEOF (IDCRTWRS)
        convert RBA (IDCBPCRB)
            IKQEDX
            IKQEOV
        data disk write (IDCRTDWR)
        data write wait (IDCRTDWW)
        write index (IDCRTWRX)
            convert RBA (IDCBPCRB)
                IKQEDX
                IKQEOV
```

Return to the main line on page 4-2 for VSAM close processing.

Restoration with File Modification

```
remap data set (IDCRTMDS)
    get extent (IDCRTGEX)
        IKQNEX
    restore EOV (IDCRTREV)
        mount next (IDCRTMTN)
            operator (IDCRTOPI)
    convert RBA (IDCBPCRB)
        IKQEDX
        IKQEOV
    data disk write (IDCRTDWR)
    data write wait (IDCRTDWW)
    remap sequence set (IDCRTMSS)
        get extent (IDCRTGEX)
            IKQNEX
        add control area (IDCRTACA)
            get next index record (IDCRTGNX)
                get extent (IDCRTGEX)
                    IKQNEX
            write index (IDCRTWRX)
                convert RBA (IDCBPCRB)
                    IKQEDX
                    IKQEOV
            read index (IDCRTRDX)
                convert RBA (IDCBPCRB)
                    IKQEDX
                    IKQEOV
            get extent (IDCRTGEX)
                IKQNEX
            write SEOF (IDCRTWRS)
                convert RBA (IDCBPCRB)
                    IKQEDX
                    IKQEOV
                data disk write (IDCRTDWR)
                data write wait (IDCRTDWW)
                write index (IDCRTWRX)
                    convert RBA (IDCBPCRB)
                        IKQEDX
                        IKQEOV
    preformat (IDCRTPFO)
        convert RBA (IDCBPCRB)
            IKQEDX
            IKQEOV
    close index (IDCRTCLX)
        write SEOF (IDCRTWRS)
            convert RBA (IDCBPCRB)
                IKQEDX
                IKQEOV
            data disk write (IDCRTDWR)
            data write wait (IDCRTDWW)
            write index (IDCRTWRX)
                convert RBA (IDCBPCRB)
                    IKQEDX
                    IKQEOV
        write index (IDCRTWRX)
            convert RBA (IDCBPCRB)
                IKQEDX
                IKQEOV
    write SEOF (IDCRTWRS)
        convert RBA (IDCBPCRB)
            IKQEDX
            IKQEOV
        data disk write (IDCRTDWR)
        data write wait (IDCRTDWW)
        write index (IDCRTWRX)
            convert RBA (IDCBPCRB)
                IKQEDX
                IKQEOV
```

Return to the main line on page 4-2 for VSAM close processing.

Summary of Executable Modules

IDCBPADE	**Add Directory Entry** Acquires the space for a directory entry in a directory block and allocates new directory blocks as necessary.
IDCBPALE	**Add Locate Entry** Acquires the space for a Locate Entry (catalog information for the Object Header) in the Locate Area.
IDCBPBBF	**Build Backup Buffers** Constructs the buffers, Buffer Definition Blocks, and buffer channel programs for the backup of an object.
IDCBPBDR	**Build RPSTAB** Builds a sector number table for RPS devices to allow fast access to sector numbers during backup or restoration.
IDCBPBDS	**Back Up Data Set** Performs the actual backup of a data set.
IDCBPBLE	**Build Locate Entry** Constructs the Locate Entry (catalog information for the Object Header) in the Locate Area.
IDCBPBPC	**Backup Close** Closes the backup file after backup and causes the printing of the cross-reference listings.
IDCBPBPO	**Backup Open** Opens the backup file for output and constructs channel programs for writing the directory and the dummy records; writes the directory onto the first backup volume; initializes the volume list.
IDCBPBPV	**Backup EOV** Writes an EOT-record onto the current backup volume, mounts the next backup volume, and writes the directory onto it; extends the volume list.
IDCBPCAU	**Convert Allocation Units** Converts space allocation specifications (TRACKS or CYLINDERS, as retrieved from the catalog) to device-independent units (RECORDS) to be saved in the tape backup file.
IDCBPCMA	**Command Analyzer** Checks the correctness of any generic name in the object or exclusion list of the BACKUP or RESTORE command.
IDCBPCRB	**Convert RBA** Converts an RBA into a disk address.
IDCBPDDR	**Data Disk Read** Modifies the disk read channel program for a buffer and schedules the reading of a buffer from an object to be backed up.
IDCBPDDW	**Data Disk Wait** Completes a disk read operation scheduled by the Data-Disk-Read Function.
IDCBPDYB	**Directory Build** Builds a directory from the BACKUP command object list, the exclusion list, and the VSAM catalog. In parallel, the Locate Area is filled with catalog information for the objects to be backed up.
IDCBPDYS	**Directory Sort** Sorts the directory by object name.
IDCBPFSR	**BACKUP Function Support Routine** Basic module invoked by the Access Method Services Executive; directs the flow of control during the BACKUP command execution.
IDCBPLVO	**Locate VSAM Object** Obtains the catalog information for an object, builds a directory entry for it, and stores its catalog information in the Locate Area.
IDCBPMDE	**Move Directory Entry** Moves an existing entry of the directory to the end of the directory.
IDCBPMSH	**Message Handler** Prepares any message to be printed during BACKUP or RESTORE command execution for printing by the Access Method Services UPRINT.
IDCBPNBV	**Next Backup Volume** Writes the dummy record terminating a part of a data object, calls backup EOV to mount the next backup volume, and writes a Continuation Header for object being backed up.
