Y28-6609-1

Program Logic

IBM System/360 Operating System Input/Output Support (OPEN/CLOSE/EOV)

Program Logic Manual

Program Number 360S-DM-508

This publication describes the internal logic of IBM System/360 Operating System input/output support. The discussion includes the relation of I/O support routines to other portions of the control program. Detailed descriptions of the open, close, and EOV routines provide the basis for the discussions of the other I/O support routines openJ, RDJFCB, Tclose, and FEOV.

Program Logic Manuals are intended for use by IBM customer engineers involved in program maintenance, and by system programmers involved in altering the program design. Program logic information is not necessary for program operation and use; therefore, distribution of this manual is limited to persons with program maintenance or modification responsibilities.

PREFACE

This publication describes the functions and organization of the input/output (I/O) support portion of System/360 Operating System. It also describes the relationship of I/O support to other portions of the operating system.

The publication is divided into sections that describe each of the major components of I/O support. Each section refers to flowcharts that show the sequence in which the functions are performed. Appendixes are included to show the formats of work areas and tables used by I/O support.

PREREQUISITE PUBLICATIONS

The reader of this publication must be familiar with the concepts described in the following publications:

IBM System/360 Operating System: Concepts and Facilities, C28-6535

IBM System/360 Operating System: Data Management, C28-6537

IBM System/360 Operating System: Introduction to Control Program Logic, Program Logic Manual, Y28-6605

SUGGESTED READING

Information related to that provided in this publication is supplied throughout the following publications:

IBM System/360 Operating System: Control Program Services, C28-6541

IBM System/360 Operating System: Job Control Language, C28-6539

Second Edition (February 1967)

This publication is a major revision of Form Y28-6609-0 and obsoletes it. In addition to incorporating information released in Technical Newsletter Y28-2125, significant changes have been made to describe the password-protection feature in the open and EOV routines.

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or Technical Newsletters.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602

ILLUSTRATIONS

FIGURES

TABLES

CHARTS

The I/O support routines are nonresident SVC routines; they reside in the SVC library (SYS1.SVCLIB) on the system residence volume, and operate from the SVC transient area in main storage. Processing programs normally specify use of I/O support via a macro-instruction whose expansion includes an SVC instruction. Execution of this SVC instruction causes CPU control to be passed through the SVC interruption handler to the appropriate SVC routine. There are seven I/O support macro-instructions each having an SVC routine that performs the I/O support function. The I/O support macro-instructions, their associated SVC numbers, and the I/O support SVC routines are listed in Table 1.

Table 1. I/O Support Macro-Instructions, SVC Numbers and Routines

Macro-Instruction	SVC Number	SVC Routine
OPEN	19	Open
OPEN (TYPE=J)	22	OpenJ
RDJFCB	64	RDJFCB
CLOSE	20	Close
CLOSE (TYPE=T)	23	Tclose
EOV	55	EOV
FEOV	31	FEOV

All these routines except RDJFCB are type 4 SVC routines; RDJFCB is type 3. A discussion of the types of SVC routines is given in the publication IBM System/360 Operating System: Introduction to Control Program Logic, Program Logic Manual.

Because type 4 SVC routines are broken down into load modules of 1024 bytes or less, functions required by more than one I/O support routine are actually performed by common load modules. For example, the open and the openJ routines are separate SVC routines, but functions common to both are performed by the same load modules.

To save time, the user can open or close more than one DCB via a single macro-instruction. If an OPEN macro-instruction specifies three DCB's, the initial load module is executed three times (once for each DCB) before the next load module is executed. Before a load module is replaced, it is executed as many times as is needed to open the specified data control blocks.

Input/output (I/O) support routines perform three functions associated with I/O operations. These functions are:

- Opening a data control block, which is initialization required before a data set can be read or written.
- Closing a data control block, which is final processing on a data set after it has been read or written.
- Processing end-of-volume conditions, which is the processing required when an end-of-volume or end-of-data set condition occurs during an I/O operation.

Opening a Data Control Block

Before any information can be read from or written into a data set, the data control block for that data set must be opened. When a processing program issues an OPEN macro-instruction, the open routine of the control program performs the open processing.

