Xerox Universal Time-Sharing System (UTS)

Sigma 6/7/9 Computers

File Management
Technical Manual

90 19 89A

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245
213 679-4511

XEROX

Xerox Universal Time-Sharing System (UTS)

Sigma 6/7/9 Computers

File Management
Technical Manual

FIRST EDITION

90 19 89A

February, 1974

Price: $6.75

Printed in U.S.A.

NOTICE

This publication documents disk and tape file management in the Universal Time-Sharing System (UTS) for Sigma 6/7/9 computers. All material in this manual reflects the C01 version of UTS.

RELATED PUBLICATIONS

Title	Publication No.
UTS Overview and Index Technical Manual	90 19 84
UTS Basic Control and Basic I/O Technical Manual	90 19 85
UTS System and Memory Management Technical Manual	90 19 86
UTS Symbiont and Job Management Technical Manual	90 19 87
UTS Operator Communication and Monitor Services Technical Manual	90 19 88
UTS Reliability and Maintainability Technical Manual	90 19 90
UTS Interrupt Driven Tasks Technical Manual[t]	90 19 91
UTS Initialization and Recovery Technical Manual	90 19 92
UTS Command Processors Technical Manual	90 19 93
UTS System Processors Technical Manual	90 19 94
UTS Data Bases Technical Manual	90 19 95

[t] Not published as of the publication date given on the title page of this manual. Refer to the PAL Manual for current availability.

The specifications of the softwar[redacted]cribed in this publication are subject to change without notice. The availability or performance of some features may depend on a specific confi[redacted]of equipment such as additional tape units or larger memory. Customers should consult their Xerox sales representative for details.

CONTENTS

ID

FUNCTIONAL OVERVIEW

A file is an organized collection of information that may be created, retrieved, modified or deleted only through the Monitor system.

A file is identified by the account number in which it was created and by its name. Access to a file may be limited to the users operating under account numbers that have been specified as being permitted to retrieve or modify that file. Access may be further restricted to those users specifying the proper password.

A file may be organized as consecutive file, keyed, or random. In a consecutive, the records may be accessed only in the sequence in which they were originally written. In a keyed file, each record has an associated name, or key. Records in a keyed file may be accessed directly by specific key values, or sequentially, according to their order in the file. A random file consists of contiguous granules rather than a group of records. Random files are accessed by granule number relative to the beginning of the file.

A disk file resides on the Monitor's secondary storage. UTS uses both the RAD and disk pack devices for secondary storage. Any combination of these devices can be defined for a UTS system at SYSGEN time. A disk pack device has dismountable volumes and can be declared either a public or private device at SYSGEN time, while a disk device, not having dismountable volumes, can only be declared a public device. A public disk pack has only one volume that can be recognized by UTS, and that volume must be mounted at all times while the system is active. A private disk pack device has any number of dismountable volumes that can be recognized by UTS. The Monitor requires that only those volumes needed for execution of the user's job be made available and be mounted. A public file resides on public devices (RAD and/or disk pack); a private file resides on private disk pack volumes.

A private volume set is defined as a collection of removable volumes that the user has grouped together containing any number of files with any type of organization (consecutive, keyed, or random). All files in a private volume set must belong to the same account. A private volume set is identified by the volume serial numbers specified in the SN option of the !ASSIGN command when the first file is written on the set. Volumes may be added to the set by entering a new volume serial number in the SN list, but a volume may not be removed.

Keyed and consecutive file space is allocated on a demand basis as the file is being created or updated, therefore, such files do not necessarily exist in contiguous areas on a RAD or disk pack and can exist on many different physical devices. Random file space is allocated when the file is opened for output. The size of a random file can never be changed. For allocation, a disk pack device is partitioned into logical units, either granule or cylinder. RADs are partitioned and allocated in granule units only. A granule unit equals 512 words and is equivalent to two sectors. A cylinder unit equals 30 granules and is equivalent to one-half of a physical disk pack cylinder. A public file can be allocated in granules or cylinder units; a private file is always allocated in cylinder units.

A file may be shared among several users providing that none of them updates the file (i.e., opens the file in the INOUT mode). For keyed and consecutive files, a user may have only one open DCB in the output or scratch mode referencing an output file (a "new" file). However, if another version of the file (a "current" file) already exists, any number of DCBs in the input mode may read that file. If the new file is closed with the SAVE option, the disk space for the current file is released and the disk space for the new file is closed. Furthermore, the user must open the new file first. For random files, a user may have only one DCB opened to a file in the output, update, or scratch mode. However, any number of DCBs can be opened to the same file in the input mode. Random files differ from keyed and consecutive files because only one version (rather than a new or current version) of a named file exists, so that DCBs opened in the input mode would be referencing the same file being written or updated.

When a file is opened for creation, the user may specify whether space allocated to the file is to be saved or released when the file is closed. If file disposition is not specified, an output or scratch file will be released and an input or update file will be saved. When the file is closed, the user can override the file disposition specified at open only to release the file. If the user does not close the file before the job step is either terminated or aborted, an output or scratch file will be released and an input or update file will be saved. A file that has been closed and saved can be declared to be temporary via M:TFILE. A temporary file is released by the Monitor when the job terminates. A temporary file may therefore be accessed by many job steps within the same job and then be released at job end. Once a file is declared to be temporary, there is no way that the file can be saved at job end, unless the file has a password. The M:TFILE process will not release a file with a password.

INTERFACES

CALs executed by the user for I/O requests are processed first by ENTRY and CALPROC. CALPROC puts the DCB address in register 6 and verifies that it is for a valid DCB. The function code from the first word of the FPT is loaded into register 8, and register 7 is set with the address of the FPT. CALPROC then transfers control to the routine that handles the particular function. When the function is completed, the file management routine returns to CALPROC which then passes exit parameters back to the user and returns via routines in SSS.

The various CAL processing routines, among other things, check assignment type and transfer control to device routines, disk file routines, or labeled tape routines.

When I/O operations are required, the file management routines normally call QUEUE1 to enqueue the request with or without end-action. The end-action routines perform various tasks such as moving a data record segment releasing a buffer. Several I/O requests for the same file may be in the queue at the same time.

SYSGEN builds the AVRTBL and allocation tables and allocates space for the CFU buffers.

Private disk pack volumes are initialized by the VOLINIT processor.

DCBs are part of the user code and are created either by the Loader or the user. CCI creates an ASSIGN/MERGE record so that a DCB can be modified by OPEN.

STEP allocates space for the FPOOL and IPOOL buffers at run-time.

OPERATIONAL OVERVIEW

The File Management System

Access to user files is via a hierarchy of disk resident Monitor files. The top file is an Account Directory, which contains a directory of all accounts that have public user disk files. There is òne account directory for all public files in the system (the Public File Account Directory). Each account has its own file directory, which contains a directory of all files in the account. Each file has a File Information Table (FIT), which is part of the file directory for random file, and part of the file itself for keyed and consecutive files, and contains all the information necessary to open a file, such as its organization, location, password, etc.

To locate a public file, the public account directory is searched for the file account number. The account number entry contains the disk address of the account's file directory. The file directory is searched for the filename. The filename entry contains the disk address of the file's FIT. The FIT contains the disk address of the file.

Private files are located via AVR and MOUNT logic.

All open files have a Current File Usage Table (CFU) which is core-resident and allocated from a pool of CFU buffers established at SYSGEN time. The CFU contains all the file information necessary to the system while the file is opened. Basically, the CFU is the core-resident subset of the file's FIT, which is disk-resident. Since more than one DCB can be opened to a file, the DCB rather than the CFU contains user-specific information about the file, such as access means, current record position, etc. An open DCB contains the address of the CFU associated with the file and, if a file is being shared in the input mode, more than one DCB will point to the same CFU.

The system has two dedicated CFUs called ACNCFU and FILCFU. ACNCFU contains file-specific information about the Public Account Directory. This information is always core-resident in ACNCFU while the system is active. FILCFU contains file-specific information about the currently referenced public or private File Directory.

A keyed file consists of two parts, a Master Index and a set of data granules. The data granules contain the records in the file, which are packed in granule-size blocks. Data granules do not contain any system information. The Master Index is a collection of hierarchical levels of index blocks where the entries in a higher level point to index blocks at the next lower level, and the entries in the lowest level point to data records.

A consecutive file consists of granules containing the data of the records preceded by four bytes of control information per record generally. (See Section JA.18) A random file is devoid of system information. Record management and format of the file is the user's responsibility. Besides the security checks required for access to a file, the only checks made by the system are to prevent the user from reading or writing past the limits of the file. Functionally and operationally, a random file is a collection of contiguous granules on the specified device type. However, if a random file is larger than a disk pack in size, the file will extend beyond volume boundaries (if private) or device boundaries (if public). The file will completely fill the first N-1 volumes/ devices required with the remaining space allocated from the beginning of the Nth volume/device. For private files, the N volumes required for the file must have consecutive volume numbers; for public files, consecutive DCT indexes. This organization is transparent to the user and I/O requests are expressed as granule displacements relative to the beginning of file. The Monitor will compensate for files that cross volume or device boundaries when calculating disk addresses.

Data granules, index blocks, and Volume Table of Contents for private volume sets require 512-word buffers for I/O. Even though a user might only be using random files, an index buffer is required to read the account and file directories. Buffers are allocated from a pool established at STEP time. Each job is provided a minimum of six buffers in this pool.

SYSGEN creates an AVRTBL entry for each disk pack device in the system. An AVRTBL entry is used to contain the serial number and mount information about the disk pack volume mounted, if any, on the associated device. When a private volume is required, the system searches the AVRTBL entries associated with private disk pack devices for the requested serial number. If the volume is not mounted, the system searches for an available disk pack device and asks the operator to mount the volume. Once the volume is mounted, its serial number is verified.

Each RAD and disk pack device has an Allocation Table (HGP), created by SYSGEN, that contains the maps which control the allocation of the granule/cylinder units in the file and symbiont storage areas of the device. Allocation tables for private disk pack devices contain information that resides on the private volume. When a volume is verified, storage-allocation information is moved from the volume to the allocation table. When a DCB's use of a volume is terminated (either when the file is closed or when a consecutive file switches to the next volume), the information from the allocation table is written back on the private volume. The Monitor ghost job ALLOCAT manages the allocation tables for public devices.

FILE MANAGEMENT PROCEDURES

File Management procedures are the user-initiated Monitor instructions for manipulating records within a file. Because M:OPEN, M:CLOSE, M:READ, and M:WRITE are complex procedures involving several modules, a brief overall description of these procedures is given in the following paragraphs.

Each of the procedures can be viewed as consisting of three basic steps: (1) analyzing the environment, which normally involves examining the FPT data to the DCB, making legality checks, and transferring control as indicated by file type, storage medium and function; (2) finding the entity, which means locating a record or a file, or finding a place to put a record or a file; and (3) performing the function of the procedure call. In the following description of the four procedure calls, only steps (2) and (3) will be explained since step (1) is essentially common to all of them and, except for special situations, requires no more detailed description than that given above.

M:OPEN

If the DCB is assigned to a public file, the Public File Account Directory is searched for the file's account number. If the DCB is assigned to a private file with keyed or random organization, all volumes in the private volume set are mounted and verified. If the DCB is assigned to a private file with consecutive organization, only the primary volume (which contains the volume set's account and file directories) is mounted and verified. Since a private volume set contains only files in a single account, the one entry in the volume set's account directory is checked against the account number specified for the file. If the account numbers are not the same, the file is not opened and an error code is returned to the user. The account directory entry, whether public or private, contains the disk address of the file directory associated with the requested account number. The appropriate file directory is searched for the specified filename. If the filename is found, the FIT is read. The password is checked and the job's account number is checked for read or write access to the file. If the DCB is a system DCB (M:) with a function of OUT or OUTIN and this is not the first time in this job that this DCB has been opened, then it is assumed that the user wants to add to the end of the file, which is a condition called file extension. If file extension applies, the function is changed to INOUT, the NEWKEY flag is set, and the file is positioned at the end-of-the-file.

If the filename is not found and function is IN or INOUT, an error return is given. If the function is OUT or OUTIN, the file must be created in the same account as specified for the job.

In all cases a user CFU is obtained. First, the active CFUs are searched to see if a file with the same name and account is already open. If not, an inactive CFU is initialized and information from the file's FIT is transferred to the CFU if the DCB's function is IN or INOUT.

If an existing CFU is located, and the file has keyed or consecutive organization, the function of the DCB being opened must be IN. If the CFU's function is also IN, the DCB is linked to the existing CFU. If the CFU's function is INOUT, the DCB cannot be opened and an error code is returned to the user. If the CFU's function is OUT or OUTIN, a new CFU is established (with information from the file's FIT) and linked to the existing CFU.

If an existing CFU is located and the file has random organization, there are several possibilities. If the CFU is IN and the DCB is IN, the DCB is hooked to this CFU. If the DCB is INOUT, another CFU is obtained and linked to this CFU. If the CFU is INOUT and the DCB is IN, another CFU is obtained and linked to this CFU. If the DCB is INOUT, an abnormal return is given.

M:WRITE

The M:WRITE procedure has four modes: (1) adding a record at the current position; (2) inserting a record in a file; (3) modifying an existing record; and (4) random writing. A record is added at current position, which defines the end, if the function is OUT or OUTIN and access is sequential, or if the function is INOUT, the last operation was not a read, and the file organization is consecutive.

A record is inserted if a key is specified and not found in the Master Index and either the function is INOUT or the access is direct. A record is modified if (1) a key is specified and found and either the function is INOUT or access is direct, or (2) no key is specified, the function is INOUT, and the last operation was a Read.

If operating in the add mode on disk, and the current file position is BOF, then the file is to be completely rewritten; therefore, all disk space previously allocated to the file is released, and a minimum of new space is obtained. If not at

BOF and file organization is sequential, and if the last operation was not a Write, all records forward of current position are deleted.

The record position is the current position in the master index, or if the file is a tape file, the current position in the blocking buffer.

If operating in the insert mode, the record position in the master index is following a smaller key than the new one, or BOF; and preceding a larger key, or EOF. If operating in the modify mode, the record position in the MI is that of the matching key.

For keyed disk files in either the add or insert mode of operation, a key entry is constructed and inserted into the master index at the current position. Trailing entries, if any, are moved down. If the insertion causes an overflow, the last entry of the index block is inserted at the beginning of either the next index block if there is one and it has room, or of a new index block that is obtained and linked in. The data is moved into the current blocking buffer. If the data does not all fit into the current blocking buffer, flags are set in the Master Index entry and blocks of 512-data words are written to disk directly from the user's buffer. If fewer than 512 words remain, the data is moved into a new blocking buffer. In every case, an entry is constructed and inserted into the Master Index for each record segment.

For keyed disk files in the modify mode, the data blocks associated with the existing record receive the data and the Master Index entries are adjusted to reflect any change in size from the original record. If the modified data record is larger than the original record (or any previous extension thereof), the excess data is handled in the same manner as continuation records in an insert operation.

M:READ

For disk files, a pointer (DCB:CMD) is maintained that points to the first Master Index entry of the last record read. When reading sequentially, CMD is moved in the current direction to the next beginning-of-record Master Index entry, unless the last operation was in the opposite direction from the current operation, in which case, CMD is already correctly positioned. With direct access, the Master Index is searched backward from current position until a smaller than specified key is found, then forward until either the specified or a larger key is found. An error return is made if the exact key is not found for either access. Encountering BOF or EOF results in an abnormal return.

The transfer of data is essentially the reverse of add and insert mode Writes, except that transmission of data terminates when either end-of-record is reached or the number of bytes requested is reduced to zero. Transfer of data is always in a forward direction, i.e., the first byte of the first record segment is transferred into the first byte of the user's buffer, etc.

M:CLOSE

The first steps in a Close are (1) to write out all buffers that were modified since they were last written, and (2) delay for completion of I/O. The last step is to close the DCB. For disk files there are seven basic procedures:

1. If the file is to be released, all of its disk space is released back to the system. If the account number or filename is not found in the appropriate account or file directory, no further action is taken. If the account number and filename are located, the entry is deleted from the file directory.

2. If an output or scratch file is to be saved and the account number is not located, a new account directory entry is created. Disk space is allocated for the new account's file directory, its address is inserted into the new account entry and the new file directory is initialized. An entry for the file is created, a FIT granule is allocated for random files, and its address is inserted into the file directory entry. The FIT is created from information in the CFU and DCB and written to the disk. This case can only happen for public files since the one account number in a private account directory is entered either when the volume is first initialized or when the first file is opened on the private volume set.

3. If an output or scratch file is to be saved and the account number is located, but the filename is not in the account's associated file directory, a new file directory entry is created. The filename is entered, a FIT granule is allocated for random files, and its disk address is saved. The FIT is created from information in the CFU and DCB and written to the disk.

4. If an output or update file is to be saved and the filename is in the located account's file directory, then a new version of the file is to replace an existing version. A FIT for the file's new version is created and written to disk. All disk space associated with the old file is released.

5. If an input file is to be saved, no further action is required.

6. If an update file is to be saved, its FIT is updated and no further action is required.

OTHER FILE MANAGEMENT CALs

The other File Management CALs are described under Module Descriptions as follows:

CAL	Routine	Module
M:CHECK	IOCHCK	IORT
M:DELREC	DELETE	DLT
M:PFILE	PFIL	POS
M:PRECORD	PRECORD	POS
M:REW	REW	POS
M:TFILE	MSRTFILE	TFILE
M:TRUNC	TRUNC	RDF
M:WEOF	WEOF	POS

SERIAL MOUNT CAPABILITY

This section describes implementation of the serial mount capability for consecutive files on a private volume set.

All files on a private volume set can cross volume boundaries. For keyed and random files, all volumes in the set must be mounted while the file is opened. For consecutive files, only one volume need be mounted at any one time. As another volume is needed, the system requests, if necessary, that it be mounted. Implementation of the serial mount capability for a consecutive file requires (1) that all entries in the Master Index on one volume point to data granules on the same volume and (2) that the disk address of the index block on the next volume (i.e., the FLINK) be available on the current volume. This means that when a consecutive file is being created (or added to) and there is no more space on the current volume, the last index block on the volume must point to an index block on the next volume. In other words, the FLINK in the last index block of the current volume must be a disk address on the next volume; but the next volume can't be mounted until the current volume is closed, and the current volume can't be closed until the last index block points to an index block on the next volume. The problem is solved by having a subset of the next volume's allocation table on the current volume. This subset is called the Next Volume's Allocation Table (NVAT) and contains 30 bits, where each bit represents a granule in cylcinder 0 of the next volume. Since a volume's Table of Contents (VTOC) is on granule 0 of cylinder 0, cylinder 0 is always marked as allocated in the next volume's cylinder bit maps. This means that the remaining 29 granules within cylinder 0 cannot be allocated from the cylinder bit maps on that volume. The granules within NVAT are only used when a consecutive file extends beyond volume boundaries. Since a cylinder unit has 30 granules, one volume can have a maximum of 29 consecutive files extending to another volume. Because new volumes can always be added to a private volume set, the NVAT on the last volume does not have any meaning until another volume is added. For example, if a private volume set has three volumes, NVAT on volume 1 points to volume 2; NVAT on volume 2 points to volume 3; and NVAT on volume 3 has no meaning. Note that there is no NVAT POINTING TO THE PRIMARY VOLUME. The remaining granules in cylinder 0 of the primary volume are allocated to the private volume set's file directory. This is done by setting NGAVAL and GAVAL in the file directory's Mini-FIT when the volume is initialized by the VOLINIT processor.

ID

IORT

PURPOSE

IORT contains primary routines used in processing M:READ, M:WRITE, M:DELREC, and M:CHECK CALs. It also contains numerous secondary functions, macro-subroutines, and end-action routines used throughout the I/O system.

SUBROUTINES

IOCHEK

Purpose: To process the M:CHECK CAL which returns the completion type (TYC) parameters on I/O completion.

Entry: B IOCHEK
(R7) = parameter list address-1,
R11 = normal return address, stack must be open by eight locations.

Exit: B PULLALLEXIT
(R8) = user address to return to (0 = CAL+1)
(R10) = error/abnormal code and DCB address

Operation: If the DCB is not open, IOCHEK exits. Otherwise, IOCHEK1 is called to wait for completion of all I/O for this DCB and to set R8 and R10 as indicated above. (See Appendix B, Monitor Error messages in the UTS Batch Processing Reference Manual.)

MSSRDWT

Purpose: To receive and do preliminary processing of M:READ and M:WRITE CALs by analyzing the FPT parameters and transferring control to the appropriate routine for further processing, and to process the M:DELREC FPT.

Entry: B MSRRDWT
(R5) = JIT address
(R7) = parameter list address (FPT address)-1
SR1 = operation code (i.e., CAL type)

Exit: To appropriate routine for further processing (see below).

Operation: Parameters present in the FPT or their defaults, where appropriate, are moved into the DCB (MSROPN is first called to open the DCB.) The various addresses and counts in the DCB are checked for legality, an error causing an exit to MSR01EXIT. If (R8) equals 13 (M:DELREC CAL), exit is made to DLT. If (R8) is greater than 17, an exit is made to T:AMRDWT in UCAL to perform an ASSIGN/MERGE. Control is transferred to a processing routine according to whether (R8) equals 16 or 17 (M:READ or M:WRITE), and ASN equals 1, 2, or 3 (disk, labeled tape, or device), as shown below.

RBLKEND This routine is now in the RDF module.

Purpose: To move record segment into user's buffer during end-action that follows a disk read initiated through QUEUE1.

Entry: BAL,SR4 RBLKEND
(R6) = BA (I/O buffer)
(R10) = BRF,Q4X, - 1,8,23

where:

BRF is the Buffer Release Flag: 0 indicates "release buffer".

Q4X is index into Q4AVL, shown below.

Q4AVL	0 15	16 31
0		user's current buffer word address
1	remaining buffer size	initial displacement

Exit: B SR4

Operation: The record segment is moved to the user's buffer and, if BRF is set (1), the DCB item, RBBI, is cleared to indicate that the buffer no longer has I/O in progress. If BRF is not set (0), the buffer is released. Either way, the Q4VAL entry is released and the routine exits.

RECTRAN

Purpose: To move a string of bytes between a user's buffer and a Monitor buffer.

Entry:
```
BAL,SR4  RECTRAN
(R1) = source buffer size (bytes), i.e., ending
       displacement
(R2) = destination buffer size (bytes)
(R3) = destination initial displacement (bytes)
(R4) = source initial displacement (bytes)
(D3) = destination buffer (word) address
(D4) = source buffer (word) address
```

Exit:
```
B  *SR4
(R3) = new destination displacement (R3 + number
       of bytes moved)
(R4) = new source displacement (R4 + number of
       bytes moved)
```

Operation: The MBS instruction is used to move the data. The number of bytes moved is the lesser of the remaining buffer sizes, i.e., R1-R4 or R2-R3, whichever is smaller.

MODEFRMA, MODEF

Either routine sets the DCT5 bit controlling mode from the DCB item, MOD. Entry at MODEFRMA is conditional, performing the operation only if the DCB item, DRC, is set; entry at MODEFRM performs the operation unconditionally. In either event, the various bits in the I/O command code (byte zero of R8) are set from the DCB items MOD, DIR, PUN, FBCD, and DRC. If DRC was set, the exit is skipping.

MSRRED

Purpose: To perform read for device DCBs.

Entry: B MSRRED

Exit: Normal/abnormal - B PULLALLEXIT
Error - B MSREXIT

Operation: If the device is not the C device, MSRRED sets up R8 for a QUEUE call and transfers control to MSROTHR (see below).

If the device is a C device, two possibilities exist: either CCBUF is empty, or it contains a record. If CCBUF is empty, MSRRED initiates a read (unless one is already in progress), and makes FORTRAN conversion, if required, on I/O completion.

The record in CCBUF either is or is not a control command. If it is a control command (! in column 1) and the user previously tried to read it (DCB item AGV), MSRRED sets up an error exit. If the user did not previously try to read the record, then MSRRED sets up an end-of-file abnormal-return exit.

If the record in CCBUF is not a control command, MSRRED moves the record to the user's buffer, initiates a read of the next record into CCBUF, calls IOCHECK1 to perform housekeeping functions and set up parameters (R8 and R10), and exits.

MSROTHR

MSROTHR performs all device reads for other than the C device (ASN = 3). It uses IOQUEUE1 to set up and make a call to COOP to read a record directly into the user's buffer.

ID

RDF

PURPOSE

RDF contains the routines that process the M:READ CAL for disk files and the M:TRUNC CAL and numerous support subroutines used by other modules, particularly those related to reading disk. RDF is in the root in a UTS System.

SUBROUTINES

ISEQUB

Purpose: To process the M:READ CAL for disk files.

Entry: B ISEQUB

Exit: Various, see operation.

Operation: If the file organization is random, ISEQUB exits to RWRAND. If this is a key read (KAD not zero), the routine calls SETUPUB to locate the key and move it into the user's key buffer; it exits to KEYER3 if the exact key is not found. If a multilevel structure exists, and if the key requested is not available in the current master index information, the code SETUPUB causes a search of the multilevel structure from the top down. If this is not a key-requested Read, but a reverse Read, PRCRD1 is called to back the key pointer to the beginning of the previous record. If this is not a key Read but a forward Read, PRCRD1UB is called to position to the next record. For both key and forward Reads, ISEQUB calls TRANSFERUB to move the record to the user's buffer and exits to MSREXIT. If BOF is encountered, the routine exits to MSREXIT with an abnormal code of X'04'. Otherwise, for reverse reads, the record is transmitted by TRANSFERUB, the positioned-before-record flag (TRN) is set, and the routine exits to MSREXIT.

TRUNC

Purpose: To process the M:TRUNC CAL by writing out modified buffers for a file and then releasing them.

Entry: B TRUNC

Exit: B MSRWRTX

Operation: If the DCB is not open or the assignment (ASN) is to device, TRUNC exits. Otherwise, the routine calls CLRBFUB to write out the index buffer, if modified, and calls RFI to release the buffer. CLRBUF is then called to write out the blocking buffer. TRUNC then calls IOSPIN to wait for I/O completion, then exits to MSRWRTX.

TRNS1

Purpose: To transfer a keyed disk file data record to or from the user's buffer according to whether EOP indicates Read or Write.

Entry:
```
BAL,SR4  TRANSFERUB  for Read
      or
B            TRNS1
D1 = X'00080000'  for Write
```

Exit:
```
Error - B   MSR01EXIT
Normal Read  - B    *SR4
Normal Write - B    RW2 new record is smaller than
                        original record.
               B    RW1 new record is equal to or
                        larger than orginal record.
```

Operation: The routine initializes various record-related items such as ARS, then transfers record segments in the direction indicated by EOP until either the end of the record is reached or the requested number of bytes is exhausted. When writing, the original record segment size (BLKSIZE) is used to control the transfer and to reset the Master Index item BLK. When reading, the current segment size (BLK) is used to control the transfer. TRNS1 transfers unblocked segments by doing I/O into or from the user's buffer, and blocked segments via UBLK. If a bad disk address is encountered, ERRLOG is called to log the error and the MSR01EXIT exit is taken. Otherwise, the normal exit is taken upon completion as indicated above. The current index entry (CMD) reflects either the index entry for the last record segment handled or for the next index entry. It reflects the entry for the last segment handled if a record Read request was for less than the record, or a Write request was for less than the record space.

CLRBBUF, CBB4, CBB5, CBB6, CBB7

Purpose: To write out a blocking buffer if it has been modified, and then release it.

Entry: BAL,R0 CLRBBUF
(R6) = DCB address

Exit: B *R0

Operation: If the buffer has not been modified, the routine sets truncation flag (TBT) and, following a delay of CBB5 for any I/O in progress, releases the buffer and exits. Otherwise, according to assignment type (disk or labeled tape), it does miscellaneous housekeeping jobs, establishes the end-action routine address for write disk (WRTBLKEND) or write labeled tape (WRTELEND), and generates a write command. At CBB6, the buffer address is set up as the I/O address, and preliminary buffer release accounting is done. At CBB4, the JIT address is added to the end-action information word. At CBB7, the I/O operation is initiated via QUEUE1 and the routine exits, actual buffer release occurring upon I/O completion by the end-action routines, unless DCB:RBBI is set. In this case, the buffer is not released.

GETBBUF

Purpose: To assure that a blocking buffer is assigned to a DCB.

Entry:
```
BAL,R0   GETBBUF
(R6) = DCB address
```

Exit:
```
B   *R0
```

Operation: If a blocking buffer is already assigned, the routine exits. Otherwise, the routine calls subroutine GFBB to obtain a buffer, sets the buffer address into BUF1, and exits.

If a buffer is not obtained, GETBBUF searches all DCBs associated with this job, looking for a blocking buffer associated with that DCB least recently operated on by a CAL which will be truncated (released) via the CLRBBUF routine and reallocated, as above, from the free buffer pool.

GETBUFM, GETSEC

Purpose: To obtain an index buffer, truncating the index buffer of some other DCB for this job, if necessary.

Entry:
```
BAL,R0   GETBUFM
           or
BAL,R0   GETSEC

(R6) = DCB address
```

Exit:
```
B   *R0
(R1) = CFU address
```

Operation: If an index buffer is already assigned, the routine loads R1 with the CFU address and exits. Otherwise, the routine calls GSBUF to obtain a buffer, sets the buffer address into BUF2, loads R1 with the CFU address, and exits.

The GSBUF subroutine attempts to obtain a buffer from the index buffer free pool and, if successful, returns the address in R14. If no buffer is available, the DCBs for the job are searched for the index buffer associated with that DCB least recently operated on by a CAL. That buffer is truncated via CLRBBUF (but not returned to the free pool) and reassigned by clearing BUF2 of the found DCB and returning the buffer address in R14.

LOCKEYUB

Purpose: To locate either the requested key or, if it does not exist, the next larger key.

Entry: BAL,R0 LOCKEYUB (D4) = word address of object key
(R4) = byte displacement of object key

Exit: If the key is found: B PULLEXIT1

If the key is not found (error exit): B PULLEXIT

Operation: If no master index exists, the error exit is taken. If the index is currently in core, the pointer (CMD) is validated by the LOCKY2 routine. If the index is not in core, its first sector is brought into core and the pointer is set to the beginning of the index. In either case, LOCKEYUB makes a preliminary comparison of keys and, if the current pointers are past the desired key, makes a reverse-search until the requested key or a smaller key is found. If the looked-for key is found, the routine exits skipping. If a larger key is found, or beginning or end of index is encountered, the error exit is taken.

PRDCRD11

Purpose: To move (position) a file in the direction by DIR, for the number of records indicated in CDA, by locating the appropriate index entry and moving the key into the user's key buffer.

Entry: BAL,R11 PRDCRD11

Exit: B *SR4

Operation: At BOF with a backward request, or at EOF with a forward request, or if the file is null, the exit is taken. Otherwise, the routine determines the current position via LOCKEYUB, and makes appropriate adjustment to number of records (CDA) if current position is indicated as being following the data record. Positioning is then performed by stepping keys (CMD) via FNDKY or FNDKYR, depending on direction. If BOF or EOF is encountered, the exit is taken. Otherwise, upon stepping the indicated number of active key entries, the key is moved from the Master Index to the user's key buffer and the exit is taken.

UBLK

Purpose: To move a record segment from the blocking buffer to the user's buffer.

Entry: BAL,R0 UBLK
(R1) = sector count
(R2) = number of valid bytes in block
(R7) = initial displacement in block

Exit: B *R0 (via BLNKIN)

Operation: If there is a valid buffer, UBLK calls BLKIN to transfer the segment and exit. If there is a buffer but it contains the wrong block, the buffer is released via CLRBBUF. If there is no buffer or if it was released, UBLK obtains a new buffer, sets up a Q4AVL end-action entry, initiates I/O with RBLKEND as the end-action routine, and exits. The record segment is subsequently transferred by RBLKEND.

WRTSEC

Purpose: To write out the current Master Index buffer and save the buffer contents in a new buffer.

Entry:
```
BAL,R0   WRTSEC
(D3) = current master index buffer address
```

Exit:
```
B    *R0
```

Operation: The current block of the Master Index is written out with end-action at WRTXEND. A new buffer is obtained and the contents' first two words of the old buffer are moved to the new buffer. The old buffer is sub- sequently released by the end-action routine.

OPENPV

Purpose: To begin a DCB's use of a private volume.

Input:
```
(R6) = DCB address
DCB:VNO = Volume number
```

Call:
```
'Label'     BAL,R0     OPENPV
            LI,R15
            B          MTPV         not mounted
```

Return 1: Returns to (R0), if the requested volume has not been mounted and verified. This return will cause MTPV in TYPR to be called and the return from MTPV will be to the point where OPENPV is called so that the volume can be opened.

Return 2: Returns to (R0)+2, if either the requested volume has been opened or the job has been aborted because the volume could not be mounted.

Output: DCB:VNO = 0, if job aborted because volume not mounted, if job aborted because volume belongs to another set, or if unable to mount the volume because unit unavailable.
DCB:PAT,VDCTX = 0, if job aborted because volume not mounted.
DCB:PAT = 0,VDCTX = foreign volume number, if job aborted because volume belongs to another set.
DCB:PAT =49,VDCTX = 0, if unit unavailable.
DCB:VNO = unchanged, if job not aborted.

AVR:NOU (of the AVRTBL entry associated with the volume) is incremented.
DCB:OVC incremented.

Registers: All nonvolatile, except R1, R6, SR1-SR3, D1, D3, D4.

Operation: The AVRTBL is searched and if the volume is not mounted and verified, OPENPV exits to MTPV in TYPR where the volume, if possible, is mounted and verified MTPV returns to OPENPV via the path described in the calling sequence. If the volume could not be mounted (either unit unavailable or job aborted), OPENPV exits (R0)+2. If the volume is mounted and verified, the "number of users" count in the volume's AVRTBL entry (AVR:NOU) and the file's "open volume count" (DCB:OVC) are incremented with interrupts disabled.

CLOSEPV

Purpose: To terminate a DCB's use of a private volume.

Input: R6 = DCB address
DCB:VNO = the volume number of the volume being closed.
DCB:PAT and DCB:VDCTX may not point to the volume in VNO.
DCB:OVC = a count of the number of opened volumes.

All data has been written to the volume (BBUD,MIUD = 0).

Call: BAL,R0 CLOSEPV

Output: DCB:OVC decremented.
AVR:NOU (of the AVRTBL entry associated with the volume) is decremented).

The volume's cylinder bit map is written onto the VTOC.

Registers: All registers nonvolatile.

Operation: The AVRTBL entry for the volume is located. The cylinder Bit Map and NVAT from the volume's allocation table (HGP) are written onto the volume's table of contents (VTOC) using the file's blocking buffer. The "number of users" count in the volume's AVRTBL entry (AVR:NOU) and the file's "OPEN VOLUME COUNT" (DCB:OVC) are decremented with interrupts disabled.

ID

WRTF

PURPOSE

WRTF contains routines used to write disk files.

SUBROUTINES

IOSFILE

Purpose: To process the M:WRITE CAL for disk files by switching to the appropriate processing routine.

Entry: B IOSFILE (from MSRRDWT in IORT)

Exit: See Operation.

Operation: If file organization is random, control is transferred to RWRAND. If the organization is consecutive, control is transferred to SEQD. If the mode function (FUN) is INOUT, control is transferred to IOSEQUB. If NEWKEY is specified and ONEWKEY is not, control is transferred to OSEQUB. Otherwise, the organization being keyed, if no key is specified or neither NEWKEY nor ONEWKEY is specified, the abnormal exit to MSR01EXIT is taken with a code of X'17'. If ONEWKEY is specified, control is transferred to ODIR.

OSEQUB

Purpose: To perform disk file Writes for sequential consecutive files and keyed files, with ONEWKEY not specified.

Entry: B OSEQUB

```
Exit:      Normal    B  MSREXIT
           Abnormal  B  KEYER1
                 or
                     B  KEYER2
```

Operation: If the Master Index is not initialized, all granules are released, the dummy key is initialized if the file is consecutive, and control is transferred to IOSEQUB4 to write the first record in the file. If the file is keyed, OSEQUB checks the key for proper value and exits to KEYER1 if the key is smaller, or to KEYER2 if the new key is equal to the key at current file position. If the file is consecutive, the dummy key is incremented. For either organization, if the last operation was not a Write, all records forward of current position are deleted. Control is then transferred to IOSEQUB4 to write out the record, or to MSREXIT if the record consists of zero bytes and file organization is consecutive.

ODIR

ODIR writes a new record if the key cannot be found in the index and ONEWKEY or NEWKEY is specified. The existing record is modified if the key is found in the index and either ONEWKEY is, or NEWKEY is not specified. Otherwise, ODIR exits to KEYER2 or KEYER4.

IOSEQUB4

This routine writes a new record via ENTKEYUB, clears the WAIT flag if less than a block was transferred, and exits to MSREXIT.

IOSEQUB

If a key is specified, control is transferred to ODIR. If the last operation was a Read, the record is rewritten via REWRITEUB. If it was not a Read, control is transferred either to OBSEQUB to write a new record if the file is consecutive or to MSR01EXIT with a code of X'15' if it is not.

RWRAND

Purpose: To perform Read or Write for a random file.

Entry: (SR1) = FPT code X'10' for Read
X'11' for Write

Exit: Normal	B	MSREXIT
Error	B	MSR01EXIT

Operation: If the requested starting or ending granule is outside the file limits, an error code of X'42' is given for a starting granule, or of X'57/44' for an ending granule, and exit is made to MSR01EXIT. For a request inside file limits, RWRAND calculates the disk address, sets up a transfer of the lesser of 15 granules for RAD, three granules for pack, or the requested size, and makes a read or write call, according to R8, to QUEUE. If the requested number of bytes is not exhausted, additional QUEUE calls are made. ARS is then set to the lesser of bytes requested or maximum value, and exit is made to MSREXIT.

INSERTUB,ENTKEYUB

Purpose: To insert key entries at current position and output data portion or record. The two entry points are identical.

Entry:
```
BAL,R0    INSERTUB
      or
BAL,R0    ENTKEYUB
```

Exit:
```
B    *R0
```

Operation: If the current data granule is partially used or the remaining record size is less than a granule, as much of the record as will fit is moved into the blocking buffer. Otherwise, an unblocked record segment is set up to be written directly from the user's buffer. Either way, a Master Index entry is established for the record segment at current position (CMD) and, if record is unblocked or buffer is full, the data granule is written out. If the record is not complete, the above is repeated.

REWRITEUB,RW1,RW2

Purpose: To rewrite an existing disk file record. RW1 and RW2 are reentry points from TRNS1.

Entry:
```
BAL,R11   REWRITEUB
```

Exit:
```
B         TRANX
```

Operation: R12 is set to indicate a write operation and TRANS1 is called to write as much of the requested data as will fit in the current record. Reentry is at RW2 if the request is for less than the existing record size. In this case, BLK of the index entry for the last segment modified is changed to reflect the new record length, and BLK is set to zero for all following index entries for the same record. Reentry is at RW1 if the request is not for less than the original record. If the request is for the exact size, control transfers to RW2, above, where BLK for all index entries for the following segments is cleared. Otherwise, if the original record consisted of zero bytes, the original entry is made null. In any event, control is then transferred to ENTKEYUB1, an alternate entry to ENTKEYUB that does not perform record initialization but does write the balance of the record. ENTKEYUB1 exits via TRANX.

IMAGEA

Purpose: To build a key entry in the Master Index at the slot pointed to by CMD.

Entry: BAL,R0 IMAGEA
(R14) = word address of Master Index

Operation: After assuring that a Master Index (MI) block is in core, the key is moved into the MI entry from the user's key buffer. MI items, BLKSIZE and BLDISP, are then set from the DCB items BLK and PBD, respectively. DABLK is set from the DCB item BCDA, if buffered, otherwise, from CDA. BLK is set from the DCB item BLK and C is set to zero or one depending on RWS being zero or nonzero. In either event, EOF is zeroed and FAK is set from the DCB item NLR. If the Associated segment is then written out and a new index buffer is obtained.

INST

Purpose: To assure that there is room in the MI at the current position and insert a key entry in the Master Index.

Entry: BAL,R0 INST
(R14) = word address of master index

Exit: B *R0

Operation: If the MI pointer (CMD) is at the end of data in the block and there is room in the block for the entry, the entry is inserted and the exit is taken. If the pointer is to the end of the block and this is not the last block in the MI, the next block is read in, the pointer is reset to the first entry, and control returns to the beginning of the routine. If the current block is the last MI block, another half-granule is obtained, the old FLINK is set, the old block is written out, the buffer is released, a new buffer is obtained and initialized, and control returns to the beginning of the routine. INST exits if another half-granule is not available. New entries to be inserted other than entries that follow all other entries in a block, are processed in the INST3 region. If there is room in the block, the remaining entries are moved down and the entry is built in the created slot. Otherwise, either the next block if it has room or a new block that is linked in if the next block does not have room, receives the last entry. from this block, and the new entry is constructed in the same manner as above.

ENTER1,ENTER2

Purpose: To insert a new entry into the Master Index or write out the associated block if the MI is full. The two entry points are identical.

Entry: BAL,R11 ENTER2

Exit: B *SR4

Operation: ENTER2 obtains a buffer and a granule and initializes the buffer. INST is then called to build and insert the entry in the MI and write out the associated data block if that buffer is full. ENTER2 also moves the EOF flag in the index if the insertion is at the end of the index.

GETIHGRAN

Purpose: To allocate an index half-granule for keyed and consecutive files on either public devices or private volume sets.