IDCBPOON	**Obtain Object Name** Obtains the true name and the master password of a cluster, alternate index, or path record whose control interval number has been specified.
IDCBPOVC	**Open VSAM Catalog** Opens the VSAM Catalog as regular data set for input.
IDCBPPXL	**Print XREF** Assembles and prints the Backup Volume and the Backup Object Cross Reference listings.
IDCBPRSL	**Reset Locate Area** Resets the Locate Area to empty so that it can be refilled with catalog information.
IDCBPRVB	**Remove Buffers** Releases and frees the virtual storage for the buffer pool for BACKUP or RESTORE.
IDCBPRVD	**Remove Directory** Frees the virtual storage acquired for the backup file directory.
IDCBPRVL	**Remove Locate Area** Frees the virtual storage acquired for the Locate Area and for catalog work areas.
IDCBPSLE	**Secure Locate Entry** Ensures that the Locate Area contains the catalog information for the next object to be backed up. If not, it refills the Locate Area with the catalog information.
IDCBPSRD	**Search Directory** Searches the directory for a specified object name.
IDCBPSXL	**Scan Exclusion List** Scans the exclusion list of BACKUP or RESTORE command to determine if an object is to be excluded from backup or restoration.
IDCBPVCL	**VSAM Close** Closes an object after backup or restoration.

IDCBPVOP	**VSAM Open** Opens an object to be backed up or to be restored for input or output; constructs the Data Set Control Header for the object.
IDCBPWOH	**Write Object Header** Writes the Object Header for an object being backed up.
IDCRTACA	**Add Control Area** Writes the sequence set record for a control area and constructs the higher-level index entries for the control area.
IDCRTBBR	**Build Restore Buffers** Constructs the buffers, Buffer Definition Blocks, Index Buffer Blocks, and buffer channel programs for the restoration of an object.
IDCRTBDX	**Build XIB** Constructs the Index Information Blocks for the index reconstruction of an object to be restored.
IDCRTBFV	**Build FVT** Builds a field vector table and the associated field parameter lists for a component necessary for the redefinition of an object.
IDCRTBRL	**Build Restore List** Builds the Restore Member List (a list of all objects to be restored).
IDCRTCLX	**Close Index** Issues and completes any outstanding index I/O operation after the restoration of a key-sequenced data set. Initiates the writing of all necessary software-ends-of-file.
IDCRTDFO	**Define Object** Defines an object in the VSAM catalog during restoration.
IDCRTDVO	**Delete VSAM Object** Deletes an old version of a VSAM object to be restored.
IDCRTDWR	**Data Disk Write** Modifies the disk channel program for a data buffer and schedules the disk write operation for the data buffer during restoration.
IDCRTDWW	**Data Write Wait** Completes a disk write operation scheduled by the Data-Disk-Write Function.
IDCRTFSR	**RESTORE Function Support Routine** Basic module invoked by the Access Method Services Executive; controls the flow during the RESTORE command execution.
IDCRTGEX	**Get Extent** Obtains an extent for an object being restored.
IDCRTGNX	**Get Next Index Record** Obtains disk space and an index buffer for the next index record and initializes it.
IDCRTMDS	**Remap Data Set** Performs actual restoration of a data set when file modification (moving files to volume of different device type, or DATARECORDS or INDEXCISIZE specified) is required.
IDCRTMSS	**Remap Sequence Set** Reconstructs sequence set records when file modification (moving files to volume of different device type, or DATARECORDS or INDEXCISIZE specified) is required.
IDCRTMTL	**Mount Later** Mounts the next or any later volume of the backup file during restoration.
IDCRTMTN	**Mount Next** Mounts the next backup volume during restoration.
IDCRTMTS	**Mount Specific** Mounts a specified volume of the backup file.
IDCRTOPI	**Operator Interaction** Issues any messages to the operator during restoration.
IDCRTPFO	**Preformat** Preformats one or more empty CIs to use as free space within a CA.
IDCRTRDS	**Restore Data Set** Performs the actual restoration of a data set when no file modification is required.
IDCRTRDX	**Read Index** Reads an index control interval into an index buffer for third- or higher-level index.
IDCRTREV	**Restore EOV** Handles the transition to the next backup volume when the end of a backup volume is reached during the restoration of an object.
IDCRTROH	**Read Object Header** Scans the backup file for a specified object and reads the Object Header for it.
IDCRTRTC	**Restore Close** Closes the backup file after completion or termination of the RESTORE command.
IDCRTRTO	**Restore Open** Opens the backup file for input and reads the directory of the mounted backup volume.
IDCRTRVX	**Remove XIB** Frees the virtual storage acquired for Index Information Blocks.
IDCRTWRS	**Write SEOF** Writes a software-end-of-file (SEOF) for a data set being restored.
IDCRTWRX	**Write Index** Schedules the writing of an index buffer.

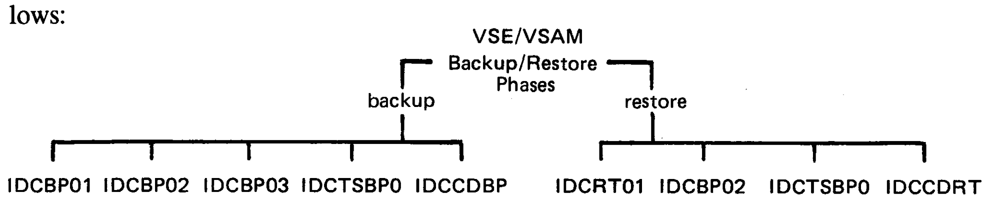
Summary of Non-Executable Modules

VSE/VSAM Backup/Restore includes modules that do not contain executable code but rather tables or pregenerated control blocks which are loaded at execution time, or which punch link books for the individual phases of VSE/VSAM Backup/Restore. The following is a list of these modules.