Open processing consists of completing control blocks that contain the characteristics of the data set to be read or written, and of bringing into main storage the access method routines that will operate on this data set.

Closing a Data Control Block

After reading or writing a data set is completed, the processing program should issue a CLOSE macro-instruction to complete the processing of that data set. The close routine releases main storage that was acquired for the I/O operations when the data control block was opened. The close routine also performs final label processing for the data set, and sets indicators so that the data set is properly disposed of when the job step terminates.

Processing End-of-Volume Conditions

When an end-of-volume or end-of-data set condition occurs, the end-of-volume (EOV) routine processes labels and determines whether processing is to continue on some other volume or data set. When an EOV condition occurs, the load modules entered are part of the sequential access method (SAM) and are described in the publication IBM System /360 Operating System: Sequential Access Methods, Program Logic Manual, Y28-6604. These SAM modules are the first ones entered: one for EOV and the other for FEOV.

OPENING A DATA CONTROL BLOCK

The DCB associated with a data set must be opened before any transfer of data between main and auxiliary storage can occur. The data control block is created when the processing program is assembled, but it may not be completed at that time.

Opening includes completing the DCB from information in the job file control block (JFCB) and the data set label or DSCB, and selecting the executor routine that brings appropriate access method routines into main storage. Executors are load modules that are entered from an I/O support routine but perform functions for a specific access method. The operation of executors is described in the program logic manuals for the access methods.

The user may specify either of two routines to open a DCB: the open routine or the openJ routine. Both are type 4 SVC routines. The difference between the two is that the open routine reads the JFCB from the job queue, but the openJ routine moves the JFCB from the dynamic area of storage to the openJ work area. The JFCB must be in main storage before openJ receives control.

Opening a DCB using the open routine requires only the execution of an OPEN macro-instruction that does not have a TYPE specification of J. Execution of the OPEN macro-instruction causes an SVC interruption (SVC 19). The SVC interruption handler passes CPU control to the open routine.

The openJ routine receives control from the SVC interruption handler when an OPEN macro-instruction with a TYPE specification of J (SVC 22) is executed. However, opening a DCB using the openJ routine requires that the JFCB be in main storage prior to execution of openJ. When the JFCB is on auxiliary storage, it may be read into main storage using the RDJFCB macro-instruction. This allows the user to modify the JFCB before the DCB is opened.

THE OPEN ROUTINE

When an OPEN macro-instruction is executed, the open routine gains control from the SVC interruption handler after program fetch has read the first load module into the SVC transient area.

The first module, and all successive open modules, retain the address of the OPEN macro-instruction's parameter list in a general register. The parameter list specifies the DCB's that are to be opened. Also maintained in a general register is the address of the DCB that is being opened since each open load module processes each DCB to be opened before passing control to the next load module.

Throughout open processing, a DCB must have its LOCK bit off (off is 1 for the LOCK bit) and its BUSY bit on (on is 1 for the BUSY bit). These bits are in the DCBOFLGS field of the DCB. If a DCB specifies otherwise, it is not processed. The LOCK bit is set on by I/O support routines preceding a user exit; thus, other I/O support routines cannot process or change the DCB until the I/O support routine that set the bit regains control and resets it. The BUSY bit is set on by the open routine to indicate that the DCB is in the process of being opened. When the BUSY bit is not on for successive open functions, that DCB is not being opened and will not be processed. The BUSY bit is turned off and the OPEN bit turned on when all open functions have been performed and the DCB is considered open.

Charts 10 through 13 show the functions and operation of open. A symbolic name is indicated above the blocks that describe each function. These names, and names of the actual open load modules that contain the functions, make up a table that resides in the first load module of open. This table enables cross-referencing between the charts and the open load modules.

The four main functions that the open routine performs are:

- Basic initialization.
- Volume mounting and volume verification.
- Merging of control block information.
- Determination of access method routines.