Input: (R6) = DCB address
(R1) = CFU address

Output: (D1) = disk address of allocated index half-granule,
or = 0, if unable to allocate.
Condition codes set to indicate value of D1.
CFU:SMI = (D1)
DCB:TYC = X'A', if unable to allocate.

Call: BAL,R0 GETIHGRAN

Registers: Volatile - SR1, D1, D2

Operation: If CFU:SMI contains the disk address of an available half-granule, GETIHGRAN exits. Otherwise, GETIGRAN is called to allocate an index granule. If a granule is allocated, its disk address is stored in CFU:SMI and GETIHGRAN exits. If a granule is not allocated, the "type of completion" status (DCB:TYC) is set to "saturated disk" and GETIHGRAN exits.

GETDGRAN, GETIGRAN

Purpose: To allocate an index granule for keyed and consecutive files on either public devices or private volume sets.

Input: (R6) = DCB address

Call: BAL,R0 GETIGRAN or GETDGRAN

Output: Same as GETDGRAN except that DCB:VNO, DCBVDCTX, and DCB:PAT remain unchanged for private consecutive files when an index granule is allocated on the next volume; in this case, DCB:SWXV is set.

Registers: All registers saved except SR1.

Operation: If a public account directory or file directory granule is being allocated, all file options (such as NOSEP and device type) are ignored, and it prefers RAD, PACK, Cylinder in that order. If a data or index granule for a public temporary file (opened with release) or a STAR file is being allocated, it prefers RAD, PACK, Cylinder, in that order, unless Cylinder allocation was specifically requested, in which case, it prefers Cylinder, PACK, RAD, in that order. For data or index granules of all other public files, the default preference is PACK, Cylinder, RAD, in that order, unless Cylinder allocation was requested, in which case the preference is Cylinder, PACK, RAD. Failing request for Cylinder allocation, the installation may set the FILEPREF flag in the Monitor, in which case, the preference is RAD, PACK, Cylinder. Finally, if neither Cylinder nor FILEPREF pertains, the NOSEP and device options may be used to prefer either PACK, Cylinder, RAD or RAD, PACK, Cylinder. Once preference has been established, the Monitor attempts to allocate, as requested, failing if:

1. there are no granules of the type requested currently available;

2. the user has depleted his authorization.

Failure to allocate on first preference causes a try for second preference, etc. Failure to allocate at all produces a X'A' TYC and I/O ERR 57.

If the file is private, a check is made to see if the cylinder that was last allocated to the file has available granules. If so, one is taken by decrementing the cylinder's available granule count (CFU:NGAVAL) and calculating the disk address of the next available granule, if any, within the cylinder (CFU:GAVAL). GETIGRAN exits with the disk address of the granule in SR1 (accounting was done when the cylinder was allocated to the file). If there are no available granules from the last cylinder, GPVCYL is called to allocate a cylinder from the current volume.

If a cylinder is available, its disk address is saved in CFU:GAVAL and the number of granules is saved in CFU:NGAVAL. Accounting is done and a granule is allocated from the cylinder as previously described. If a cylinder is not available on the current volume and a file directory granule is being requested, GETIGRAN exits with a zero in register SR1, indicating that a granule is not available. Private file directory granules can only be allocated on the primary volume (which would be the current volume, in this case). If a cylinder is not available on the current volume and a user file granule is being requested, a check is made to determine whether the file has keyed or consecutive organization.

If the private file has keyed organization, all volumes in the set are mounted and opened to the file. GPVCYL is called to allocate a cylinder from other volumes in the set, starting with the last.

If the private file has consecutive organization, only the current volume will be opened to the file. GNVAT is called to allocate a granule from the next volume's Cylinder 0 Allocation Table (NVAT). If an NVAT granule is available, the switch-volume flag (DCB:SWXV) is set and GETIGRAN exits with the disk address of the allocated NVAT granules.

INITMI,INITMI1

Purpose: To initialize the first three words of the index half-granule in DCB:BUF2, which has just been allocated by GETIHGRAN.

Entry: If entered at INITMI, the FLINK is set to zero.
If entered at INITMI1, the FLINK is set to (D1).

Input: (R1) = CFU address
(D1) = disk address of FLINK (INITMI1)
(D3) = DCB:BUF2
CFU:SMI = disk address of index half-granule in DCB:BUF2
(SR4)= disk address of BLINK

For private consecutive files, DCB:VNO, DCB:PAT, and X:DCTX should point to the volume that the disk address in CFU:SMI points to.

Call: BAL,R0 INITMI or BAL,R0 INITMI1

Output:
(CFU:CDAM) = disk address of index half-granule in DCB:BUF2
(CFU:SMI) = disk address at next available half-granule or zero, if none
DCB:MIUD set
(D1) = 0, if entered at INITMI

Registers:
Volatile = R2, R3, R4, R5, R7, SR1, SR2, SR4, D2
Nonvolatile: = R1, R6, SR3, D1 (INITMI), D3, D4

Operation: The counters "slides" (CFU:SLIDES) and "consecutive slides" (DCB:RDLO), which control the (re)building of a higher level index structure, are incremented. If the "consecutive slides" counter (DCB:RDLO) equals or exceeds the limiting value specified (DCB:LRDLO), the flag that signals "CLOSE to reconstruct the higher level index" is set (i.e., CFU:SLIDES is set equal to 255).

The first three words of the level -0 index block are initialized and DCB:CDAM (if user file) or CFU:CDAM (if AD or FD) is set equal to the disk address of the index block.

If the index block being initialized is the first half of an index granule, the disk address of the second half is calculated and saved in CFU:SMI. If the index block is the last half of an index granule, CFU:SMI is set equal to zero.

ID

DLT

PURPOSE

The DLT module contains routines to delete records and to release granules from a file.

SUBROUTINES

DELETE

Purpose: To process the M:DELREC CAL.

Entry:

```
OVERTO    DLTSEG,4
      or
BAL,SR4   DLT

(R7)  = FPT+1
(SR1) = FPT code (X'0D')
```

Exit:

```
B     PULLALLEXIT
```

Operation: DELETE exits to MSRWRTX if the DCB is not for a disk file in the INOUT mode. Otherwise, an M:READ format parameter list is established in TSTACK and MSRRDWT is called to process the list and to validity checks. MSSRDWT then exits to DEL to delete the record and DEL ultimately returns to DELETE which then exits to PULLALLEXIT.

DEL

Purpose: To delete a specified record from a disk file by nullifying its Master Index entries.

Entry:

```
BAL,SR4   DEL
```

Exit:

```
B    *SR4
```

Operation: The current entry is made null by nullifying its index entry (key length) set to zero. If there are continuation entries, KL being set to zero makes them null. If the entry points to no data (DABLK equals zero), the entry is deleted by sliding up the index. If the deleted entry was at EOF, the EOF flag is moved back to the last active entry, and if there are no active entries, all granules are released via REL.

DELA

Purpose: To remove the current (CMD) index entry.

Entry:

```
OVERLAY   DLTSEG,7
      or
BAL,SR4   DELAA
```

Exit:

```
B    *SR4
```

Operation: DELA sets up the index and pointers and enters the DEL routine at DEL1 where the index is deleted and EOF checks are made.

DELF,DELO

Purpose: To delete an entry from a file directory.

Entry:

```
BAL,SR4    DELF
      or
OVERTO     DLTSEG,6
```

Exit:

```
B    *SR4
```

Operation: At DELF the FIT address is entered into the Free Half-Granule Pool (FSP). At DELO, the current file directory block is brought into core. Then, at an alternate entry to DELA, pointers are set up and the entry is deleted. Entry at DELO assumes that SR4 has been placed into the TSTACK.

REL

Purpose: To release all granules or cylinders allocated to a file, file directory, or account directory.

Entry:
```
OVERLAY    DLTSEG,1
         or
BAL,SR4    REL
```

Exit:
```
B     *SR4
```

Operation: If the CFU indicates no first disk address (i.e., no granules allocated) REL exits. Otherwise, an index buffer and a blocking buffer are obtained and a table of granules or cylinders to be released are initialized in the blocking buffer. Master Index blocks are then read into the index buffer. If the index is for a user's file (IMT = 4), each entry is checked for being the beginning of a granule (BLDISP = 0) and, if so, the granule is released. (NOTE: this procedure assures that all data granules are released but no granules are released twice.) In any event, if the index block is either at the top of a granule or at the top of a cylinder, the disk address of that granule is entered into the granules-to-be-released table via the SGAIBB subroutine. After all index blocks have been processed, if there is a granule or cylinder is entered into the granules-to-be released table, as are the FSP blocks. Top of a granule or cylinder is entered into the granules-to-be released table, as are the FSP blocks. In any event, each granule in the granules-to-be-released table is then released as are any granules constituting that table.

SPOOLUB

Purpose: To obtain a half-granule from a Free Half-Granule Pool (FSP).

Entry:
```
OVERLAY    DLTSEG,2
        or
BAL,SR4    SPOOLUB
```

Exit:
```
B     *SR4
(R8) = disk address of half-granule obtained, or zero,
       if no FSP.
```

Operation: If there is no FSP or if an error is detected in reading a half-granule, zero is returned. Otherwise, if the first half-granule of the FSP has entries, the last entry is returned and the FSP half-granule is adjusted. If the first FSP half-granule is empty, the FSP half-granule is returned, and the next half-granule's links (if any) and the pointer to the FSP in the associated FILCFU or ACNCFU are adjusted.

SETFPOOL

SETFPOOL adds an entry to the Free Half-Granule Pool (FSP) from D4. If an additional half-granule is needed for the FSP, the half-granule is used for the FSP instead of being added to the FSP.

SGAIBB,SGABB

Stores a granule address from D1 into the next entry of the granules-to-be-released tables (see REL above). If the last entry in the table has been used, the SGABB routine is entered, a new granule is obtained for the next section of the table, as is a granule for the first section of the table if it was not previously obtained, granules are linked, and the completed block is written out.

RELRBG

Releases the granule or cylinder whose address is in SR1 and adjusts the JIT item for permanent or temporary disk space.

ID

OBSE

PURPOSE

OBSE contains subroutines used by the various Open modules.

SUBROUTINES

CHKFLACN

Purpose: To check for valid file name and account entries in the DCB and to set the FLD and ACD pointers and USR flag in the DCB.

Entry: BAL,R0 CHKFLACN

Exit: Normal - B *R0
Error - B OPENR03

Operation: If there is no key buffer, the error exit is taken. If the next file option (NXTF) is specified, the filename variable length parameter (VLP) must allow eight words. Otherwise, the filename must be in the range of one to 31 bytes long. The error exit is taken if the VLP is too short or is nonexistent. In any case, FLD is set with the relative address of the filename VLP, and ACD is set with the relative address of the account number. If the user did not specify an account number, the DCB is set with the user's job account number. The DCB item USR is then set if the account number in the DCB is not the user's account number. The normal exit is then taken.

SECCHK

Purpose: To determine if the user may access the file in the manner specified.

Entry: BAL,R0 SECCHK
(R7) = address of VLPs in FIT (for finding password).
(R14) = address of VLPs in FIT (for finding Read/Write accounts).

Exit: Access allowed - B PULLEXIT
Access denied - B PULLEXIT

Operation: If this is a monitor request, TEL request, or if user has CO privilege, the allowed exit is taken. If FIT indicates a password and the DCB does not contain a matching password, the access-denied exit is taken. If it is the user's own file (i.e., USR = 0), the access-allowed exit is taken. If write access ALL is found, it is treated as the user's file. Otherwise, the allowed-access is taken only if the user's account number is found in the list of write-access numbers or if the user's account number is found in the list of read-access numbers and the requested access is IN.

SETACOG

Purpose: To set access, organization, and maximum key length for the file being opened.

Entry: BAL,R11 SETACOG
(R7) = address of VLPs in FIT

Exit: B *R11

Operation: If the file organization is random and the function (FUN) is OUT or OUTIN, the function is changed to INOUT. If the function is IN or INOUT, organization (ORG) and maximum key length (KEYM) in the DCB are set from the corresponding items in the FIT. If the organization is consecutive, access (ACS) is set to sequential and the length of key portion of MI entries (SCR) is set to 4. If KEYM is zero, SCR is set to 12. Otherwise, SCR is set to the lesser of KEYM+1 or 32.

ID

OPN

PURPOSE

The OPN module contains the routines that process the M:OPEN CAL and perform the Open for disk files and devices.

SUBROUTINES

MSROPN

Purpose: To process the M:OPEN CAL by transferring data from the FPT to the DCB and transferring control to the appropriate routine to complete the Open.

Entry: OVERTO OPNSEG,0
(R7) = address of FPT+1

Exit: Various (see operation)

Operation: If the file is open, MSROPN sets the abnormal code of X'2E' and exits to MSR01EXIT. Otherwise, it moves the fixed length parameters present in the FPT to the DCB. If there are variable length parameters present in both the FPT and the DCB, the routine moves the FPT parameter into the DCB for each parameter present in both. Any FPT parameter that is longer than the space allocated in the DCB is truncated. In all cases, control is then transferred, according to assignment, as follows:

ASN Value	Location
1 (disk)	OPNFIL
2 (labeled tape)	MSRLBT
3 (device)	OPENDEV
Other	OPENER01

OPENDEV

Purpose: To open a DCB assigned to a device.

Entry: B OPENDEV

Exit: Error - B OPENER01
Normal - B MSRWRTX

Operation: If the operational label text flag is set, the label is converted from the character text format to an OPLB value. If lines per logical page (LVA) in the DCB are zero, this item is set from the DCT. If the DCB is assigned to tape and not via an operational label, OPNT is called. Otherwise, the routine sets the function code from the DCT and, if the DCB is for tape, sets packed binary mode. In all cases, record-related items are initialized and exit is to PEOF if file extension applies; otherwise, exit is to MSRWRTX.

OPNFIL

Purpose: To perform Open of a disc file.

Entry: B OPNFIL

Exit: Normal - B MSRWRTX
ABN/ERR - various paths to MSRWRTX

Operation: The account directory* is searched for the account number or the first name or the next name if the NXTA option applies. If the account number is found, the file directory* is searched for the filename, or the first name or next name if the NXTF option applies. If the account number or name is not found, a new file is opened if function is OUT or OUTIN; OPNFIL makes another search if INOUT and SYNON is specified, or an abnormal code of X'02' is set, and exits to OPER. If the file is private (i.e., if ASN = 1 and a serial number has been specified), OPV is called in the OPNL module to open the file on the private volume set. If the filename is found, the FIT is read and the filename is moved to the DCB. Organizattion, as specified in the DCB and FIT, is tested for conflict (abnormal code of X'14') or if organization is unspecified and the FIT indicates random, organization is set to random. If SYNON is specified, either an abnormal code of X'08' is given if NXTF or the base filename is found and the FIT is read. An abnormal code of X'14' is given if the security checks fail. If file extension applies, the file extension flag is set and the function is forced to INOUT. The access organization and key length are set and a CFU entry is obtained and initialized via HOOKUP2 and HOOKUP1. If the file is already open (i.e., the CFU already exists), either the number of users is incremented if the organization is random or the DCB and CFU functions are IN; or a new CFU is set up and linked to the original if the CFU function is OUT and the DCB function is IN; or an abnormal code of X'14' is returned. If exit is not abnormal, either both filename and account number are set into the CFU or only the account if organization is OUT or OUTIN and REL is specified. The record-release items are then initialized and the routine exits to PEOF if the file extension flag is set, otherwise, to MSRWRTX.

*The account and file directories are not searched for scratch files (OUTIN & REL) or STAR file (the FIT for STAR files is pointed to by the JIT).

If the filename is not found and a new file is to be created, the function must be OUT or OUTIN and the DCB:USR flag must be set or an abnormal return is made. Otherwise, if not a random file, the FIT is initialized. If the file is random, GRAND is called to allocate space.

HOOKUP2

The CFU whose address is in R1 is initialized by clearing words one to eight and, if the file is random, setting FDA. RLIM is put into the CDAM field. Exit is skipping if OUT or OUTIN.

HOOKUP1

Various CFU items are set from the FIT. If the filename in the FIT is not correct, or if any of the disc addresses involved are bad, the FD entry is deleted via DELO and an error X'75/03' is returned via OPER. The calls to HOOKUP1 always immediately follow a call to HOOKUP2, thus, HOOKUP1 is never entered if function (FUN) is OUT or OUTIN.

SCANCFU

An empty CFU is obtained and, if no other DCB is open to the file, the CFU address is saved in the DKEY item of the DCB and in R1, and the routine exits skipping. If a CFU already exists for the file, exit is not skipping and the address returned is either that of the unlinked CFU, if no secondary CFU exists, or that of the IN mode CFU of a linked pair.

FINDFIL

If a filename is specified in the DCB, that file is searched for in the file directory associated with the account number. If no filename is specified, either the next FD entry is assumed or the first FD entry if the account number is not that of the current FD. The exit is skipping if either the name is found or execution is in the "next entry" mode and not at EOF for the FD.

FINDFIL1

If the file is for output and release, no directory search is necessary and the nonskipping exit is taken. Also, if the account number is not found, the nonskipping exit is taken. If the account number is unchanged from the last reference, or if the account number is found and its file directory disc address (DA) is valid, the DA is saved and the skipping exit is taken. If a bad DA is found, the error is logged, the account number is deleted via DELF, and exit is via OPER with a code of X'75/0A'.

GETFI

The file information table (FIT) for a file is read into core. If the disc address in the file directory (FD) is bad, the entry is logged and deleted from the FD.

OPFIND

The OPLBT1 is searched for a match with the label in D1 and the routine exits skipping if the label is found.

EXTCHK

If file extension applies to this open of this DCB, the skipping exit is taken.

TRANFLNE1

The filename is transferred into the DCB from the location in D3.

TRNINFO

The FPARAM data is moved to the user's buffer if FPARAM is a legal, nonzero location.

ID

CLS

PURPOSE

The CLS module contains routines to process the M:CLOSE CAL, to perform the close of disk files and devices, and to process disc errors.

SUBROUTINES

MSRCLS

Purpose: To process the M:CLOSE CAL.

Entry: OVERTO CLSSEG,0

Exit: Various - see operation.

Operation: If the file is not open, the abnormal code of X'0A' is returned via MSR01EXIT. If the DCB is the OC DCB, exit is taken via MSRWRTX. Otherwise, after delaying for I/O to complete, control is transferred to routine based on assignment (ASN) as follows:

ASN	Location
0 (null)	MSRWRTX
1 (disk)	CLSFIL
2 (labeled tape)	CLSLBL
3 (device)	CLSDEV
4 (core file)	CLSFIL

CLSDEV

If the DCB is assigned to paper punch or card punch and special formatting is indicated, a !EOD record is output. If it is assigned to device and the device is tape, CLSTP is called. Otherwise, I/O is forced to completion, DCB words 15-21 are cleared as are miscellaneous flags in the DCB and, if not assigned to disk, the device or operational label is cleared.

CLSFIL

Purpose: To perform the Close of a DCB assigned to disk, and to close the associated file if there are no other users of the file.

Entry: B CLSFIL
(R7) = word address of FPT+1

Exit: Various - see operation.

Operation: After flushing and releasing the buffers associated with the DCB, the routine waits for I/O to complete. If there are other users of the file via the same CFU, CLSFIL closes the DCB and exits to MSRWRTX.

If the file is OUT or OUTIN with "save" not specified on both Open and Close, or if this is the current user's file with REL specified but not SYNON, the file space is released via RELRAND if the file is random, otherwise, via REL in DLT. Then for OUT/OUTIN files at CLSFIL2, the CFU link, if any, is cleared, the CFU is released, and the DCB is closed as above.

For IN and INOUT files being released, the file directory is first searched for SYNON files and all those associated with the original files are deleted. Then, for both SYNON and nonSYNON files, the FD entry is deleted, the CFU is released, the DCB is closed, and the MSRWRTX exit is taken. If the file is IN and not under the user's account, it is closed in a way similar to the OUT/OUTIN closing described above, except that the file is not released.

If REL is specified for an IN file not the user's file, or for an INOUT file that is a SYNON file, CLSFIL updates the FIT of the original file from the CFU at CL2 and writes it out. If the SYNON name is in the file directory, it is deleted, the CFU is released, the DCB is closed, and the MSRWRTX exit is taken. If REL is not specified and the file is an INOUT file*, the FIT is updated at CL2, the CFU is released, DCB closed, and MSRWRTX exit taken. If REL is not specified and file function is IN*, then the FIT is not updated and the CFU links are cleared at CLSFIL2 before these other actions are taken.

For the OUT/OUTIN keyed files with "save" specified, further consideration depends on whether or not a multilevel structure exists. If so, then the number of new granules added since the higher level structure was built must exceed a user-specified limit and the absolute value 2 for further consideration. If no multilevel structure exists, then the number of master index half-granules must exceed 2 for further consideration. If all necessary conditions are met, the routine builds or rebuilds multilevel structure, as required, by passing control to the overlay MUL via OVERLAY MULSEG,0.

Three possibilities exist if the file function is OUT/OUTIN with "save" specified on both Open and Close: (1) the account may not be in the account directory (AD); (2) the file may not be in the file directory (FD); and (3) both may be present. If the CFU is linked (SCIW= 0), the file is released. Otherwise, if the account is not in the AD, it is entered into it at CLSFIL4A and an FD is created. After handling the AD, CLSFIL builds and saves the FIT at CLSFIL4B and enters the filename into the FD if it was not previously entered. At CLSFIL3A, the CFU is released, the DCB closed, and the MSRWRTX exit taken. If the file is already in the FD, a new FIT is built, the old file is released, and the Close proceeds as at CLSFIL3A above, provided there are no SYNON files in existence. If there are SYNON files, the FD is searched at CL6 and all those SYNON files associated with the original file are deleted from the FD, the new FIT is updated, and Close proceeds as before.

*For these cases and for STAR files, a directory search is not made.

EPWRTA,EPWRT

EPWRTA sets up filename as key. Both entry points then insert the key as an entry in an index. If there is no CFU:SMI value, the disc is presumed to be saturated and the MSR01EXIT is taken with a code of X'56'. Otherwise, the disc address in D1 is set into the index entry as the FIT address (DABLK).

FILIMG

The FIT is constructed from the DCB and CFU information.

FINDFILS,FINDFILINIT,FINDFIL

The FD for the DCB specified account is searched for the filename in the DCB (for FINDFILS), in the CFU (for FINDFILINT), or as pointed to by KAD (for FINDFIL). If found, exit is skipping.

BADA

Purpose: To process disc address errors.

Entry:
```
OVERTO   CLSSEG,8
      or
B        BADA

(SR1) = FLINK/BLINK disc address
(D3)  = File Information Block (FIB) address
```

Exit: Various - see operation.

Operation: BADA determines the type of CFU (AD, FD, or MI) that has failed by comparing the CFU address of the DCB with the address of FILCFU.

If the CFU type is AD and the disc address if FLINK (i.e., sector 1), BADA exits to the routine RECOVER with code X'1A' in R15. If the bad disc address is not sector 1, BADA reports the error code/subcode X'75/06' and researches the sector by branching to REDSEC1.

If the CFU type is MI and the disc address is for the Free Sector Pool, the FSP in the CFU is zeroed and the error code/subcode X'75/00' is reported. BADA exits via REDSEC. Otherwise, error code/subcode X'75/02' is reported and BADA exits through MSR01EXIT.

If the CFU type is FD, the error code/subcode X'75/04' is reported and the sector is re-searched by branching to REDSEC1.

BALINK

Purpose: To process BLINK/FLINK errors.

Entry:

```
OVERTO   CLSSEG,9
       or
B          BALINK
```

(R2) = 0 if BLINK, 1 if FLINK
(R3) = BLINK or FLINK disc address

Exit: B *SR4

Operation: BALINK determines the type of CFU involved in the failure (AD, FD, MI).

If the CFU is AD and the DA = 0, the RECOVER routine is entered. Otherwise, the error code/subcode X'75/06' is reported and BALINK exits to REDSEC1.

If the CFU is the Master Index and the disc address is for the FSP, then the error code/subcode X'75/02' is reported and BALINK exits to MSR01EXIT.

If the CFU is FD, the error code/subcode X'75/04' is reported and BALINK exits to REDSEC1.

For both BADA and BALINK, if the JIT (J:CLS) indicates that a close file operation is in progress, exit is made to REDSEC1 to finish up the close operation.

ID

TFILE

PURPOSE

The TFILE module contains only the routine that processes the M:TFILE CAL.

SUBROUTINES

MSRTFILE

Purpose: To process the M:TFILE CAL.

Entry: BAL,SR4 TFILE
(R7) = word address of FPT+1

Exit: B *SR4

Operation: The file is closed by calling MSRCLS (see above, CLS). A file named T is then opened with parameters KEYED, DIRECT, INOUT, SAVE, KEYM = 31. If the Open is unsuccessful, implying that the TFILE file has not yet been created for this job, it is opened as OUT. A call is then made to MSRRDWT (see above, IORT) with R8 containing X'11' to indicate Write and R7 containing the TFILE parameter list. This parameter list, interpreted by Write, causes a zero data length record to be written into file T with a key that is the filename of the file being made temporary. The routine then closes file T by calling MSRCLS.

ID

POS

PURPOSE

The POS module contains routines to process the M:PFIL, M:PRECORD, M:REW, and M:WEOF CALs.

SUBROUTINES

PFIL

Purpose: To perform the M:PFIL procedure.

Entry: B PFIL
(R7) = word address of FPT+1

Exit: Error - B PULLALLEXIT
Normal - B MSRWRTX

Operation: The file is opened if not open. If the file cannot be opened, the error exit is taken. If the file is random, the MSRWRTX exit is taken. If the DCB is for device and is not tape, the exit is taken. If the DCB is for (unlabeled) tape, a skip forward or backward file command, according to FPT, is executed via TAPEOP and the MSRWRTX exit is taken with WAT set. If the DCB is for disk and the FPT indicates EOF, the sectors of the file index are read until the last record is found, the current Master Index Displacement (CMD) is set to the end of the last entry and backed one entry, and the exit is taken. If the FPT indicates BOF, the CMD is set to zero, which is an uninitialized condition equivalent to BOF, and the exit is taken. If the DCB is for labeled tape and the FPT indicates BOF, CLSLBL is called to close the DCB with the PTL (position to label) option. Upon return, the DCB is reset to open, two forward-space file commands are executed via TAPEOP, and the exit is taken.

If the FPT indicates EOF for labeled tape, PFIL checks the last operation for Write, exiting if it is found. Otherwise, a forward-space file command is executed and CHKEOF is called to check the labels and do a volume switch if not EOF. If the EOF sentinel is found, the routine exits. Otherwise, a volume switch having been made, the forward-space file command and call to CHKEOF are repeated.

REW

Purpose: To perform the M:REW procedure for a disk file, labeled tape, or unlabeled tape.

Entry: BAL,SR4 REW

Exit: B MSRWRTX

Operation: If the DCB is open and is for disk or labeled tape, PFIL1 in the PFIL routine is called to position to beginning-of-file. If the DCB is for device and is assigned to tape, the DCB is opened, if necessary, and a rewind command is executed via QUEUE. If the DCB is closed and is not assigned to device, the routine exits via *SR4. Otherwise, exits are via B MSRWRTX.

WEOF

Purpose: To perform the M:WEOF procedure.

Entry: B WEOF

Exit: B MSRWRTX

Operation: The DCB is opened if not open and, if the function is IN or the DCB is not assigned to device, the exit is taken. If the device is paper tape or card punch, an !EOD record is written. If the device is a printer, PAGE is called to skip to top of form. If a magnetic tape, a tape mark is written via TAPEOP. For all other devices, the routines exit directly.

PRECORD

Purpose: To perform the M:PRECORD procedure.

Entry: B PRECORD
(R7) = word address of FPT+1

Exit: B MSRWRTX

Operation: The DCB is opened if not open and the FPT parameters are moved into the DCB. If the DCB is for a disk file, PRDCRD11 is called to skip records in the designated direction, and ARS is set with the number remaining (e.g., if EOF or BOF encountered). If the file is empty, BOF or EOF is set depending on direction. If the DCB is for labeled tape, repeated read-zero-byte calls are made to READL until the prescribed number of records has been skipped in the designated direction, or an abnormality (e.g., BOF, EOF) is encountered. ARS is set with remaining record count.

ID

OPNL

PURPOSE

The OPNL module contains routines that open input labeled tapes, routines that allocate disk space for random files, and the routine to open a file on a private volume set.

SUBROUTINES

GRAND

Purpose: To allocate space for a random file.

Entry: BAL,SR4 GRAND

Exit: B *SR4

Operation: If a background user's disk space limit would be exceeded, or if the space is not available, the abnormal code of X'01/0B' is passed and RAND exits. Otherwise, the space is allocated via GNBG or via GNDPBG for a disk pack. If SAVE was specified, the CFU is set up, the DCB is marked open, and MSRCLS is called to insert the file in the FD. In either case, RAND then exits.

OPV

Purpose: To open a file on a private volume set.

Input:
(R3) = word displacement of the INSN/OUTSN list in DCB:VLP
(R6) = DCB address
(R7) = address of the VLP area of the DCB
(SR4) = return address to RDF

Call: Only called by OPN at OFIL2. OPV returns to OPN at OFIL3 with the error code in R3.

```
Output:     (R3) = 0, if error return and SR3 contains an
                      error code;
                 = 1, if job aborted because a volume couldn't
                     be mounted;
                 = 2, if normal return.
            DCB:RNDEV = X'B', if device type not specified by user.
            DCB:VSND  = word displacement to DCB serial number table
                        (the INSN/OUTSN list). Note that VSND points
                        to the INSN/OUTSN control word, not to the
                        first serial number.
            DCB:PRIV SET; DCB:SWXV  RESET
```

Registers: All registers volatile except R6.

Operation: OPV is called by OPEN if the DCB is assigned to a file and serial numbers are specified. The routine opens the primary volume to the file by either calling OPENPV (if the volume is already mounted and verified) or MTPV (if the volume is not mounted and verified).

If the operator aborts the job instead of mounting the primary volume, OPV exits with an abort code. If a unit is not available to mount the primary volume, OPV exits with an error code. If the primary volume gets opened, its volume table of contents is read, if not already input. If the file's function is OUT, OUTIN, INOUT, any new volume serial numbers specified in the DCB are added at the end of the primary volume serial number table. Whatever the file's function, the primary volume serial number table is put into the DCB so that volumes in the set will be referenced in the proper order. If there is not enough space in the DCB for the serial number table, OPV exits with an error code.

If the file has keyed or random organization, all the secondary volumes in the set (if any) are opened. If a unit is not available to mount on one of the volumes, OPV exits with an error code. When OPV exits with an error code, OPEN will close all volumes opened to the file before returning the error code to the user. If the operator aborts the job because a new serial number that belongs to another private volume set was specified, it is purged from the primary volume's serial number table. All volumes opened to the file are closed and OPV exits with an abort code. If the operator aborts the job instead of mounting one of the secondary volumes, all volumes opened to the file are closed and OPV exits with an abort code.

When all volumes are opened to a keyed or random file or when the primary volume is opened to a consecutive file, the volume set's account directory is input. If the volume does not have an account number and the file's function is OUT or OUTIN, the file's account number is put into the account directory. If the volume does not have an account number and the file's function is IN or INOUT, OPV exits with an error code. If the file's account number is not the same as the private volume set's account number, OPV exits with an error code. If the account numbers are the same, OPV exits with a normal return.

ID

TYPR

PURPOSE

The TYPR module contains routines that perform four functions: (1) a multiple entry point routine to construct and output OC messages relating to magnetic tape handling; (2) a routine to print or type messages to the operator that do not require a response; (3) a routine to type an OC message and accept a response from the operator; and (4) routines to mount a private disk pack volume.

SUBROUTINES

MTPV

```
Purpose:   To mount a private volume and to verify the
           serial number.

Input:     (R6)     = DCB:ADR
           DCB:VNO  = volume number of the volume to be mounted
           DCB:BBUD = 0

Call:      BAL,SR4   MTPV

Output:    DCB:VNO = 0, if job aborted because volume not mounted;
                        if job aborted because volume belongs to
                        another set; or if unable to mount the volume
                        because unit unavailable.
           DCB:PAT,VDCTX = 0, if job aborted because volume not mounted.
           DCB:PAT = 0, VDCTX = foreign volume number if job aborted
                                because volume belongs to another set.
           DCB:PAT = 49, VDCTX = 0, if unit unavailable.
           AVR:SN = serial number of the requested volume.
           AVR:AVR  set.
           AVR:PRIM  set, if DCB:VNO = 1.
           AVR:VER  set while the serial number is being verified and
                    the bit map is being moved to the allocation table.
                    The field gets reset when AVR gets set.
           DCB:BUF1 = address of 512-word buffer containing VTOC.
           DCB:BBUD set, if VTOC modified.
           DCB:BCDA = disc address of VTOC (in private volume format).
```

Return: B *SR4

Registers: All volatile except R1,R6,SR1-SR4,D1,D3,D4.

Operation: The AVRTBL table is first searched to see if the requested volume is already mounted and verified. If so, MTPV exits. If not, the AVRTBL table is searched to see if the requested volume has been premounted by the operator. If so, the volume is verified, as described in the next paragraph. If not, the AVRTBL table is searched for (1) a free unit, (2) a unit with another volume that has been premounted, (3) a unit with a verified volume that has no users but is not a primary volume, or (4) a unit with a verified volume that has no users and is a primary volume. If a unit is not located, MTPV exits to RECOVER. If a unit is located, a MOUNT message is output to the operator. For cases 3 and 4, a DISMOUNT message is output prior to the MOUNT message. The system waits 20 seconds for the operator to respond with either a MOUNT or an ABORT key-in. If the operator aborts the job, MPTV exits with an abort code. If the operator responds with the requested MOUNT key-in, the volume is verified, as described in the next paragraph. Otherwise, the AVRTBL search begins again to see if the operator has mounted the volume on another unit. If not, another MOUNT message will be output.

A volume is verified by checking that the serial number on the volume matches the serial number specified for the unit and by moving the volume's cylinder bit map and NVAT table to the Allocation Table (HGP) entry associated with the unit. If the volume has not been initialized by the VOLINIT processor (and hence does not have a serial number), the message 'XXXX NOT INIT' is output to the operator and the MOUNT message 'REEL NO ERROR' is output to the operator, and the MOUNT message is repeated. If the volume mounted is not the first volume to be mounted for the file, MTPV checks to see if the volume belongs to the set. If not, the message 'XXXX NOT IN SET' is output to the operator and the MOUNT message is repeated. This gives the operator a chance to either mount the correct volume (assuming that all volumes do not have unique serial numbers) or abort the job. If the volume mounted is the first volume to be mounted for the file but is not a primary volume, it is not verified. The serial number of the primary volume is taken from the secondary volume and used instead as the volume to be mounted

ID

MUL

Purpose: The primary function is to build or rebuild a multilevel key structure. To decrease the size of the OPN overlay, the routines OPN and OPENDEV have been moved into this overlay. The description of these routines is found in the OPN Section of this documentation.

Entry:	MULMUL	-	OVERLAY call via T:OV.
	OPV	-	OVERLAY call via T:OV to open a file or a private volume set.
	OPENDEV	-	OVERLAY call via T:OV to open a DCB to a device.

DESCRIPTION

This module is entered at symbolic location MULMUL to build or rebuild a multilevel key structure. If only three level 0 half-granules exist, the top of the structure is set to the middle of these; otherwise, more than three must exist for a multilevel structure to exist. If the structure is being rebuilt, granules previously used in the multilevel structure are reused as long as they are available; otherwise, new granules are obtained via calls to GETDISK as required.

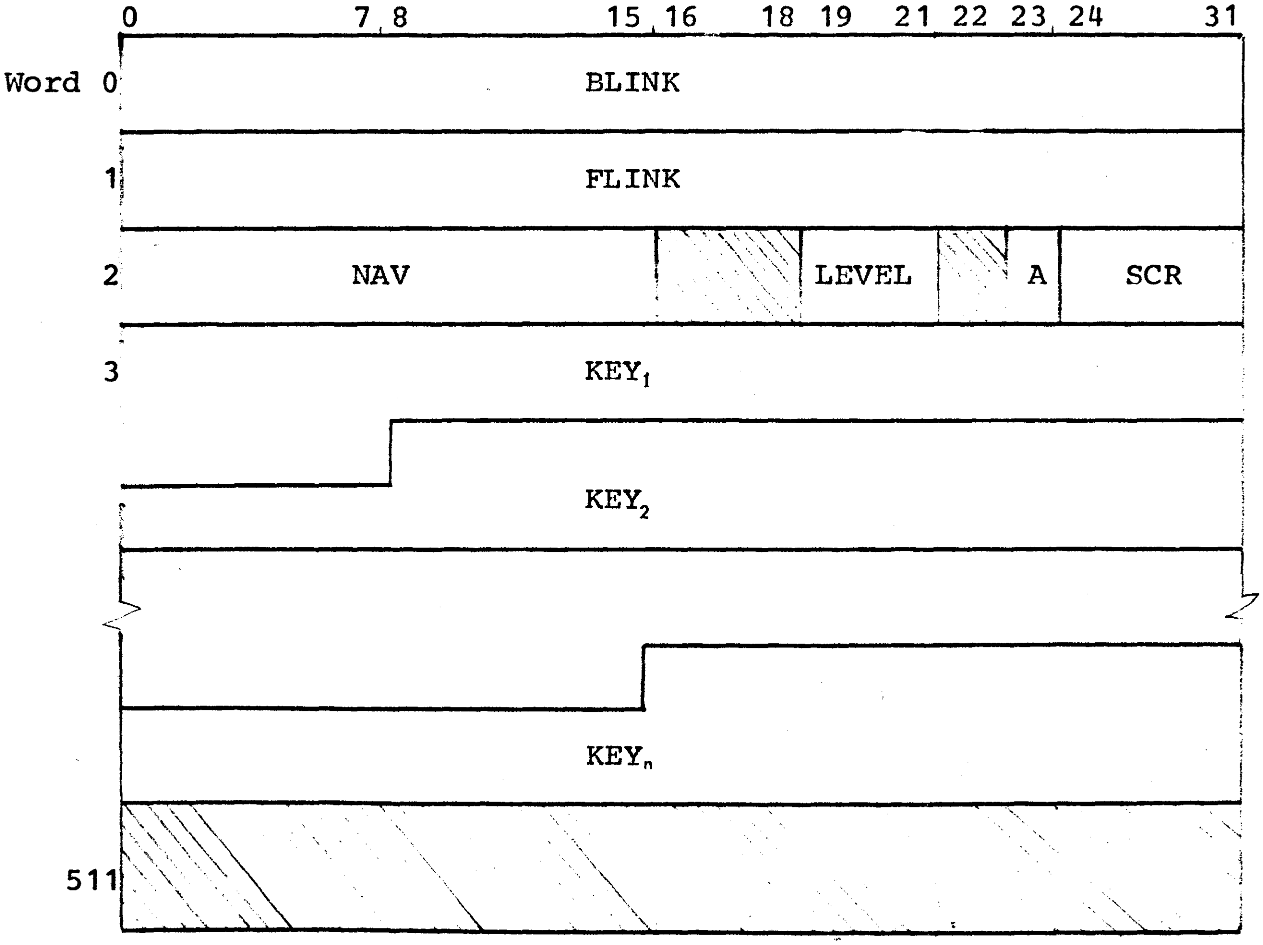
Level 0 is read sequentially starting at the lowest key position, and first keyin each level 0 half-granule is extracted via the code at symbolic location MOVEAKEY and placed in a level 1 granule along with a pointer to the appropriate level 0 sector. Level 1 granules are backward and forward linked in the usual manner. When level 0 has been completely processed, return is made to close, unless three or more level 1 granules were built, in which case, the above-described process is repeated for successively higher levels until we arrive at a level with two or fewer granules.

DATA AND TABLE FORMATS

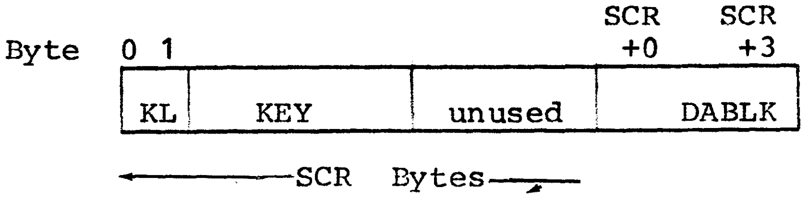
MASTER INDEX FORMAT FOR A HIGHER LEVEL INDEX

The formats of the higher-level granule and the higher-level key entry are detailed below.

Master Index



Key Entry



Where:

KL contains the number of bytes in the key. If the key entry in the level 0 half-granule pointed to by DABLK is not the first appearance of that key at level 0, the high order bit of KL (at level 1) is set.

KEY contains the key from the first key entry in an index half-granule on the lower level.

SCR is a field in the DCB and equals the maximum key length +1.

DABLK contains the disk address of the index granule on the lower level.

BLINK contains the disk address of the preceding index granule at this level, or zero, if none.

FLINK contains the disk address of the next index granule at this level, or zero, if none.