IDCBPBCA	**Backup Catalog Area** Pregenerated Backup Catalog Area containing all work areas, catalog parameter lists, field parameter lists, and channel programs for catalog access during the execution of the BACKUP command.
IDCBPBRB	**Backup/Restore Block** Pregenerated Backup/Restore Block; all fields initialized as required for the execution of BACKUP or RESTORE commands.
IDCBPBST	**Buffersize Table** Contains the tables necessary to determine the (optimal) buffersize to be used for the backup of an object.
IDCBPIOM	**I/O Module** Contains the DTFMT, MTMOD, and DTFCN declarations used for the opening, closing, and end-of-volume handling of the backup file or for operator messages.
IDCCDBP	**Backup Command Descriptor** Contains the command descripor to be used by the Access Method Services Reader/Interpreter to analyze a BACKUP command and to construct the appropriate Function Data Table.
IDCCDRT	**Restore Command Descriptor** Contains the command descriptor to be used by the Access Method Services Reader/Interpreter to analyze a RESTORE command and to construct the appropriate Function Data Table.
IDCCMZ3	**IDCTSBP0 Link Book** Punches phase, include, entry, and end statements for the link book for phase IDCTSBP0, which contains the static text entries for VSE/VSAM Backup/Restore.
IDCCMZ4	**IDCBP01 Link Book** Punches phase, include, entry, and end statements for the link book for phase IDCBP01, which contains the functional support routines for the BACKUP command.
IDCCMZ5	**IDCBP02 Link Book** Punches phase, include, entry, and end statements for the link book for phase IDCBP02, which contains the pregenerated Backup/Restore Block, the Backup Catalog Area, and the Restore Catalog Area.
IDCCMZ6	**IDCBP03 Link Book** Punches phase, include, entry, and end statements for the link book for phase IDCBP03, which contains the buffersize table for VSE/VSAM Backup/Restore.
IDCCMZ7	**IDCCDBP Link Book** Punches phase, include, entry, and end statements for the link book for phase IDCCDBP, which contains the command descriptor for the BACKUP command.
IDCCMZ8	**IDCRT01 Link Book** Punches phase, include, entry, and end statements for the link book for phase IDCRT01, which contains the functional support routines for the RESTORE command.
IDCCMZ9	**IDCCDRT Link Book** Punches phase, include, entry, and end statements for the link book for phase IDCCDRT, which contains the command descriptor for the RESTORE command.
IDCRTRCA	**Restore Catalog Area** Pregenerated Restore Catalog Area containing all work areas, catalog parameter lists, field vector tables, and field parameter lists required for catalog access during the execution of the RESTORE command.
IDCTSBP0	**Backup/Restore Static Text Module** Contains the format structures for the VSE/VSAM Backup/Restore messages to be printed by means of the Access Method Services UPRINT function.

Chapter 5: Phase Structure

VSE/VSAM Backup/Restore consists of seven phases used by the BACKUP and RESTORE commands as follows:



IDCBP01	**BACKUP FSR** Contains all executable modules for the BACKUP command.
IDCBP02	**Pregenerated Control Blocks** Contains the pregenerated control blocks and nonreentrant I/O routines for the BACKUP and RESTORE commands.
IDCBP03	**Buffersize Tables** Contains the buffersize tables used during backup.
IDCCDBP	**BACKUP Command Descriptor** Contains the command descriptor to be used by the Access Method Services Reader/Interpreter to analyze a BACKUP command and to construct the appropriate Function Data Table.
IDCCDRT	**RESTORE Command Descriptor** Contains the command descriptor to be used by the Access Method Services Reader/Interpreter to analyze a RESTORE command and to construct the appropriate Function Data Table.
IDCRT01	**RESTORE FSR** Contains all executable modules for the RESTORE command.
IDCTSBP0	**Backup/Restore Static Text** Contains the format structures for the messages issued by VSE/VSAM Backup/Restore.

Phase-to-Module Relationship

This section lists which modules belong to the individual phases for VSE/VSAM Backup/Restore. They are listed in the order in which they are included at link-edit.