The null or dummy data set is treated as a special case by the I/O support routines. No device is allocated to a null data set; thus, no volume positioning, label processing, or volume mounting is performed. The open routine recognizes a null data set by finding the characters NULLFILE in the DSNAME field in the JFCB associated with the DCB being opened. Information concerning the dummy data set is given in the publication IBM System/360 Operating System: Sequential Access Methods, Program Logic Manual.

BASIC INITIALIZATION

The open routine performs basic initialization functions for all DCB's in the parameter list. These functions are:

- Obtaining main storage for a work area for each DCB to be opened.
- Determining the size of the where-to-go (WTG) table.
- Obtaining storage for and setting up the WTG table.
- Reading the JFCB for each DCB to be opened.

Upon receiving control, the open routine inspects each DCB in the parameter list and counts each DCB as an entry for the variable section of the WTG table. The open routine turns on the BUSY bit in each DCB. (A DCB that is already OPEN at inspection, i.e., has its OPEN bit on, is not processed and is not "busy"; however, an entry for it is counted to maintain the parallel structure of the WTG table and the parameter list.)

For each DCB to be opened, open obtains main storage by use of the GETMAIN macro-instruction. This storage is used as a work area, the address of which is stored temporarily in the DEB address field of the DCB. After the WTG table has been built, the work area address is transferred from the DCB to the corresponding DCB entry in the WTG table. The work area is used for setting up control blocks and channel programs that are required for reading and writing header and trailer labels and DSCB's, and for tape positioning. (The work area for each DCB is described in Appendix A.)

After open has determined how many DCB's are in the parameter list, storage for the WTG table is obtained via a GETMAIN macro-instruction. The number of bytes requested is 32+8(n+1), where n is the number of DCB's specified in the parameter list. The total number of bytes required is in the WTG table size counter.

The WTG table is a communication area for the modules of open. Figure 1 illustrates the table, and Appendix C provides a detailed description. The table has two parts: a thirty-two byte basic section of standard format and a variable number (n+1) of eight-byte entries.

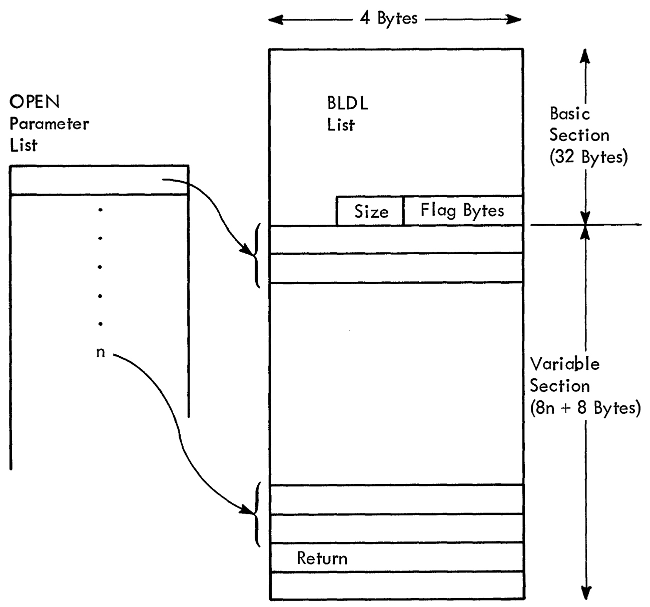


Figure 1. Where-To-Go (WTG) Table

The basic section of the table contains a twenty-nine byte list equivalent to that produced by the BLDL macro-instruction. This is followed by one byte which gives the double-word size of the entire WTG table, and by two bytes which serve as flag bytes (the WTGPATH).

The bits of the WTGPATH are set on or off to indicate the path through the open modules. Each open module determines the next open module by testing the bits of the WTGPATH. The path is controlled by the following:

- Device type (tape, direct-access, etc.) upon which the volume for the data set currently being initialized resides.
- Labeling characteristics.
- Type of processing required.
- Number of volumes specified by the JFCB. (If there are more than five volumes indicated, JFCB extensions must be read into main storage.)

Each eight-byte entry of the variable length section contains the IDTTR of the required access method executor in the first five bytes and the address of the work area associated with a particular DCB in the remaining three bytes. When a DCB is not being opened at this time, its entry contains binary zeros. The last eight-byte entry is always the IDTTR of the open load module which is to regain control from the access method executors after their processing is complete.