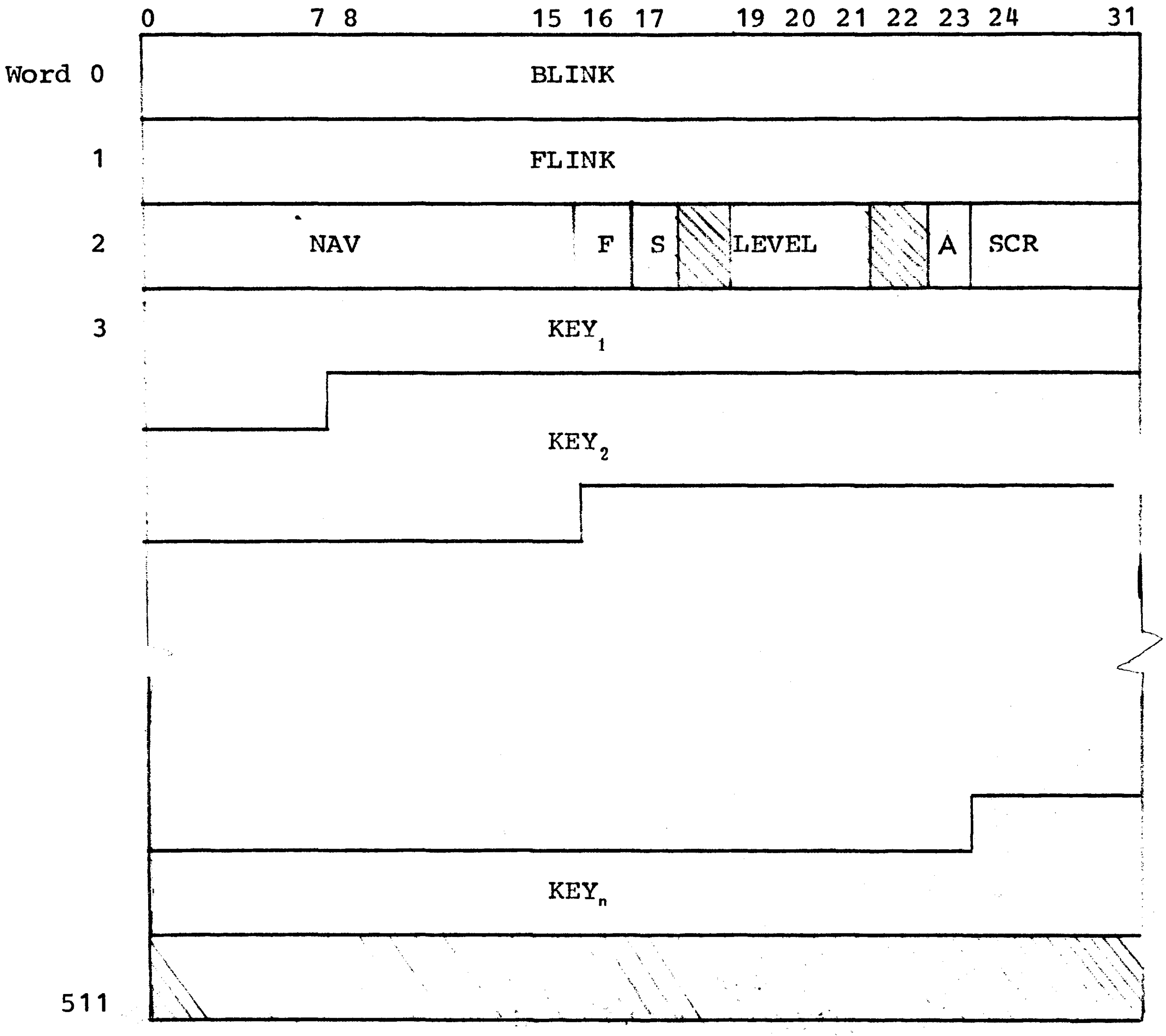
NAV contains the number of significant bytes in the index granule.

LEVEL contains the level of the index entries (the lowest is level 0, the next level is 1, and so forth).

A only meaningful at level 0.

The level 0 index granule format is detailed below followed by the level 0 key entry.

Master Index Format



where:

S = 1 if full-granule size; 0 if half-granule size

BLINK contains the disk address of the preceding index granule, or zero, if none.

FLINK contains the disk address of the next index granule, or zero, if none.

NAV contains the number of significant bytes in the index granule (i.e., points to the next available byte in the index granule).

LEVEL contains the level of the index entries (i.e., contains 0).

A is the added flag and indicates whether or not this index half-granule has been added since the current higher level index structure was created (0 means no, 1 means yes).

KEY is the key entry, shown on the next page.

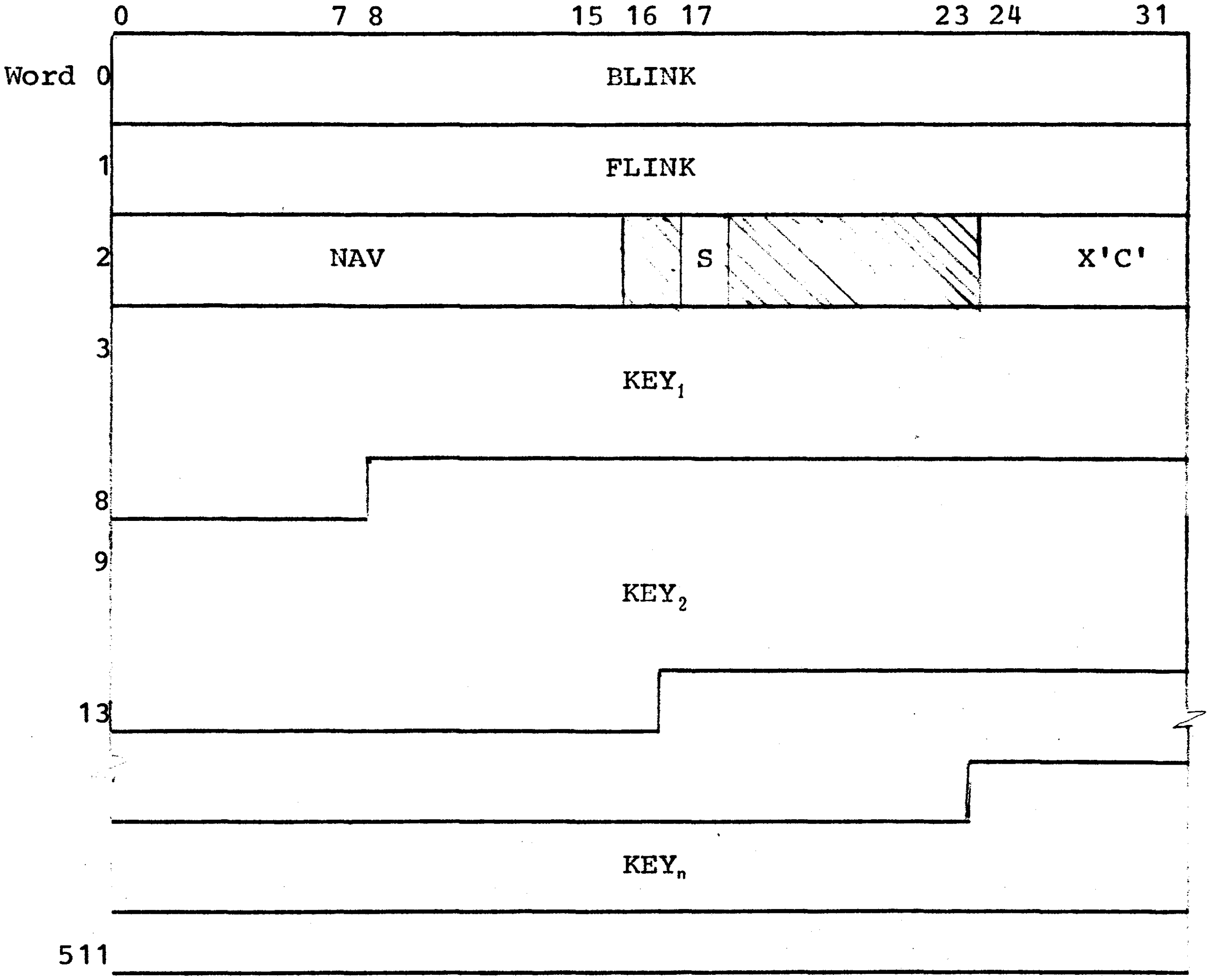
SCR 1 + KEYM (see description on 14-38)

F = 1 if FIT is in this granule; 0, otherwise. The FIT will occupy the final 80 words, if present.

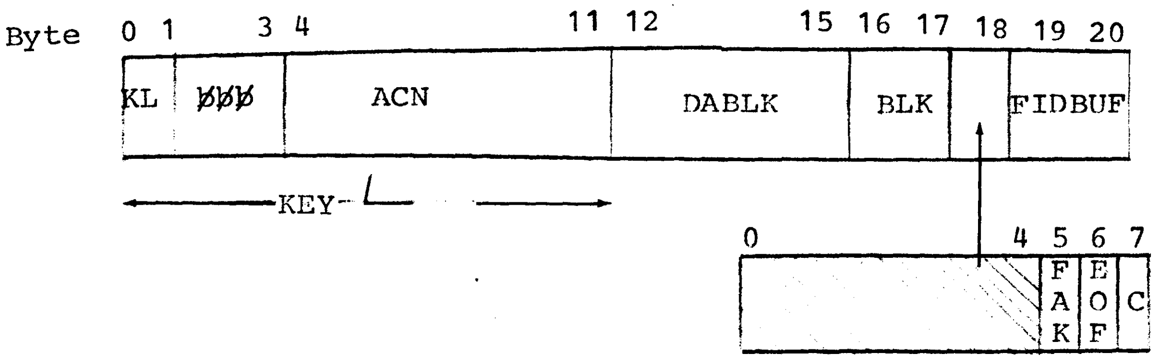
An account directory consists of a Master Index. An account directory index granule consists of key entries that contain an account number and the disk address of the file directory associated with the account. There is one account directory for all public files in the system. The permanent information about the public file account directory is contained in the ACNCFU. ACNCFU is not used for private account directories.

MASTER INDEX FORMAT FOR THE PUBLIC FILE ACCOUNT DIRECTORY

Master Index Granule Format



Key Entry Format



where:

S=0 for full-granule size; =1, for half-granule size

KEY=11 the number of bytes in the key

ACN contains an eight-byte EBCDIC account number.

DABLK contains the disk address of the file directory associated with this account.

BLK=0 has no meaning for an account directory.

FAK=1 indicates that this entry is the first appearance of the key.

EOF indicates whether or not this key entry is the last in the file (0 means not EOF, 1 means EOF).

C=0 has no meaning for an account directory.

FIDBUF=0 has no meaning for an account directory.

BLINK, FLINK, and NAV have the same meaning as in the Master Index for Level 0.

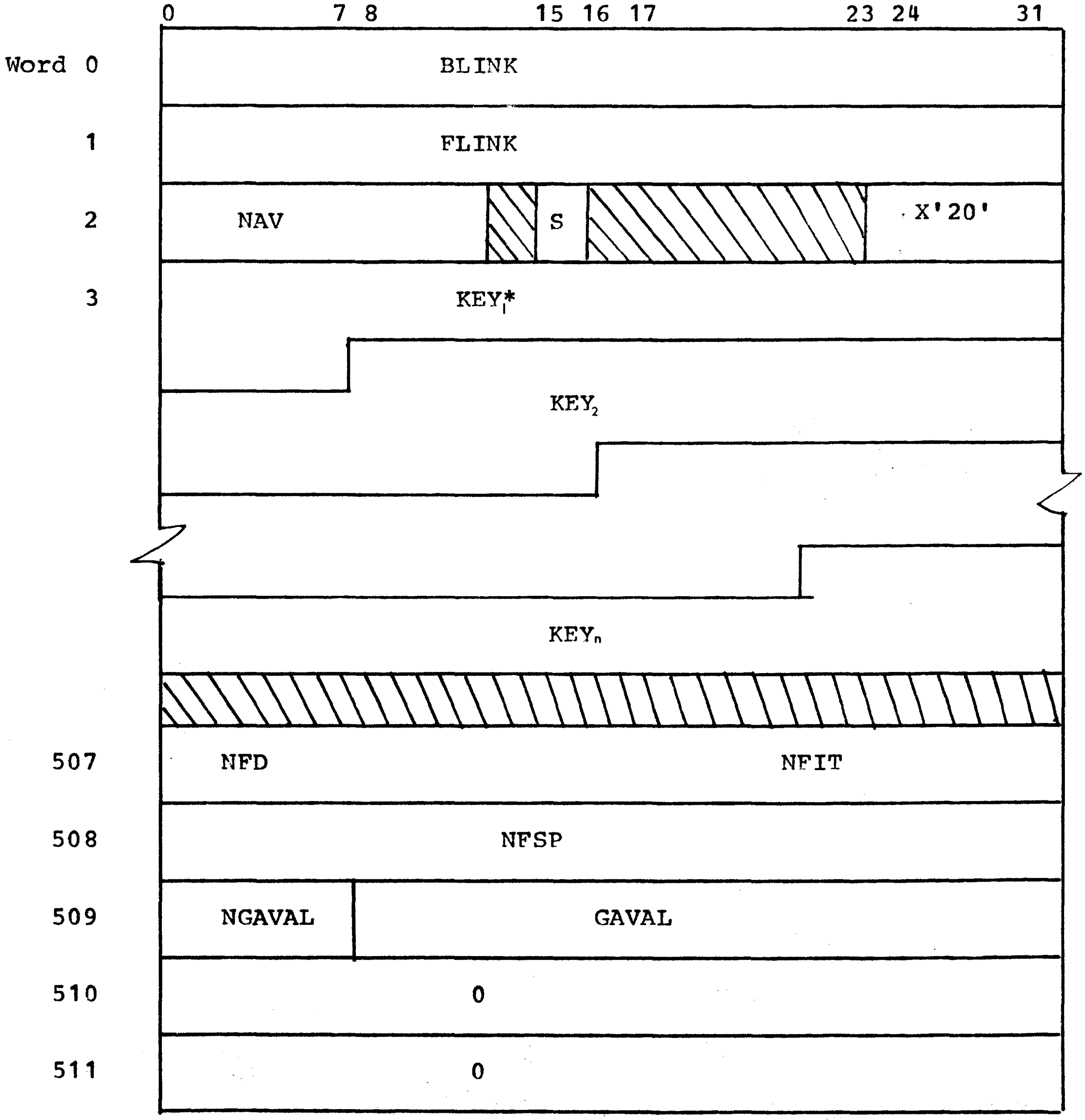
MASTER INDEX FORMAT FOR A PRIVATE VOLUME SET ACCOUNT DIRECTORY

	0 7	8 15	16 23	24 31
0	BLINK=0			
1	FLINK=0			
2	NAV = X'0021'		▨	X'2C'
3	KL=X'0B'	X'40'	X'40'	X'40'
4 5	ACN contains eight-byte EBCDIC account number.			
6	DABLK = X'00010004'			
7	BLK=X'		EOF,FAK=X'06' FIDBUF=X'00'	
8	FIDBUF=X'00'	▨	▨	▨
255	▨	▨	▨	▨

FILE DIRECTORY

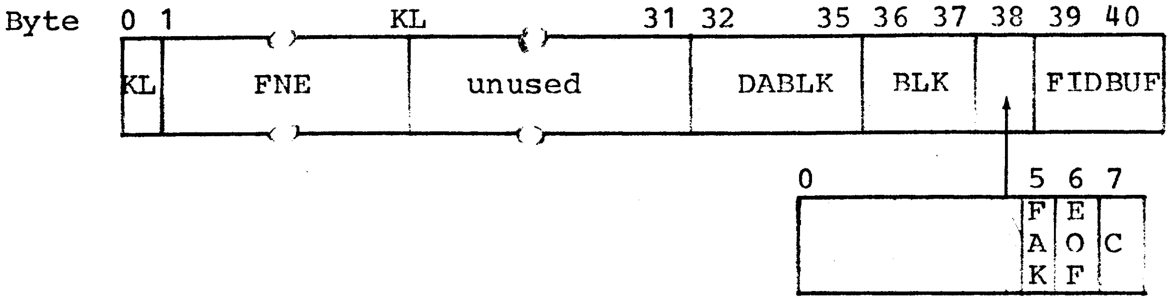
A file directory consists of a Master Index (MI) and a set of File Information Tables (FITs). A file directory index granule consists of key entries that contain the name of a file in the associated account and the disk address of the file's FIT. A FIT is located on a granule allocated to the file and contains all the information necessary to open a file. Information about the file directory itself (its mini-FIT) is contained in the last three words of the first block of its Master Index. The information from the mini-FIT is maintained in the FILCFU when the file directory is being updated. Public and private file directories have the same formats. However, since there is only one account per private volume set, there is also only one file directory. The file directory always begins on granule 2 of the primary volume in the set.

Master Index Granule Format



*The first entry in the file directory is a null entry for use with the NXTF option.

Key Entry Format



where:

S=1 for full-granule size; =0, for half-granule size

KL contains the number of bytes in the key.

FNE contains an EBCDIC file name.

DABLK contains the disk address of the file's FIT.

BLK Descriptors

The first byte contains the dynamic descriptors:

bit 0 = 1
bit 1 = 0 in file directory; but set to 1 in DCB field if we are creating a new synonymous file.
bit 2 = 0 in file directory; but set to 1 in DCB field if a synonymous file is being processed.
bit 3 = unused
bit 4 = 1 if file has been modified since it was backed up by FILL.
bit 5 = 1 if file has been modified since it was last backed up by an incremental.
bit 6 = 1 if file has been modified since it was last backed up by a SAVEALL.
bit 7 = 1 if file has been modified since the last FILL.

The second byte contains the static descriptors:

bit 0 = 1 if file has a password.
bit 1 = 1 if the file is a SYNON.
bit 2 = 1 if the file is a random file.
bit 3 = unused
bit 4 = 1 if file is not to be automatically backed up.
bit 5 = 1 if access date is not be updated.
bit 6 = 1 if file is not to be semi-automatically deleted.
bit 7 = 1 unassigned.

FAK=1 indicates that this entry is the first appearance of the key.

C=0 has no meaning for a file directory.

FIDBUF=0 has no meaning for a file directory.

BLINK, FLINK, and NAV have the same meaning as in previous MI formats.

NGVAL, GAVAL have the same meanings as in the FILCFU (see File Directory CFU (FILCFU)) and are written to the FD from the FILCFU when the file is closed and the FD is updated. These two fields (the mini-FIT) are written on the last three words only of the first half-granule in a file directory.

NFD # of granules allocated to this file directory.

NFIT # of granules allocated to FITs for random files in this account.

NFSP #of granules currently available in the free sector pool of this directory.

FILE INFORMATION TABLE (FIT) FORMAT FOR AN ORIGINAL FILE

	0 7	8 15	16 23	24 31*	
9 words	FNE (in TEXTC format)				
Optional Entries	03	LE=0	NDW=2	NAW=2	1
	Password				
	05	LE=0	NDW=X'10'	NAW=X'10'	
	Read ACNs				
	06	LE=0	NDW=X'10'	NAW=X'10'	
	Write ACNs				
	04	LE=0	NDW=2	NAW=2	
	Expiration Date				
	0F	LE=0	NDW=2	NAW=2	
	Last Access Date				
	10	LE=0	NDW=2	NAW=2	
	Backup Date				
	0E	LE=0	NDW=2	NAW=2	
	Creation Date				
	0A	LE=0	NDW=3	NAW=3	
	Modification Date				
	0D	LE=0	NDW=1	NAW=1	
	File Size				
	0C	LE=0	NDW=7	NAW=7	2
	E | FDA				
	TDA				
	NGAVAL	GAVAL			
	CCBD		O	SLIDES	
	SMI				
	SREC				
	LDA			NOSEP CYL	
	09	LE=1	NDW=3	NAW=3	
	ORG	KEYM		NOSEP CYL	3
	LSIDES		LRDL0	SPARE	
	NSF		FID		

*The FIT starts in word 4 for consecutive files and in the 80th word from the end for keyed or random files.

NOTES:

1. These coded entries are optional; presence of the entry is indicated by the byte 0 code. If present, entries occur in this order.

2. See CFU format for meaning of fields.

3. See DCB for meaning of fields.

where:

ACN is account number. Maximum of 16 total Read and Write ACNs. Each ACN is an eight-byte EBCDIC entry with trailing blanks. If there is no Read ACN entry, any ACN can read the file. If there is no Write ACN entry, no one can write in the file except the ACN that created the file.

Dates are of the form MMDDHHYY.

where:

MM is numerical month.
DD is day of month.
HH is hour of day.
YY is last two digits of the year, all in EBCDIC BYTES.

E = 1 if file contains no record.

File Size contains the number of index and data granules allocated to keyed and consecutive files; or the number of data granules allocated for random files.

FNE contains the EBCDIC name of the file.

NAW contains the number of available words in the entry (not including the control word).

NDW contains the number of significant data words in the entry (not including the control word).

NSF contains the number of names synonymous to this file.

LE is the last entry flag and indicates whether or not this parameter is the last entry in the FIT (0 means no, 1 means yes).

Password is an eight-byte EBCDIC entry with trailing blanks.

FILE INFORMATION TABLE (FIT) FORMAT FOR A SYNONYMOUS FILE

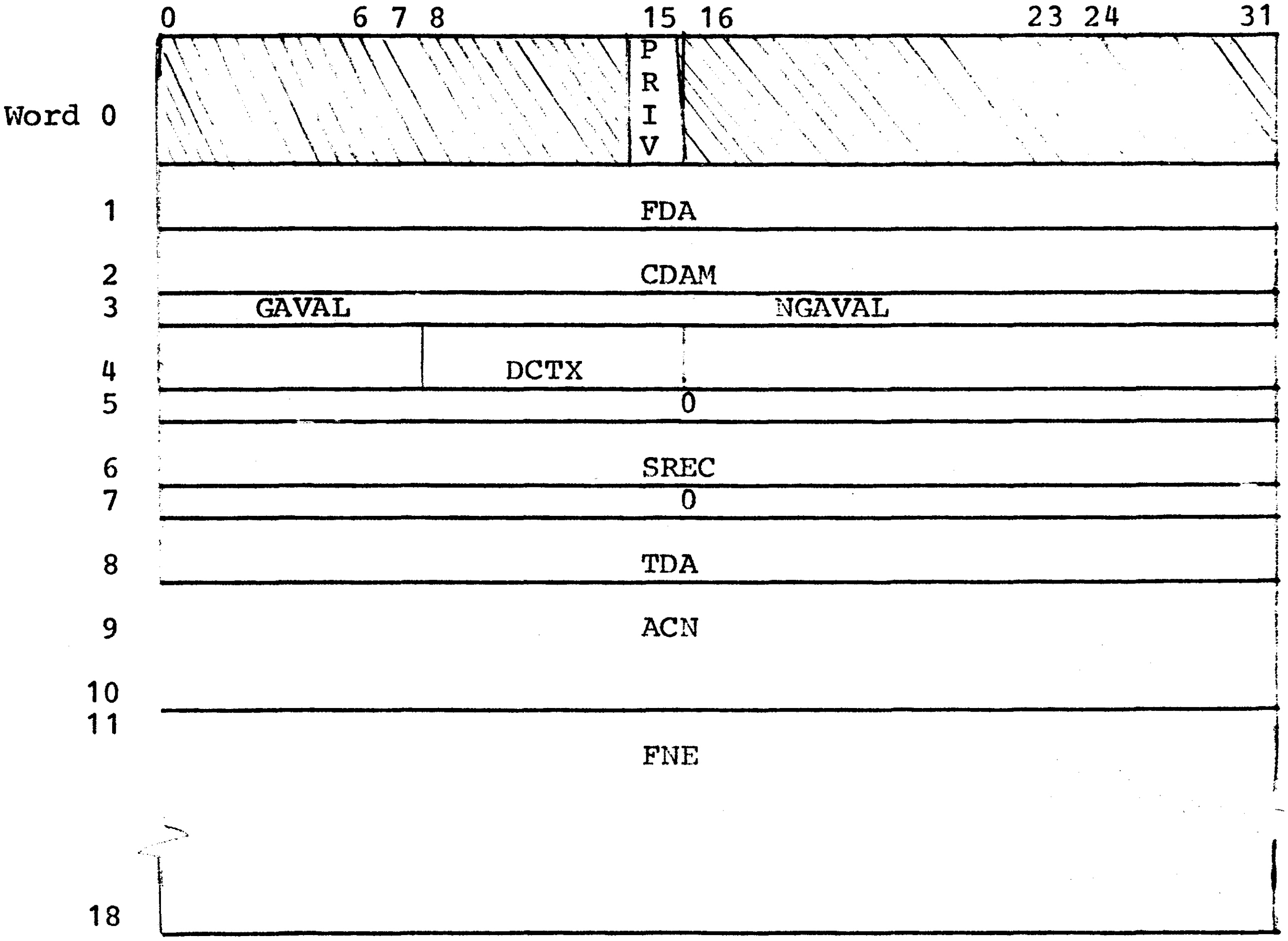
0 7	8 15	16 23	24 31
0B	LE=1	NDW=8	NAW=8
FNE (TEXTC Format)			
0B	LE	NDW	NAW
SYNON FILE NAME (TEXTC Format)			*

A synonymous file does not have a FIT, but if a synonymous file is accessed on a NXTF open with FPARAM, an X'08' error return is made with the above information passed as the FPARAM.

*This entry replaces the optional entries on the original file.

FILE DIRECTORY CFU (FILCFU)

FILCFU Format



FILCFU Contents

FIELD NAME	WORD	MEANING
ACN	9	The eight-byte EBCDIC account number associated with this file directory.
CDAM	2	Disk address of the current granule in DCB:BUF2.
DCTX	4	DCT index of the primary volume of a private volume set.
FDA	1	Disk address of the first granule in either a public account file directory or a private volume-set file directory.
FNE	11	Used by CLOSE as a temporary stack when errors are encountered while a user CFU is being closed. FNE+2 is used to indicate whether or not FD:SMI, FD:GAVAL, FD:FSP have been put into FILCFU for this account (0 means no, 1 means yes).
GAVAL	3	Disk address of the next available granule in the last cylinder allocated to this file directory; or zero if none.
NGAVAL	3	Number of available granules in the last cylinder allocated to this file directory.
PRIV	0	Private file flag, indicating whether the File Directory assigned to the FILCFU is public or private (0 means public, 1 means private).
SREC	6	Disk address of the file's FIT, when a file is being closed and the FD is searched to locate the FIT.
TDA	8	Zero, which means that a higher level index does not exist.

USER FILE CFU

User File CFU Format

	0	1	2	3	4	6	7	8 14	15	16 23	24 31
0	B	A	O	U	▨	W	R	NOU	PRIV	FUN	SLIDES
1	E	FDA									
2	CDAM										
3	NGAVAL							GAVAL			
4	CCBD									SCFU	
5	0										
6	SREC										
7	LDA										
8	TDA										
9 10	ACN										
11 • • • 18	FNE										

User File CFU Contents

FIELD NAME	WORD	MEANING
A	0	Active flag, indicating whether or not OPEN has assigned a DCB to the CFU (0 means no, 1 means yes).
ACN	9	Eight-byte EBCDIC account number (left-justified and blank-filled) of the file to which the CFU is assigned.
B	0	Busy flag, indicating whether or not OPEN has reserved the CFU, pending further checks, to be assigned to a file (0 means no, 1 means yes).
CCBD	4	Either the byte displacement to the next available byte in the last data granule of the file (SREC), which means that the blocking buffer was truncated; or 0, which means that the last data granule in the file (SREC) contains 512 words. Only used in the output mode.
CDAM	2	Number of granules in the file; only has meaning for random files.
E	1	1, if file contains no records.
FDA	1	Disk address of the file's first index granule at level 0.
FNE	11	EBCDIC name of the file (in TEXTC format) the CFU is assigned to.
FUN	0	Function of the file assigned to the CFU (0 means null, 1 - IN, 2 - OUT, 4 - INOUT, 8 - OUTIN).
GAVAL	3	Disk address of the next available granule in the last cylinder allocated to the file; zero if none.
LDA	7	Disk address of the file's last index granule at level 0.

NGAVAL	3	Number of available granules in the last cylinder allocated to the file.
NOU	0	Number of DCBs assigned to the CFU.
O	0	Level-1 flag indicating whether or not a level-1 index exists in a keyed file (0 means no, 1 means yes).
PRIV	0	Private file flag, indicating whether the file assigned to the CFU is public or private (0 means public, 1 means private).
R	0	1, if file has been read during this open.
SCFU	4	Word address of the secondary CFU associated with the file (the secondary CFU's SCFU entry will point to the original CFU).
SLIDES	0	Either (1) a tally of the number of index half-granules allocated at level 0 since the current multi-level index structure was created, or if none exists, since the file was first opened; or (2) a tally of the number of index granules allocated at the current level while the multi-level index structure is being (re)created; or (3) the value 255, which means that a new multi-level index structures should be built when the file is closed (unless LSLIDES in the DCB equals 255 and a level-1 index exists).

SREC	6	Disk address of the last data granule in the file; only used in the output mode. For random files, SREC contains the FIT location.
TDA	8	Either (1) the disk address of the first index granule at the top of the multi-level structure; if one exists; or (2) the disk address of the middle index half-granule, if there are three level-0 index half-granules and the file is keyed; or (3) 0, which means that either the file is consecutive, or that the file is keyed and there are, at most, two index half-granules.
W	0	1, if file content has been modified during this open.

CONSECUTIVE FILES

All information in the consecutive file is contained in full-granule sized records. There are no Master Index half-granules used. Many files consist only of control granules which will be flinked and blinked and will contain all record data and control information. Each record is broken into segments with length less than or equal to 2048 bytes. Record segments which are shorter than 2033 bytes are blocked into the control granules in monitor blocking buffers for input or output; but record segments of greater length are written from, or read to, the user's buffer directly.

Zero length records are treated as no record on output. Records which are less than 2049 bytes long appear as one segment, unless the record is shorter than 2033 bytes but longer than the amount of space available in the final granule of the file at the time it was written. If such is the case, the record is broken into two segments: the first, filling out the current granule and the second, starting a new final granule. Records with length greater than 2048 bytes are broken into a sufficient number of 2048-byte segments to reduce the remainder to less than 2049 bytes. This remainder is treated as described above.

Record Segments

Each record is broken into segments 2048 bytes.

Case 1. Record Segment 2033 bytes

Control Word:

bit 0 = 0 (for blocked)

bit 1 = FAK =1, for initial segment of record
0, if not initial

bit 2 = C =0, for last segment of record,
1, if otherwise

bit 3 = 0

bits 4-15 = # of data bytes in segment (call it m)

bits 16-31 = word position of previous record segment control word in this granule; or if this is the first segment in the granule, 0.

The next $\frac{m+3}{4}$ words contain the data of the segment.

Case 2. Record Segment 2033 bytes

Control Word:

bit 0 = 1 (for unblocked)
bit 1 = FAK
bit 2 = C
bit 3 = 1, if preceding word in the granule is a backspace control word (see below).
0, if this is the first record segment of the granule or if the preceding record segment in the granule is also unblocked.
bits 4-7 = # of data bytes in the segment less 2033.
bits 8-31 = generalized disk address of the granule containing the data starting at byte 0.

Case 3. Backspace Control Word. (This word is used only when a granule contains a blocked segment followed by an unblocked segment whose control word is in the same granule. The word is inserted following the data of the blocked segment and preceding the control word of the unblocked segment.)

bits 0-15 = 0
bits 16-31 = word position of previous record segment control word in this granule.

Control Information for Control Granules

Word 0 Blink
Word 1 Flink
Word 2

bit 0 = 0, if no case 2 entries appear in the granule;
granule;
1, otherwise
bit 1 = 0, for granule not full;
1, if otherwise
bits 16-31 = If bit 1=0, the word position of the next available word in the granule.
If bit 1=1, the word position of the last segment control word in the granule.

The initial granule of a file contains the FIT for that file in words 4-83 of the granule. Word 3 contains a dummy segment control word of type described in Case 1 above with FAK = 0, C = 0, and M = 320.

If a record is deleted from a consecutive file, the FAK bit in the first (or only) segment control word for the record is reset to 0.

The following statements apply to the use of granules for consecutive files:

1. If the remainder of a record to be written is at least 2033 bytes long, up to 2048 bytes will be written unblocked except in one very rare circumstance (see Paragraph 4 below).

2. If the remainder of a record to be written is less than 2033 bytes long, it will be written as one or two blocked segments. If the remainder will not fit entirely in the appropriate granule, as much as will fit is placed in that granule, and the remainder is placed in the succeeding granule as a continuation record segment.

3. All bytes of each granule are used except:

   a. some number of bytes in the final granule of the file beyond the end-of-file;

   b. up to 15 bytes at the end of a data granule for an unblocked segment;

   c. the final four bytes of a control granule will contain a backspace control word if a blocked segment would have otherwise started there;

   d. up to three bytes per blocked final segment of a record will be unused if the record length is not congruent to zero modulo 4.

4. If the data granule of an unblocked record segment would fall into a different volume of a private volume set from the volume containing the control word for that segment, as many bytes from the start of the segment as will fit into the control granule are written in a blocked fashion to fill out the control granule.

The following changes to the CFU (FIT) and DCB are made:

1. TDA in the CFU (FIT) contains the number of records in the file.

2. FDA and LDA in the CFU (FIT) now contain the appropriate granule addresses as opposed to half-granule addresses.

3. SMI in the CFU is not used for consecutive files.

4. The TRN bit in word 5 of the DCB is 1 only if the most recently executed operation on the file was a read backwards.

5. The fourteenth word of the DCB (W14) contains one of the following:

   a. 0 if at BOF
   b. the contents of TDA, if at EOF
   c. the sequential record number of the record most recently read or written.

6. The nineteenth word of the DCB (W19) contains the direction (+ or -) and the number of records that must be skipped from the position indicated in W14 prior to a data transfer operation (read, write, or delete).

7. The CMD halfword in word 20 of the DCB contains a word position in the granule pointed to by DCBCDAM (see below).

8. DCBCDAM in word 21 of the DCB contains a disk address of a granule reflecting (in conjunction with CMD and W14) the location in the file at which the most recent data transfer operation took place.

It should be noted that all positioning operations for consecutive files will be done with no I/O. Positioning operations are PRECORD, PFIL, and OPEN with extension. When these operations are encountered, the appropriate modification is made to W19 of the DCB. Only when a data transfer operation is about to take place will the positioning be effected; and at that time, there will be three known points in the file which can be used as a starting point (beginning-of-file, end-of-file, and the position reflected by W14, CMD, and DCBCDAM of the DCB). The starting position chosen will be the one which requires the fewest record skips to be made.

On a delete forward operation on a consecutive file, all vacated granules will be returned to the availability pool at that time rather than when the file is released. This is in contrast to the current method.

ID

Tape File Management

INTRODUCTION

A file is an organized collection of information that is created and accessed only through the Monitor. The Monitor deals with three kinds of tapes; labeled, ANS, and unlabeled tape. Only labeled and ANS tapes contain "files" in the sense discussed in this chapter. Labeled and ANS tapes conform to rigid formats including system-imposed sentinels descriptive of the tape and the files. Unlabeled tape is devoid of system information; record management and file format are the user's responsibility.

The functional and operational overviews apply to labeled and ANS tapes. Information relevant to unlabeled tapes is noted when appropriate.

STRUCTURE AND IDENTIFICATION OF TAPES

Sentinels, Files, Volumes

In the discussion which follows, certain definitions are in order. A file is a logical concept. It is a collection of information. A volume can be equated to a single reel of tape and is more easily thought of as a physical entity. Thus, one file may span several volumes. One volume may contain several files. A label or sentinel is a record at the beginning or end of a volume or a file which serves to identify or delimit, that volume or file. A tape set is an ordered set of volumes. A block is a physical tape record.

System (Monitor) information, called 'sentinels' are automatically placed at the beginning and end of volumes, and at the beginning and end of each file by the system. With the exception of header and trailer labels, the user has virtually no control over the presence or content of these sentinels. Except the end-of-file sentinel, he may not cause or prevent sentinels from being written nor is he informed when they are read by the system.

Sentinels serve to identify the volumes and delimit or describe the files. A brief description of the major sentinels is shown below. See Section VH.16 for a detailed description of sentinel format and content.

MAJOR SENTINELS

LABELED	ANS	DESCRIPTION
:LBL	VOL1	Identifies the volume with a unique serial number.
:BOF	HDR1	Identifies the beginning of a file (or a portion of a file if the file spans several volumes).
:EOF	EOF1	Identifies the end of a file. (For ANS, an EOF1 followed by no other major sentinel signifies end of tape set)

:EOV	EOV1	Signifies physical end of a volume with the continuation of the present file on the next sequential volume. (There is no :EOV/EOV1 if this volume is the last in a tape set.)
:EOR	----	Signifies end of file set, which in the absence of :EOV implies end of tape set.

Data Formats

Blocked Records

Blocking is the process of packing several logical records into one physical record(a block)for output. Deblocking extracts the logical records from the physical record for input. The advantages of blocking are to reduce actual I/O to a small percentage of user I/O requests and to make more efficient use of the tape medium (reduce tape consuming inter-record gaps).

For unlabeled tapes, there is no record blocking. Each read or write request transfers one physical record to/from the user's buffer.

For ANS tapes also, there is no record blocking. One physical record (one block) is transferred per read/write request. However, the presence of the HDR2, EOF2, and EOV2 records will assist the user should he wish to block his data. The COBOL run-time subroutines for example, intercept the data and, on the basis of blocking information in the HDR2, perform blocking/ deblocking. HDR2 is always created on output files (from information in the DCB via the ASSIGN and/or OPEN command) and updates the DCB data (if present) on input.

For labeled tape, record blocking and deblocking is automatic whenever the system is asked to perform an I/O transfer of less than 512 words. The operation is similar to blocking disk records. When a labeled tape file is created, the Monitor automatically introduces a 512-word core buffer between the user's buffer and tape. Each write causes the user's record to be transferred to the next available location in the blocking buffer along with Monitor-introduced control information describing the record. Thus I/O does not actually occur until the Monitor buffer is full. During the Read of an existing file, a reverse procedure is performed. A 512-word blocked record is read by the system into a Monitor blocking buffer, and the user-requested record is located within the block and transferred to the user's buffer.

If a user write-request is sufficiently large so that, after filling up the existing monitor buffer, more than 512-words remain to be written from the user's buffer, then the system will write the remainder of the record directly from the user's buffer (after writing a small control record describing what is to follow). This large record is called an unblocked record. When unblocked records are read, they are read directly into the user's buffer. The user may not bypass blocking or control the block size. Backspace or forespace operations are performed by reading each block and determining the number of records within the block.

Keyed Records (labeled tape only)

If the file organization is keyed, the user may read individual records by specifying the record's key. That is, he need not position the tape head within the file before accessing a record. Two types of file organization (keyed and consecutive) are permitted as a means of backing up the RAD/DP file system. Thus, while keyed organization on LT permits users to access individual records by key, this was not the intent of providing keyed organization. This dual organization also permits RAD/DP files of keyed or consecutive organization to be copied to labeled tape and later recopied back to RAD/DP with the recopied version identical to the original.

Classification of Tapes

In a system where a number of users are being processed in rapid succession, it is not practical for an individual user to assign his tape to a particular physical tape unit. One program might specify input from unit 1 and output to unit 2, while the immediately following program might want to use both of these units for scratch tapes. Actually, neither user really cares which tape unit he uses; the user only wants to be sure that he is reading and writing the proper tapes and has scratch tapes available as needed. Therefore, in the UTS Monitor, user assignments are made to the tapes rather than directly to the tape drives. This implies that the selection of the drive is relegated to the Monitor and the operator. To perform this selection and subsequently housekeep the tape resources, a tape identification system is called for. Every tape to be used must be identified to the Monitor. Depending on the kind of tape needed, this identification either may occur when the Monitor needs the tape and requests it or may occur in advance without a request from the Monitor.

Tapes can be divided into two categories. 1) Specific tapes that have input data on them or that are to be saved as output data. All input tapes are specific. 2) Scratch tapes that are of no concern to the user either before or after his job. Specific labeled or unlabeled tapes are identified by a serial number that appears on the !ASSIGN card or in M:OPEN as (SN, sn).

All ANS tapes are specific tapes, in that the system protects an unexpired ANS tape from inadvertent destruction. The protection feature eliminates the possibility of ANS tapes being members of the "scratch pool" since every usage would involve a time-consuming protection check. ANS tapes are identified either by the serial number or the file name of the first file on the volume. For labeled or ANS tapes, the serial number is the reel number that will be written on the tape. For unlabeled tape, the serial number is known by its presence in the DCB (via the ASSIGN or OPEN).

Scratch tapes, on the other hand, have no information which can identify them. It is not important which reel of tape is used as a scratch tape, and the Monitor can use the same tape for one job's scratch tape that is used for the previous job. Therefo after some scratch tapes have been made available, the Monitor will continue to use them and will not ask for more unless there are not enough for some job. The "scratch pool" is the group of scratch tapes which have been made available to the Monitor (by the operator) for scratch usage.

The "function" parameter on the !ASSIGN or M:OPEN is used among other things, to further identify the tape to the system. Each !ASSIGN card or M:OPEN should indicate the function, i.e., how the tape is going to be used.

(IN)	for input tapes
(OUT)	for output tapes
(OUTIN)	for scratch tapes
(INOUT)	for update tapes

IN and INOUT imply that the tape already exists. It is therefore a specific tape. It must have a serial number or, if ANS, a file name. Otherwise the error message I/O ERROR 46 will occur when an attempt is made to read it. OUT or OUTIN imply that the tape is to be created. A labeled tape which is OUT or OUTIN needs a serial number only if it is going to be saved after the the job.

If a tape is not ANS, has no serial number, and is being used OUTIN, it is by definition a scratch tape. It will be allocated to the user from the scratch pool.

In summary, the serial number, the file name, and the function used in various combinations will identify the tape to the system and the operator for the purpose of selecting a specific tape or a tape from the scratch pool.