Phase Name	**Module Name**
IDCBP01	IDCBPFSR
	IDCBPMSH
	IDCBPSLE
	IDCBPVOP
	IDCBPBDR
	IDCBPVCL
	IDCBPBBF
	IDCBPWOH
	IDCBPBDS
	IDCBPDDR
	IDCBPDDW
	IDCBPCRB
	IDCBPNBV
	IDCBPBPV
	IDCBPLVO
	IDCBPSXL
	IDCBPBLE
	IDCBPCAU
	IDCBPALE
	IDCBPOON
	IDCBPRSL
	IDCBPCMA
	IDCBPDYB
	IDCBPADE
	IDCBPSRD
	IDCBPMDE
	IDCBPOVC
	IDCBPBPO
	IDCBPBPC
	IDCBPPXL
	IDCBPDYS
	IDCBPRVB
	IDCBPRVL
	IDCBPRVD
IDCBPO2	IDCBPBRB
	IDCBPBCA
	IDCBPIOM
	IDCRTRCA
IDCBP03	IDCBPBST
IDCRT01	IDCRTFSR
	IDCBPMSH
	IDCRTROH
	IDCRTDFO
	IDCRTBFV
	IDCRTDVO
	IDCBPVOP
	IDCBPBDR
	IDCBPVCL
	IDCRTBBR
	IDCRTBDX
	IDCRTRDS
	IDCRTDWR
	IDCRTDWW
	IDCBPCRB
	IDCRTACA
	IDCRTWRX
	IDCRTGNX
	IDCRTRDX
	IDCRTWRS
	IDCRTGEX
	IDCRTCLX
	IDCRTREV
	IDCRTPFO
	IDCRTMDS
	IDCRTMSS
	IDCRTMTN
	IDCRTOPI
	IDCRTMTL
	IDCRTMTS
	IDCRTRTO
	IDCBPCMA
	IDCRTBRL
	IDCBPSXL
	IDCRTRVX
	IDCBPRVB
	IDCRTRTC

IDCTSBP0	IDCTSBP0
IDCCDBP	IDCCDBP
IDCCDRT	IDCCDRT

Phase-to-Link Book Relationship

This section lists the link books for VSE/VSAM Backup/Restore and the phases that can be linked by means of the individual link books. In order to not have to relink unnecessary phases of VSE/VSAM Backup/Restore in case of a required fix, a separate link book is provided for each phase.

Phase Name	Link Book Name
IDCTSBP0	IDCCMZ3
IDCBP01	IDCCMZ4
IDCBP02	IDCCMZ5
IDCBP03	IDCCMZ6
IDCCDBP	IDCCMZ7
IDCRT01	IDCCMZ8
IDCCDRT	IDCCMZ9

Chapter 6: Macro Directory

VSE/VSAM Backup/Restore has the following macros:

IDCDFB00	**Backup/Restore Block (BRB)** Generates a dummy section or actual code for the Backup/Restore Block.
IDCDFB01	**Directory Control Header (DCH)** Generates a dummy section or actual code for the Directory Control Header.
IDCDFB02	**Directory Block Header (DBH)** Generates a dummy section of the Directory Block Header.
IDCDFB03	**Directory Entries (DE)** Generates dummy sections of the external (EDE) and internal (IDE) directory entries.
IDCDFB04	**Locate Area Control Header (LCH)** Generates a dummy section or actual code for the Locate Area Control Header.
IDCDFB05	**Locate Area Block Header (LBH)** Generates a dummy section for the Locate Area Block Header.
IDCDFB06	**Data Set Control Header (DSH)** Generates a dummy section or actual code for the Data Set Control Header.
IDCDFB07	**Component Definition Block (CDB)** Generates a dummy section or actual code for a Component Definition Block, which is part of the Data Set Control Header.
IDCDFB08	**Buffer Pool Header (BPH)** Generates a dummy section or actual code for the Buffer Pool Header.
IDCDFB09	**Buffer Definition Block (BDB)** Generates a dummy section for the Buffer Definition Block.
IDCDFB10	**Request Control Section (RCS)** Generates a dummy section or actual code for Request Control Sections, which are part of the Buffer Definition Block.
IDCDFB11	**Index Buffer Block (XBB)** Generates a dummy section for the Index Buffer Block.
IDCDFB12	**Backup File Header (BFH)** Generates a dummy section or actual code for the Backup File Header.
IDCDFB13	**Tape Command Parameter List (TCP)** Generates a dummy section or actual code for the Tape Command Parameter List.
IDCDFB14	**VSAM Data Set Work Area (VDW)** Generates a dummy section or actual code for the VSAM Data Set Work Area.
IDCDFB15	**Volume List (VL)** Generates dummy sections for the layouts of a Volume List Block (VLB) and a Volume List Entry (VLE).
IDCDFB16	**Channel Command Word (CCW)** Generates a dummy section and equates for a channel command word.
IDCDFB17	**DTFMT Layout (DTF)** Generates a dummy section for the layout of a DTFMT.
IDCDFB18	**Volume Label (VOL1)** Generates a dummy section for the layout of a VOL1 label.
IDCDFB19	**I/O Module Header (IOH)** Generates a dummy section for the layout of the header portion of the module IDCBPIOM.
IDCDFB20	**GENL Parameter List (GENL)** Generates a dummy section for the GENL parameter list to be used for a LOAD macro with TEXT=NO.
IDCDFB21	**Fix List (FXL)** Generates a dummy section for the fix list to be used during the construction of the buffer pools for BACKUP and RESTORE.
IDCDFB22	**Inter-Module Trace Table (MTT)** Generates a dummy section describing the layout of the Access Method Services Inter-Module Trace Table.
IDCDFB23	**Map Data Set Work Area (MWK)** Generates a dummy section of the work area used by IDCRTMDS and IDCRTMSS during restoration of a data set when file modifications are made (moving files to volume of different device type, or specification of DATARECORDS or INDEXCISIZE).
IDCDFB24	**Map Volume Characteristics Table (VCT)** Generates a dummy section describing the structure of the Volume Characteristics Table blocks and entries.
IDCDFB30	**Backup Catalog Area (BCA)** Generates a dummy section or actual code for the Backup Catalog Area.
IDCDFB31	**Restore Catalog Area (RCA)** Generates a dummy section or actual code for the Restore Catalog Area.
IDCDFB32	**Locate Control List (LCL)** Generates a dummy section or actual code for the Locate Control List, a sub-structure of the Backup Catalog Area.