To read the JFCB from the job queue to the work area, the open routine must construct control blocks (DCB, DEB, ECB, and IOB) and a channel program within the work area. Thus, the work area contains the information necessary to read the JFCB by use of the EXCP macro-instruction.

If an I/O error occurs, the ABEND routine (SVC 13) is entered. (The ABEND routine is described in the publication _IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual_, Y28-6612.)

JFCB extension blocks are read into the work area when (1) there are more than five volume serial numbers and a user is opening for RDBACK, or (2) there are more than five volume serial numbers, MOD is specified and the user is opening for OUTPUT or OUTIN.[1]

Open obtains the address of each JFCB from the task input/output table (TIOT). The address obtained is a relative track address (TTR) and must be converted to the full device address (MBBCCHHR). The convert routine that performs this conversion is described in the publication _IBM System/360 Operating System: Sequential Access Methods, Program Logic Manual_.

VOLUME MOUNTING AND VERIFICATION

After basic initialization, open's next function is volume mounting. During mounting open determines whether required volumes are mounted on devices allocated to a data set. Open requests mounting of the required volumes and then checks them by examining volume labels.

The mount function is divided into general mounting for all DCB's to be opened and parallel mounting for DCB's which specify either the basic direct or indexed sequential access methods.

General Mounting

Open's actions during mounting depend initially on the type of volumes needed by the processing program. If specific volumes are needed, open uses the volume serial numbers specified in the data set's JFCB to check for correct volume mounting. (Serial numbers are required when INPUT, INOUT, or RDBACK is specified.)

Open performs general mounting only for the first volume of a data set to be processed. If the user's parameters specify RDBACK or if MOD is specified in the DD statement with an OPEN for OUTPUT or OUTIN, the first volume to be processed will have its volume serial number specified last in the JFCB. Otherwise, the volume needed is the first one indicated by the JFCB.

When deferred mounting is specified, the job scheduler performs no mounting for the data set during job step initiation. The open routine issues the initial mounting messages. Deferred mounting is described in _IBM System/360 Operating System: Concepts and Facilities_.

Whether mounting has been deferred or not, open determines whether the correct volumes are mounted. If correct mounting is not found, open issues mount messages and rechecks for correct mounting.

After open has determined that mounting is correct, the routine locates the proper data set on the volume. For tape, open positions the volume; for direct-access, open locates the data set control block (DSCB) and reads it into main storage.

During mounting, open uses the SRTEDMCT field of the unit control blocks (UCB) associated with a data set. The high order bit of this one-byte field serves as a mount switch and is set on when a mount message is issued concerning the unit. When the switch is on, no further mount messages are issued for that unit until the mounted volume has been checked. The switch is reset to 0 when the proper volume has been mounted.

The low-order seven bits (the data management count) are the binary number of DCB's which are open for data sets on the mounted volume. For direct access, the count is incremented by 1 when open ascertains that the mounted volume is the one specified in the JFCB. For tape, an attempt to begin processing a second data set on a volume results in abnormal termination of the job step. Thus, data management counts for tape devices will be either 0 or 1.

DEFERRED MOUNTING: Open checks all the UCB's allocated to the data set for the serial number of the first volume to be processed. When the volume serial number is in the UCB, the job scheduler has initiated mounting, and only volume checking is required. When the serial number is not found and the mount switches in the allocated UCB's are off, the user has requested deferred mounting. Open checks these same UCB's for one whose data management count is 0 and which does not specify a volume that is reserved, public, permanently resident, or used for system residence. Should none of the allocated de-

----
[1]All JFCB's and DSCB's are read for concatenated BPAM data sets.

vices be available, the job step is abnormally terminated.

When open has found a suitable unit, the routine issues a mount message to the operator designating the volume serial number specified in the JFCB. When the JFCB does not indicate a volume serial number (e.g., for OUTPUT or OUTIN data sets), open's mount message specifies a scratch volume.

Open sets the mount switch in the associated UCB to indicate that a mount has been requested and that volume checking is required.