ANS TAPES

ANS LABEL FORMATS

Volume Header Label

Sp. Nf.

VOL1	vol. serial number	access ('0')	blank	owner ident (blank)	blank	blank
1-4	5-10	11	12-37	38-51	52-79	80

File Header Labels

First Header Label

HDR1	file name	volume number first volume of set	volume sequence number	file sequence number	generation number (blank)	version of generation (blank)	creation date (byy-ddd)	expiration date (byy-ddd)	access ('0')
1-4	5-21	22-27	28-31	32-35	36-39	40-41	42-47	48-53	54

block count ('000000')	programming system identifer (Xerox/UTS-D00)	blanks
55-60	61-73	74-80

Second File Header

HDR2	format (F=fixed length D=variable lng.dec. V=variable length U=undefined)	block length	record length	density ('2')	backward read continuation flag ('0')	blank
1-4	5	6-10	11-15	16	17	18-50

buffer offset ('00')	blank
51-52	53-80

Trailer Labels

End of Volume

EOV1	Same as HDR1	block count	Same as HDR1
1-4	5-54	55-60	60-80

EOV2	Same as HDR2
1-4	5-80

End of File

EOF1	Same as EOV1
1-4	5-80

EOF2	Same as EOV2
1-4	5-80

User Labels

User Header Labels

UHL1	user specified
1-4	5-80

User Trailer Labels

UTL1	user specified
1-4	5-80

ANS LABEL GROUPS

Volume Header Group (VHG)

VOL1

File Header Group (FHG)

HDR1
HDR2
UHL1 optional
Tape Mark

End of File Group (EFG)

EOF1
EOF2
UTL1 optional
Tape Mark

End of Volume Group (EVG)

EOV1
EOV2
UTL1 optional
Tape Mark

End of Data Group (EDG)

EOF1
EOF2 } End of File Group
UTL1 optional
Tape Mark
Tape Mark Extra tape mark

ANS Tape File Structures

Single File Volume

VHG
FHG
data records
tape mark
EDG

Multi-File Volume

VHG
FHG-1
data records-file 1
tape mark
EFG-1
FHG-2
data records - file 2
tape mark
EFG-2
•
•
•
•
data records - file n
tape mark - n
EDG

Multi-Volume File

Volume 1
VHG
FHG
data records
tape mark
EVG
Volume 2
VHG
FHG
data records
tape mark
EVG
•
•
•
•

Volume n

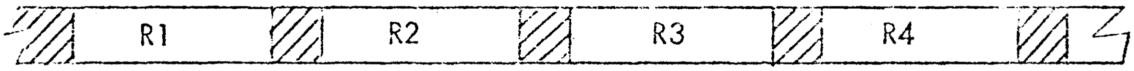
VHG
FHG
data records
tape mark
EDG

Multi-File Multi-Volume

VHG Volume 1
FHG
data records - file 1
tape mark
EFG
FHG
data records - file 2
tape mark
EVG
VHG Volume 2
FHG
data records - file 2
tape mark
EVG
VHG Volume 3
FHG
data records - file 2
tape mark
EFG
FHG
data records - file 3
tape mark file 3
EFG
FHG
data records file 4
tape mark
EVG (assuming end of tape coincides with last data block)
VHG
FHG
tape mark (file complete on Volume 3)
EDG

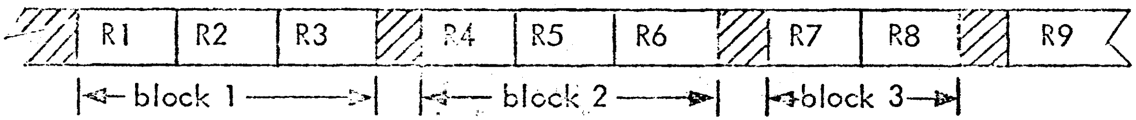
ANS DATA FORMAT

Fixed Length (F)



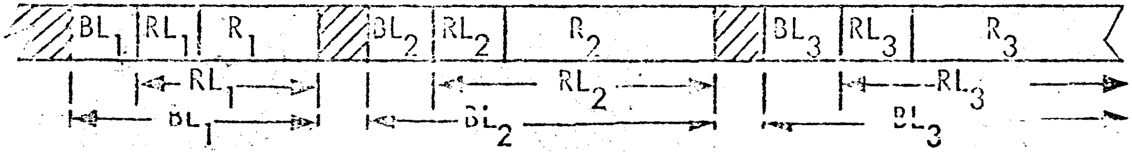
1 record/block, constant record length

Fixed Length Blocked (FB)



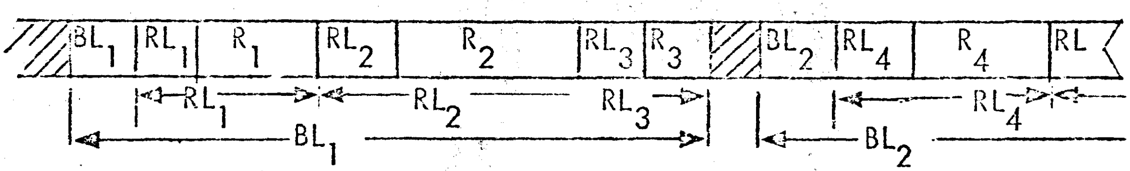
max. records/block = block size/record size
block size > record size

Variable Length (V)



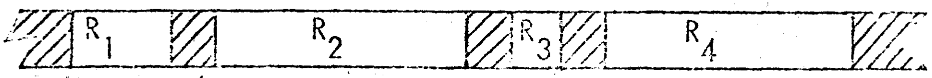
1 record/block, block size = record size + 4

Variable Length Blocked (VB)



record size < block size

Undefined (U) *



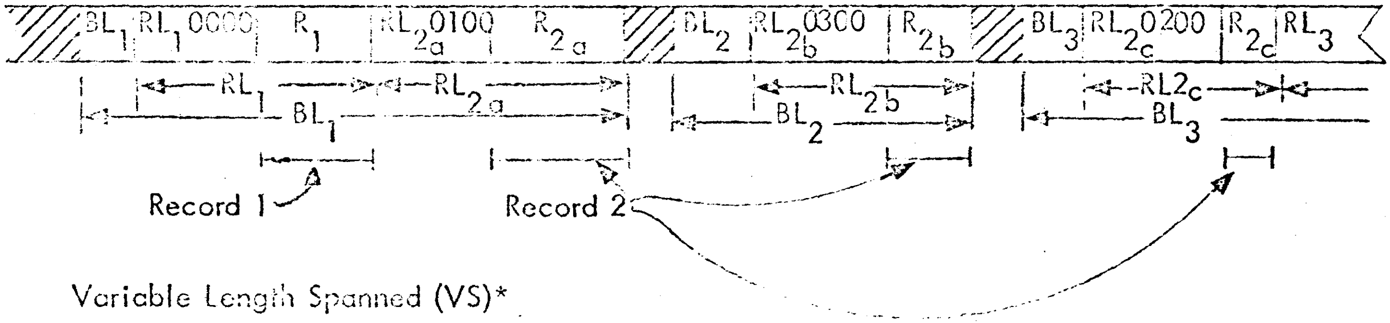
1 record/block, record length calculated from I/O data

Rn = data record "n".
BLn = 2 bytes expressing number of bytes in block "n" as a binary number, followed by 2 bytes of binary zeroes
BLn = same as BLn for record "n".

Variable Length Blocked Spanned (VBS)*

Similar to VB except the third byte of the "RL" field is coded as follows:

00** - this is the first, last, and only section of this record
01 - this is the first section of a record that spans blocks
03 - this is a middle section of a record that spans blocks
02 - this is the last section of a record that spans blocks



Variable Length Spanned (VS)*

Combines the Variable Length (V) and Spanned (S) features in a manner comparable to VBS, above.

*Spanning need not be specified for input files as each record specifies whether or not spanning has occurred.

**These flags may also be viewed as: bit 6 set = this is not first section
bit 7 set = this is not last section

LABELED TAPE

Labeled Tape, General Format and Sentinels

Shown below are two labeled tapes containing two volumes of file A, having a total of four records, and the one-record file B. The various sentinels are explained following the tape format illustrations. All sentinels begin on a word boundary.

Tape 1
Label Sentinel (:LBL)
Identification Sentinel (:ACN)
Tape Mark
Beginning of File A (:BOF)
User's Label (optional)
Tape Mark
Record 1 of File A
Record 2 of File A
Record 3 of File A
Tape Mark
End of Volume (:EOV)
Trailer Label (optional)
Tape Mark
End of Reel (:EOR)
Tape Mark
Tape Mark

Tape 2
Label Sentinel (:LBL)
Identification Sentinel (:ACN)
Tape Mark
Beginning of File A (:BOF)
User's Label (optional)
Tape Mark
Record 4 of File A
Tape Mark
End of File A (:EOF)
Tape Mark
Beginning of File B (:BOF)
Tape Mark
Record 1 of File B
Tape Mark
End of File B (:EOF)
Trailer Label (optional)
Tape Mark
End of Reel (:EOR)
Tape Mark
Tape Mark

where

:LBL identifies the reel number of the tape (SN). Reel number are four alphanumeric characters in length. Sentinel length is 12 bytes, including four padding bytes. The format is shown below.

:	L	B	L
X	X	X	X
4 PADDING BYTES			

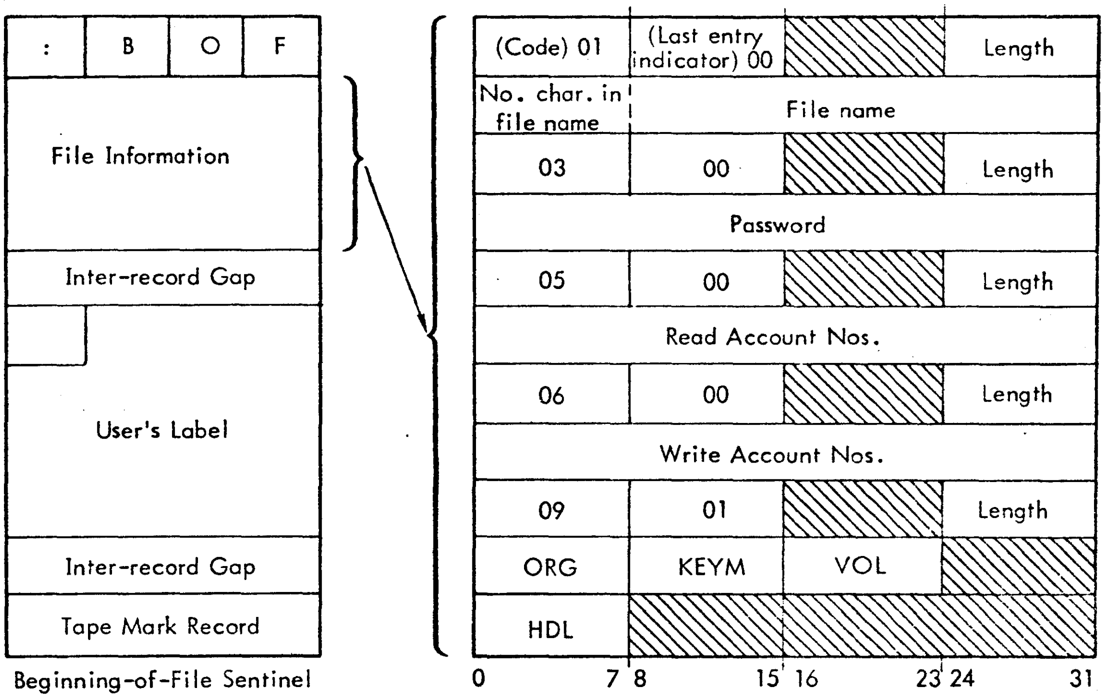
Label Sentinel

:ACN identifies the owner of the tape (ACCNT #), the expiration date, and the creation date, in that order.

The account number is eight alphanumeric characters in length, left-justified and in EBCDIC code. The dates are of the form $m_1m_2d_1d_2$b̸b̸y_1y_2, where m_1m_2 is the numerical representation of the month, d_1d_2, the day, b̸b̸ are blanks, and y_1y_2 are the last two digits of the year. The digits are in EBCDIC and the blanks must appear. Sentinel length is 28 bytes followed by a physical end-of-file (tape mark record). The format of the ACN Sentinel, also referred to as the identification sentinel, is shown below.

:	A	C	N
a_1	a_2	a_3	a_4
a_5	a_6	a_7	a_8
m_1	m_2	d_1	d_2
b̸	b̸	y_1	y_2
m_1	m_2	d_1	d_2
b̸	b̸	y_1	y_2
Inter-record Gap			
Tape Mark Record			

:BOF is the beginning-of-file sentinel consisting of the file-information record, the user's label (if specified), and a physical end of file (tape mark record). The format of the :BOF Sentinel is shown below.



The file-information record, as shown, contains several control words and the information associated with each of these. The control words have the following form:

Code	LEI	▨	Length

where

Code identifies the type of information following the control word. The codes are:

01 - file name. The file name may be a maximum of 31 characters. An additional byte is used to state the length of the file name.

03 - password (2 words, left-justified).

05 - Read account numbers.

06 - Write account numbers.

Each account number is left-justified, blank-filled, and two words long. The total number of Read and Write accounts must not exceed 16. Read accounts identify those who may have only read access to the file. Write accounts identify those who may read and write the file. None or All are also allowed.

09 - miscellaneous information, such as:

ORG - gives file organization, i.e., keyed or consecutive.

KEYM - specifies maximum length of the keys. Keys may not be greater than 31 bytes. An additional byte is used to specify the length of the key. On consecutive files, the length of the dummy key is assumed to be three; therefore, KEYM is ignored. On keyed files, if KEYM=0, maximum length is assumed to be 11.

VOL - on multi-reel files, this entry specifies the position of this tape in the file. For example, VOL=2 implies this is the second tape of the multi-reel file. VOL=1 indicates the beginning of the file. Every file begins with VOL=1 (including single-reel files).

HDL - specifies the length of user's label. If HDL=0, then no user's label exists and the following record must be a physical end-of-file.

LEI is the last-entry indicator; this entry in the control word indicates the end of the file information. The control words, along with the information they define, do not have to be in a particular order, but LEI must equal 0 if the file information entry is not the last one and must equal 1 if the entry is the last.

Length specifies the length, in words, of the information associated with a particular entry (i.e., following the code word).

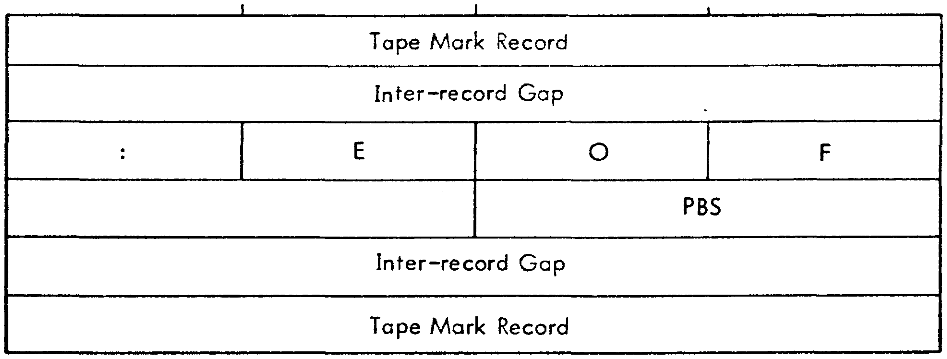
:EOV is the End-of-Volume sentinel:

<table>
<tr><td>:</td><td>E</td><td>O</td><td>V</td></tr>
<tr><td colspan="2"></td><td colspan="2">PBS</td></tr>
<tr><td colspan="4">Inter-record Gap</td></tr>
<tr><td colspan="4">Tape Mark Record</td></tr>
</table>

:EOR is the End-of-Reel sentinel:

<table>
<tr><td>:</td><td>E</td><td>O</td><td>R</td></tr>
<tr><td colspan="2"></td><td colspan="2">PBS</td></tr>
<tr><td colspan="4">Inter-record Gap</td></tr>
<tr><td colspan="4">Tape Mark Record</td></tr>
</table>

:EOF is the End-of-File sentinel:

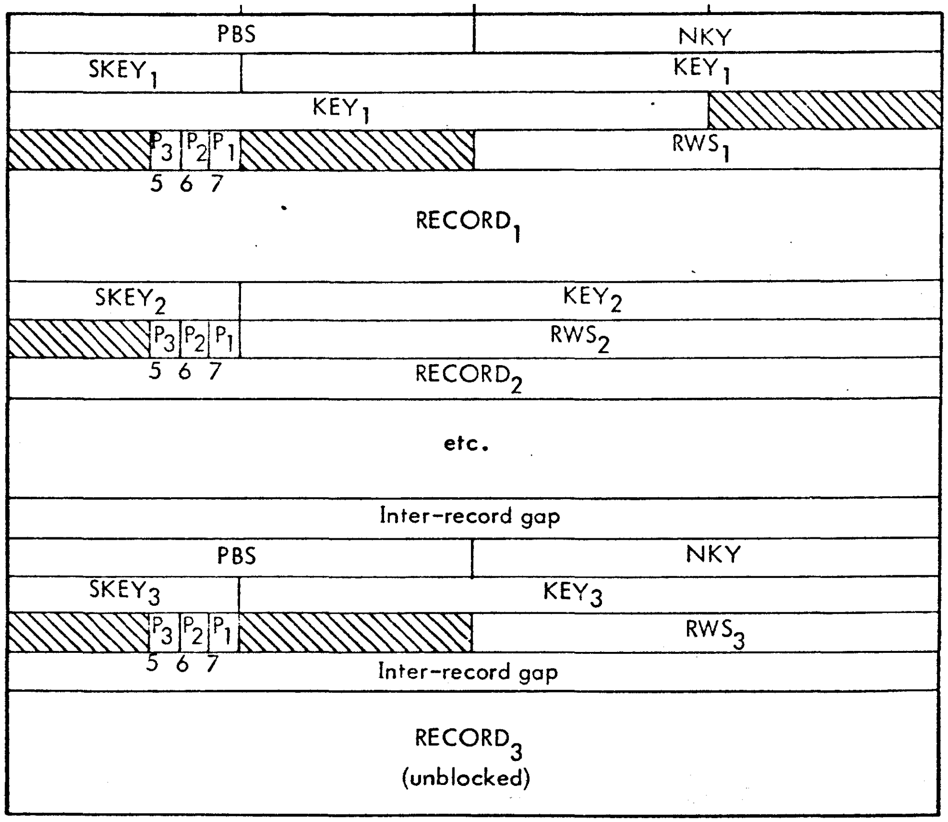


where

PBS is the previous block size, in bytes.

RECORD FORMAT

The labeled-tape record format for blocked records is shown below. The truth table following the diagram shows the possible combinations, of blocked/unblocked continued/not continued records. Figure 15-1 shows a number of examples of various types of records.

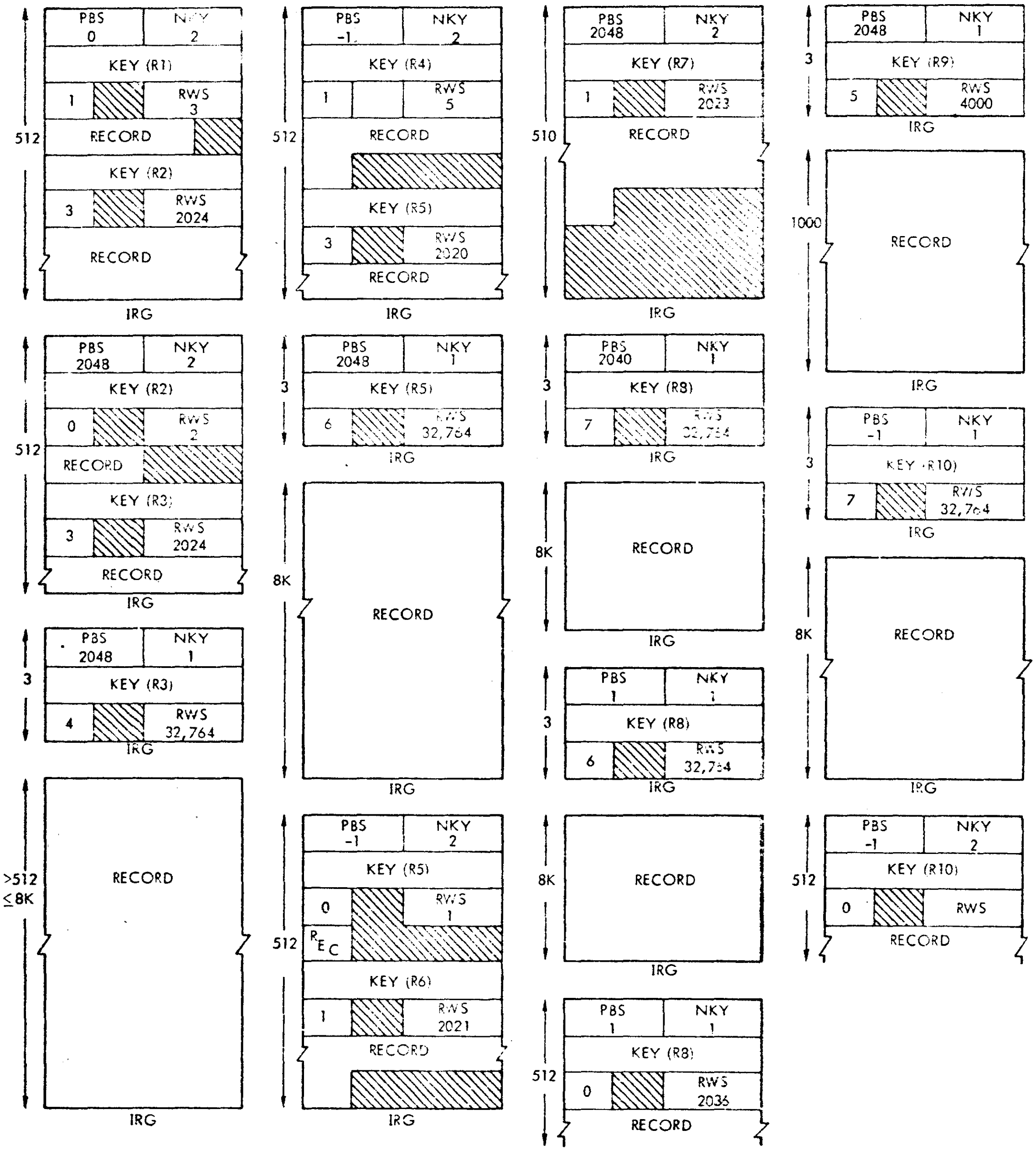


where

NKY is number of entries in block.

PBS is previous block size.

†Must start on word boundary.



Notes:
1. All numbers are decimal.
2. A new record will not be started in a block unless there is room for at least one full data word.
3. PBS — -1 whenever prior block is unblocked.
4. All keys are assumed to be three bytes long.

Examples of Different Types of Records

SKEY is size of key.

RWS is size of record in block.

P_3 is 1 if record is unblocked, 0 if blocked.

P_2 is 1 if record is continued into next block, 0 if record is not continued.

P_1 is 1 if this is first part of record, 0 if it is not first part.

Truth Table. Combinations of Record Types

Unblocked P3	Continued P2	1st. Segment P1	Meaning of Contol Bits
0	0	1	Record is wholly contained within this 512 word block.
0	1	1	1st record segment of N segments. Continued next block.
0	1	0	Not possible.
0	0	0	Nth record segment (N >1) of N. If N >2, ∴ N-1 was unblocked.
1	0	0	Physical record following is Nth segment of N (N > 1).
1	0	1	Physical record following is complete.
1	1	1	Physical record following is not complete. Size = 8K words.
1	1	0	Physical record following is not complete. Size = 8K words. This is ith segment of N (1 > i < N).

USER-SYSTEM INTERFACE: USER VIEWPOINT

Volume Creation and Access - The First Open

Volumes are created or verified by the execution of the first M:OPEN to the tape. At this time, the DCB will have been updated to reflect the !ASSIGN and the M:OPEN and the user will have supplied three crucial parameters:

1. TYPE

   One of the following must appear on the !ASSIGN card; or in the M:OPEN statement referring to a system DCB.

(ANSLBL, name)	for ANS tapes
(LABEL, name)	for labeled tapes
(DEVICE, 9T)	for 9-track unlabeled tapes
(DEVICE, 7T)	for 7-track unlabeled tapes

   It is also possible to specify both LABEL and DEVICE. Specifying both will indicate which kind of drive will be used for the labeled tape.

2. FUNCTION (IN, OUT, OUTIN, or INOUT)

3. SERIAL NUMBER

   Specific tapes require serial numbers unless they are ANS. ANS tapes require either a serial number or a file name.

Processing of the first OPEN will cause the tape to be selected and mounted, and the initial sentinels (:LBL/VOL1) to be created or verified.

If the function is output, :BOF/HDR1 is created. If the function is input, the :BOF/HDR1 sentinel containing the file name matching the one in the DCB is located at the completion of the OPEN. The tape is positioned for a read/write of the first data block.

Protection: If the function is output, the physical tape is subjected to a protection check. If an operator assigns an unexpired ANS tape for output, the monitor gives the operator a warning message. The operator may then direct the monitor to write over the unexpired tape. In an ANS protected system, the write over instruction from the operator will be rejected unless the user has a privilege level of at least X'CO'. (The protection mode is a SYSGEN parameter.)

Volume Sequence Check (ANS Tapes): In general, since a file may span several volumes, a request to process any file involves a volume sequence number check to ensure that the sections of the file which exist on different tapes are being accessed in their proper logical order. Volume sequence numbers begin with 1 and increment by 1 for each new volume which is spanned by the file. Processing of a file continues until an EOF1 label is encountered.

File Searching: File searching occurs over an entire volume set when the user requests that a specific file is opened. The tape file system uses the :BOF/HDR1 sentinel for ID purposes. Volume switching during the search is automatic. Thus, the user need not be concerned with file position. On unlabeled tape, the user must locate a desired file via explicit positioning operations.

Reading and Writing Records

ANS Noise Records: To comply with the IBM/ANS convention, all records of length less than 18 bytes read from tapes specified as ANS-formatted are bypassed as noise records. Any request for output of a record containing less than 18 bytes is written as an 18-byte record. The trailing bytes contain whatever data follows the requested data in core.

Volume Switching: Volume switching is automatic when reading or writing a file that is split between two volumes. Although the user can explicitly control volume switching, it is not necessary that he do so. On unlabeled tape, the user must explicitly request volume switching when informed that a physical end-of-tape reflective marker has been detected.

Concatenation (ANS Tapes Only): Concatenation of n files is the input process whereby n files of the same file name and format are treated as one logical file. The EOF1 labels are prevented from terminating the reading of records from the logical file until the nth EOF1 label is encountered. The concatenation parameter is then satisfied and the end-of-file status is reported to the user.

Data Transfer: An M:READ or M:WRITE transfers the information between the tape and the user's buffer. Since labeled tapes are blocked, this transfer is intercepted by the blocking logic, which packs or unpacks the logical records into a blocking buffer. The contents of the blocking buffer are then the physical record. (cf. Blocking/ Deblocking above).

M:REW: If the DCB associated with a labeled tape file is open, the file is positioned to its beginning, otherwise, the tape is not moved. If the DCB is associated with an ANS tape and is open, the tape is positioned following the VOL1 sentinel and the DCB is closed. If the DCB is associated with unlabeled tape, it is opened if closed, and the tape is positioned to loadpoint.

Closing: If the CLOSE REM option is specified, the tape is rewound and a message is typed requesting the operator to dismount the reel. Otherwise the tape heads remain at the current file and, if a DCB is opened using that tape, one of two actions occurs:

1. On input or update, the tape is scanned forward for the desired file.
2. On output, the tape is positioned to the end of the current file and the new file is written at that position.

Input tapes closed by the Monitor CLOSE routine are always saved. The REM (remove) option, if specified, is honored on all tapes; the PTL (position to label) option is honored on labeled tapes only. It positions the tape in front of the tape mark preceding the :BOF sentinel. The PTV option is honored on labeled and ANS tapes only. It positions the tape following the :LBL/VOL1 sentinel.

Output, update and scratch tapes closed by the Monitor CLOSE routine are handled as indicated below.

SAVE option is specified in CLOSE	REM is specified	The tape is rewound, SAVE and DISMOUNT messages are output
	PTL is specified	The file is positioned to the beginning-of-file; if the beginning-of-file is on another tape; SAVE and DISMOUNT messages are output.
	No options	No action is taken. (Tape head is somewhere between tape marks surrounding records, depending on last record read or written, and direction.)
SAVE option is not specified in M:CLOSE for OUT or OUTIN tapes.	OUTSN is not specified	The tape is rewound and remains a scratch tape. Any other scratch tapes saved (due to CVOL) for the file are released for other use.
	OUTSN is specified	The tape is rewound; DISMOUNT message is output if REM specified in CLOSE FPT.

SYSTEM-OPERATOR INTERFACE - OPERATOR VIEWPOINT

Note: The following is an overview of the operator-tape system interaction. Detailed operator protocol is available in the UTS Operations Manual.

Mounting Tapes (Scratch and Specific)

Scratch Tapes: When the Monitor needs a scratch tape (i. e., a request for a non-ANS, OUTIN tape with no serial number), it types a message on the operator's console:

!!SCRATCH ndd

where n is the channel letter and dd is the device number (for example, A80, A81, etc.). The operator should then mount a scratch tape on the specified unit and key in the following via an interrupt.

SCRATCH ndd, sn

where sn is a serial number and is, in this context, simply an identifier for the operator. Once the Monitor has selected a tape drive and requested a scratch tape on it, a tape must be placed on that drive and identified via a keyin. It is not possible to use AVR on a scratch tape. If an AVR is attempted after a scratch request or if some other tape is mounted, the Monitor cycles to the next available unit and requests a scratch on that. This process can be endless if the proper SCRATCH keyin is not received.

Alternatively, the operator can enter scratch tapes into the scratch pool before they are actually needed (by using the same SCRATCH keyin), and he can select any available tape drive. After the operator selects a tape drive, the Monitor will continue to use that drive as a scratch tape for all jobs and will not request more unless there are not enough for some job.

Note that for each job, the Monitor limits the number of scratch tapes that the user can assign. The default value is a SYSGEN parameter, but it can be changed with a LIMIT or RES control command. If the limit is exceeded, the job will be aborted.

Note that any scratch file can be assigned to the disk rather than to the tape, since the information is usually needed only during the job.

Specific Tapes: A specific tape is always identifieu either by serial number (labeled, ANS, and unlabeled) or file name(ANS only). When the monitor needs such a tape, it types the message:

1. MOUNT ndd, sn

   OR

2. ANSSCRATCH ndd, filename, VSN (All ANS OUT tapes opened without serial number)

   OR

3. ANSMOUNT ndd, $\left\{\begin{matrix}\text{SN}\\\text{filename}\end{matrix}\right\}$, VSN (All ANS tapes not covered by 2.)

where ndd is the tape drive

SN is the serial number

VSN is the volume sequence number

The operator should place the tape on the requested drive and respond with a keyin message to confirm to the Monitor that he has mounted the proper tape on the proper drive.

MOUNT ndd {,SN}

OR

ANSMOUNT ndd {,SN}

OR

ANSSCRATCH ndd {,SN}

For the ANS operator keyins (ANSMOUNT or ANSSCRATCH), the presence or absence of a serial number varies with the circumstances.

Automatic Volume Recognition (AVR)

If the tape drive is equipped with an ATTENTION button, for some tapes there is a simpler way to respond to a MOUNT or ANSMOUNT request. This is called automatic volume recognition (AVR) and requires no operator interrupt or typing. The operator merely pushes the RESET, ATTENTION, and START buttons (in that order) and this process alerts the Monitor that a tape has been mounted on that tape drive. AVR consists of a sequence of operations designed to read introductory sentinels (ANS or labeled) of a volume, and leave that volume positioned properly for the first open. It follows that only a tape which already has labels is amenable to AVR. Tapes of this category are:

1. Labeled IN or INOUT tapes.
2. ANS IN or INOUT tapes.

If the label is incorrect (or missing), the Monitor types one of the following messages:

!!REEL NO ERR or !!AVR ERR

The operator must recover by mounting the correct tape and using either AVR or MOUNT. The latter message will also result from trying to use AVR on a scratch tape.

Premounting Tapes

Tapes may be mounted in advance (i. e., before the Monitor needs them) with certain restrictions.

1. Any scratch tape may be premounted with a SCRATCH ndd, sn keyin.
2. Any tape which can use AVR can be premounted with AVR (see AVR above).
3. Any labeled tape can be premounted with the MOUNT ndd, sn keyin.
4. Any unlabeled specific tape can be premounted with a MOUNT ndd, sn keyin.
5. ANS input tapes can be premounted with the ANSMOUNT ndd, sn keyin providing the system is in the "semi-protected" mode. (In the "protected" mode, no serial numbers are permitted from the operator's console).

AVR Table Bit Definitions

AVRTBL (AVRTBLSIZ=number of tapes, BATAPE=first tape DCT index) (EQUed in PASS2)

Entry Format for Labeled Tapes

0 31	32					in use		39	40 47	48 63
Serial Number	PUB	POS	AVR	SCR	HLD	PTL	UPL	OPN	NOU	TPOS

Serial Number	Four-byte EBCDIC serial number of volume mounted.
PUB	Device public (1) or private (0).
POS	Tape at beginning (1) yes (0) no.
AVR	Volume serial number verified, yes (1) or no (0).
SCR	Scratch tape mounted, yes (1) or no (0).
HLD	Volume can be dismounted, yest (1) or no (0).
PTL	PTL option specified in last M:CLOSE, yes (1) or no (0).
UPL	Labeled tape, out or outin.
OPN	Tape position known by system, yest (1) or no (0).
NOU	Number of DCBs open to files on the volume.
TPOS	Number of tape marks between load point and present position of tape.
in use	Set when drive is allocated.

ANS-AVR TABLES

The ANS-AVR tables contain filename, volume sequence number, Julian expiration date, and flags.

These tables, generated by SYSGEN, are parallel to the tape drive segment of the device control tables and are indexed by device index minus the first tape device index.

AVRFNMT - Six Words

0 17	18	19 23
FILENAME	VSN	EXPIRATION

FILENAME name of the first file on the tape volume (TEXTC).

VSN volume sequence number (binary)

EXPIRATION Julian expiration date (five bytes representing YYDDD in EBCDIC.

ANSFLGS - One Byte

0	1	2 3	4	5	6	7
AT	TC	ERR	AK	RO	BLP	MS

AT tape is ANS (0 = no, 1 = yes).

TO type of DCB (0 = NON-ANS, 1 = ANS)

ERR error flags (00 = no error, 01 = NOT ANS, 10 = NOT EXPIRED, 11 = ANS VOL).

AK ANS key-in processed (0 = no, 1 = yes).

RO Access protection (0 = unrestricted, 1 = read only).

BLP BLP option specified (0 = no, 1 = yes).

MS MOUNT OR SCRATCH (0 = Mount, 1 = Scratch)

AVRSID - One Word

0   31
V1SN

V1SN - The serial number of the first volume of the tape set.

Tape Protection

If an unexpired ANS tape is threatened by a pending output operation, one of the following messages will appear:

NOT ANS — An ANS DCB is about to write on a non-ANS tape.

ANS VOL — A non-ANS DCB is about to write on an unexpired ANS tape

NOT EXPIRED — An ANS DCB is about to write on an unexpired ANS tape.

In the semi-protected mode, or if the user has a privilege of at least X'CO', the operator can override the warning with:

```
OVER   ndd   ,{sn}
```

VOLUME ACQUISITION

In general, acquisition of any volume (a physical tape reel) is handled by the interaction of KEYIN, TYPR, AVR, and the TAPECHK-USECHK subroutines.

This interaction is as follows: every OPEN and volume switch (to labeled, ANS and unlabeled tape) is driven to TAPECHK-USECHK. TAPECHK-USECHK (located in the OPNL module) is the allocation center for all tapes. A determination is made to establish whether the user already has the tape. This is done by locating the serial number or the file name in AVR or AVRFNMT respectively. If the serial number is located and the bit in J:ASPIN matching the tape drive is set, the user "has" the tape and the drive. Control returns to the calling OPEN routine.

If the user does not have the tape, control passes to TYPR which selects a drive, posts the appropriate tape request message, and blocks the user. During this time, one of the following events may happen: (1) an AVR sequence is performed on the tape drive, or (2) a keyin is entered on the operator's console, or (3) no action is taken.

If no action is taken, the user will be unblocked after a system specified time quantum. TYPR repeats the tape request message and blocks the user again.

If AVR or keyin occurs on the requested drive, the user is unblocked and TYPR returns to USECHK. USECHK recycles the AVR or AVRFNMT to match the serial number or file name to the one in the user's DCB. Having established a match and authorized the allocation per user and system limits, USECHK sets the J:ASPIN bit and returns to the calling OPEN routine.

With regard to protection check, USECHK branches to PROTCHK (in LBLT). PROTCHK examines all non-IN-tapes to determine suitability for writing, e. g., ANS/non-ANS, expired/unexpired, etc. If the tape is unsuitable, the appropriate message is posted through TYPR, and the user is blocked until some operator action occurs (KEYIN or AVR). Control returns to PROTCHK, then to USECHK. The action must be appropriate to the circumstances (i. e., protective or semi-protective mode, ANS or non-ANS tape, open by serial number or file name, etc.). These checks (on the propriety of the operator response) are conducted at the KEYIN and USECHK levels.

FILE IDENTIFICATION

File identification is the recognition or generation of file header and trailer labels. These functions are performed during open and close. OPNL, OPLO, and LBLT are the modules for these functions. File identification begins with M:OPEN from the user. OPN merges the FPT into the DCB and then drives to OPNL, OPLO, or OPNTP for input, output or unlabeled tape, respectively.

Output

When an output file is opened, OPLO first calls TAPECHK to guarantee the existence of the tape. Then, if the tape is newly mounted, introductory sentinels are written (:LBL and :ACN or VOL1) and AVR:___is set. :BOF or HDR1/HDR2 sentinels are written (optional). User labels are written. When an output file is closed, LBLT (which contains the close routines for all tapes) writes the standard ending sentinels (:EOR, :EOF, :EOV, or EOF1, EOF2, EOV1, EOV2) and the tape (optionally) positions the user's trailer sentinels.

Input

When an input file is opened, OPNL searches the volume set until the requested file (in the DCB) is located (in :BOF or HDR1). The user's label is transferred if present and requested. File attributes are transferred to the DCB. The position of the tape (i. e., number of tape marks from the beginning) is updated.

VOLUME SWITCHING

A file may span several tapes. Therefore, during the reading or writing of the file blocks, it may become necessary to switch from the current volume to the one which contains the next file section. Volume switching is this process. It amounts to a pseudo close (of the current volume) and a pseudo open of the next one. Accordingly, volume switching code is found in the open and close modules for labeled and ANS tapes; i.e., OPNL, OPLO, and LBLT.

Volume switching is triggered by one of **five conditions:**

1. An explicit M:CVOL from the user (CVOL in LBLT).
2. Encountering an end-of-tape reflective spot during output (WRITEL in LBLT).
3. Encountering a :EOV or EOV1 during input (reading), i. e., the file spans the next tape (CHKEOF in ARDL).
4. Encountering two tape marks on ANS input tapes during an open input operation (OPNL).
5. Failure to find a file on the current volume during the open searching process (input) OPNL.

An explicit M:CVOL drives directly to CVOL in LBLT. A pseudo close is performed. If the last operation was write, trailer labels (:EOV, :EOR or EOV1) are written. A "switch volume" flag is set in the DCB and the open routines (OPNL or OPLO) are called. The open routines access the "next" tape (if there is one) through TAPECHK-USECHK. If the function is output, the :LBL, :ACN, :BOF, or VOL, HDR1, HDR2 sentinels are written with information indicating the volume number relative to the volume set. Control returns to CVOL which exits. If the function is input, OPNL finds the file and verifies the file section number (VOL in BOF or VSN in HDR1) then returns to CVOL.

Conditions 2-5 enter CVOL in LBLT at the Monitor entry point CVOLA, the only significant difference in the processing being the linkage.

DATA TRANSFER

Processing an M:READ or M:WRITE begins in IORT with the decoding of the FPT and transfer of control to RDL (input) or WRITEL in LBLT (output). Reprocessing of the request occurs, viz; checking the last operation for unusual conditions such as end-of-file or end-of-tape indicators.

If the DCB is ANS, control is transferred to MSROTHR in IORT where the entire block is transferred to/from the user's buffer. The post-processing of ANS requests occurs in POSTANS (ANSTP) which checks for unusual conditions.

If the DCB is labeled tape, the read or write operation is translated into the blocking process (see Blocking). If the operation is "read with a key", the proper record must be located. Several blocks may need to be inspected before the proper one is obtained. Keyed reads are processed in ARDL.

MODULE DESCRIPTIONS - SUMMARY

The module summary identifies the modules which address themselves primarily to tape file management. The tape functions which they perform are noted below.

ANSTP Contains resident ANS routines. The main ones are: (a) special ANS AVR logic, and (b) post processing of ANS data transfers. This module is not included in a non-ANS system.

ARDL Contains routines to do keyed read and read reverse on labeled tape; special file/volume end and concatenation logic for ANS.

AVR (Automatic Volume Recognition) Performs volume identification by inspecting the front sentinels of ANS or labeled tape.

IORT Central module for intercepting reads and writes. Drives device I/O (input). Contains numerous secondary functions used throughout the I/O system.