IDCDFB33	**Define Control List (DCL)** Generates a dummy section or actual code for the Define Control List, a sub-structure of the Restore Catalog Area.
IDCDFB34	**Catalog Parameter List (CTGPL)** Generates a dummy section or actual code for a Catalog Parameter List.
IDCDFB35	**Catalog Field Vector Table (CTGFV)** Generates a dummy section or actual code for a Catalog Field Vector Table.
IDCDFB36	**Catalog Field Parameter List (CTGFL)** Generates a dummy section or actual code for a Catalog Field Parameter List.

IDCDFB37	**Catalog Cluster Record (CCR)** Generates a dummy section for the layout of a catalog cluster record.
IDCDFB38	**Extension Record (EXR)** Generates a dummy section for the layout of a catalog extension record.
IDCDFB39	**Group Occurrence Pointer (GOP)** Generates a dummy section for the layout of a Group Occurrence Pointer.
IDCDFB40	**Object Header (OHD)** Generates dummy sections for the elements of the Object Header, such as Object Header Control Portion (OHC), the Object Header Catalog Dictionary (OCD), or the entries of the Catalog Information Area.
IDCDFB41	**Dummy Record (DRD)** Generates a dummy section for the layout of a dummy record.
IDCDFB42	**Restore Member List Entry (RLE)** Generates a dummy section for the layout of a Restore Member List Entry.
IDCDFB43	**Index Information Block (XIB)** Generates a dummy section for the layout of an Index Information Block.
IDCDFB44	**Index Header (XHD)** Generates a dummy section for the layout of the header of an index record.
IDCDFB50	**Function Data Table (FDT)** Generates dummy sections for the layout of the elements of the Function Data Table for the BACKUP and RESTORE commands.
IDCDFB60	**Message Codes (MSC)** Generates equates for all internal message codes and condition codes used by VSE/VSAM Backup/Restore.
IDCDFB70	**Module Initialization** Generates code for the module initialization of all VSE/VSAM Backup/Restore modules.
IDCDFB71	**Module Termination** Generates code for the termination of all VSE/VSAM Backup/Restore Modules.
IDCDFB72	**Error Code Setting** Generates code for the setting of the internal error codes and the condition codes used by VSE/VSAM Backup/Restore.
IDCDFB73	**Execute I/O** Generates code for the issuance of an EXCP.
IDCDFB74	**Wait I/O** Generates code for waiting for the completion of an I/O operation.
IDCDFB75	**Re-Entrant Load** Generates code for the re-entrant loading of the phase IDCBP02 containing the Backup/Restore Block, the Backup Catalog Area, the Restore Catalog Area, and the DTF I/O modules.
IDCDFB76	**Convert Time** Converts the time of day and the date into printable format.
IDCDFB77	**Convert RBA** Generates code for RBA conversion (IDCBPCRB).
IDCDFB78	**Next Backup Volume** Generates code for the Next-Backup-Volume function (IDCBPNBV).
IDCDFB79	**Restore EOV** Generates code for the Restore-EOV function (IDCRTREV).
IDCDFB80	**Message Handler** Generates code for the Message Handler function (IDCBPMSH).
IDCDFB81	**Add Control Area** Generates code for the Add-Control-Area function (IDCRTACA).

Chapter 7: Diagnostic Aids

VSE/VSAM Backup/Restore invokes Access Method Services functions. Accordingly, the diagnostic aids for Access Method Services apply as far as VSE/VSAM Backup/Restore supports the diagnostic capability. For corresponding detail, use *VSE/VSAM Access Method Services Logic.*

Trace Tables

VSE/VSAM Backup/Restore supports inter-module trace points. At the beginning of each module (except where critical to performance) the trace-ID of the module is stored in the Inter-Module Trace Table. Upon exit from a module, the caller's trace-ID is restored so that the Inter-Module Trace Table correctly reflects the flow of control through the VSE/VSAM Backup/Restore modules.

Intra-module trace points are not supported by the VSE/VSAM Backup/Restore modules because the individual modules are small.

Trace Point to Module Cross Reference

The following list contains the trace points set by VSE/VSAM Backup/Restore modules. The trace points are set at the beginning of these modules. In general, the trace-ID corresponds to the last three letters of the module name, padded with one blank.

The trace-IDs for the modules IDCBPFSR and IDCRTFSR are an exception. They are equal to the last 4 characters of the phase names for the BACKUP FSR (IDCBP01) and the RESTORE FSR (IDCRT01) and are BP01 and RT01 respectively.