VOLUME VERIFICATION: Whether a mount message has been issued by open or by the job scheduler, open checks the mounted volume. When volume serial numbers are required, the serial number from the JFCB is compared with the serial number from the label. When job scheduler has initiated the mount, the volume serial number is found in the UCB.[1] When the mount was initiated by open, open must read in the volume label to obtain the serial number. When the volume has been recognized as the correct volume, open inserts the volume serial number into the UCB. For tape volumes, open also inserts the file sequence number into the UCB.

Volume checking has the following variations:

- For volumes with standard labels, open obtains the volume serial number from the label. If this number is the same as the number specified in the JFCB, the required volume has been mounted. Open sets the mount switch off and increments the data management count by one. When the two serial numbers are not the same, open reissues a mount message to the operator and checks the new volume.

- For volumes with nonstandard labels, open ascertains that standard labels do not exist for the volume. Open passes control to the user's nonstandard label processing routines. When the user returns control, he indicates whether the volume is correct. If the volume is not correct, open reissues mounting messages and again asks for checking by the user routines. After the correct volume is mounted, open sets the mount switch off and increments the data management count.

- When unlabeled volumes are specified, open ascertains that standard labels do not exist for the volume. The mount switch is set off, and the data management count is incremented.

When OUTPUT or OUTIN tape volumes are specified, the open routine checks the mounted volume for the specified density (when dual density devices are used) and label characteristics. When the label type and/or density of the volume does not agree with the user's specifications, control is transferred to a label editor module. The user may either utilize the IBM-supplied routine or supply an editor module.[2] The standard routine requests replacement of the current volume with a volume containing the specified label type and sets the mount switch on. Control is returned to the mount-verification module for volume checking.

When an OUTPUT, OUTIN, or INOUT tape volume has been verified, the open routine reads the sense bytes to check for file protection. When the sense bytes do not show file protection, the open routine proceeds to its next function. When the volume is protected and the mode is OUTPUT or OUTIN, the routine issues a message to the operator requesting the insertion of a file protect ring. When the mode is INOUT, the message issued gives the operator the option of inserting a protect ring.

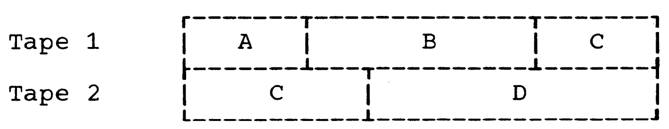
TAPE POSITIONING: After tape volumes have been verified, the open routine positions the volumes at the data set specified by the user and, when possible, verifies data set labels.[1] (The user's nonstandard label routines position tapes before returning control to the mount-verification module.) Open positions tape using the data set sequence numbers supplied by the user. These sequence numbers are available as counters in the UCB -- SRTEFSCT and SRTEFSEQ. The physical sequence (SRTEFSCT) is the relative position of the data set on its resident volume. The logical sequence (SRTEFSEQ) is the sequence of a data set in a group of related data sets. When the data sets are contained on a single volume, these two numbers are equal. For multi-volume, multi-data set aggregates, they may differ.

Figure 2 shows the physical and logical sequences of the data sets residing on a multi-volume, multi-data set aggregate. This aggregate consists of four data sets contained on two tape volumes.

---

[1]When SYSIN or SYSOUT is specified, the volumes are not processed by open except for incrementing the data management count.

[2]Information on writing volume label editor routines is given in the publication <u>IBM System/360 Operating System: System Programmer's Guide</u>.



Data Set	Logical Sequence	Tape 1 -- Physical Sequence	Tape 2 -- Physical Sequence
A	1	1	-
B	2	2	-
C	3	3	1
D	4	-	2

Figure 2. Multi-Volume, Multi-Data Set Aggregate

Unlabeled Tape Positioning: The open routine positions unlabeled tapes by comparing the physical sequence number in the UCB to the logical sequence number in the JFCB. For output data sets if the physical sequence number does not equal the logical sequence number, the data set sequence number in the JFCB is replaced by the physical sequence number (this is required only for multi-volume, multi-data set aggregates).