KEYIN Interprets all operator keyins. Legal tape requests cause user to be unblocked. SOLICIT is marked to indicate that a keyin has been received.

LBLT Performs output functions on labeled and ANS tapes; volume switching, sentinel writing, protection checking, and writing of data records (labeled tape only). Also closes all tapes.

OPLO Contains the routines which open output tapes and perform volume switching.

OPNL Contains the routines which open input tapes. Includes TAPECHK and USECHK which are used by all open functions (input/output) for all tapes (labeled, ANS and unlabeled).

POS Performs tape positioning functions.

RDL Performs pre-processing input functions for labeled and ANS tapes.

TYPR Outputs Monitor tape requests and tape protection warnings; blocks the user until AVR or operator keyin indicates that the tape is available.

ID

ANSTP - contains resident routines for processing ANS tapes.

The ANSTP module is present only in systems that include the ANS optional feature. It contains both open and closed subroutines that are functionally part of other modules. These routines are described within the other module descriptions.

ID

ARDL - contains routines to do keyed reads and read-reverse on labeled tape.

ARD2

Purpose: To perform keyed reads of labeled tape.

Entry: B ARD2

Exit: Normal B MSREXIT
Error B KEYER3

Operation: See following flowchart. If the last block was truncated, it is reread. If a blocking buffer is not allocated, one is obtained and a blocked record is read. The routine determines direction of search by comparing the user-requested key with the key at the current position in the blocking buffer. ARD2 then searches the buffer in the established direction, calling on RDBLK or RDBLKR when necessary to read a new block. When a match is found, or when it is determined that the desired key does not exist, the desired key or the closest key to it is transferred to KBUF.

ARD3

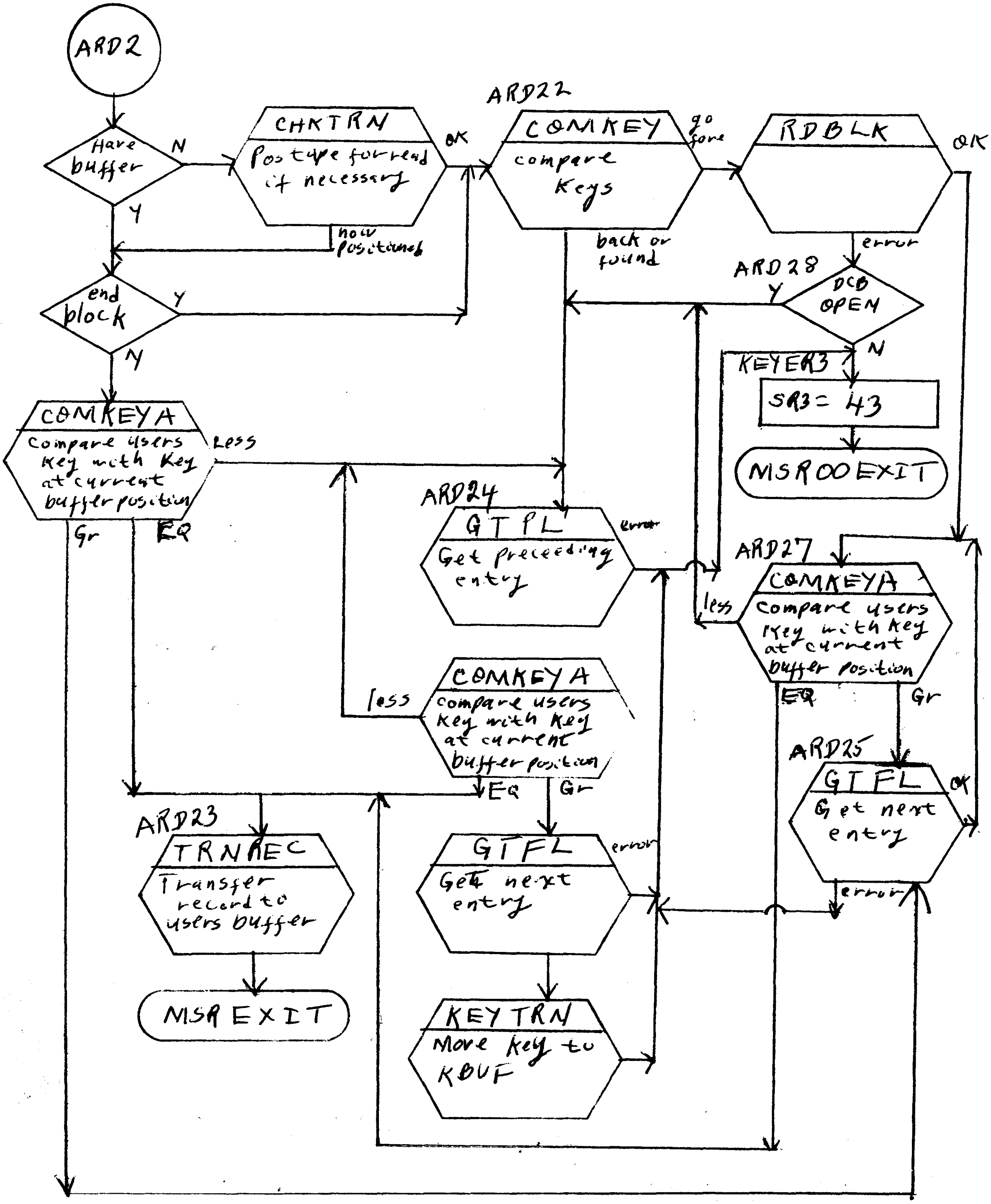
Purpose: To perform backwards Read of labeled tape.

Entry: B ARD3

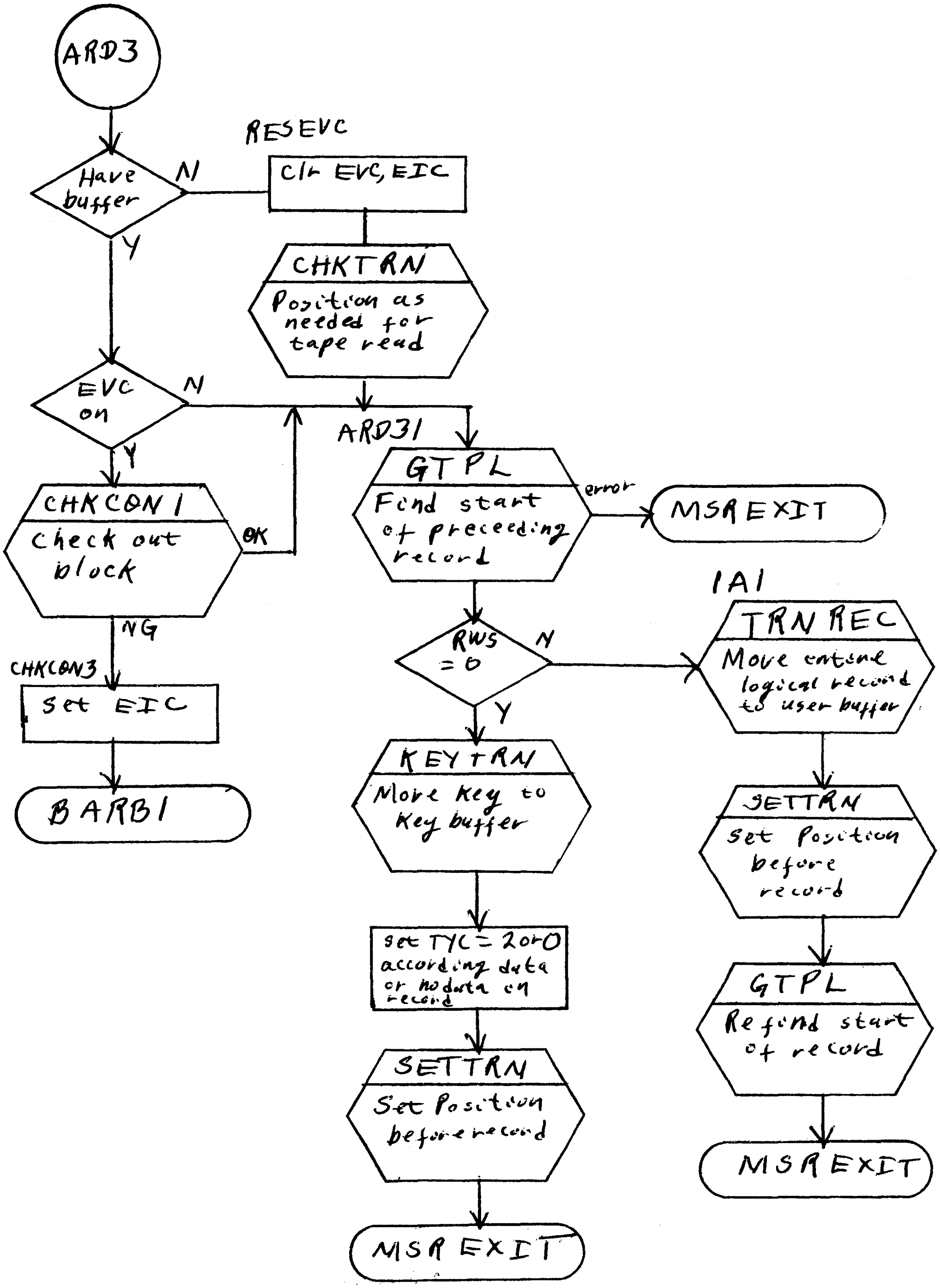
Exit: B MSREXIT

Operation: See following flowchart. If the last block was truncated, it is reread. If not, a blocking buffer is obtained and a reverse read (RDBLKR) performed if necessary.

ARDL-ARD2


ARD2
Have buffer
N
Y
CHKTRN
Pos tape for read if necessary
now positioned
OK
end block
Y
N
ARD22
COMKEY
compare Keys
go fore
back or found
RDBLK
OK
error
ARD28
DCB OPEN
Y
N
KEYER3
SR3 = 43
MSR00EXIT
COMKEYA
Compare users key with key at current buffer position
Less
Gr
EQ
ARD24
GTPL
Get preceeding entry
error
ARD27
COMKEYA
Compare users key with key at current buffer position
less
EQ
Gr
COMKEYA
compare users key with key at current buffer position
less
EQ
Gr
ARD25
GTFL
Get next entry
OK
error
ARD23
TRNREC
Transfer record to users buffer
MSR EXIT
GTFL
Get next entry
error
KEYTRN
Move key to KBUF

ARDL-ARD3


ARD3
Have buffer
N
Y
RESEVC
Clr EVC, EIC
CHKTRN
Position as needed for tape read
EVC on
N
Y
ARD31
CHKCON1
Check out block
OK
NG
CHKCON3
Set EIC
BARB1
GTPL
Find start of preceeding record
error
MSREXIT
IA1
TRNREC
Move entire logical record to user buffer
RWS = 0
N
Y
KEYTRN
Move key to key buffer
set TYC = 2 or 0 according data or no data in record
SETTRN
Set Position before record
MSR EXIT
SETTRN
Set Position before record
GTPL
Refind start of record
MSR EXIT

GTPL is entered to locate the record prior to the present position in the block, and TRNREC is employed to transfer the record to the user's buffer. Finally, GTPL is again entered to reposition DCB:CMD which was advanced by TRNREC.

RDBLKR

This subroutine skips the tape backward, if necessary, to position it correctly if the last block was truncated or to skip unblocked records, and reads the preceding record into the blocking buffer. If necessary, it obtains the preceding volume by changing CIS and calling CVOLA, positioning the tape to precede the tape mark at end of data, and reading the preceding record. If the read is successful, the routine exits skipping.

GTPL

This subroutine scans the blocking buffer forward to find the record preceding the current one. BCDA is used to count the number of records in the block that must be skipped. If a new block is required, RDBLKR is called. Exit is skipping if the record is found.

CHKEOF

Purpose: To process tape marks on labeled and ANS tapes.

Entry: OVERLAY LBLTSEG, 6

Exit: Normal OBSR4
Abnormal MSRO1EXIT

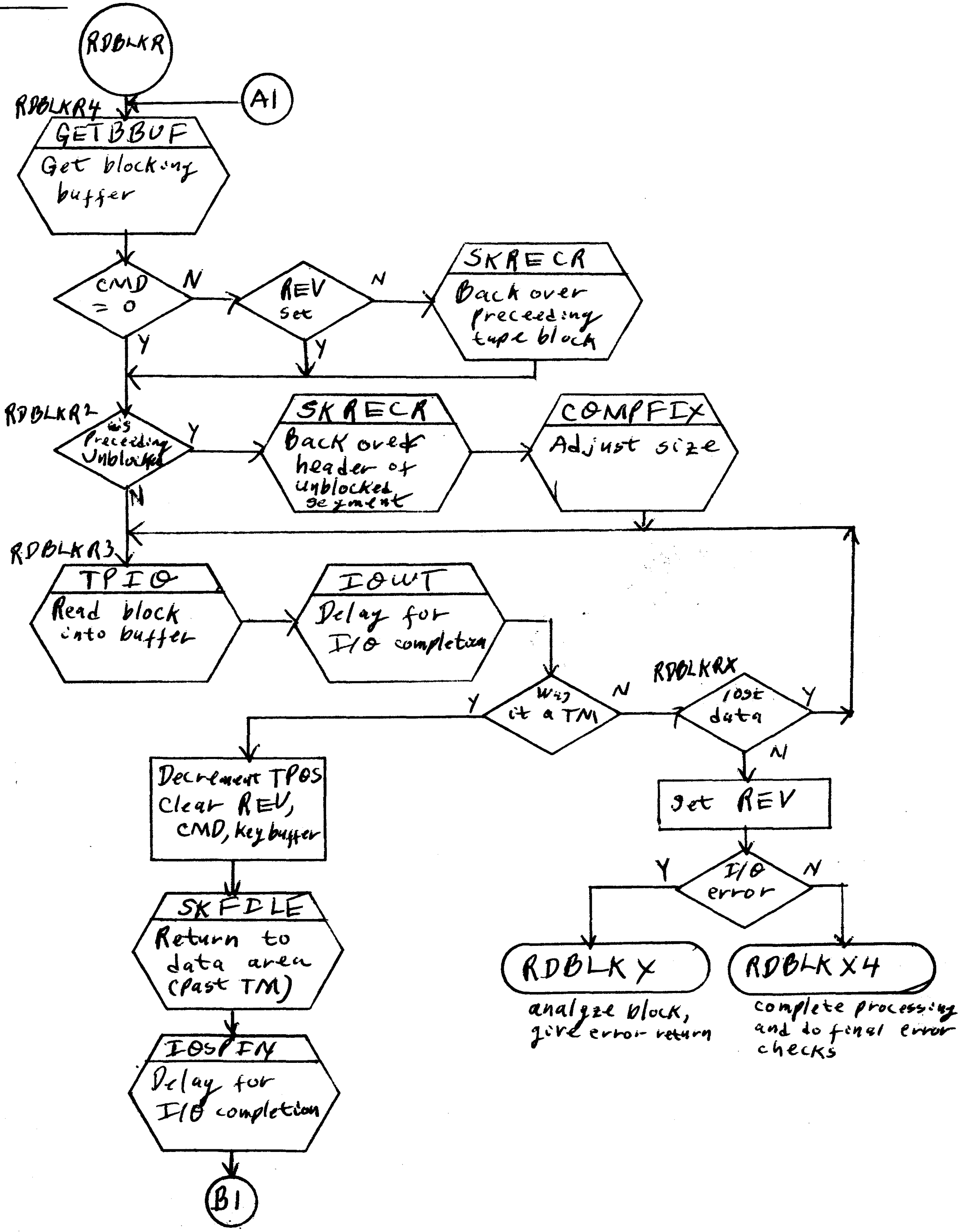
CHKEOF

CHKEOF is called when a tape mark is encountered while reading forward on labeled tape. If end-of-file is present and CONCAT <2, it returns an abnormal EOF. If end-of-volume is present and the user has specified the ULBL option, it returns an abnormal end tape and passes the user label, if present. Otherwise, it performs a CVOL and exits skipping. CHKEOF is called from POSTANS by:

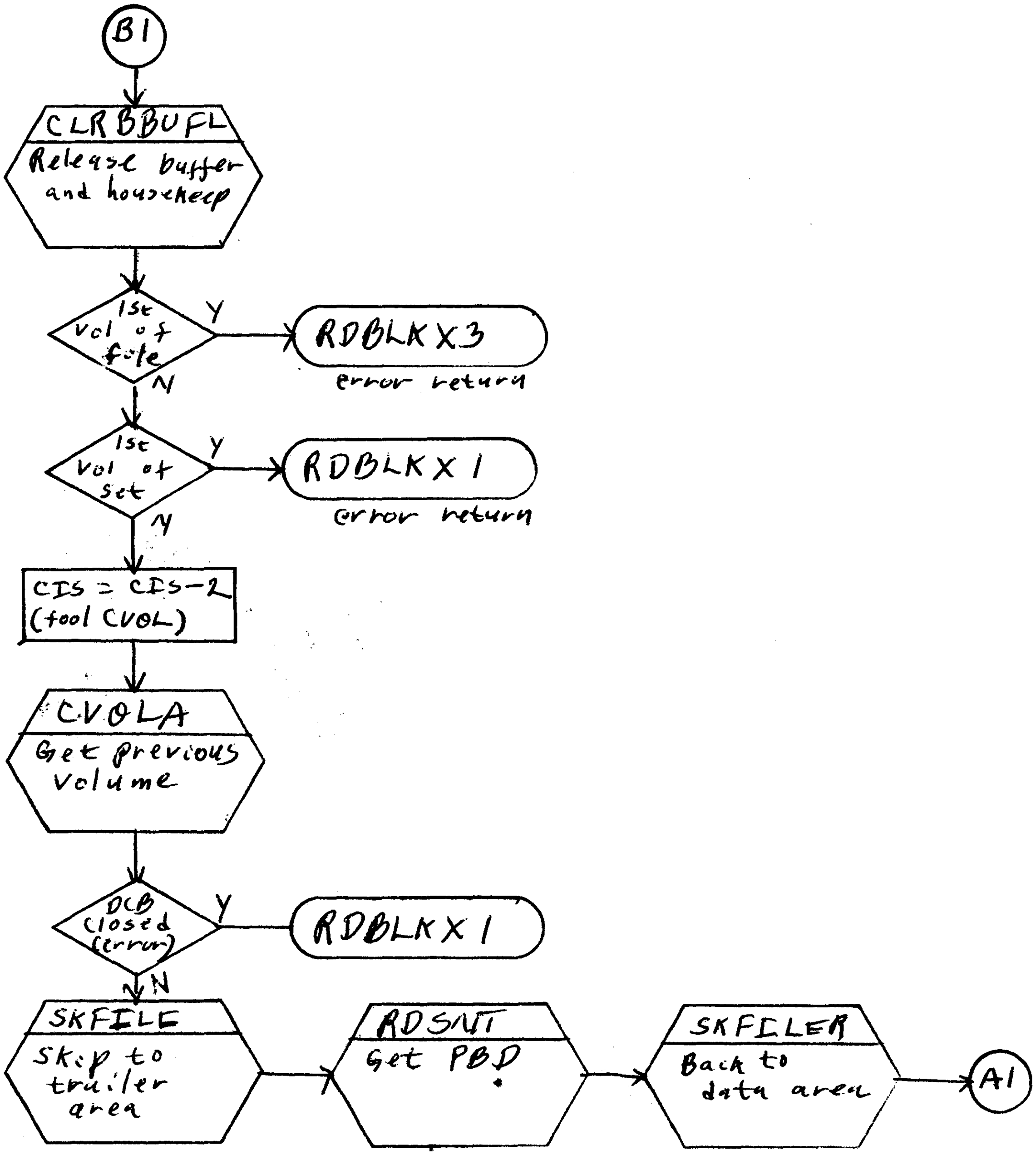
OVERLAY LBLTSEG, 6

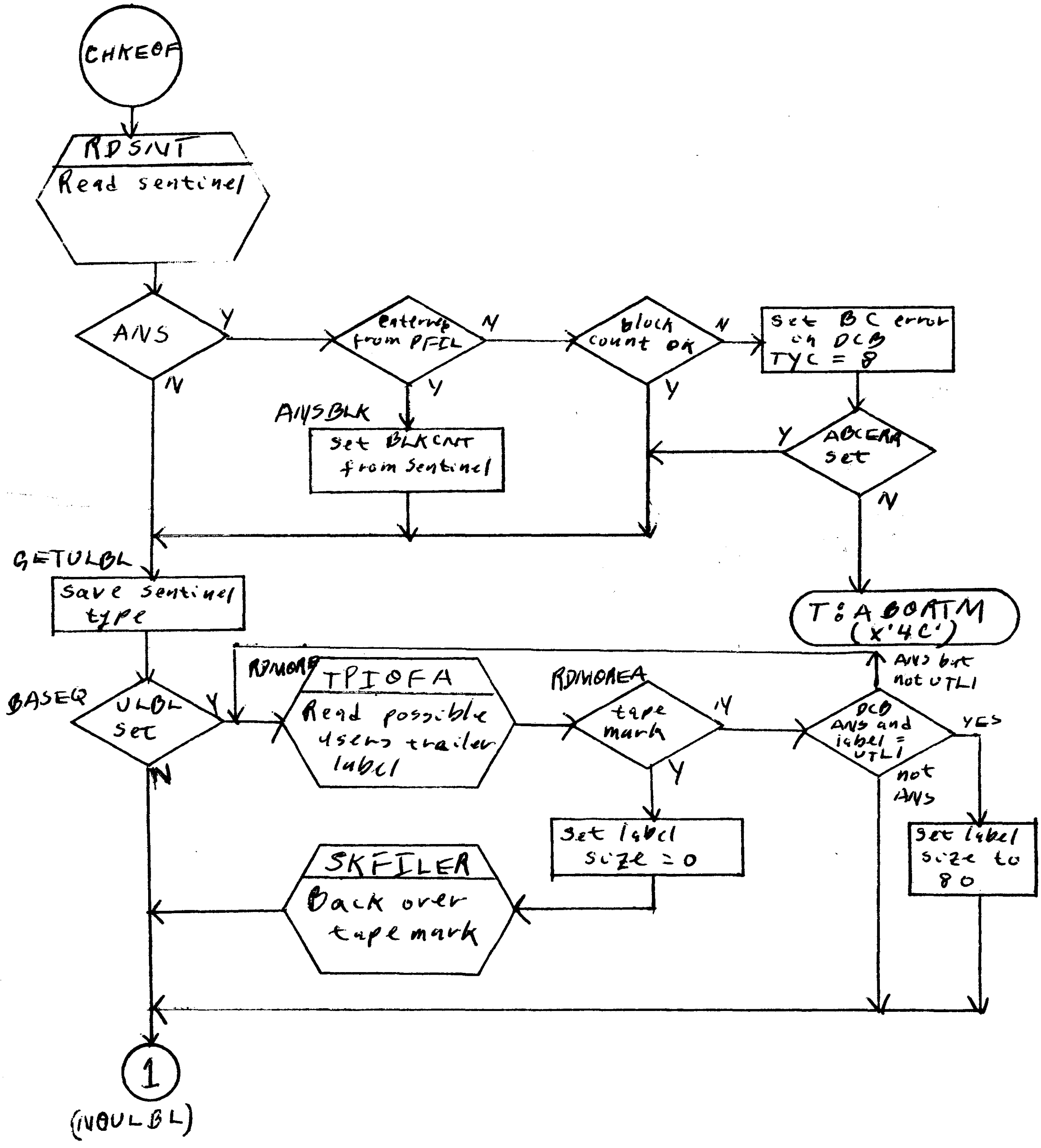
and only R6 is guaranteed to be unchanged.

ARDL-RDBLKR

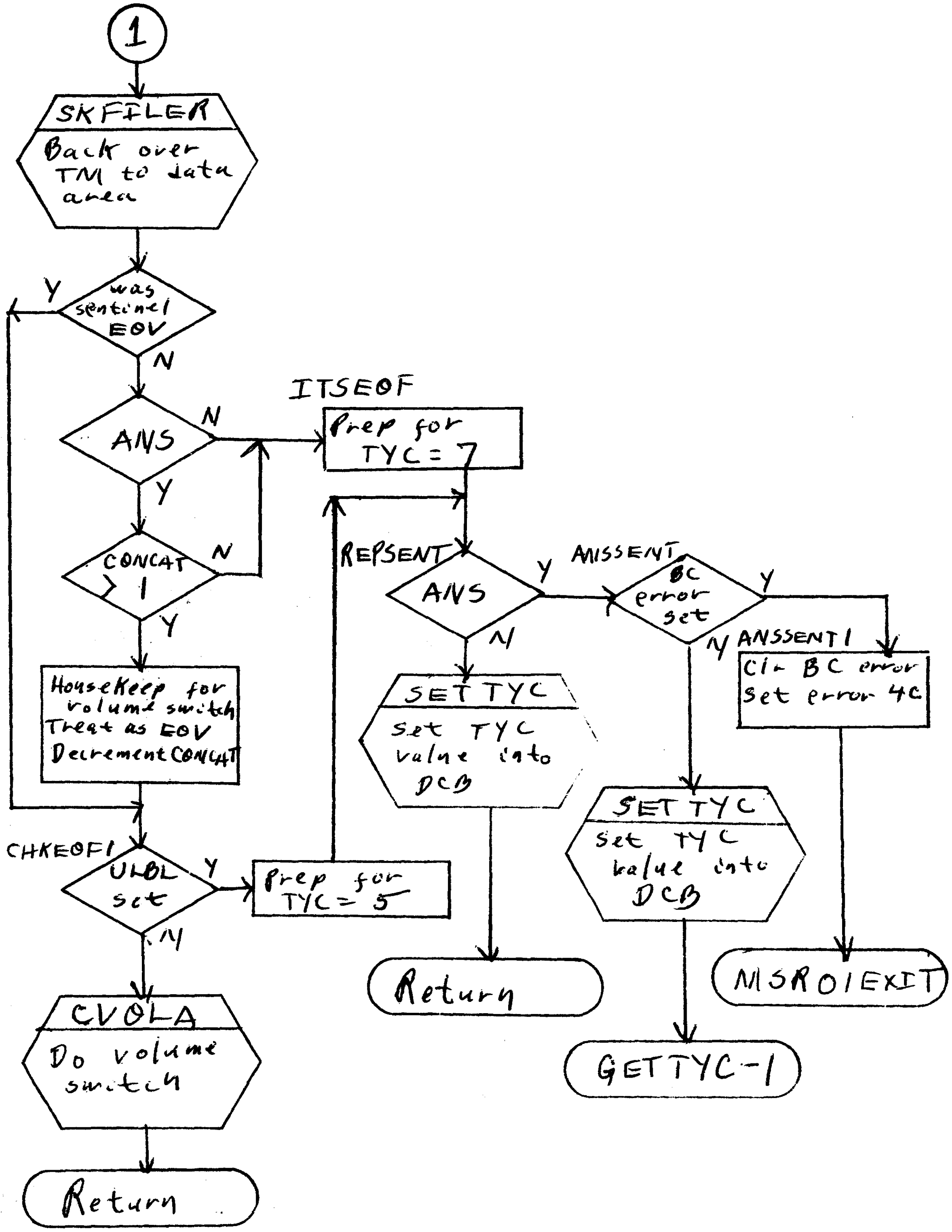

RDBLKR
A1
RDBLKR4
GETBBUF
Get blocking buffer
CMD = 0
N
REV set
N
SKRECR
Back over preceeding tape block
Y
Y
RDBLKR2
was preceeding unblocked
Y
SKRECR
Back over header of unblocked segment
COMPFIX
Adjust size
N
RDBLKR3
TPIO
Read block into buffer
IOWT
Delay for I/O completion
RDBLKRX
Y
Was it a TM
N
lost data
Y
N
Decrement TPOS
Clear REV,
CMD, key buffer
set REV
Y
I/O error
N
SKFILE
Return to data area (past TM)
RDBLKX
analyze block, give error return
RDBLKX4
complete processing and do final error checks
IOSPEN
Delay for I/O completion
B1

ARDL-RDBLKR


B1
CLRBBUFL
Release buffer and housekeep
1st Vol of file
Y
N
RDBLKX3
error return
1st Vol of set
Y
N
RDBLKX1
error return
CIS = CIS-2
(fool CVOL)
CVOLA
Get previous Volume
DCB closed (error)
Y
N
RDBLKX1
SKFILE
Skip to trailer area
RDSNT
Get PBD
SKFILER
Back to data area
A1


CHKEOF
RDSNT
Read sentinel
ANS
Y
N
entered from PFIL
N
Y
ANSBLK
set BLKCNT from sentinel
block count ok
N
Y
set BC error on DCB
TYC = 8
ABCERR set
Y
N
T:ABORTM (X'4C')
ANS bit not UTL1
GETULBL
Save sentinel type
BASEQ
ULBL set
Y
N
RDMORE
TPIOFA
Read possible users trailer label
RDMOREA
tape mark
N
Y
DCB ANS and label = UTL1
YES
not ANS
set label size = 0
SKFILER
Back over tape mark
set label size to 80
1
(NOULBL)

ARDL-CHKEOF


1
SKFILER
Back over TM to data area
was sentinel EOV
Y
N
ANS
N
Y
CONCAT > 1
N
Y
Housekeep for volume switch
Treat as EOV
Decrement CONCAT
CHKEOF1
ULBL set
Y
N
Prep for TYC = 5
CVOLA
Do volume switch
Return
ITSEOF
Prep for TYC = 7
REPSENT
ANS
Y
N
SET TYC
Set TYC value into DCB
Return
ANSSENT
BC error set
Y
N
SET TYC
Set TYC value into DCB
GETTYC-1
ANSSENT1
Clr BC error
Set error 4C
MSR01EXIT

In the initial region of CHKEOF, the sentinel is read. If the DCB is ANS, DCB:BLKCNT is then set from the sentinel if BUF is zero (zero indicates entry from PFIL), or checked against the sentinel if BUF is nonzero. If there is a block count error, the abnormal code is set, and CHKEOF exits to MSRO1EXIT.

In the user trailer label routine, if the DCB is ANS and the label read is not a tape mark and does not start 'UTL1', the next record is read, i.e., the label read is presumed to either EOV3 or EOF3, which are optional system labels not used by UTS.

In the NOULBL rgion, if the label is EOF, the DCB is ANS, and DCB:CONCAT is greater than one, concatenation must be performed. In such case, CVI and SETID are cleared, DCB:CONCAT is decremented, and the CHKEOF1 routine is entered to treat the EOF like an EOV.

The RDSNT subroutine is used by CHKEOF to read the next sentinel record. It has a parallel path added for ANS tape. A Monitor buffer is obtained and the EOF1/EOV1 is read into the buffer. A ':EOF' or ':EOV' is set into TSTACK to simulate the labeled tape sentinel. (For labeled tape, eight bytes are read into TSTACK.) This must be simulated as CHKEOF will switch based upon the sentinel read. The block count field is fetched from the EOV1/EOF1 label, converted to binary and saved in the second TSTACK word, similar to PBS for labeled tape. The next record is then read into the buffer and, if it is not 'EOF2' nor 'EOV2', the tape is backed over it as, presumably, only the EOF1/EOV1 is present. The Monitor buffer is then released.

ID

AVR	Automatic volume recognition
SRCHAVR	Search AVR table for serial number match
SETNEW	Set new AVR table entry and wakeup user
GETUSER#	Get user number from system i.d.

AVR
PURPOSE

To extract revelant tape possessing information from tape labels and post in AVR tables.

USAGE

AVR is entered from the interrupt handler as the result of a RESET, ATTENTION START tape drive sequence or as part of the processing of an IN mode open request if the tape had not been previously processed by AVR.

CALL

```
 BAL, SR4  AVR
 R7=DCTX
```

DATA BASE

The AVR tables are described in Section VC. 04.

DESCRIPTION

Initially, it is determined if the AVR request is solicited. If not, the drive must not be in use. A monitor buffer is obtained via GMB and the first four bytes of the tape are read. If the initial record of the tape begins VOL1, the tape assumed to be ANS labeled and ANS processing is required (see ANSTP, JB. 30. 01). Otherwise the following sequence is queued via NEWQ: rewind, skip file forward, backspace file, backspace two records and read with end action. This record must begin :LBL. The monitor buffer is released via RMB, the AVR bit (Y2) is set and SRCHAVR is entered.

ERRORS

The message AVR ERROR is typed on the OC if the AVR request is unsolicited and drive is busy or the label read after the AVR sequence is not :LBL. In each case the AVR table is cleared.

SRCHAVR

PURPOSE

To determine if a given serial number is in the AVR table.

USAGE

BALRO SRCHAVR
R6 = i.d.
D3 = serial number
D4 = flags
R5 = DCTX

DESCRIPTION

The input serial number is compared with each AVR table entry until a match is found or the table is exhausted. If a match is found, the user associated with the drive is awakened if possible. If no match is found, SETNEW is entered immediately. The tape AVRed may satisfy a request for a different drive. In this case AVRTBL and AVRID are moved to reflect the unit switch and solicit is reset. SETNEW is entered.

SETNEW
PURPOSE

To set a new AVR table entry and wake up user.

USAGE

```
CALL:
BAL, RO      SETNEW
R1 = AVRX
R6 = USER#
D3 = serial.#
D4 = flags
```

DESCRIPTION

The solicit entry is cleared and D3, D4 are stored in AVRTBL. If the user id is non zero and the user is asleep, a wakeup event is reported.

GETUSER#
PURPOSE

To return a user number given a system id.

USAGE

```
CALL:
BAL, 7       GETUSER#
R6 = system id number
RETURN
B     0,7    not found
B     1,7    found
```

DESCRIPTION

The system id is compared with table PLH:SID. If a match is found the PLB:USR entry parallel to PLH:SID is returned.

ID

IORT - contains numerous subroutines in support of all DCB related service CALs.

The IORT module is described in the file management section of this manual.

ID

LBLT - The LBLT module contains routines that perform output operations for labeled and ANS tapes. This includes volume switching, sentinel writing, protecting checking and writing of data records (labeled tape only). LBLT also closes labeled, ANS, and unlabeled tapes.

CLSLBL, CLSTP

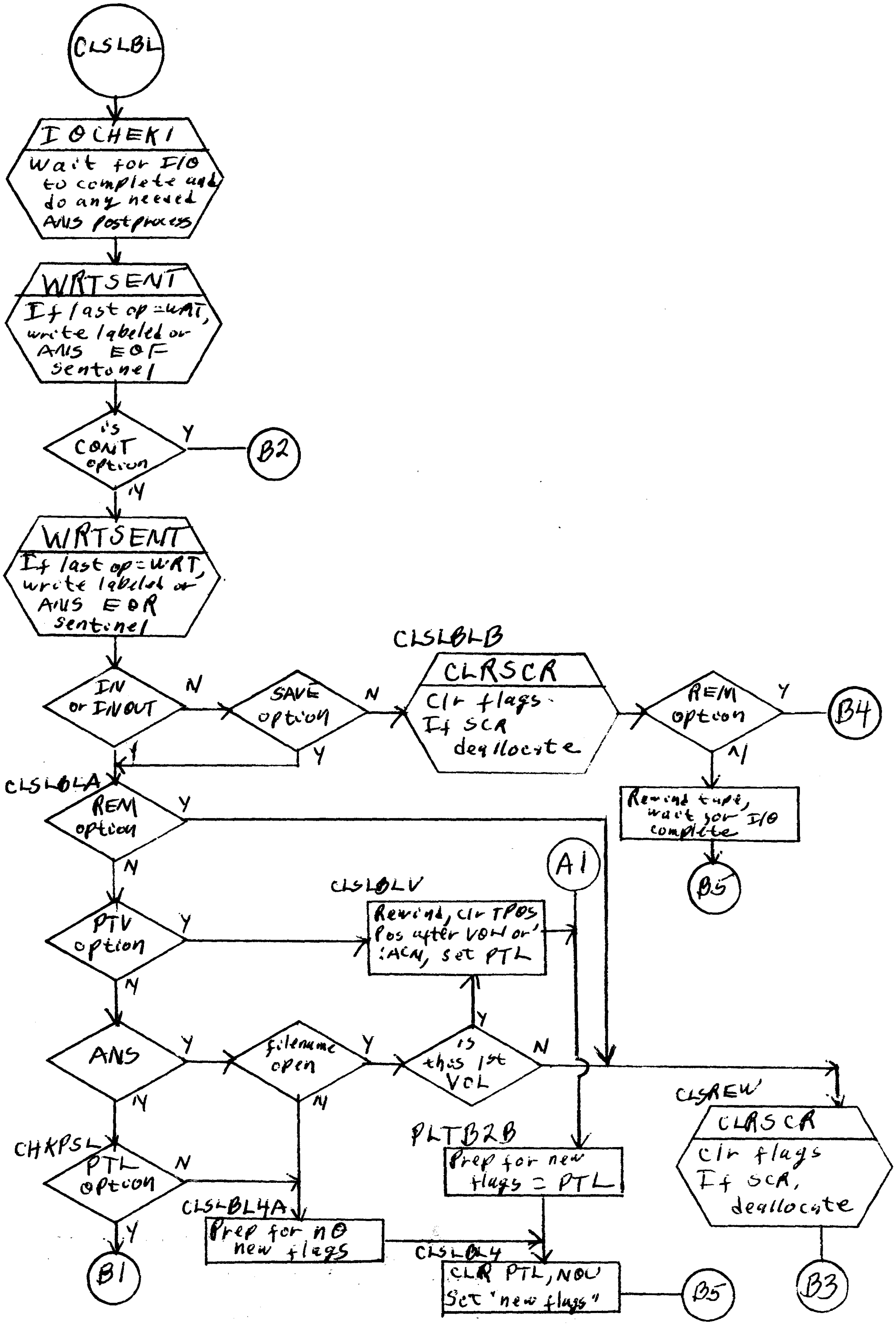
Purpose:	To perform M:CLOSE procedures for tapes.		
Entry:	OVERTO, LBLTSEG,4	(CLSTP)	labeled tape
	OVERTO, LBLTSEG,3	(CLSLBL)	labeled and ANS tapes
Exit:	(R7) = word address of FPT+1		
	Normal - B PULLALLEXIT		
	Error - B MSR01EXIT		

Operation: If entered at CLSTP (unlabeled tape) and if the last operation was write and DRC indicates special formatting, two tape marks are written and the tape is positioned between them. A parameter is set up to treat both the REM and PTL options as REM. If scratch is indicated, the parameter is processed as OUT or OUTIN; otherwise, processing is as for IN or INOUT. Both processing modes are described below.

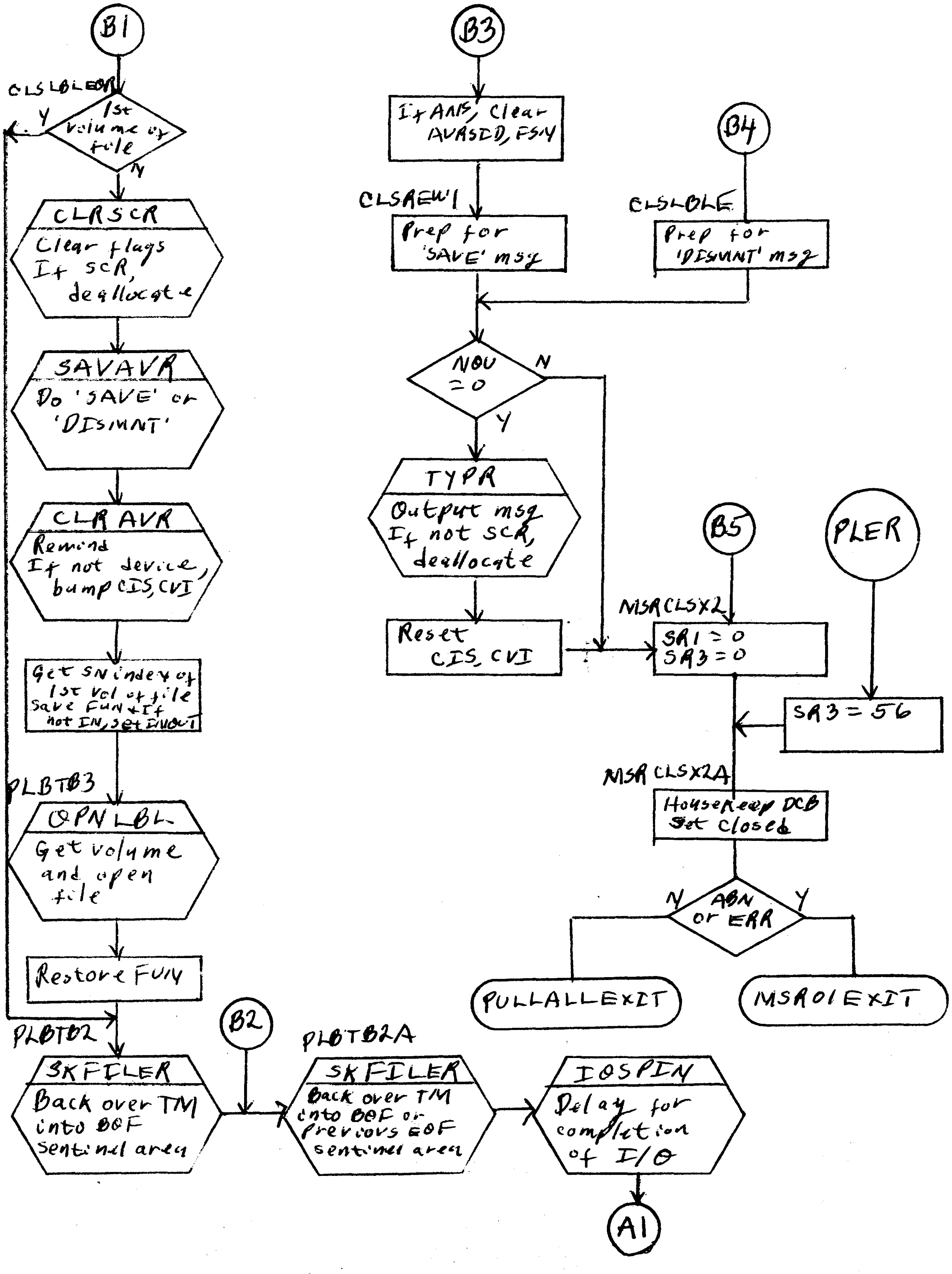
If entered at CLSLBL, WRTSENT is used to write :EOF, :EOR/EOF1, EOV1 sentinels if the last operation was a Write.