Trace Point	Module Name	Function
ACA	IDCRTACA	Add Control Area
ADE	IDCBPADE	Add Directory Entry
ALE	IDCBPALE	Add Locate Entry
BBF	IDCBPBBF	Build Backup Buffers
BBR	IDCRTBBR	Build Restore Buffer
BDR	IDCBPBDR	Build RPSTAB
BDS	IDCBPBDS	Backup Data Set
BDX	IDCRTBDX	Build XIB
BFV	IDCRTBFV	Build CTGFV
BLE	IDCBPBLE	Build Locate Entry
BPC	IDCBPBPC	Backup Close
BPO	IDCBPBPO	Backup Open
BPV	IDCBPBPV	Backup EOV
BP01	IDCBPFSR	BACKUP FSR
BRL	IDCRTBRL	Build Restore List
CAU	IDCBPCAU	Convert Allocation Units
CLX	IDCRTCLX	Close Index
CMA	IDCBPCMA	Command Analyzer
CRB	IDCBPCRB	Convert RBA
DFO	IDCRTDFO	Define Object
DVO	IDCRTDVO	Delete VSAM Object
DYB	IDCBPDYB	Directory Build
DYS	IDCBPDYS	Directory Sort
GEX	IDCRTGEX	Get Extent
GNX	IDCRTGNX	Get Next Index Record
LVO	IDCBPLVO	Locate VSAM Object
MDE	IDCBPMDE	Move Directory Entry
MDS	IDCRTMDS	Remap Data Set
MSH	IDCBPMSH	Message Handler
MSS	IDCRTMSS	Remap Sequence Set
MTL	IDCRTMTL	Mount Later
MTN	IDCRTMTN	Mount Next
MTS	IDCRTMTS	Mount Specific
NBV	IDCBPNBV	Next Backup Volume
OON	IDCBPOON	Obtain Object Name
OPI	IDCRTOPI	Operator Interface
OVC	IDCBPOVC	Open VSAM Catalog
PFO	IDCRTPFO	Preformat Function
PXL	IDCBPPXL	Print XREF
RDS	IDCRTRDS	Restore Data Set
RDX	IDCRTRDX	Read Index
REV	IDCRTREV	Restore EOV
ROH	IDCRTROH	Read Object Header
RSL	IDCBPRSL	Reset Locate Area
RTC	IDCRTRTC	Restore Close
RTO	IDCRTRTO	Restore Open
RT01	IDCRTFSR	RESTORE FSR
RVB	IDCBPRVB	Remove Buffers
RVD	IDCBPRVD	Remove Directory
RVL	IDCBPRVL	Remove Locate Area
RVX	IDCRTRVX	Remove XIB
SLE	IDCBPSLE	Secure Locate Entry
SRD	IDCBPSRD	Search Directory
SXL	IDCBPSXL	Scan Exclusion List
VCL	IDCBPVCL	VSAM Close
VOP	IDCBPVOP	VSAM Open
WOH	IDCBPWOH	Write Object Header
WRS	IDCRTWRS	Write SEOF
WRX	IDCRTWRX	Write Index
none *	IDCBPDDR	Data Disk Read
none *	IDCBPDDW	Data Disk Wait
none *	IDCRTDWR	Data Disk Write
none *	IDCRTDWW	Data Write Wait

* No trace point provided because module's performance is critical.

Dump Points

VSE/VSAM Backup/Restore does not support dump points.

Abort Codes

The following list identifies the ABORT codes set by modules of VSE/VSAM Backup/Restore.

Module Name	Code	Cause
IDCBPFSR	28	No virtual storage available to load the Backup/Restore Block.
	80	The Backup/Restore Block was not found in the system libraries.
IDCRTFSR	28	No virtual storage available to load the Backup/Restore Block.
	80	The Backup/Restore Block was not found in the system libraries.

How to Find the Backup/Restore Block

For all modules of VSE/VSAM Backup/Restore, register 13 points to the Backup/Restore Block. Offset 72-75 should contain the characters 'BRBb̸', the identifier for the Backup/Restore Block.

If register 13 does not point to the BRB because a service invoked by VSE/VSAM Backup/Restore has control, you can find the BRB by scanning down the right side of a dump for the identifier 'BRBb̸' at offset 72 of the Backup/Restore Block.

How to Find the GDT and FDT from the BRB

The Backup/Restore Block points to the Global Data Table and the Function Data Table for the executed command.

The field labeled BRBGDT points to the Global Data Table. The field labeled BRBFDT points to the Function Data Table. The field BRBREQ identifies the command being executed:

4 - BACKUP command being executed.
8 - RESTORE command being executed.

How to Find the Inter-module Trace Table

After you have found the Global Data Table from the Backup/Restore Block, you can find the Inter-Module Trace Table address at offset 8 of the Global Data Table.

How to Determine the Active Module

If register 13 points to the Backup/Restore Block, you can determine which module of VSE/VSAM Backup/Restore is active: In general, register 12 is used as base register. If you subtract X'12' from the value in register 12, the result points to the name of the module that is in control.

Exceptions are the modules IDCBPDDR and IDCBPDDW for BACKUP and IDCRTDWR and IDCRTDWW for RESTORE. For them, after subtracting X'12' from register 12, the result points to the module name of the caller, IDCBPBDS or IDCRTRDS, respectively.

How to Determine the Position in the Function Tree

Many modules of VSE/VSAM Backup/Restore are called from different locations. If you want to determine where you are in the function tree (see Chapter 4), do as follows:

The Backup/Restore Block contains a save area pool used to store the registers of the calling functions. The inter-module trace-ID of the caller is saved in front of the registers. The BRB save area pool starts at the label BRBSAP. The field BRBNSA of the Backup/Restore Block points to the next available position.

After you find which module is active (by subtracting X'12' from register 12, as described before), determine how many registers it stores (macro IDCDFB70). By subtracting the size of a trace-ID and the size of the registers stored from the address contained in the field BRBNSA, you come to the trace-ID of the calling module. By looking up how many registers it, in turn, stores, you can come to the trace-ID of its caller. Continue until you reach the beginning of the save area pool. This process is illustrated in Figure 7-1.

How to Determine the Last Message

The field BRBERC of the Backup/Restore Block contains the internal message code of the last message printed or being printed by VSE/VSAM Backup/Restore. See macro IDCDFB60 for message codes. The field BRBMID contains the trace-ID of the module that caused the message to be issued.

How to Determine the Last and the Maximum Condition Codes

The fields BRBLCC and BRBMCC contain the last condition code and the maximum condition code set by any VSE/VSAM Backup/Restore module. The field BRBERCNT indicates the number of errors encountered thus far by VSE/VSAM Backup/Restore.

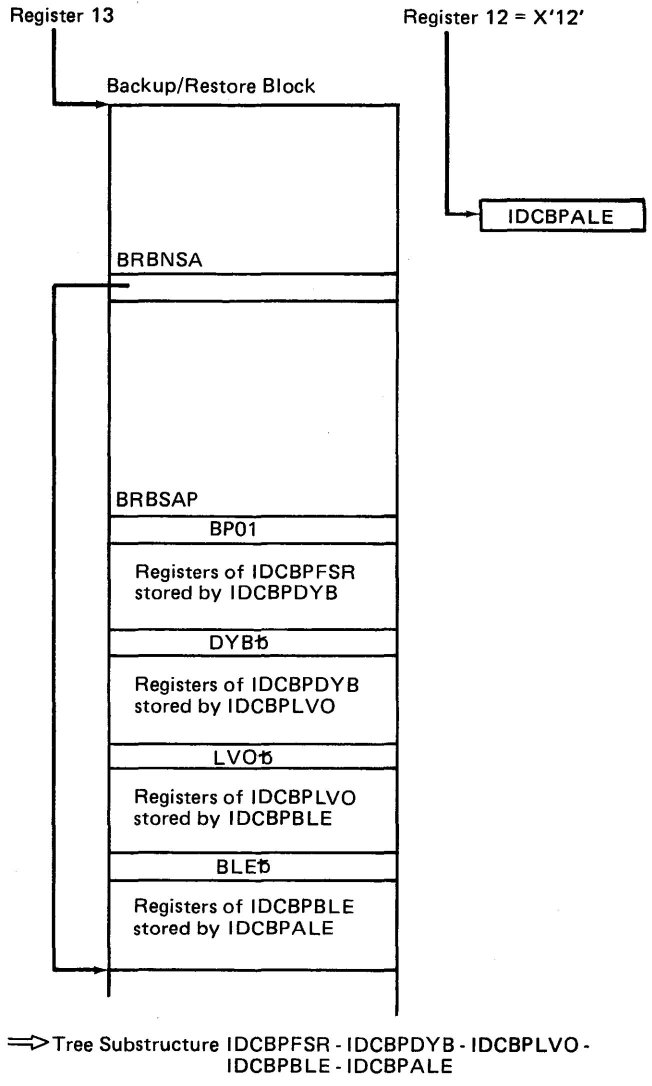

Register 13
Register 12 = X'12'
Backup/Restore Block
IDCBPALE
BRBNSA
BRBSAP
BP01
Registers of IDCBPFSR stored by IDCBPDYB
DYBb̸
Registers of IDCBPDYB stored by IDCBPLVO
LVOb̸
Registers of IDCBPLVO stored by IDCBPBLE
BLEb̸
Registers of IDCBPBLE stored by IDCBPALE
⇒Tree Substructure IDCBPFSR - IDCBPDYB - IDCBPLVO - IDCBPBLE - IDCBPALE

Figure 7-1. Determining the VSE/VSAM Backup/Restore Flow of Control

Note: The chain of modules derived by this method is different from the flow of control represented by the Inter-Module Trace Table. The chain derived by the method just described represents the last module invoked at each level of the function tree described in Chapter 4.

Message-to-Module Cross Reference

Message	Text	Module
IDC400A	MOUNT VOLUME xxx OF BACKUP FILE ON SYS004=cuu	IDCRTMTN IDCRTMTS
IDC401I	BACKUP VOLUME REQUIRED FOR file-id	IDCRTMTL
IDC402A	MOUNT VOLUME xxx OR HIGHER ON SYS004=cuu	IDCRTMTL
IDC403I	TIME STAMP MISMATCH. BACKUP FILE CREATED ON date AT hh:mm:ss	IDCRTMTL IDCRTMTN IDCRTMTS
IDC0001I	FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS xxx	IDCBPFSR IDCRTFSR
IDC3003I	FUNCTION TERMINATED. CONDITION CODE IS nnn	IDCBPFSR IDCRTFSR
IDC3004I	FUNCTION TERMINATED. INSUFFICIENT MAIN STORAGE.	IDCBPADE IDCBPALE IDCBPBBF IDCBPBDS IDCBPBPO IDCBPBPV IDCBPLVO IDCBPOON IDCBPVOP IDCBPWOH IDCRTBBR IDCRTBDX IDCRTBFV IDCRTBRL IDCRTDFO IDCRTGEX IDCRTMDS IDCRTMSS IDCRTRDS IDCRTRDX IDCRTROH IDCRTRTO IDCRTWRS IDCRTWRX
IDC01300I	BACKUP FILE CREATED ON date AT hh:mm:ss	IDCBPPXL
IDC01301I	RESTORE'S BACKUP FILE CREATED ON date AT hh:mm:ss	IDCRTRTO
IDC01302I	SUCCESSFUL RESTORATION OF file-id	IDCRTFSR
IDC01303I	SUCCESSFUL DELETION OF file-id — ENTRY TYPE=x	IDCRTDVO
IDC01304I	SUCCESSFUL DEFINITION OF file-id	IDCRTDFO
IDC01305I	PASSWORDS SUPPRESSED FOR file-id	IDCBPWOH
IDC11306I	NO OBJECT FOR entryname	IDCBPBPC IDCBPDYB IDCRTBRL
IDC11307I	SKIPPING RESTORATION OF file-id	IDCRTFSR
IDC11345I	CANNOT CONVERT ALLOCATION UNITS FOR file-id	IDCBPCAU
IDC21308I	CANNOT CLOSE file-id	IDCBPVCL
IDC21309I	**VSAM CLOSE ERROR IS nnn	IDCBPVCL
IDC31310I	INVALID GENERIC NAME file-id	IDCBPCMA
IDC31311I	ERROR EXPANDING GENERIC NAME entryname	IDCBPDYB
IDC31312I	**VSAM PHYSICAL ERROR RETURN CODE IS nnn	IDCBPDYB
IDC31313I	PASSWORD CONFLICT FOR file-id	IDCBPDYB
IDC31314I	**CONFLICTING OBJECT IS file-id	IDCBPDYB
IDC31315I	CANNOT LOCATE CATALOG	IDCBPOVC IDCRTDFO
IDC31316I	**VSAM CATALOG RETURN CODE IS nnn REASON CODE IS IGG0CLxx-mmm	IDCBPLVO IDCBPOVC IDCRTDFO IDCRTDVO