If the function is IN or INOUT, or if the tape is unlabeled non-scratch, and if REM is specified, the tape is rewound, the operator is instructed to dismount and save the tape, the DCB is closed and the routine exits. If REM was not specified but PTL was (for labeled tape only), the proper volume is mounted via operator interaction and OPNLBL if the current volume is not the first volume of the file, and is backed over two tape marks to position it to

LBLT-CLSLBL



CLSLBL
IOCHEKI
Wait for I/O to complete and do any needed ANS postprocess
WRTSENT
If last op = WRT, write labeled or ANS EOF sentinel
is CONT option
Y
B2
N
WRTSENT
If last op = WRT, write labeled or ANS EOR sentinel
CLSLBLB
IN or INOUT
N
SAVE option
N
CLRSCR
Clr flags. If SCR deallocate
REM option
Y
B4
N
Y
Y
CLSLBLA
REM option
Y
Rewind tape, wait for I/O complete
B5
N
A1
CLSLBLV
PTV option
Y
Rewind, clr TPOS Pos after VOH or ACN, set PTL
N
Y
ANS
Y
filename open
Y
is this 1st VOL
N
N
N
CLSREW
CLRSCR
Clr flags If SCR, deallocate
CHKPSL
PTL option
N
PLTB2B
Prep for new flags = PTL
CLSLBL4A
Prep for no new flags
CLSLBL4
CLR PTL, NOU Set "new flags"
Y
B1
B5
B3


B1
CLSLBLEOR
1st volume of file
Y
N
CLRSCR
Clear flags
If SCR, deallocate
SAVAVR
Do 'SAVE' or 'DISMNT'
CLRAVR
Remind
If not device, bump CIS, CVI
Get SN index of 1st vol of file
Save FUN + If not IN, set INOUT
PLBTB3
OPNLBL
Get volume and open file
Restore FUN
PLBTB2
SKFILER
Back over TM into BOF sentinel area
B2
PLBTB2A
SKFILER
Back over TM into BOF or previous EOF sentinel area
IOSPIN
Delay for completion of I/O
A1
B3
If ANS, Clear AVRSID, FSN
CLSAEW1
Prep for 'SAVE' msg
B4
CLSLBLE
Prep for 'DISMNT' msg
NOU = 0
N
Y
TYPR
Output msg
If not SCR, deallocate
Reset CIS, CVI
B5
PLER
MSRCLSX2
SR1 = 0
SR3 = 0
SR3 = 56
MSRCLSX2A
Housekeep DCB
Set Closed
ABN or ERR
N
Y
PULLALLEXIT
MSR01EXIT

the front of the TM preceding the :BOF sentinel. Then, whether PTL is specified or not, the DCB is closed and the routine exits.

If PTV is present, CLSLBLV is entered. If PTV is not present and the DCB is ANS, the PTL check is bypassed. If, in addition, the DCB was opened by filename, the REM option is forced if CIS>1 (multi-volume file), or the PTV option is forced if CIS=1 (single volume file). These default options permit the reopening of files that could not otherwise be reopened.

The CLSLB LV routine rewinds the tape. Then, if the tape is ANS, the first record (VOL1) is skipped, AVR:POS cleared and AVR:AVR set. If the tape is labeled, a skip file, skip file reverse sequence is performed to position after the :ACN. In any event, the AVR:PTL flag is set, IOSPIN is called and the routine exits to MSRCLSX2.

At CLSREW, if the DCB is ANS, DCB:SETID and DCB:FSN are cleared.

If the function is OUT or OUTIN, or if the tape is an unlabeled scratch and SAVE is specified, the tape is treated as IN and INOUT, as described above. Otherwise, if REL and REM are specified, the operator is instructed to dismount the tape. In any event, the tape is rewound, the D CB is closed and the routine exits.

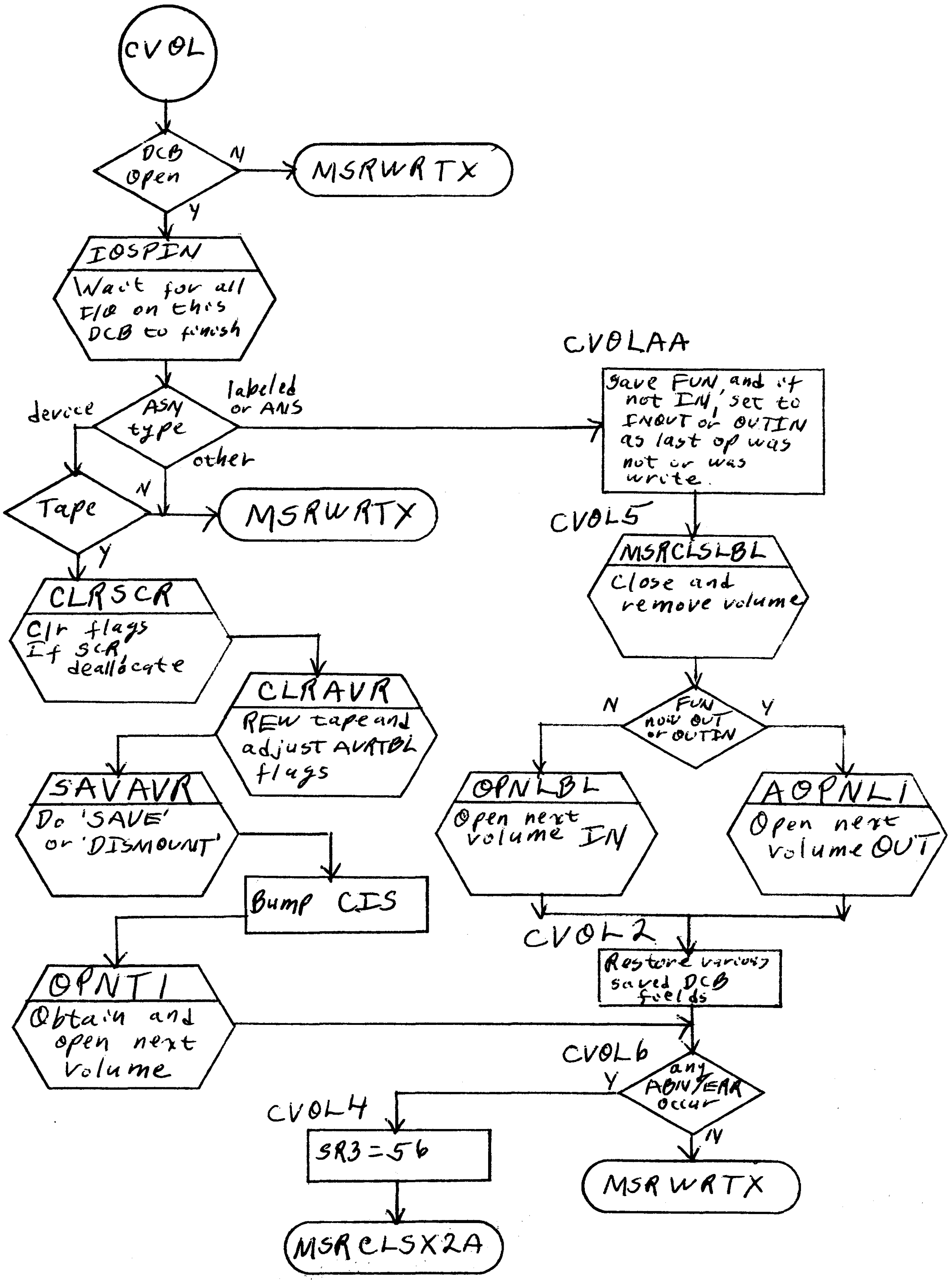
CVOL

Purpose: To perform the M:CVOL procedure.

Entry: OVERLAY LBLTSEG, 2

Exit: B MSRWRTX

LBLT-CVOL


CVOL
DCB Open
N
MSRWRTX
Y
IOSPIN
Wait for all I/O on this DCB to finish
CVOLAA
Save FUN, and if not IN, set to INOUT or OUTIN as last op was not or was write.
device
ASN type
labeled or ANS
other
Tape
N
MSRWRTX
Y
CVOL5
MSRCLSLBL
Close and remove volume
CLRSCR
Clr flags If SCR, deallocate
CLRAVR
REW tape and adjust AVRTBL flags
N
FUN now OUT or OUTIN
Y
SAVAVR
Do 'SAVE' or 'DISMOUNT'
OPNLBL
Open next volume IN
AOPNLI
Open next volume OUT
Bump CIS
CVOL2
Restore various saved DCB fields
OPNTI
Obtain and open next volume
CVOL6
Y
any ABN/EAR occur
N
CVOL4
SR3=56
MSRWRTX
MSRCLSX2A

Operation: See following flowchart. If the DCB is opened to unlabeled tape, the tape is rewound and dismounted with save instructions to the operator and the next volume is obtained via OPNT1.

If the DCB is opened to labeled or ANS tape, MSRCLSLBL is called to close the volume, then OPNLBL or OPNLO, depending on function, is called to open the next volume.

WRITEL

Purpose: To perform M:WRITE for labeled tape or drive M:WRITE to WRTD if ANS.

Entry: OVERTO LBLTSEG,1

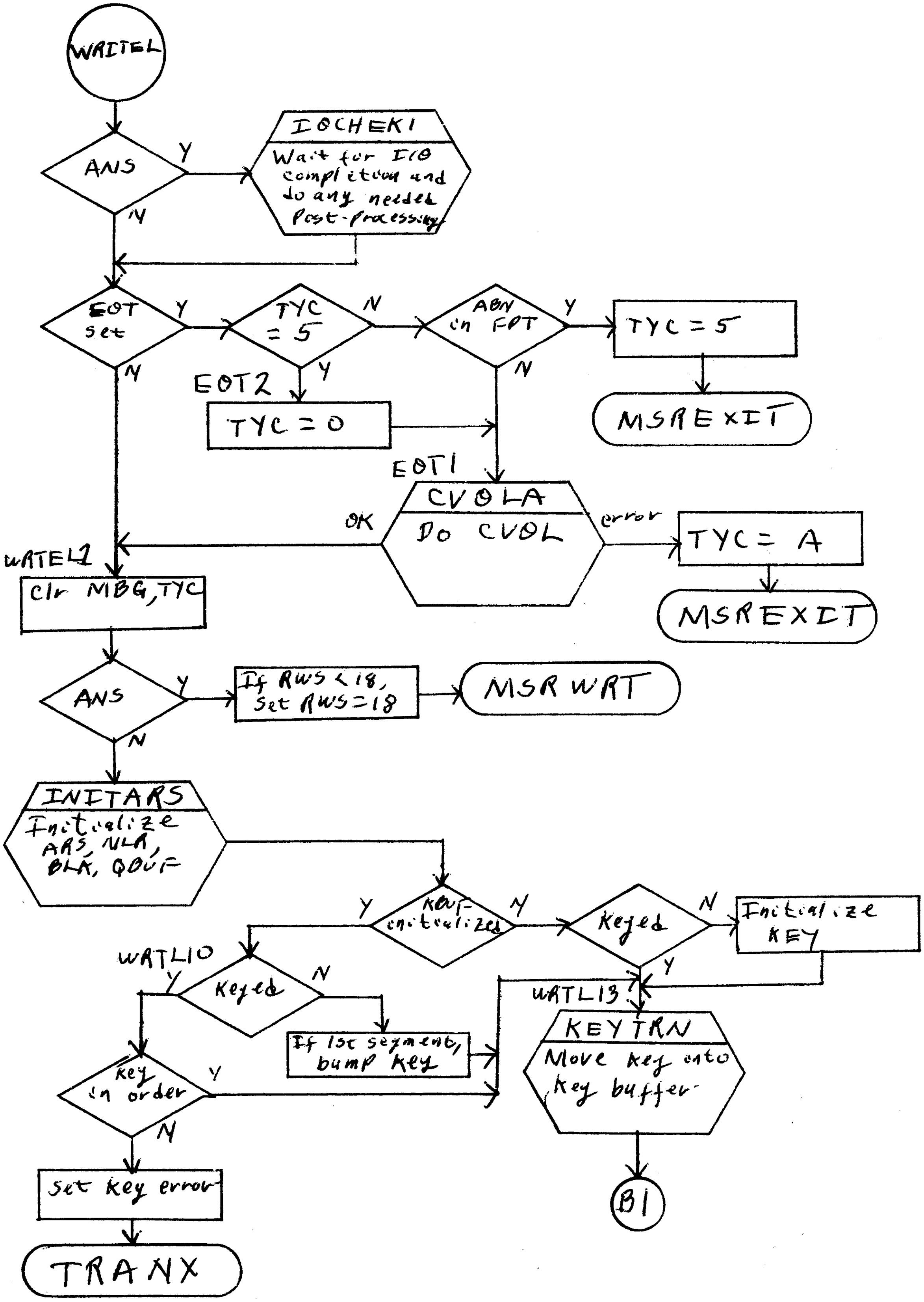
Exit: B MSRWRTX

Operation: See following flowchart. If the EOT flag is set, CVOLA is called to do a volume switch, and the exit is taken if CVOLA detects an error or abnormality.

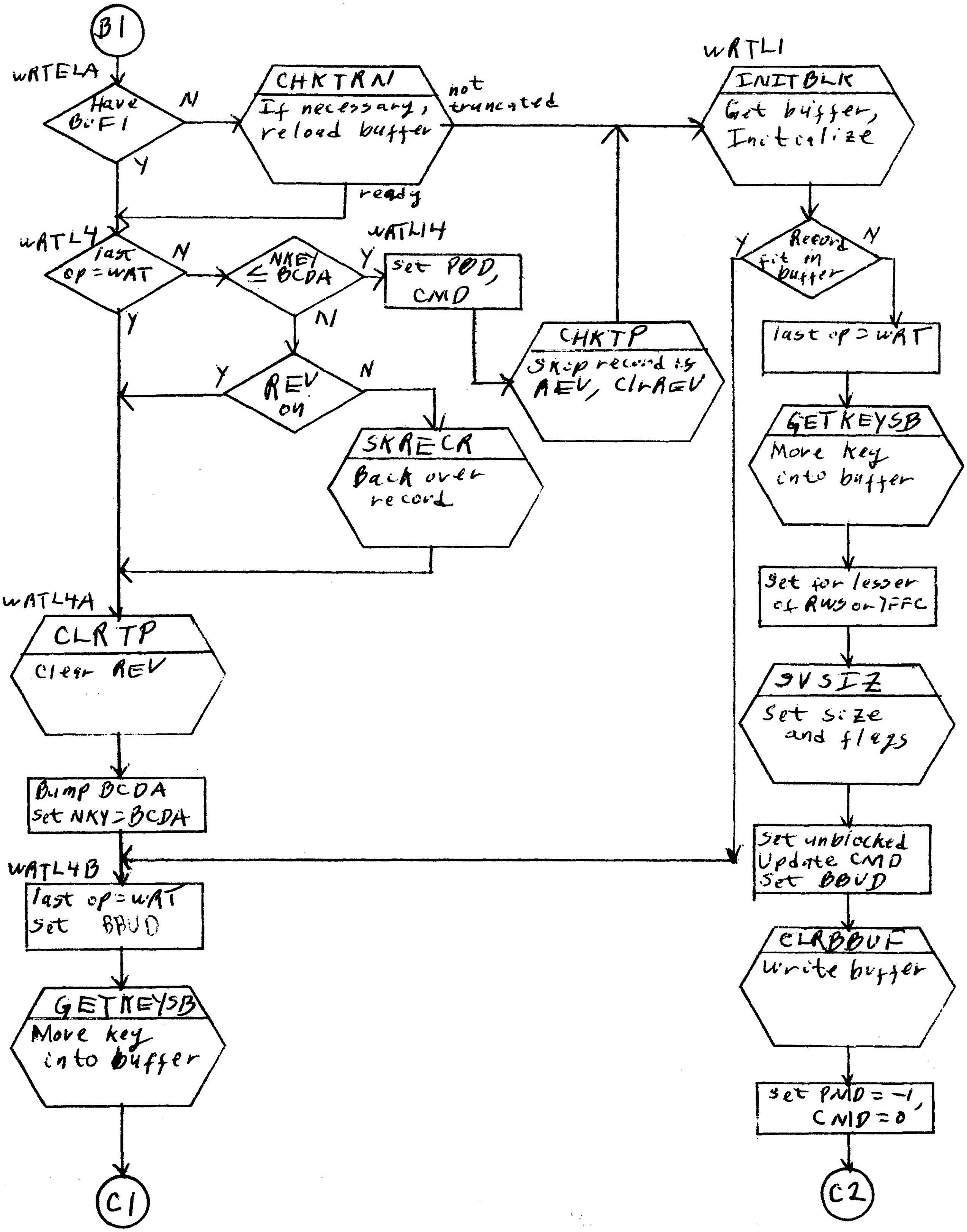
If the DCB is ANS, the block count must be >18. It is set to 18 if this condition is not met. Control is then passed on to WRTD to perform the data transfer for ANS DCBs.

If the DCB is for labeled tape, various record-related items are initialized and legality checks are made depending on key value, organization and last operation. If the organization is consecutive, the dummy key is incremented. If there is no blocking buffer, one is obtained. If a new block is to be started, the buffer is initialized. Following block initialization, if the balance of the record will not fit in the buffer, the header is written out and an unblocked record of the lesser, the remaining request of 8191 words, is written.

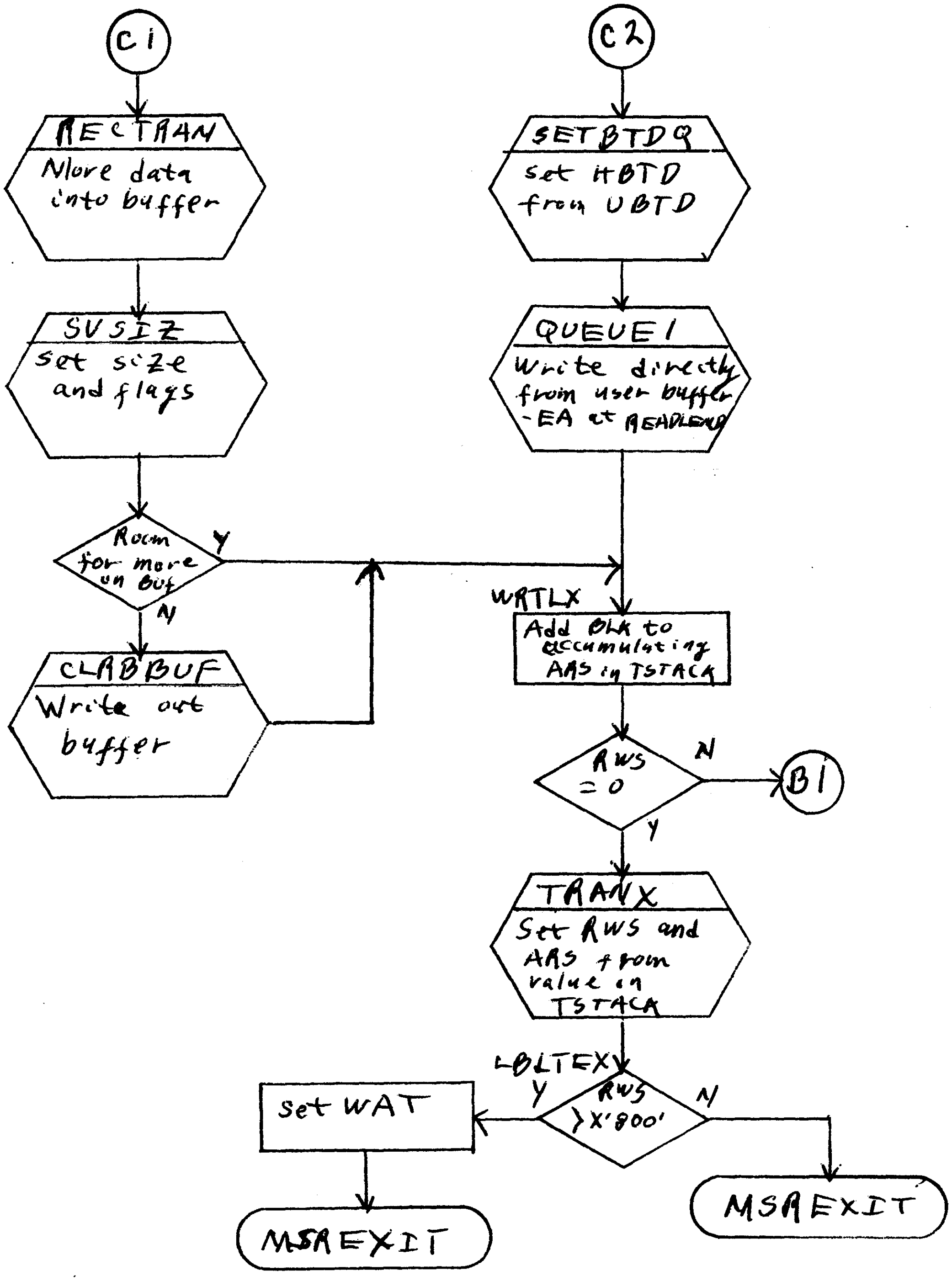
LBLT-WRITEL


WRITEL
ANS
Y
N
IOCHEKI
Wait for I/O completion and do any needed post-processing
EOT set
Y
N
TYC = 5
N
Y
ABN in FPT
Y
N
TYC = 5
MSREXIT
EOT2
TYC = 0
EOT1
CVOLA
Do CVOL
OK
error
TYC = A
MSREXIT
WRTEL1
Clr MBG,TYC
ANS
Y
N
If RWS < 18, Set RWS=18
MSR WRT
INITARS
Initialize ARS, NLR, BLK, QBUF
Y
KEY-initialized
N
Keyed
N
Y
Initialize KEY
WRTL10
Y
Keyed
N
If 1st segment, bump key
WRTL13
KEYTRN
Move key into key buffer
Key in order
Y
N
Set key error
B1
TRANX

LBLT-WRITEL


B1
WRTELA
Have BUF1
N
Y
CHKTRN
If necessary, reload buffer
not truncated
ready
WRTL1
INITBLK
Get buffer, Initialize
Record fit in buffer
Y
N
WRTL4
last op = WRT
N
Y
NKEY ≤ BCDA
Y
N
WRTL14
Set POD, CMD
CHKTP
Skip record if REV, ClrREV
last op = WRT
REV on
Y
N
SKRECR
Back over record
GETKEYSB
Move key into buffer
Set for lesser of RWS or 7FFC
WRTL4A
CLRTP
Clear REV
SVSIZ
Set size and flags
Bump BCDA
Set NKY = BCDA
Set unblocked
Update CMD
Set BBUD
WRTL4B
last op = WRT
Set BBUD
CLRBBUF
Write buffer
GETKEYSB
Move key into buffer
Set PMD = -1, CMD = 0
C1
C2

LBLT-WRITEL


C1
RECTRAN
More data into buffer
SVSIZ
set size and flags
Room for more in Buf
Y
N
CLRBBUF
Write out buffer
C2
SETBTDQ
set HBTD from UBTD
QUEUE1
Write directly from user buffer -EA at READLEND
WRTLX
Add BLK to accumulating ARS in TSTACK
RWS = 0
N
B1
Y
TRANX
Set RWS and ARS from value in TSTACK
LBLTEX
Y
RWS >X'800'
N
set WAT
MSREXIT
MSREXIT

If not the beginning of a block and the last operation was not a Write, the tape is repositioned, if necessary, by skipping forward or backward one block. The key and data, or all the data that will fit, are then moved into the blocking buffer. If the buffer does not have room for another record header and at least four data bytes, the buffer is written out. Whether blocked or not, the routine exits if all requested bytes have been accounted for. Otherwise, the process is repeated starting with the check for the presence of a blocking buffer.

WRTSENT

For labeled tapes, if the last operation (EOP) was a Write, the sentinel whose address is in R9 is written to tape along with previous block size. If the sentinel is not EOR, it is preceded and followed by tape marks. If the sentinel is EOR, two tape marks are written after the record. Then the tape is backed over four tape marks.

For ANS tapes, 'EOF1' and 'EOF2' are written in lieu of ':EOF', 'EOV1' and 'EOV2' in lieu of ':EOV'. Then, a single tape mark and three file backspaces are issued in lieu of the :EOR sequence. The size and identifier 'UTL' for ANS user trailer labels are verified.

VOLHEDANS

Purpose: To perform auxiliary OPEN functions for ANS tapes.

Entry: OVERLAY LBLTSEG, 11

Exit: B *SR4

Operation: This routine logically is part of the OPEN input function but resides in LBLT in order to conserve space in the OPEN segment.

For ANS tapes, DCB:CVI is verified against the volume sequence number in HDR1. DCB:FSN is set from HDR1. The byte count of the user's header label buffer is set to zero. The next record is read. If it is a tape mark, skip file reverse and exit. If it is HDR2, move the balance of file attributes into the DCB and continue reading. If it is not HDR2, it is a user's header label and is transferred (if requested) to the user's buffer.

PROTCHK

Purpose: To protect unexpired ANS tapes against accidental destruction.

Entry: OVERTO LBLTSEG, 9

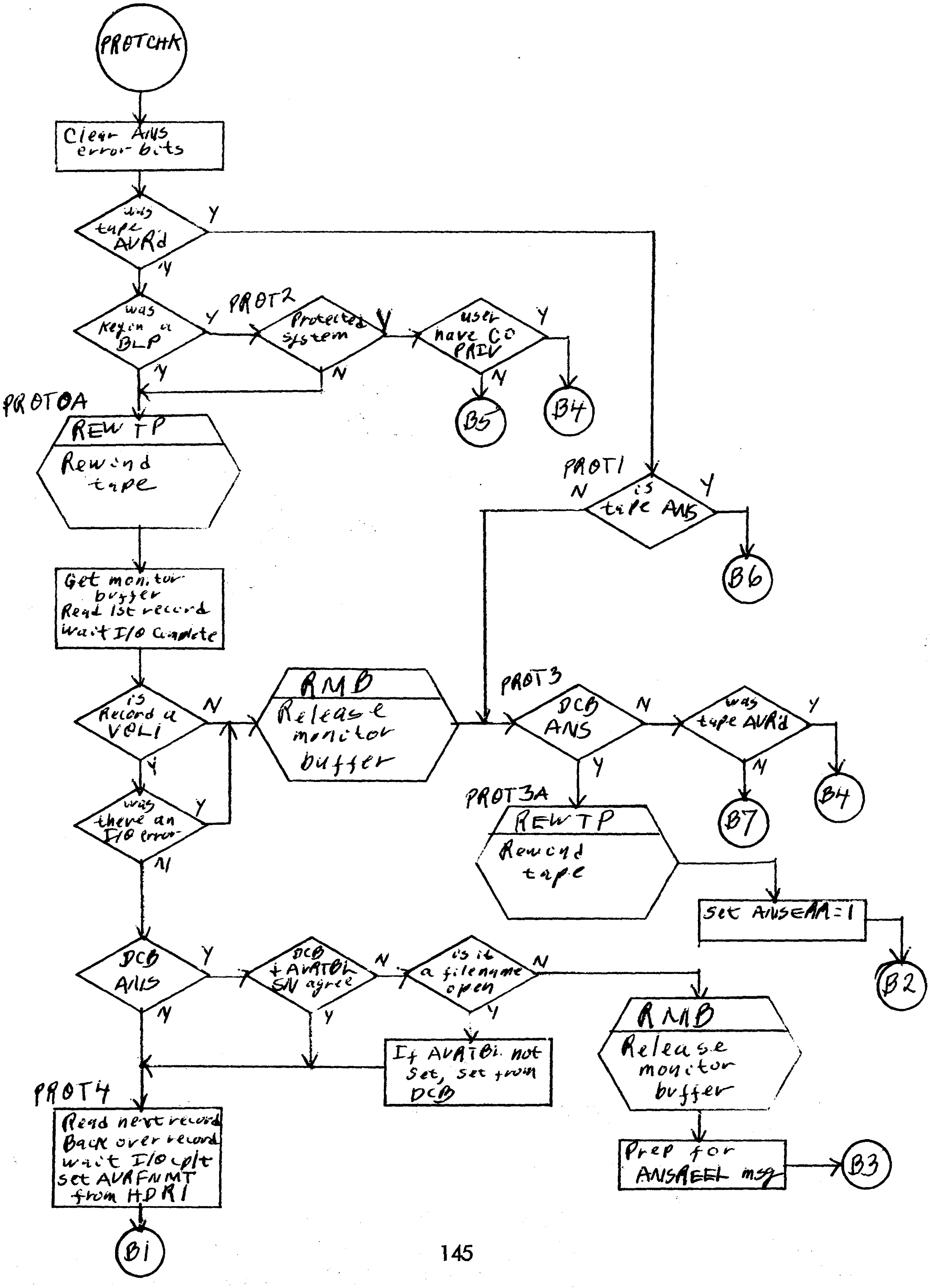
Exit: B *SR4

Operation: PROTCHK is a routine that performs the write protection checks for ANS tapes. It is logically a part of the USECHK routine of the OPNL module, but is included in the LBLT module and the LTAPE segment to minimize the chance of overflowing the OPEN segment. PROTCHK is called only when a volume is being assigned. If a tape is rejected by PROTCHK, SR1 is set to -1. Otherwise, SR1 is set to 0. PROTCHK is called by:

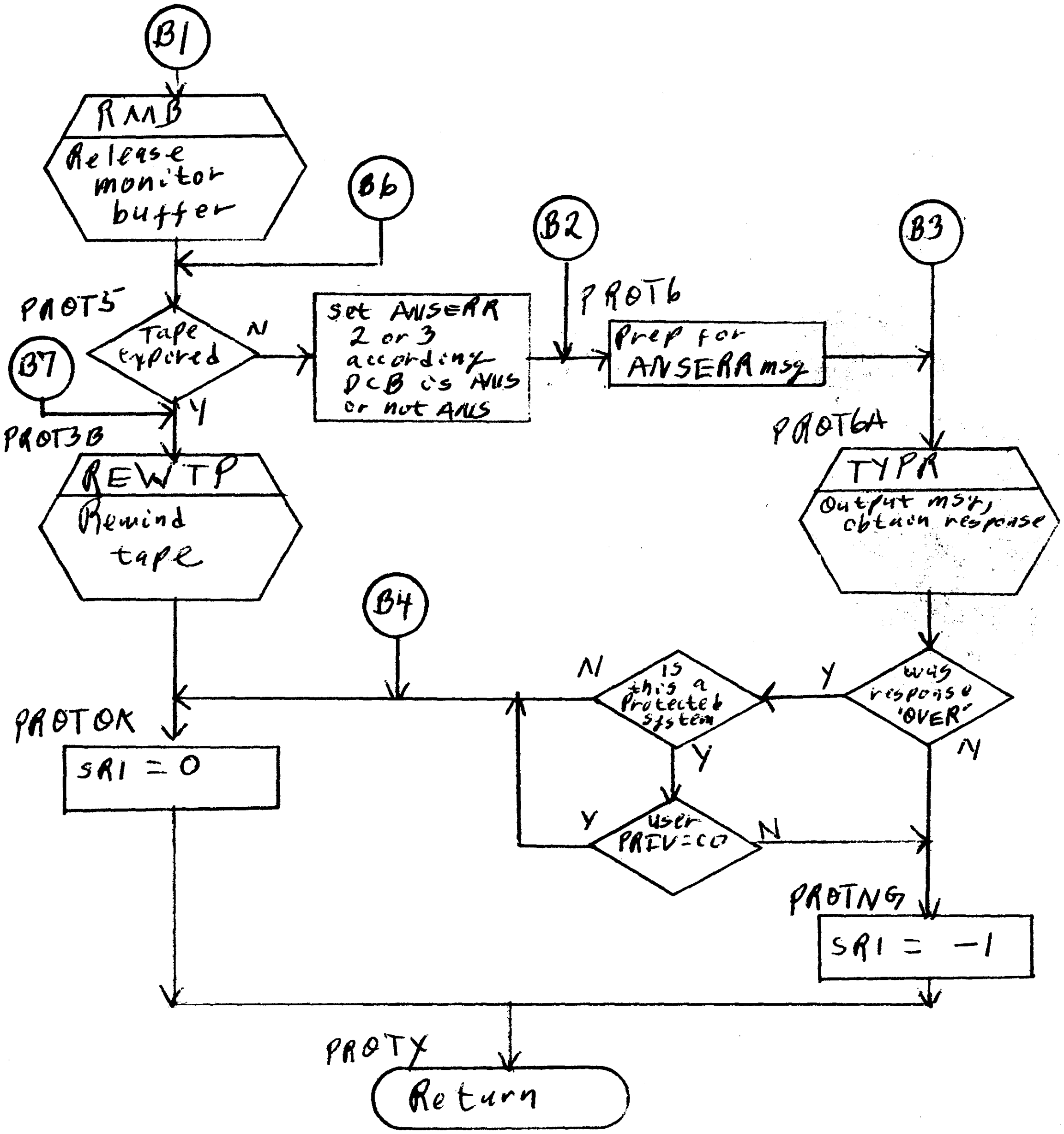
OVERTO LBLTSEG, 9

If the tape has been mounted with BLP option, further processing is bypassed and the tape is rejected unless the user's privilege is at least X'C0'. If the tape was not AVRed, the tape is rewound, the first two records are read and positioned after the first record to determine if the sentinel is a VOL1(ANS) and to set the AVRFNMT entry. In either event, if the tape is not ANS and the DCB is ANS, TYPR is called to output the 'NOT ANS' message and await the operator response. If neither tape nor DCB is ANS, the tape is accepted and is rewound if it was not AVRed.

LBLT-PROTCHK



LBLT-PROTCHK



If the serial numbers do not match, TYPR is called to output 'ANS REEL NO. ERROR' message. If the tape is an expired ANS, it is accepted. If the tape is unexpired ANS, TYPR is called to output the 'NOT EXPIRED' message (if the DCB is ANS) or the 'ANS VOL' message (if not an ANS DCB), and to await the operator's response. If the operator responds 'OVER' and either the installation is not in the "protected" mode or the user's privilege is at lease X'CO', the tape is accepted. Otherwise, the tape is rejected.

WRTANSLBL Write HDR1-HDR2, EOF1-EOF2, and EOV1-EOV2 ANS tape labels.
Input Registers: R7 = first label i.d. i.e. HDR1, EOF1, EOV1
= byte address of first label in TEXT format.
Calling Sequence: BAL, SR4 WRTANSLBL
Return: B *SR4

CVNJULIAN Convert Gregorian date to Julian date.
Input Registers: R5= address of DATE or DCB expiration date.
Calling Sequence: BAL, SR4 CVNJULIAN
Return: B *SR4
Output Registers: SR1-SR2 = EBCDIC date (YYDDD) right justified.

CNVBIN Convert EBCDIC to binary
Input Registers: R0= EBCDIC number
Calling Sequence: BAL, SR4 CNVBIN
Return: B *SR4
Output Registers: R1= binary number.

CNVDEC Convert binary to EBCDIC.

Input Registers: R1= binary number, R2 = number of opsitions to convert.

Calling Sequence: BAL, SR4 CNVDEC

Return: B *SR4

Output Registers: SR1-SR2= EBCDIC number left-justified.

ID

OPLO - processes the open CAL and the open phase of volume switching. for both labeled and ANS tapes.

OPNLO, AOPNL1

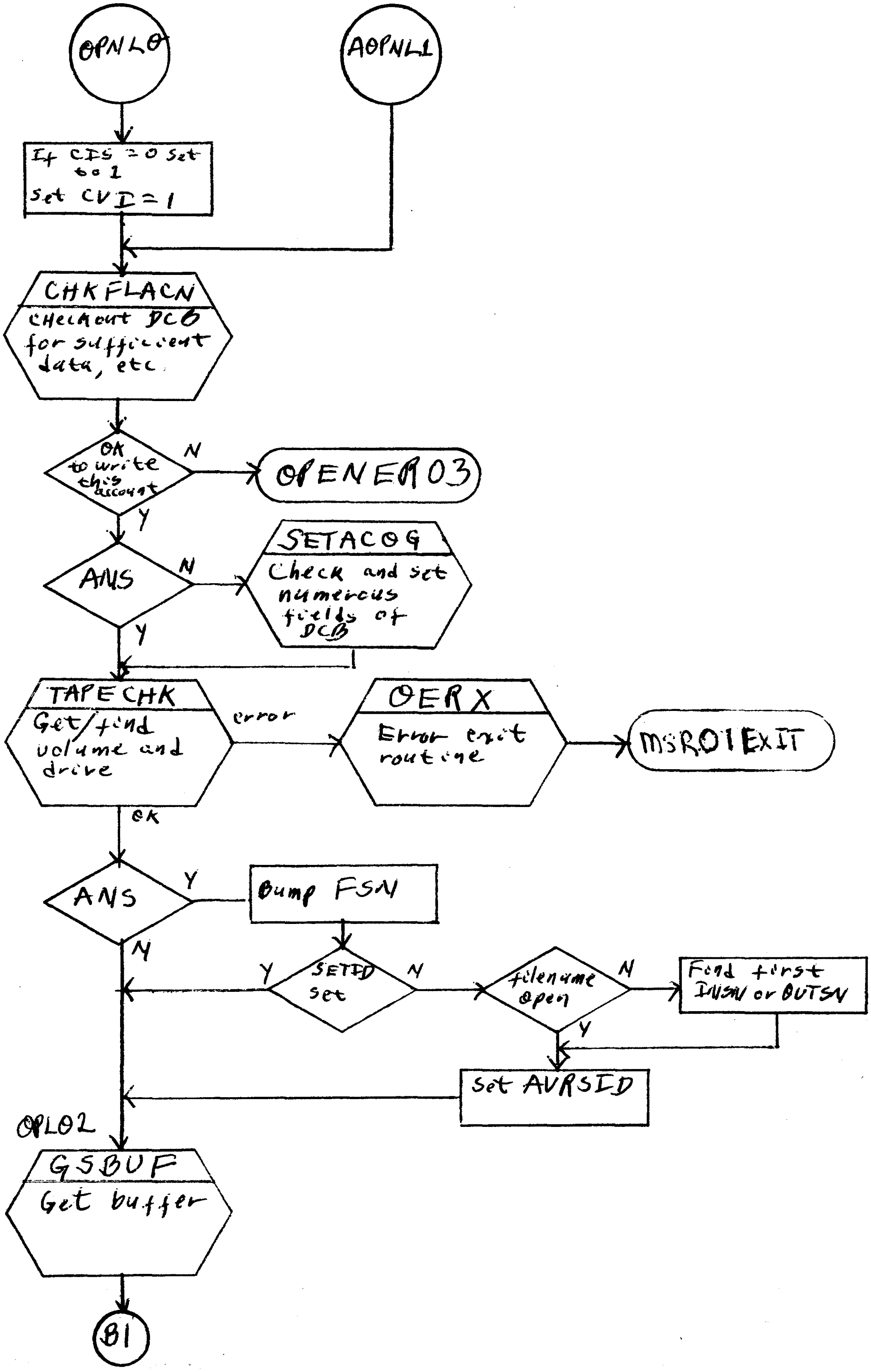
Purpose: To perform the open for output ANS and labeled tapes.