IDC31317I	CANNOT OPEN VSAM CATALOG	IDCBPOVC
IDC31318I	CATALOG VOLUME ERROR	IDCBPOON
IDC31319I	CATALOG EXTENT ERROR	IDCBPOON
IDC31320I	CATALOG I/O ERROR	IDCBPOON
IDC31321I	CANNOT RETRIEVE CATALOG INFORMATION FOR file-id	IDCBPLVO
IDC31322I	CANNOT LOCATE ASSOCIATION OF file-id	IDCBPLVO

IDC31323I	CANNOT LOCATE BASE CLUSTER OF file-id	IDCBPLVO
IDC31324I	CANNOT OPEN file-id	IDCBPVOP
IDC31325I	**VSAM OPEN ERROR IS nnn	IDCBPOVC IDCBPVOP
IDC31326I	NO BACKUP OF file-id — CANNOT BE RESTORED	IDCBPVOP
IDC31327I	EXTENT ERROR FOR file-id	IDCBPBDS IDCRTMDS IDCRTPFO IDCRTRDS IDCRTRDX IDCRTWRS IDCRTWRX
IDC31328I	VOLUME ERROR FOR file-id	IDCBPBDS IDCRTMDS IDCRTPFO IDCRTRDS IDCRTRDX IDCRTWRS IDCRTWRX
IDC31329I	DISK I/O ERROR FOR file-id	IDCBPBDS IDCRTACA IDCRTCLX IDCRTMDS IDCRTPFO IDCRTRDS IDCRTWRS
IDC31330I	BACKUP FILE I/O ERROR	IDCBPBDS IDCBPBPC IDCBPBPO IDCBPBPV IDCBPNBV IDCBPWOH IDCRTMTN IDCRTMDS IDCRTRDS IDCRTREV IDCRTROH IDCRTRTO
IDC31331I	USECLASS ERROR FOR file-id	IDCRTDFO
IDC31332I	NO DNAME FOR UNIQUE COMPONENT OF file-id	IDCRTDFO
IDC31333I	CANNOT FIND OBJECT file-id	IDCRTFSR
IDC31334I	CANNOT DELETE OLD VERSION OR ASSOCIATION OF file-id	IDCRTDVO
IDC31335I	CANNOT DEFINE file-id	IDCRTDFO
IDC31336I	CANNOT RESTORE SAM ESDS file-id	IDCRTDFO
IDC31337I	CANNOT RESTORE file-id ON SPECIFIED VOLUME	IDCBPVOP
IDC31338I	CANNOT EXTEND file-id	IDCRTGEX
IDC31339I	MORE THAN 255 INDEX LEVELS FOR file-id	IDCRTACA
IDC31340I	BACKUP FILE IN ERROR	IDCRTMDS IDCRTRDS IDCRTREV IDCRTROH IDCRTRTO
IDC31341I	INCOMPLETE BACKUP COPY OF file-id	IDCRTREV
IDC31342I	RESTORE TERMINATED. FAILURE TO MOUNT BACKUP VOLUME.	IDCRTOPI
IDC31343I	FUNCTION TERMINATED. MAXIMUM NUMBER OF ERRORS EXCEEDED.	IDCBPDYB IDCBPFSR IDCBPLVO IDCRTFSR
IDC31344I	CANNOT DEFINE file-id ON SPECIFIED VOLUME	IDCRTDFO

Index

VSE/VSAM Backup/Restore Feature Logic (File No. S370-30) Printed in U.S.A. LY24-5213-1

VSE/VSAM Backup/Restore
Feature Logic
LY24-5213-1

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

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.*

	Yes	*No*
• Does the publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

• What is your occupation? ______________________

• How do you use this publication:

As an introduction to the subject?	☐	As an instructor in class?	☐
For advanced knowledge of the subject?	☐	As a student in class?	☐
To learn about operating procedures?	☐	As a reference manual?	☐

Your comments:

If you would like a reply, please supply your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

LY24-5213-1

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 G60
P. O. Box 6
Endicott, New York 13760

Fold   Fold

If you would like a reply, *please print:*

Your Name ______

Company Name ______ *Department* ______

Street Address ______

City ______

State ______ *Zip Code* ______

IBM Branch Office serving you ______

IBM®