Entry:	B	OPNLO	(for Open)
	B	AOPNL1	(for CVOL)

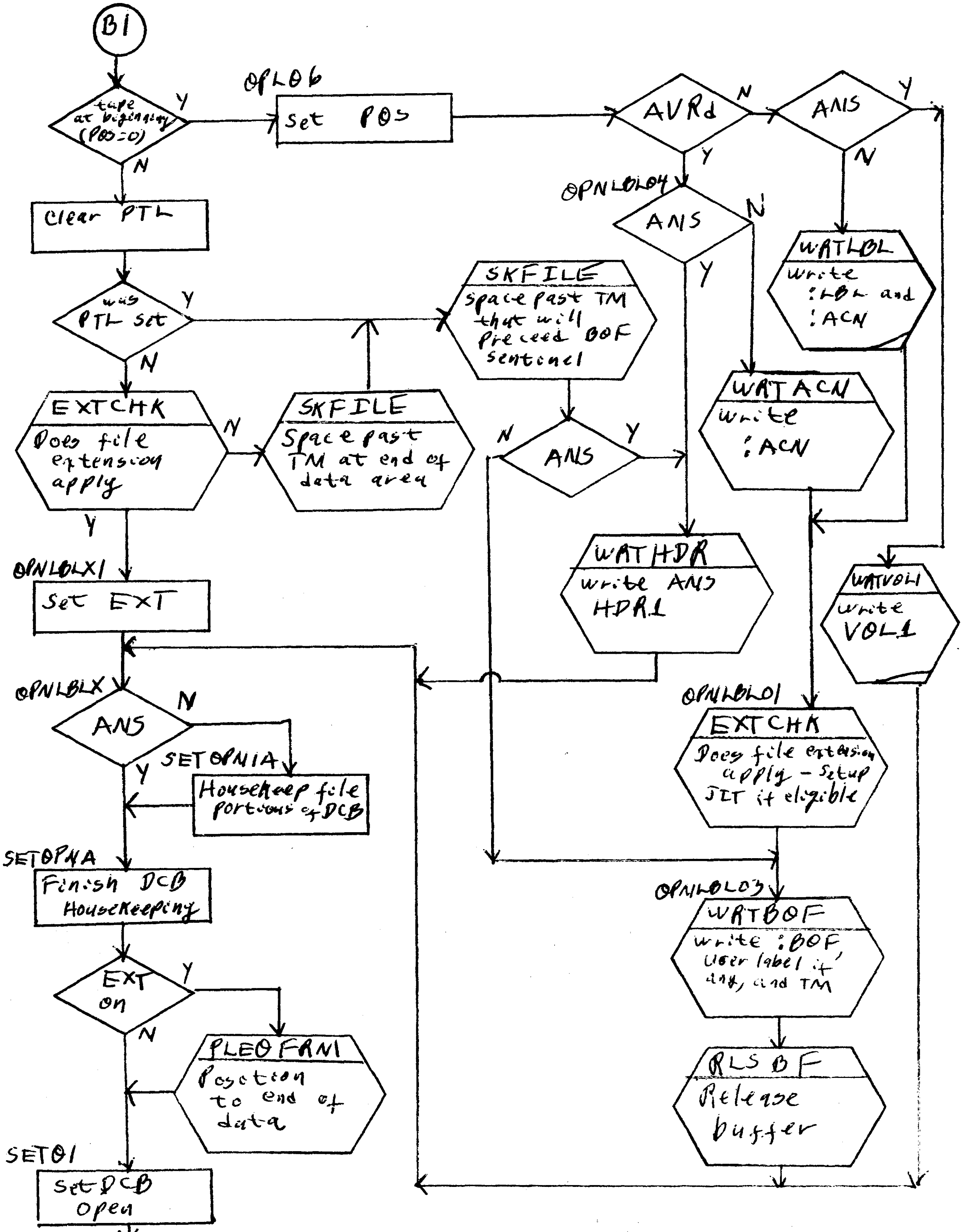
Exit:	Normal	B	OPNX
	Error	B	OPENER03

Operation: An open request enters at OPNLO and initializes CIS and CVI. The open part of a CVOL enters at AOPNL1, bypassing the initialization of CIS and CVI. The DCB is checked for sufficiency of data and the error exit to OPENER03 is taken if the current user may not write the specified account. If not ANS, additional initialization of DCB fields such as access and organization is performed. TAPECHK is then called to find or get the correct volume and do appropriate accounting. If ANS, various ANS fields are initialized in both the DCB and the AVR related tables. If volume headers are needed, the :ACN or :LBL and :ACN are written for labeled tape, as appropriate, or VOL1 for ANS tape. The tape is then positioned, taking into consideration the possible PTL option on the last previous close on this volume and whether file extension applies. Except for the case of file extension, the :BOF sentinel or HDR1 and HDR2 sentinels are written plus any requested user label and a tape mark. The DCB initialization is then completed. If file extension applies, a PFIL (EOF) is performed to assure correct position. The DCB is then set open and the routine exits via OPNX.

OPLO-OPNLO


OPNLO
AOPNL1
If CIS = 0 Set to 1
Set CVI = 1
CHKFLACN
Checkout DCB for sufficient data, etc.
OK to write this account
N
OPENER03
Y
SETACOG
ANS
N
Check and set numerous fields of DCB
Y
TAPECHK
Get/find volume and drive
error
OERX
Error exit routine
MSR01EXIT
ok
ANS
Y
Bump FSN
N
Y
SETID set
N
filename open
N
Find first INSN or OUTSN
Y
Set AVRSID
OPLO2
GSBUF
Get buffer
B1

OPLO-OPNLO


B1
tape at beginning (POS=0)
Y
N
OPLO6
Set POS
AVRd
N
Y
ANS
Y
N
OPNLOLO4
ANS
N
Y
Clear PTL
was PTL set
Y
N
SKFILE
Space past TM that will preceed BOF sentenel
WRTLBL
write :LBL and :ACN
EXTCHK
Does file extension apply
N
Y
SKFILE
Space past TM at end of data area
N
ANS
Y
WRTACN
write :ACN
WRTHDR
write ANS HDR1
WRTVOL1
write VOL1
OPNLOLX1
Set EXT
OPNLBLX
ANS
N
Y
SETOPN1A
Housekeep file portions of DCB
OPNLOLO1
EXTCHK
Does file extension apply - Setup JIT if eligible
SETOPNA
Finish DCB Housekeeping
OPNLOLO3
WRTBOF
write :BOF, user label if any, and TM
EXT on
Y
N
PLEOFRN1
Position to end of data
RLSBF
Release buffer
SETO1
Set DCB open
OPNX

ID

OPNL - processes open and volume switch requests on the input path for labeled and ANS tapes. It contains routines which allocate: disk space for random files (GRAND) and tapes for labeled, ANS and device tape DCBs (TAPECHK).

OPNLA, OPNLBL

Purpose: To perform the M:OPEN procedure for an input labeled or ANS tape.

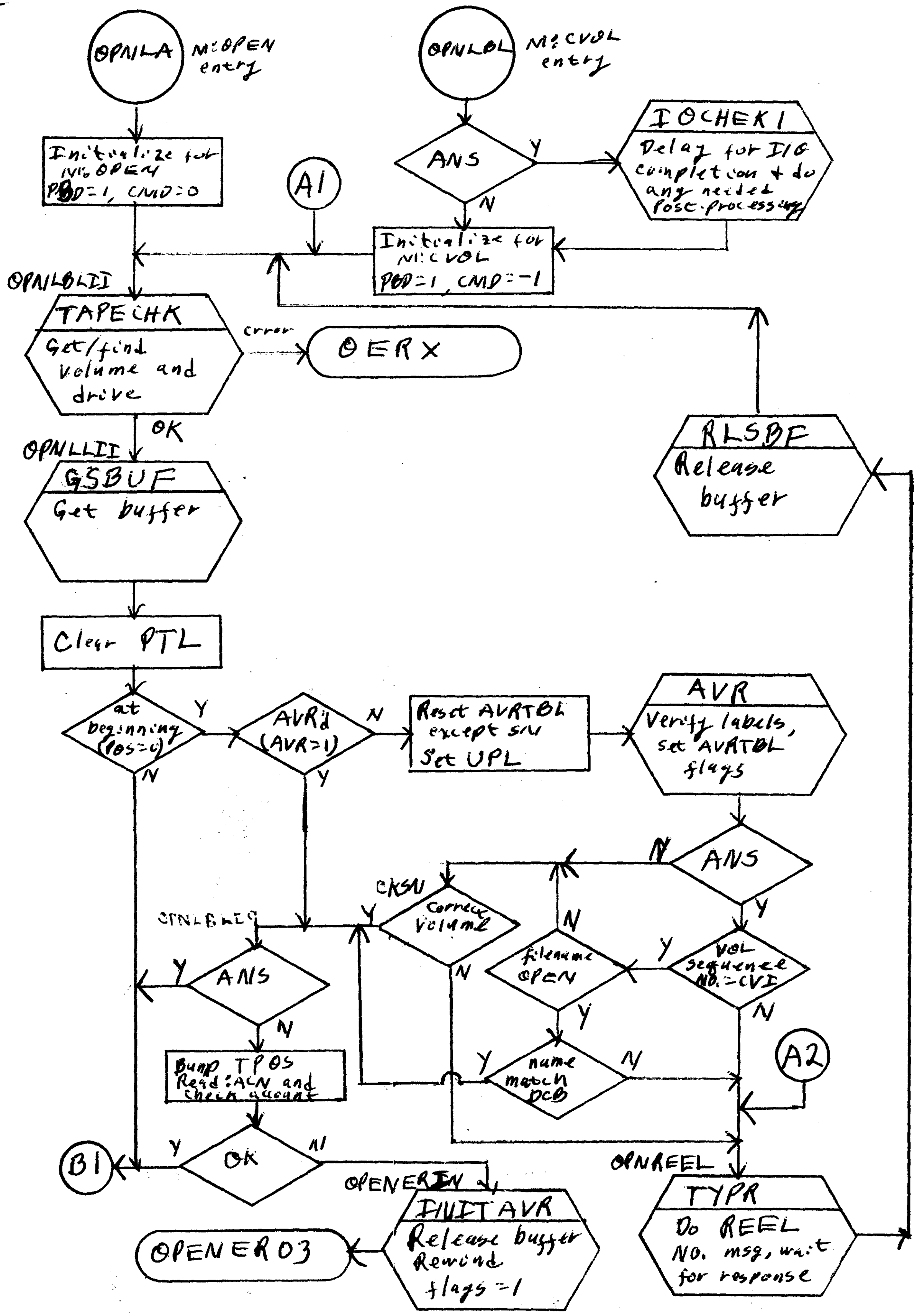
Entry: B OPNLA
B OPNLBL

Exit: Normal B MSRWRTX
Error B MSR01EXIT

Operation: An open request enters at OPNLA and a volume switch at OPNLBL. The primary difference is OPNLA initializes the reel pointers to the file set while OPNLBL retains the pointers. At OPNLA, if DCB:CIS = 0, it is initialized to 1 and CIS is then used as a displacement in the DCB SN list to locate the initial reel to be read. SETID is cleared and CVI is set to 1. File sections whose volume sequence number (CVI) is greater than 1 will be accessed through a volume switch, thus entering at OPNLBL. TAPECHK is called to issue the volume to the user if it is not already assigned to him. AVR is called if the operator has not done a physical AVR. The :ACN sentinel is read and a file mark is skipped if labeled tape.

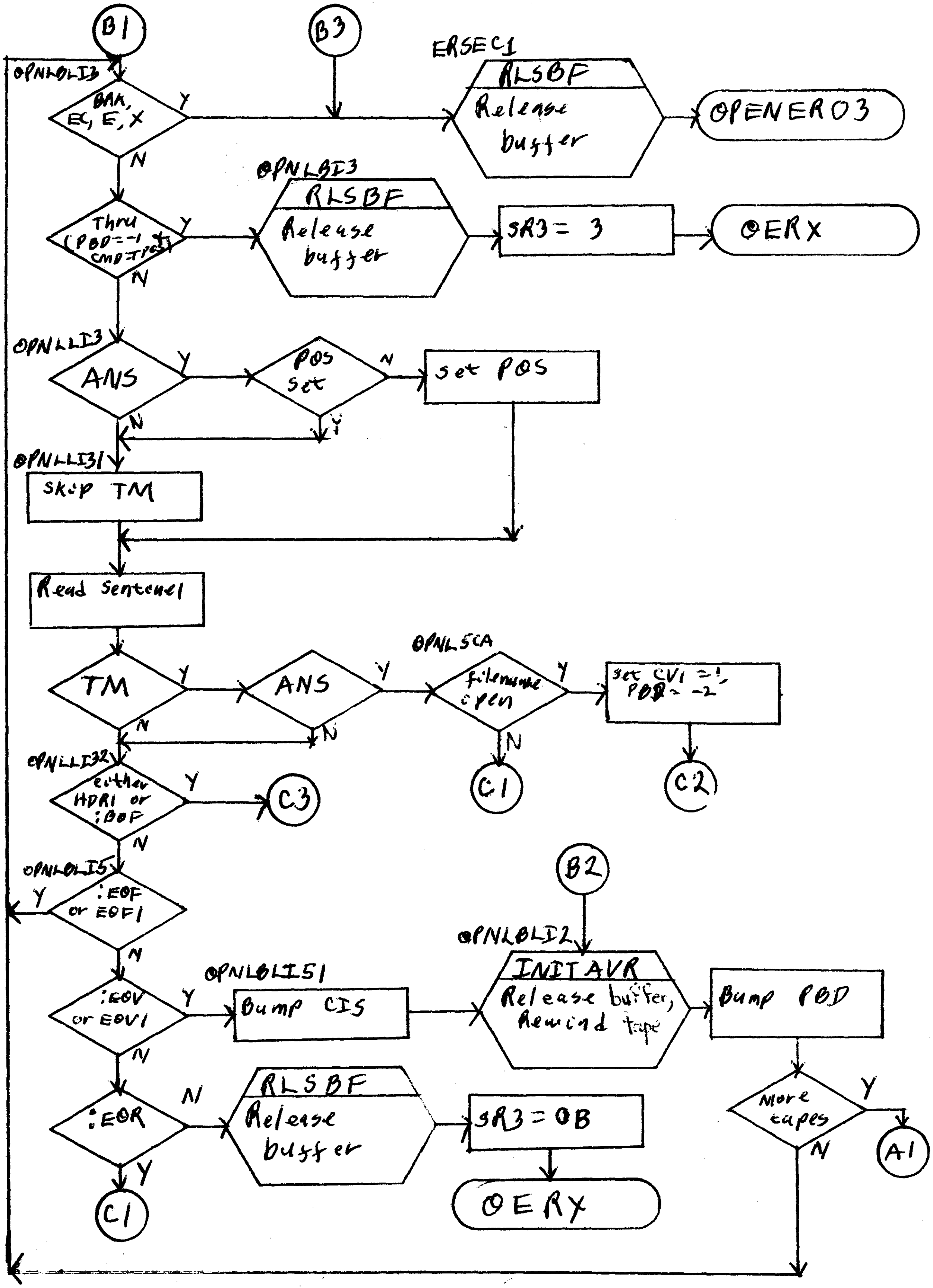
The file search on the current volume now begins. The next sentinel is read. A tape mark on an ANS tape indicates that the end-tape-set condition has occurred without finding the file. In this case, control passes to OPNL5C to determine the feasibility of a volume switch.

OPNL-OPNLA

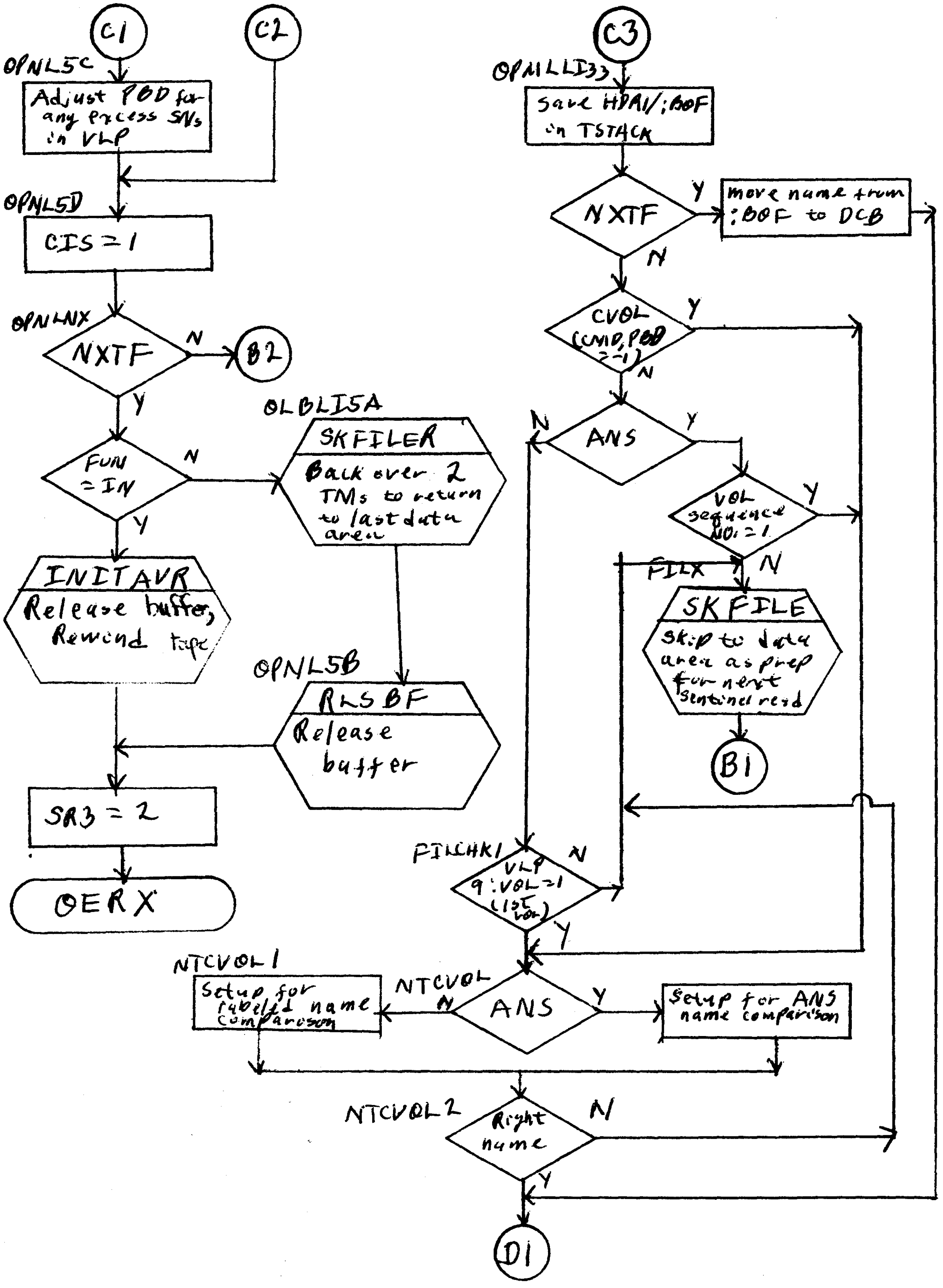


OPNLA
M:OPEN entry
OPNLBL
M:CVOL entry
ANS
Y
N
IOCHEK1
Delay for I/O completion + do any needed post-processing
Initialize for M:OPEN PBD=1, CMD=0
A1
Initialize for M:CVOL PBD=1, CMD=-1
OPNLBLII
TAPECHK
Get/find volume and drive
error
OERX
OK
OPNLLII
GSBUF
Get buffer
Clear PTL
at beginning (POS=0)
Y
N
AVR'd (AVR=1)
N
Y
Reset AVRTBL except SN Set UPL
AVR
Verify labels, set AVRTBL flags
RLSBF
Release buffer
ANS
N
Y
CKSN
Correct Volume
Y
N
OPNLBREQ
ANS
Y
N
Vol sequence NO.=CVI
Y
N
filename open
N
Y
name match DCB
Y
N
A2
Bump TPOS Read :ACN and check account
OK
Y
N
B1
OPENERIN
OPNREEL
FINITAVR
Release buffer Rewind flags=1
OPENER03
TYPR
Do REEL No. msg, wait for response

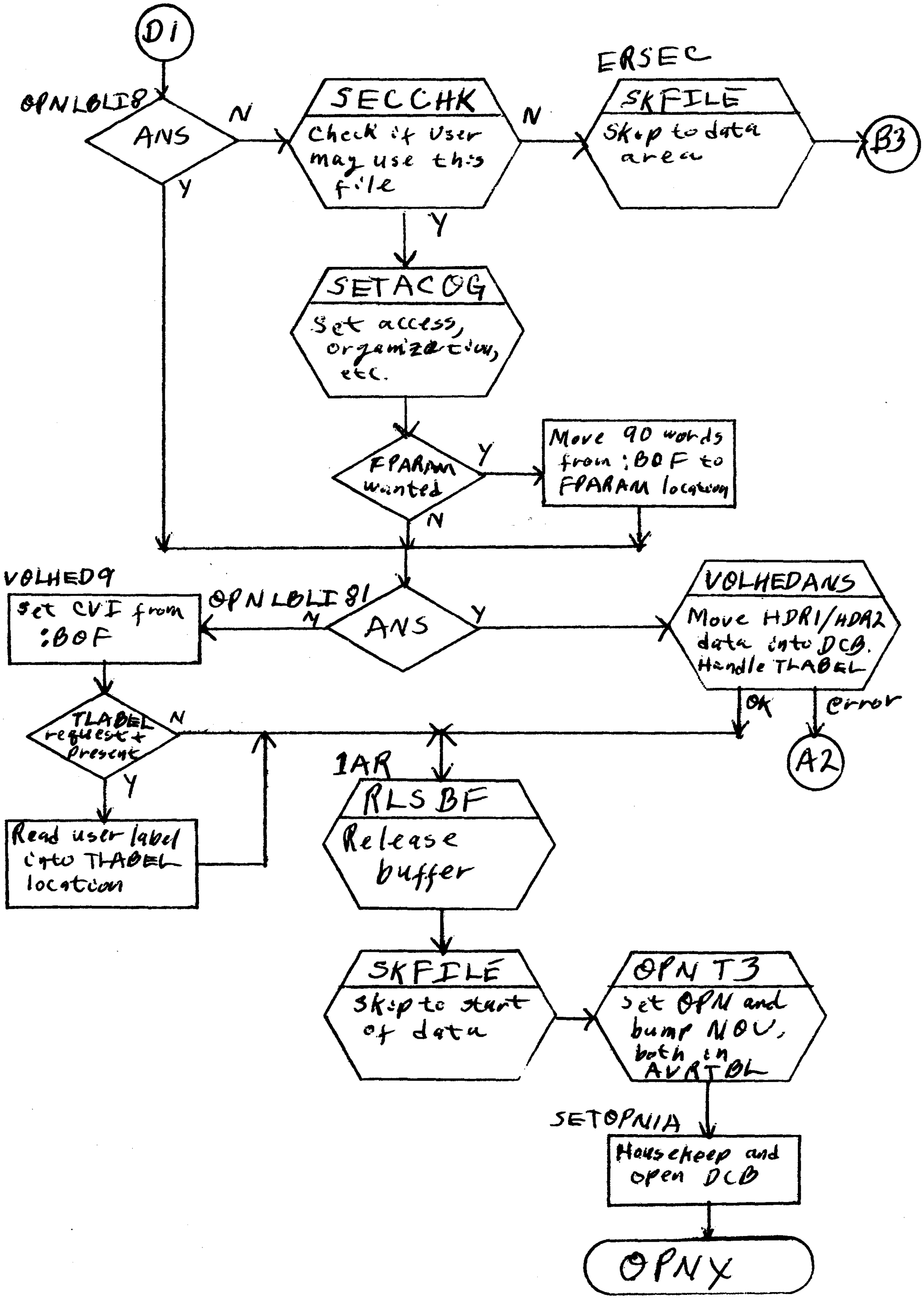
OPNL-OPNLA


B1
B3
ERSEC1
OPNLOLI3
BAK, EC, E, X
Y
N
RLSBF
Release buffer
OPENER03
OPNLBI3
Thru (PBD=-1 + CMD=TPE)
Y
N
RLSBF
Release buffer
SR3 = 3
OERX
OPNLLI3
ANS
Y
POS set
N
set POS
N
Y
OPNLLI31
skip TM
Read Sentenel
TM
Y
ANS
Y
OPNL5CA
filename open
Y
set CV1 = 1, PBD = -2
N
N
N
C1
C2
OPNLLI32
either HDR1 or ;BOF
Y
C3
N
OPNLOLI5
:EOF or EOF1
Y
N
B2
OPNLOLI2
OPNLOLI51
:EOV or EOV1
Y
Bump CIS
INITAVR
Release buffer, Rewind tape
Bump PBD
N
More tapes
Y
A1
N
:EOR
N
RLSBF
Release buffer
SR3 = OB
OERX
Y
C1

OPNL-OPNLA


C1
C2
C3
OPNL5C
Adjust POD for any excess SNs in VLP
OPNL5D
CIS = 1
OPNLNX
NXTF
N
B2
Y
OLBLI5A
SKFILER
Back over 2 TMs to return to last data area
FUN = IN
N
Y
INITAVR
Release buffer, Rewind tape
OPNL5B
RLSBF
Release buffer
SR3 = 2
OERX
OPNLLI33
Save HDAI/;BOF in TSTACK
NXTF
Y
move name from ;BOF to DCB
N
CVOL (CVID,POD =-1)
Y
N
ANS
N
Y
VOL sequence NO: = 1
Y
FILX
N
SKFILE
Skip to data area as prep for next sentinel read
B1
FILCHK1
VLP 9:VOL=1 (1st vol)
N
Y
NTCVOL1
Setup for labeled name comparison
NTCVOL
N
ANS
Y
Setup for ANS name comparison
NTCVOL2
Right name
N
Y
D1

OPNL-OPNLA


D1
OPNLBLI8
ANS
N
Y
SECCHK
Check if User may use this file
N
Y
ERSEC
SKFILE
Skip to data area
B3
SETACOG
Set access, organization, etc.
FPARAM wanted
Y
N
Move 90 words from :BOF to FPARAM location
VOLHED9
Set CVI from :BOF
OPNLBLI81
N
ANS
Y
VOLHEDANS
Move HDR1/HDR2 data into DCB. Handle TLABEL
OK
error
A2
TLABEL request + present
N
Y
Read user label into TLABEL location
1AR
RLSBF
Release buffer
SKFILE
Skip to start of data
OPNT3
Set OPN and bump NOV, both in AVRTBL
SETOPN1A
Housekeep and open DCB
OPNX

If the sentinel is an :EOF or EOF1, the search is continued on the same volume. If the sentinel is an :EOV or EOV1, DCB:CIS is incremented, DCB:PSD is decremented, and if PBD is not 0, the file search continues on the next tape, i.e., a volume switch is performed.

If the sentinel is a :EOR, DCB:CIS is initialized to 1, DCB:PBD is decremented, and if PBD is not 0, the file search continues on the first volume. If the sentinel is not recognizable, an error return to the user occurs. If the sentinel was a :BOF or HDR1, and describes the correct file, the AVR entry is marked open and the NOU field is incremented. A security check is made and FPARAM is processed (labeled tape only).

If DCB:SETID = 0, it is set from HDR1. If DCB:SETID ≠ 0, it is verified against HDR1. File information is transferred from :BOF/HDR1 into the DCB.

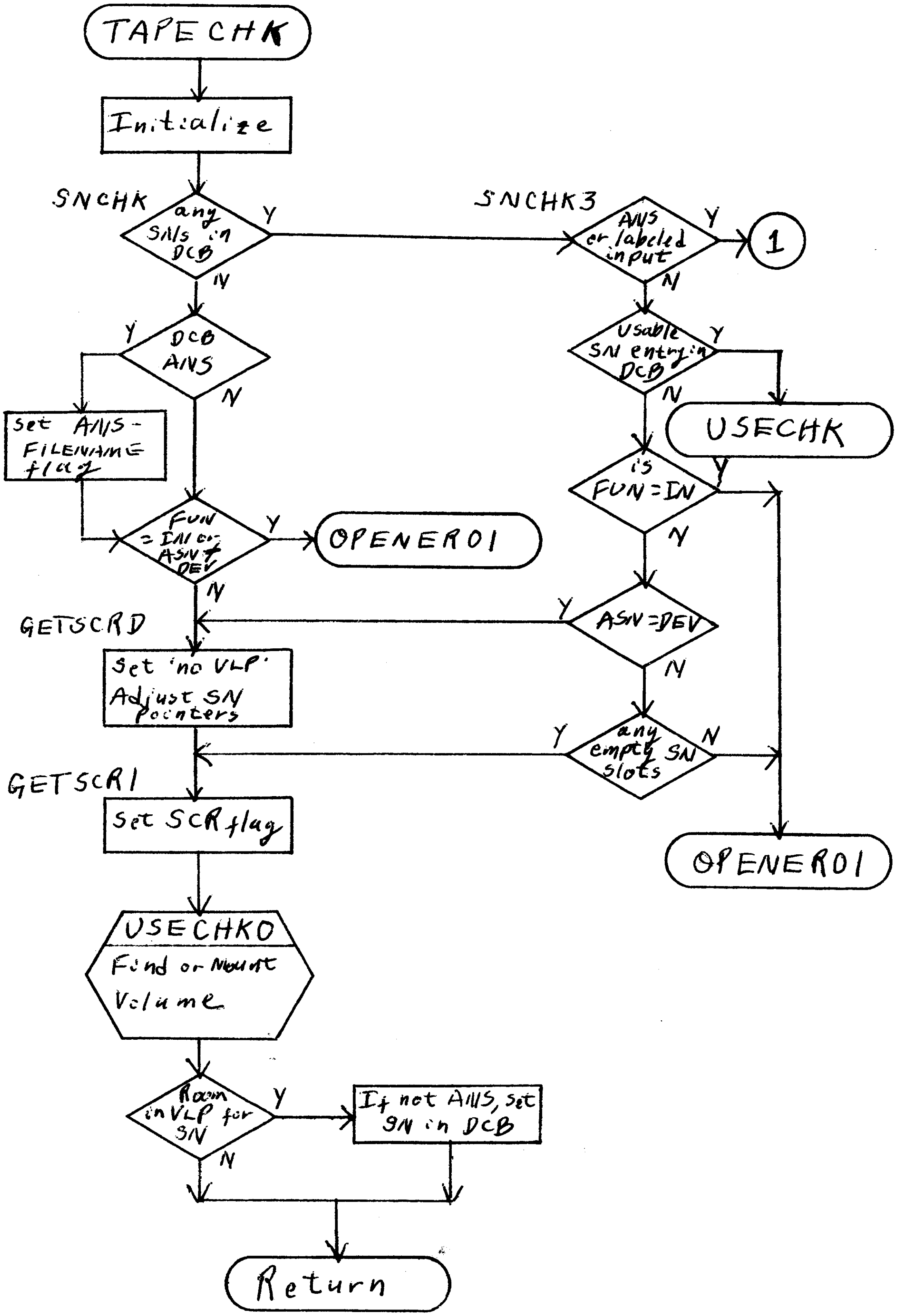
VOLHED is called to transfer the user's label if necessary. The DCB is house-kept and marked open and OPNL exits. If the :BOF/HDR1 is not correct, the next :BOF/HDR1 is interrogated, unless PBD is 1, indicating that this is last reel to be searched, and AVR:TPOS=DCB:CMD≠0, indicating that the tape heads are back at the starting point. The search for the correct :BOF/HDR1 continues until the tape heads are back at the starting point on the original volume.

TAPECHK

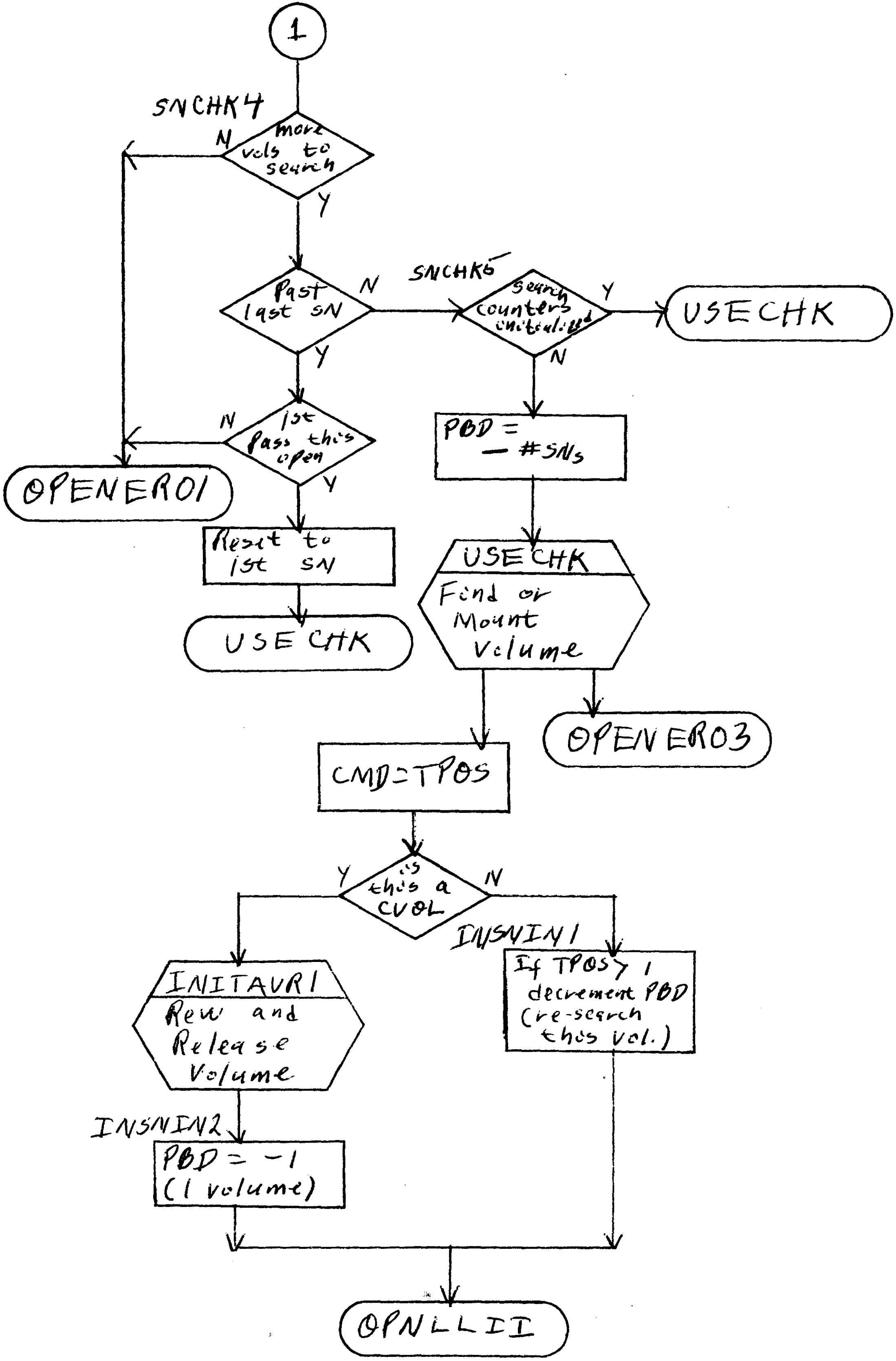
Purpose: TAPECHK is a subroutine that obtains the proper volume when any tape DCB is opened. It performs various checks upon the DCB, tape sentinels, and system and user limits, and accounts for the allocation of the tape drive. In performing these functions, TAPECHK uses the USECHK subroutine.

Entry: BAL, R5 TAPECHK
EXIT: Normal B 1,5
Error B 0,5

OPNL-TAPECHK



OPNL-TAPECHK



Operation: USECHK is called to cause the reel to be mounted if it is not already there. The number of SNs in the DCB list is placed in DCB:PBD which will determine the number of volumes to be searched in order to find the requested file. If there are no SNs in the DCB and the DCB is ANS, the filename open flag (DCB word 0, bit 14) is set and processing is continued.

If OPNL was called from CVOL, the current volume is rewound and DCB:PBD is set to 1 to limit the file search to the current volume. If not, the tape mark count between load point and the current position of the tape heads (AVR:TPOS), is copied into DCB:CMD. If TPOS is nonzero, DCB:PBD will be incremented which will permit the front part of the current volume to be searched if the file cannot be located beyond the current position on the current volume or on any subsequent volume.

USECHK

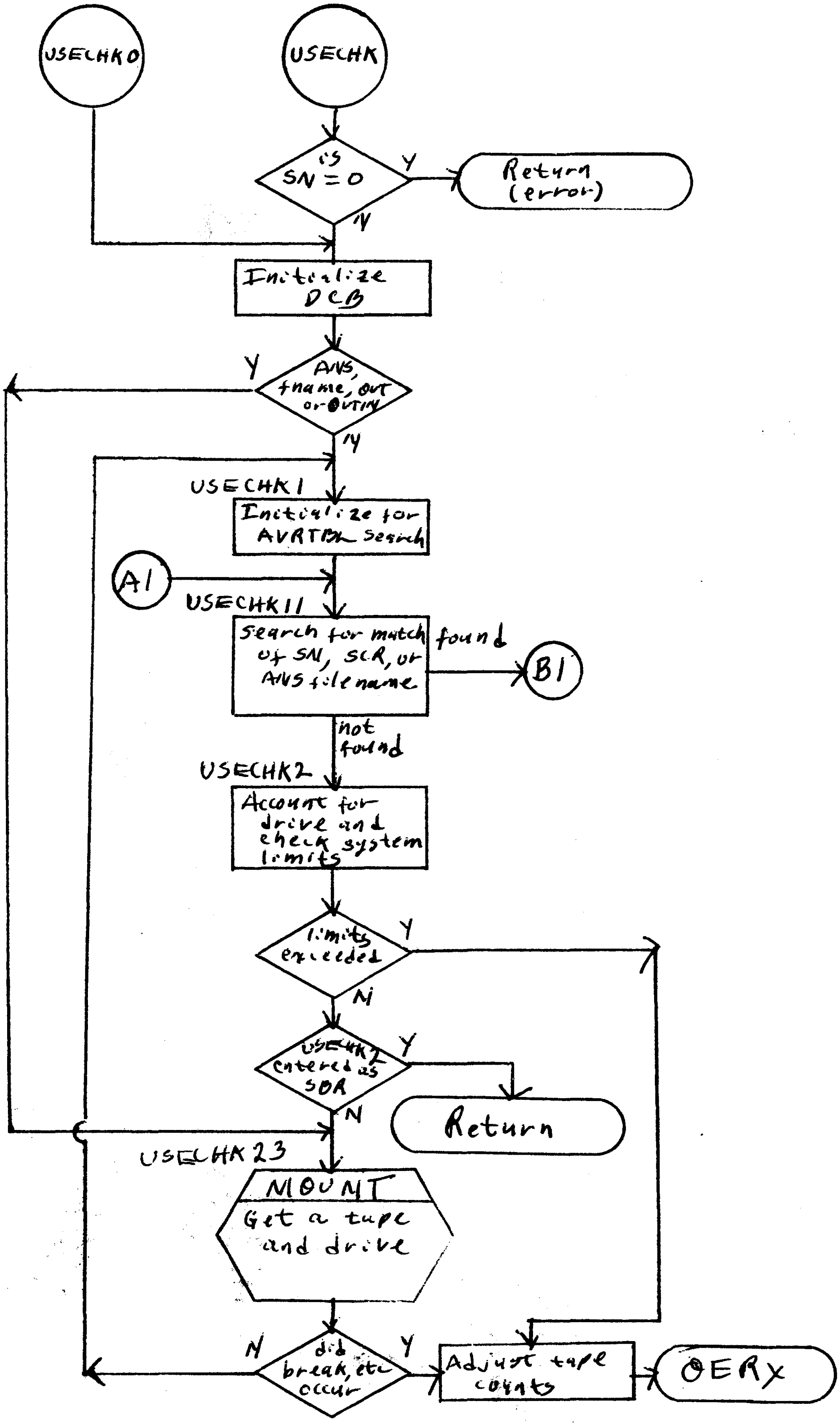
Purpose: Obtains a scratch or specific SN tape, checks system and user limits and account for the allocation of the tape drive. The alternate entry to USECHK0 is used in requesting a scratch. In performing its function, USECHK calls the MOUNT subroutine of TYPR.

Entry: BAL, R5 USECHK

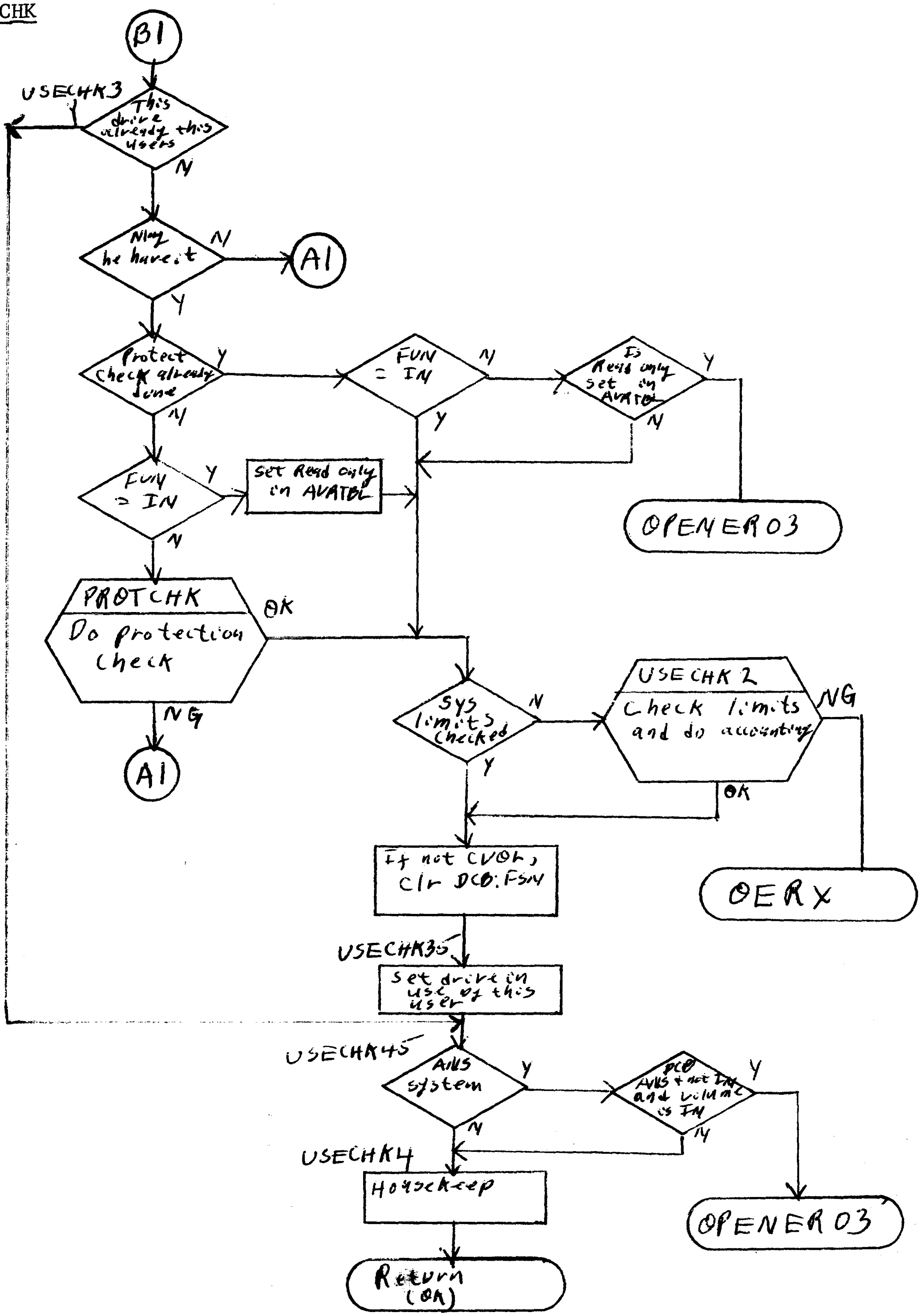
Exit:

Operation: This routine checks on whether the tape, whose reel number is in the buffer at R7, word index in R3, is found in the AVRTBL or is mounted via MOUNT. It exits skipping if the tape is successfully mounted, otherwise, exits in line.

OPNL-USECHK


USECHK0
USECHK
is SN = 0
Y
N
Return (error)
Initialize DCB
ANS, fname, out or OUTIN
Y
N
USECHK1
Initialize for AVRTBL search
A1
USECHK11
Search for match of SN, SCR, or ANS filename
found
B1
not found
USECHK2
Account for drive and check system limits
limits exceeded
Y
N
USECHK2 entered as SDR
Y
Return
N
USECHK23
MOUNT
Get a tape and drive
did break, etc occur
N
Y
Adjust tape counts
OERX

OPNL-USECHK



B1
USECHK3
This drive already this users
N
May he have it
N
A1
Y
Protect check already done
Y
FUN = IN
N
Is Read only set in AVATBL
Y
N
Y
N
FUN = IN
Y
Set Read only in AVATBL
N
OPENER03
PROTCHK
Do protection check
OK
NG
A1
Sys limits checked
N
USECHK2
Check limits and do accounting
NG
Y
OK
OERX
If not CVOL, clr DCB:FSN
USECHK35
Set drive in use of this user
USECHK45
AVS system
Y
DCB AVS + not IN and volume is IN
Y
N
N
USECHK4
Housekeep
OPENER03
Return (OK)

In the initialization sequence, if the request is for an ANS tape to be opened by filename as OUT or OUTIN, the preliminary AVR search is bypassed and MOUNT is called. If the open is ANS by filename, the AVR search is for filename rather than for SN or scratch. If the open is ANS by filename and is OUT or OUTIN, the tape is not accepted if the AVR entry is not for the current user or if the solicit flag is not set. Write protection checking for output tapes not previously verified is performed by calling PROTCHK. PROTCHK is logically a part of USECHK and the OPNL module, but was placed in LBLT to minimize the possibility of the OPN segment overflowing. Then, immediately prior to setting drive in use by this user (J:ASPIN), if the operation is not a CVOL, DCB:FSN is cleared so that it will be properly initialized.

ACNTCHK

The next record is read and, if it is the :ACN sentinel and the account number matches the account number in the DCB, the skipping exit is taken.

VOLHED

The VOLume field of the :BOF/HDR1 sentinel is transferred to DCB:CVL. If the DCB is ANS, VOLHEDANS is called. If the user has specified the TLABEL option and his buffer has a non zero size indicated, the user label is read into the user's TLABEL buffer.

TRNINFO

If the user has specified FPARAM and that address is valid, the contents of the BOF sentinel are moved to the user's FPARAM buffer.

SETOPNIA, SETOPNA

When entered at SETOPNIA, DCB words 15-21, except 17, are cleared as are BUF1 and the first word of the key buffer (addressed by KBUF). For both entry points, DCB items EGV, AGV EOP, and TRN are cleared, the DCB is set open, and the routine exits to MSRWRTX.

ID

POS – The POS module contains routines to process the M:PFIL, M:PRECORD, M:REW, and M:WEOF CALs.

Purpose: To perform the M:PFIL procedure.

Entry: B PFIL
(R7) = word address of FPT+1

Exit: Error B PULLALLEXIT
Normal B MSRWRTX

Operation: See following flowchart. The file is opened if not open. If the file cannot be opened, the error exit is taken. If DCB:ASN is disk and the file is random, the MSRWRTX exit is taken. If the DCB is for device and not tape, the exit is taken. If the DCB is for unlabeled tape, a skip forward or backward file command, according to the FPT, is executed via QUEUE and the MSRWRTX exit is taken with WAT set.

If the DCB is ANS and BOF is indicated, no action occurs and the exit is taken. If the DCB is for labeled tape and the FPT indicates BOF, LBLT is called to close the DCB with PTL (position to label) option. Upon return, the DCB is reset to open, two forward-space file commands are executed via QUEUE, and the exit is taken. If the FPT indicates for ANS or labeled tape, the routine exits if the last operation was a Write. Otherwise, a forward space file command is executed and CHKEOF is called to check the labels. (Note: DCB:BUF is cleared as a flag to CHKEOF that the function is M:PFIL.) If the EOF sentinel is found, the routine exits. Otherwise, a volume switch is made and a forward space file command is again given.

REW

Purpose: To perform the M:REW procedure for a disk file, labeled tape, ANS tape, or unlabeled tape.

Entry: B REW

Exit: B MSRWRTX

Operation: See following flowchart. If the tape is ANS, the tape is positioned as if an AVR sequence had been performed. That is, the CLOSE(PTV) FPT is set up in TSTACK, pointed to by R7, and MSRCLS is called. If the DCB is open and is for disk or labeled tape, PFIL in PFIL routine is called to position to beginning of file. If the DCB is for device and is assigned to tape, the DCB is opened, if necessary , and rewind command is executed via QUEUE. If the DCB is closed and is not assigned to device, the routine exits via OBSR4. Otherwise, exits are via B MSRWRTX.

WEOF

Purpose: To perform the M:WEOF procedure

Entry: B WEOF

Exit: B MSRWRTX

Operation: If the DCB is ANS, M:WEOF is ignored by exiting to MSRWRTX. The DCB is opened if not open and, if the function is IN or the DCB is not assigned to device, the exit is taken. If the device is paper tape or card punch, a !EOF record is written. If the device is a printer, PAGE is called to skip to top of rom. If a magnetic tape, a tape mark is written via QUEUE. For all other devices the routine exits directly.

PRECORD

Purpose: To perform the M:PRECORD procedure.

Entry: B PRECORD
(R7) = word address of FPT+1

Exit: B MSRWRTX

Operation: See following flowchart. If the DCB is ANS, M:PRECORD is ignored by exiting to to MSRWRTX. The DCB is opened if not open and FPT parameters are moved into the DCB. If the DCB is for device and is assigned to (unlabeled) tape, a position request is passed to QUEUE. All other devices return directly. If the DCB is for disk file, PRDCRD11 is called to skip records in the designated direction, and ARS is set with the number of records remaining (e.g., if EOF or BOF is encountered). If the file is empty, BOF or EOF is set depending on direction. If the DCB is for labeled tape, repeated read-zero-byte calls are made to READL until either the prescribed number of records has been skipped in the designated direction, or an abnormality (e.g., BOF, EOF) is encountered. ARS is set with any remaining record count.

ID

RDL - Contains routines related to the forward reading of labeled tape.

READL

Purpose: To process read requests for labeled tape.

Entry: OVERTO LBLTSEG, 0
B READL

Exit: B MSREXIT

Operation: See following flowchart. If the last operation was Write, RDCLS is called to write the trailer sentinels. If the request is for a keyed Read, ARD2 is called; if for a Read-reverse, ARD3 is called; (:EOF/EOF1 and :EOR/EOV1) if for an ANS Read, MSRRED is called.

Otherwise, the routine transfers the record via TRNREC after obtaining and loading a buffer via CHKTRN and RDBLK, if necessary. Exit is then made to MSREXIT.

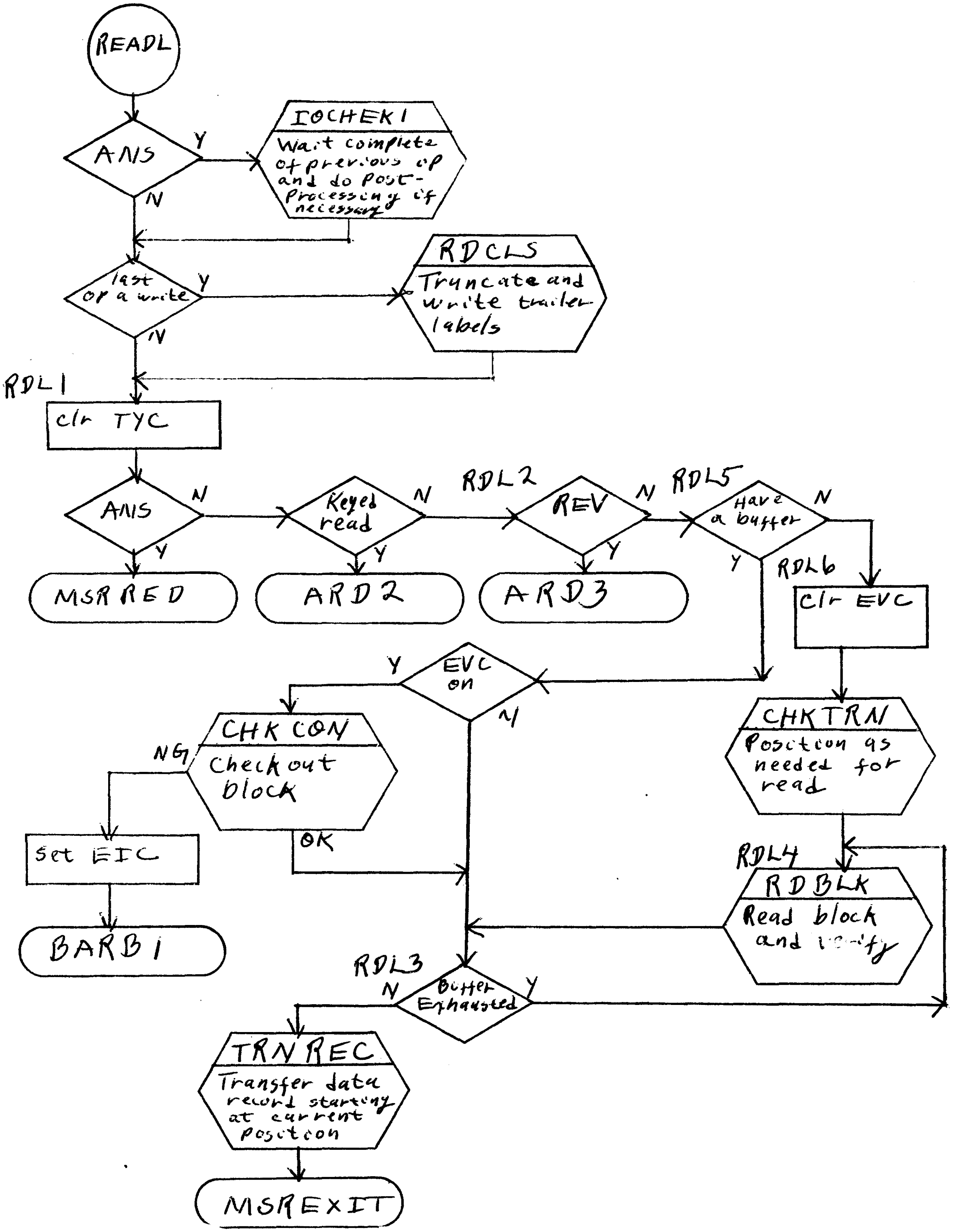
RDBLK

Purpose: To read a labeled tape block in the forward direction.

Entry: BAL, RO RDBLK

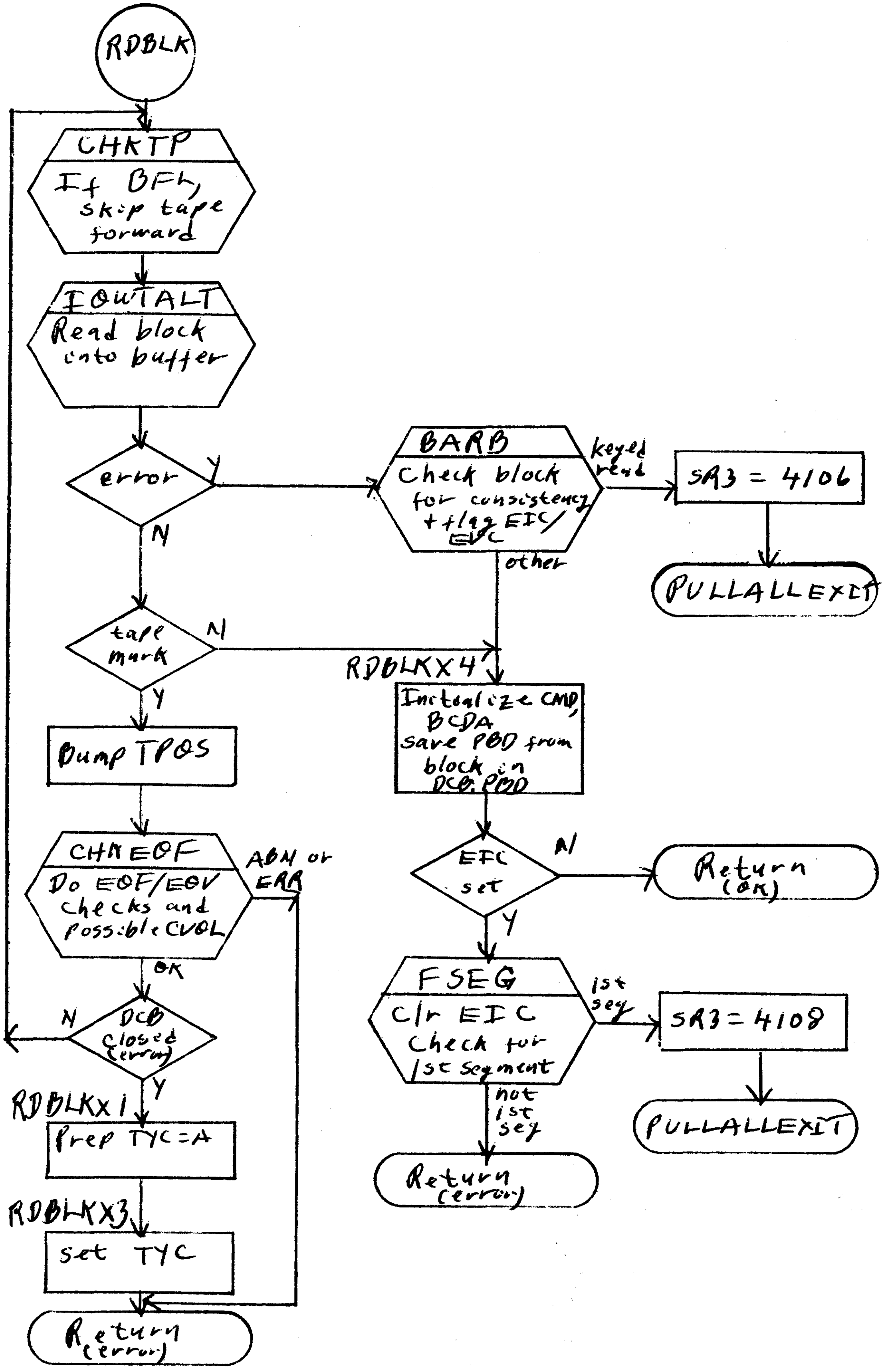
Exit: If EOF or unable to switch to the next volume - B PULLEXIT
If normal complete - B PLLEXIT1

RDL-READL



READL
ANS
Y
N
IOCHEK1
Wait complete of previous op and do post-processing if necessary
last op a write
Y
N
RDCLS
Truncate and write trailer labels
RDL1
clr TYC
ANS
N
Y
MSRRED
Keyed read
N
Y
ARD2
RDL2
REV
N
Y
ARD3
RDL5
Have a buffer
N
Y
RDL6
clr EVC
EVC on
Y
N
CHKCON
Check out block
NG
OK
Set EIC
BARB1
CHKTRN
Position as needed for read
RDL4
RDBLK
Read block and verify
RDL3
Buffer Exhausted
N
Y
TRNREC
Transfer data record starting at current position
MSREXIT

RDL-READL


RDBLK
CHKTP
If BFL, skip tape forward
IOWTALT
Read block into buffer
error
Y
N
BARB
Check block for consistency + flag EIC/EVC
keyed read
SR3 = 4106
PULLALLEXIT
other
tape mark
N
RDBLKX4
Initialize CMP, BCDA save PBD from block in DCB.PBD
Y
Bump TPOS
CHKEOF
Do EOF/EOV checks and possible CVOL
ABN or ERR
EFC set
N
Return (OK)
Y
FSEG
Clr EIC Check for 1st Segment
1st seg
SR3 = 4108
OK
N
DCB closed (error)
Y
RDBLKX1
Prep TYC=A
not 1st seg
PULLALLEXIT
Return (error)
RDBLKX3
set TYC
Return (error)

Operation: A blocking buffer is obtained, correct tape position is assured by skipping a block if the last Read was reverse, the tape is read with end action at READLEND, and IOWT is used to delay for I/O completion. If a tape mark was read, CHKEOF determines if the next sentinel is an :EOF (for abnormal exit) or an :EOV. If :EOV, a volume switch is initiated and, if successful, the entire RDBLK routine is re-executed, various buffer pointers are set and the normal exit is taken. If unable to switch volume, the abnormal exit is taken.

GTFL

Purpose: To locate the next key in the forward direction for a labeled tape.

Entry: BAL, RO GTFL

Exit: If EOF: B PULLEXIT
If found: B PULLEXIT

Operation: Pointers are stepped to the next record area. If end-of-block is encountered, the next block is read. If EOF is encountered, the EOF exit is taken.

TRNREC

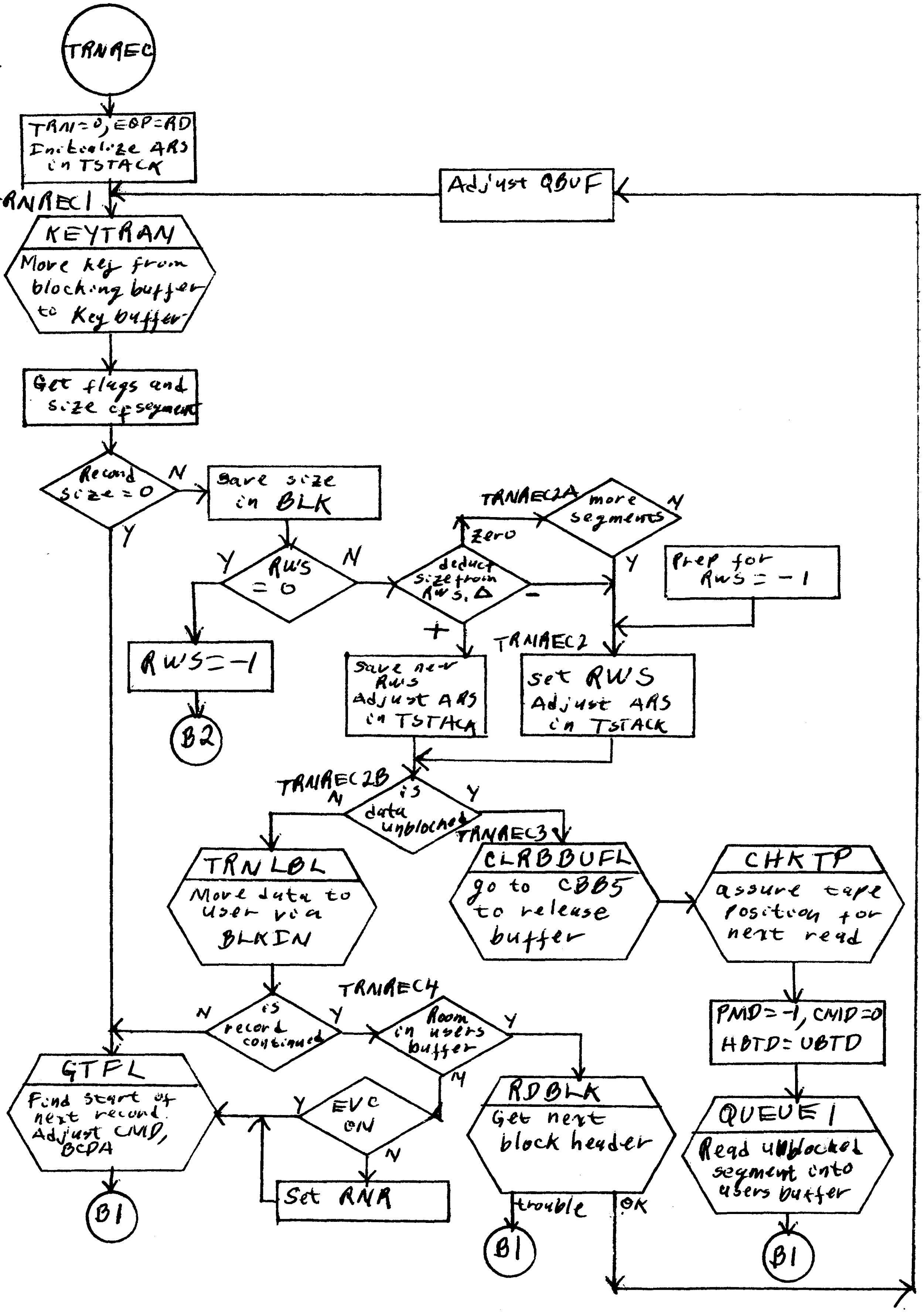
Purpose: to transfer a labeled tape record to the user's buffer.

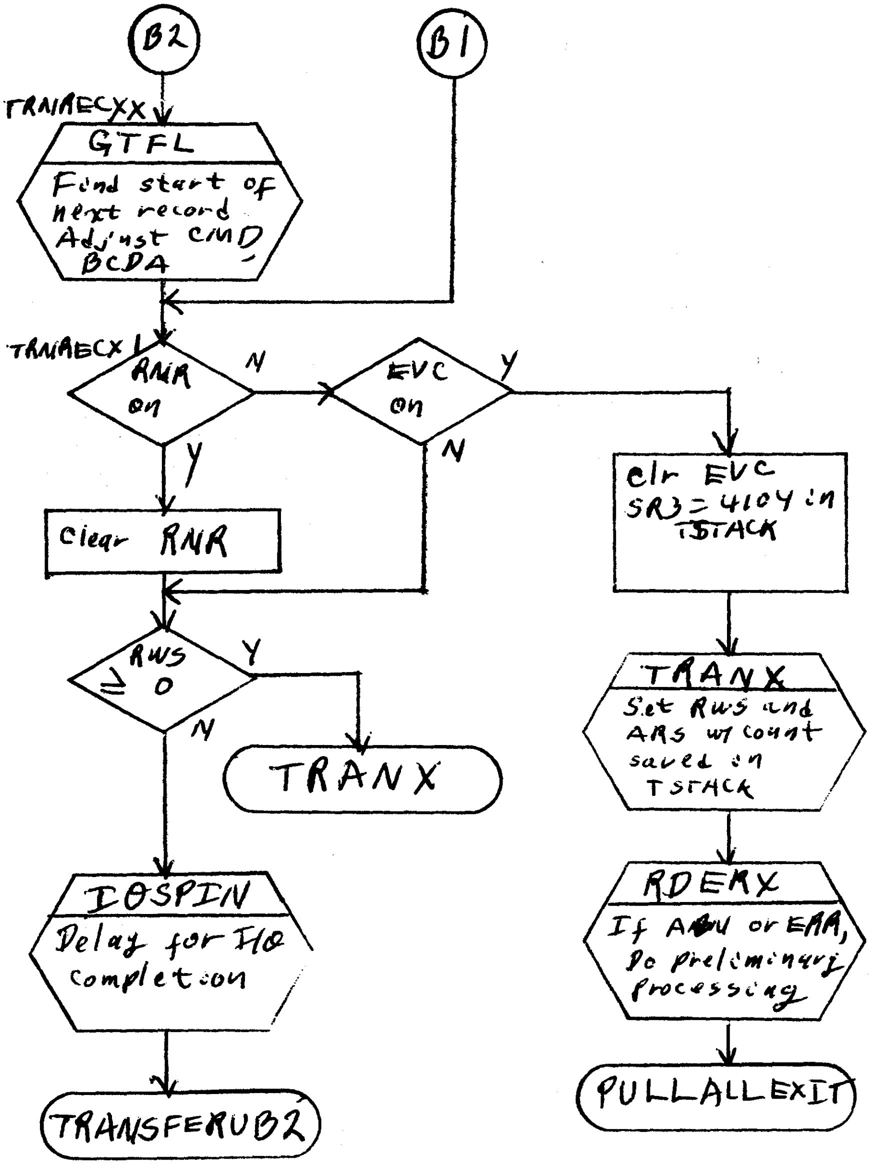
Entry: BAL, RO TRNREC
(R15) = word answer of user's buffer
(R4) = byte displacement in user's buffer

Exit: B *RO

Operation: See following flowchart. The current key is moved into the user's key buffer via KEYTRAN. If the record size or requested size is zero, the balance of the record is skipped and exit is via TRANX or TRANSFERUB2, respectively. If the control record indicated that the data is unblocked, the unblocked record is read directly into the user's buffer. If the record is continued and the user's byte count is not exhausted, the next

RDL-TRNREC


TRNREC
TRN=0, EOP=RD
Initialize ARS in TSTACK
TRNREC1
Adjust QBUF
KEYTRAN
Move key from blocking buffer to Key buffer
Get flags and size of segment
Record size = 0
N
Y
Save size in BLK
RWS = 0
Y
N
RWS=-1
B2
deduct size from RWS, Δ
zero
-
+
TRNREC2A
more segments
N
Y
Prep for RWS = -1
TRNREC2
Save new RWS
Adjust ARS in TSTACK
Set RWS
Adjust ARS in TSTACK
TRNREC2B
N
is data unblocked
Y
TRNREC3
TRNLBL
Move data to user via BLKIN
CLRBBUFL
go to CBB5 to release buffer
CHKTP
assure tape position for next read
TRNREC4
N
is record continued
Y
Room in users buffer
Y
N
PMD=-1, CMD=0
HBTD=UBTD
GTFL
Find start of next record. Adjust CMD, BCDA
EVC ON
Y
N
Set RNR
RDBLK
Get next block header
trouble
OK
QUEUE1
Read unblocked segment into users buffer
B1
B1
B1


B2
B1
TRNRECXX
GTFL
Find start of next record Adjust CMD BCDA
TRNRECX1
RNR on
N
Y
EVC on
Y
N
Clear RNR
clr EVC SR3 = 4104 in TSTACK
RWS ≥ 0
Y
N
TRANX
TRANX
Set RWS and ARS w/ count saved in TSTACK
IOSPIN
Delay for I/O completion
RDERX
If ABN or ERR, Do preliminary processing
TRANSFERUB2
PULLALLEXIT

block is read and TRNREC is reentered. If the user's byte count is exhausted and the record is continued, control records (and subsequent unblocked date records if necessary) are skipped.

If the control record indicates the data is blocked, the user-requested number of bytes are transferred to the user's buffer. If the current block contains only part of the requested data (i.e., the record is continued), the next block is read and TRNREC is reentered. If the user's byte count is exhausted and the record is continued, subsequent block(s) are read until the next logical record is in position to be read (i.e., the pointer to the current displacement in the blocking buffer is at the next record; or the blocking buffer is released and the next logical record is first in the next block to be read).

TYPR

Functional Overview

The TYPR module contains routines that perform four functions:

(1) A multiple entry point routine to construct and output OC messages related to magnetic tape handlings.

(2) A routine to print or type a message to the operator that does not require response.

(3) A routine to type an OC message and accept a response from the operator.

(4) Routines to mount and verify a private disk pack volume.

Interfaces

When a user is performing an OPEN and a desired tape or disk pack is not available, TYPR is entered to find an available unit and output the corresponding message to the operator. The user is blocked and then unblocked after a response has occurred via AVR sequence or KEYIN.

There are also entry points to output messages to the OC, which are called at appropriate situations by the monitor.

ID

MOUNT— MOUNT1 - DISMNT - REEL - SAVRL - ANSERR

PURPOSE

To output a message to the operator. If entered at MOUNT, an available tape drive must first be found.

USAGE

BAL, SR4	MOUNT	INPUT R3=REEL # or -1 (scratch)
BAL, SR4	MOUNT1	
BAL, SR4	REEL	
BAL, SR4	DISMNT, SAVRL, ANSERR	INPUT R2= AVR INDEX R3 = REEL #

DATA BASE

J:BASE - To store output messages

AVRTBL - The FLAGS and REEL # indicate the status of the unit.

SOLICIT - To indicate a unit request is posted.

AVRNOU - Set to 0 under various error conditions.

AVRID - To mark a drive as belonging to a particular user.

SL:9T, SL:7T S:OSPIN, S:BSPIN - System limits and in use indicators.

JB:C9T, JB:C7T, J:ASPIN, JB:PMTS, JB:TMTS - User limits and in use indicators.

ANSFLGS - Indicate status of ANS units.

AVRFNMT - TYPR moves the DCB file name here for ANS DCBs.

S:CUN - The user's system id used to identify the user with an AVR slot.

UH:FLG -Examined for abnormal conditions on the unblocked user.

ROUTINES

MSROCTY - To output an OC message.

TYPR1 - To clear the ANS tables when clearing the AVR tables.

TYPR2 - If an ANS tape request, it sets ANSFLGS:AT (tape is ANS) and ANSFLGS:MS (scratch request).

TYPR5 - To restore the ANS reel # to R3 before exiting.

TYPR6 - To move the filename from the DCB to the AVRFNMT for ANS tapes.

CVTINDX - To obtain the EBCDIC channel; create the appropriate message and store it into the message buffer (J:BASE+1).

CHKANS1 - To check if a DCB is ANS.

GETTPDR - Finds an available AVRTBL slot. The mask YC5FF determines availability, i.e., AVR:PUB, POS, PTL, OPN, NOU must be zero.

DESCRIPTION

There are three basic functions to be performed when a tape is to be mounted.

(1) Find an available AVRTBL slot and mark it for this user.

(2) Create and type the OC message.

(3) End action procedure on this request.

(1) Find an available AVRTBL slot and mark it for this user

MOUNT is the multiple entry point for this function. If R3 = -1, a scratch tape is desired and R1 is initialized with 'SCRA'. GETTPDR finds an available drive (see ROUTINES); if it is unable to find one, there is a software check '49'. Then an optimal available drive is searched for via GETTPDR, that is, with a zero reel # or also for a non-ANS request, a scratch drive (AVR:SCR=1).

The AVR index of the selected drive is put into DCB:DSI and TYPR2 sets the ANSFLGS. AVR:PTL (drive is pre-allocated) and the reel # are set into AVRTBL. If it is a labeled tape and the type of OPEN is OUT or OUTIN, AVR:ULBL is set. For ANS DCBs, TYPR6 deposits the file name into AVRFNMT.

SOLICIT is set and S:CUN (the user id) is placed into AVRID for this AVR slot. CVTINDX places the mount message into J:BASE+1 and exits to MOUNT1 to type the OC message.

(2) Create and type the OC message

REEL, DISMNT, SAVRL, ANSERR are various entry points which set up their respective messages in registers and give control to CVTINDX to place it into J:BASE+1.

At MOUNT1 and PLIST is set up in TSTACK and MSROCTY is called to output the OC message. The message is then examined to determine if an operator response is desired and the user should be blocked. If the user is not blocked, the location AVRCLEAR is entered to reset the necessary tables and exit.

(3) End action procedure on this request

The user is unblock when a system time quatum expires or if a keyin or AVR sequence is performed. The routine AVRWTCHK decides what to do with the unblocked user. If any UH:FLG-BRK, EC, ABRT, ERR are set, AVRCLEAR is entered. SOLICIT ≠ 0 implies that no action has been taken on requested unit and the request is reissued. When SOLICIT = 0, the AVRTBL:TPOS field is examined for a possible unit switch. If it is non-zero, the pointer residing in AVRTBL:TPOS is used to adjust DCB:DSI and the old AVRTBL slot is cleared. JB:PMTS or JB:TMTS is incremented for pack or tape respectively and MOUNT3 is given control to exit typr.

AVRCLEAR is given control to clear the AVRTBL slot and disassociates a user from that slot. First a flag is set for private packs to indicate that the operator aborted the job. If AVRID ≠ S:CUN, the AVR and user tables are not cleared and TYPR is exited. This can't happen for tapes, but may happen for shared packs. If AVRID = S:CUN, the the parallel AVR associated tables are cleared and TYPR1 is entered

If the user owns the pack (J:ASPIN ≠ 0 for this AVRTBL slot), the user and system counts are decremented (see DATA BASE). MOUNT3 resets R3 to the correct reel # or -1 scratch request and the exit is taken.

ID

MSROCTY - MSRTYPR

PURPOSE

To print or type a message to the operator that does not require a response.

USAGE

BAL, SR4	MSROCTY	INPUT R7 = ADR of PLIST
BAL, SR4	MSRTYPR	SR1 = 1 if PRINT option
		= 2 if TYPE option

DATA BASE

J:DCBLINK

M:OC - The DCB used by MSROCTY

M:UC -The DCB used by MSRTYPR for on-line jobs.

ROUTINES

CHKBIT1 - To check the presence of a buffer in the PLIST.

CKLIMIT - To check the access protection of a user specified buffer.

GETBTD - To get the message byte displacement from the DCB.

RESBTD - To restore the byte displacement to the DCB.

MSRRDWT -The routine to do the actual output.

DESCRIPTION

The DCB address is set M:UC if entered at MSRTYPR by an on-line job. Otherwise it is set to M:OC. The buffer is obtained via CHKBIT and CKLIMIT is called. PLIST is constructed in TSTACK and if print is specified, the DCB chain in JIT is searched for M:LL. The byte displacement is set and MSRRDWT outputs the message.

ID

MSRKEY

PURPOSE

To type a message on the OC device and read a response from the operator.

USAGE

BAL, SR4 MSRKEY INPUT R7 = ADR of PLIST

DATA BASE

MSRTYPR	- To output the message.
SETBTDQ1	- To set DCB:BTD to a one.
CKLIMIT	- To check the access protection of a user's buffer.
QUEUE	- To read the response.
KEYINEND	- Sets DCB:ARS into the user's buffer.

DESCRIPTION

MSRTYPR is called to type the message to the OC. The response DCB is set, depending on the type of user; the DCB and PLIST is set up for the read from the output PLIST with CKLIMIT being used for buffer checking. QUEUE is cal led to read the response and KEYINEND sets the first byte of the user's buffer to the size of the actual response.

ID

MTPV

PURPOSE

To mount a private volume and verify the serial number.

USAGE

```
LI, SR4        RETURN
B              MTPV
```

INPUT R6 = DCB:ADR

DCB:VNO=volume number of the volume to be mounted

DCB:BBUD = 0

OUTPUT

DCB:VNO =O, if job aborted because volume not mounted; if job aborted because volume belongs to another set; or if unable to mount the volume because unit unavailable.

DCB:PAT, VDCTX = 0 , if job aborted because volume not mounted.
DCB:PAT = 0, VDCTX = foreign volume number if job aborted because volume belongs to another set.

DCB:PAT = 49, VDCTX , 0, if unit unavailable.
AVR: SN = serial number of the requested volume.
AVR:AVR set.
AVR:PRIM set, if DCB:VNO = 1.
AVR:VER set while the serial number is being verified and the bit map is being moved to the allocation table. This field gets reset when AVR gets set.
DCB:BUF1 = address of 512 word buffer containing VTOC.
DCB:BBUD set, if VTOC modified.
DCB:BCDA = disk address of VTOC (in private volume format).

DATA BASE

AVRTBL - the reel # and flcgs are checked to obtain a unit.
SL:SP, JB:CSP - an operator abort must decrement these system and user counts.

OPERATION

The AVRTBL table is first searched to see if the requested volume is already mounted and verified. If so, MTPV exits. If not, the AVRTBL table is searched to see if the requested volume has been premounted by the operator. If so, the volume is verified, as described in the next paragraph. If not, the AVRTBL table is searched for (1) a free unit, (2) a unit with another volume that has been premounted, (3) a unit with a verified volume that has no users but is not a primary volume, or (4) a unit with a verified volume that has no users and is primary volume. If a unit is not located, MTPV exits with a status of "unit unavailable". If a unit is located, a MOUNT message is output to the operator. For cases 3 and 4, a DISMOUNT message is output prior to the MOUNT message. The system waits 20 seconds for the operator to respond with either a MOUNT or an ABORT keyin. If the operator aborts the job, MPTV exits with an abort code. If the operator responds with the requested MOUNT keyin, the volume is verified, as described in the next paragraph. Otherwise, the AVRTBL search begins again to see if the operator has mounted the volume on another unit. If not, another MOUNT message will be output.

A volume is verified by checking that the serial number on the volume matches the serial number specified for the unit and by moving the volume's cylinder bit map and NVAT table to the Allocation Table (HGP) entry associated with the unit. If the volume has not been initialized by the VOLINIT processor (and hence does not have a serial number), the message 'XXXX NOT INIT' is output to the operator and the MOUNT message 'REEL NO ERROR' is output to the operator and the MOUNT message is repeated. If the volume mounted is not the first volume to be mounted for the file, MTPV checks to see if the volume belongs to the set. If not, the message 'XXXX NOT IN SET' is output to the operator and the MOUNT message is repeated. This gives the operator a chance to either mount the correct volume (assuming that all volumes do not have unique serial numbers) or abort the job. If the volume mounted is taken from the secondary volume and used instead as the volume to be mounted.

(1) Is a tape drive allocated to this user?

If DCT4 indicates the existence of tape drives, S:CUN=AVRID indicates that this user has been allocated this drive and the AVRID is cleared. If the drive is not scratch, the AVRTBL doubleword is cleared and the dismount message is output. In either case, SL:9T or SL:7T is decremented and control is passed to check the disk pack allocation.

(2) Is a disk pack allocated to this user?

Every pack in the system is checked to see if the corresponding bit in J:ASPIN is set, ie. the user has been allocated a give drive. If J:ASPIN is set, AVRNOU is decremented and if it is then zero, this drive can be released. This consists of resetting S:OSPIN or S:BSPIN; clearing the AVRID (unless locked); and if this pack is shared, SL:SP must be decremented.

(3) Is this user associated with a batch partition?

PLB:USR is compared to S:CUN and if no match is found, the exit is taken. If the user is in a partition, the resources allocated to this user are cleared (see DATA BASE). SL:SP, SL:9T, etc. are decremented by the maximum values (JB:MSP, JB:M9T, etc.) from the user's JIT. Also, JB:SLNK must be followed with SL:SP being decremented for each serial number in the list (this is because this list represents disk packs that have been taken away from the system but not yet associated with this user). The exit is taken back to STEP.

ID

T:DSMNT

PURPOSE

Tape and disk pack drives which are allocated to the user are returned to the system.

ENTRY

BAL,11 T:DSMT - Entered by STEP during the LOGOFF procedure.

DATA BASE

DCT4 Determines if tape drives are present in the system.

S:CUN Determines if this user is the owner of a particular drive.

AVRTBL The flag bits must be cleared and the serial number is used for the dismount message.

ANSFLGS Cleared for a tape drive associated with this user.

AVRFNMT Contains the filename needed for the dismount message.

AVRNOU Decremented when a disk pack is released.

AVRID Identifies the owner of a pack or tape drive.

S:OSPIN, S:BSPIN Reset when a disk pack is released.

SL:SP, SL:9T, SL:7T, SL:C Decremented after the user has been disassociated.

PLB:USR Determines if this user is in a partition.

PLD:ACT, PLH, SID, PLH;FLG, PL:JIF, PLH:FLG Partition tables that are cleared when the user disassociates from a partition.

LSERIAL SL:SP + CURB is decremented for each drive linked to this user.

DESCRIPTION

A message buffer is set up in the user's TSTACK and then the following cases are examined.

(1) Is a tape drive allocated to this user?

(2) Is a disk pack allocated to this user?

(3) Is this user associated with a batch partition?

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

XEROX

Reader Comment Form

We would appreciate your comments and suggestions for improving this publication.

Publication No.	Rev. Letter	Title	Current Date

How did you use this publication?

□ Learning □ Installing □ Sales
□ Reference □ Maintaining □ Operating

Is the material presented effectively?

□ Fully Covered □ Well Illustrated □ Well Organized □ Clear

What is your overall rating of this publication?

□ Very Good □ Fair □ Very Poor
□ Good □ Poor

What is your occupation?

Your other comments may be entered here. Please be specific and give page, column, and line number references where applicable. To report errors, Please use the Xerox Software Improvement or Difficulty Report (1188) instead of this form.

Your Name & Return Address

Thank You For Your Interest. (fold & fasten as shown on back, no postage needed if mailed in U.S.A.)

Staple Staple

Fold

First Class
Permit No. 229
El Segundo,
California

BUSINESS REPLY MAIL

No postage stamp necessary if mailed in the United States

Postage will be paid by

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

Attn: Programming Publications

Fold