For RDBACK, the tape is positioned as above and in addition, for:

- One volume - forward space files followed by a backward space file is issued to position the tape at the end of the appropriate data set.
- More than one volume - the last volume specified in the JFCB is positioned as if the data set sequence number in the JFCB were a 1. The sequence number from the JFCB replaces the logical sequence number in the UCB.

Input Standard Labeled Tape Processing: Labeled tape positioning is the same as for unlabeled except that the HDR1 (Data Set Label 1) is read and the sequence number from the label is inserted into the UCB. This number is checked against the number in the JFCB.

After tape positioning, the tape is left positioned in front of the HDR1. However, for RDBACK and for OUTPUT with MOD specified, the tape is left in front of the tape mark preceding the trailer label.

For all DCB's in the parameter list with mounted tape volumes that have standard labels, the open routine determines whether the labels specify the correct data set name. (For open to verify labels, the OPEN parameters must specify INPUT, INOUT, or RDBACK.) Open then fills in the zero fields in the associated JFCB's with fields specified in the labels; open also positions the tapes at the first data set record.

To verify that the tape is correct for the data set, open inspects HDR1. Open uses the least significant nonblank characters (with a maximum of 17) of HDR1 to determine the data set name. The label data set name is then compared with the data set name in the JFCB. If the names do not agree, control is passed to the ABEND routine. When they do agree, HDR2 is read into the work area.

(If the DCB is being opened for INOUT, the retention date is checked to make certain that a current data set is not destroyed. If the date has passed, open processing continues. If not, a message is written to the operator, and if the tape is not verified by him, the task is terminated.)

After the open routine verifies the data set, open checks HDR1 for specification of a password-protected data set. When the data set is protected, control is passed to the open security module. This routine establishes its work area and then searches the SYSRES VTOC for the DSCB of the password data set. If the search is unsuccessful, an abnormal job step termination is requested. When the password DSCB is found, the routine initializes a counter used to limit to two the number of attempts to obtain the correct password from the operator; it then passes control to the password reader routine. (The latter may be installation supplied.)

The password reader increments the counter by one and then prepares the operator message to request the data set password. The message identifies the data set by giving the job and step names and the DDNAME. These names are obtained from the data set TIOT entry. When the routine has issued the message, it returns control to the security module.

The security module searches the password data set for the one supplied by the operator. When the correct password is submitted, the security module reads in the data portion of the password entry. The routine compares the mode byte of the entry to the specified method of opening for the data set. When modes agree, the user count from the data portion of the password entry is incremented by one. If the modes are not the same, the job step is abnormally terminated.

When the password submitted by the operator is not found, the open security routine determines whether one or two attempts have been made. When one attempt was made, the security module passes control to the password reader routine to initiate a second request. When the operator has given two incorrect passwords, the job step is abnormally terminated.

When the open security routine has successfully verified all security-protected data sets to be opened, it releases the work area and passes control to the next open module.

HDR2 follows HDR1 and contains data set characteristics. This header is merged with the JFCB. The zero fields in the JFCB that may be filled in by fields from the HDR2 are converted from the BCD (7 track) or EBCDIC (9 track) form of the HDR2 to the binary form of the JFCB. The tape is then positioned (forward space file) to the first data record (if RDBACK is specified, a backspace file operation is performed.)

Output Standard Labeled Tape Processing: When the tape is positioned to receive a new data set, the open routine checks the tape for a header label. If no label is found, one is created for the data set. If a header label is found (indicating that a data set is already on that part of the tape), it is checked to determine whether it may be overlaid. The open routine checks for unexpired and for security-protected data sets. If the expiration date of the data set on the volume has not passed, the open routine issues a message to the operator. If the operator replies that the tape is not to be used, he may mount a scratch volume.

When the expiration date has passed, the open routine checks for data set security. If the header label indicates security, a check is made for the correct data set name as for input tape, and then control is passed to the security routine. If the correct password is given, the new label is prepared.

The new header label is constructed from the information in the JFCB. The HDR1 and HDR2 fields are determined, and where necessary, binary fields specified in the JFCB are translated to the character forms of the label. A tape mark is written following the label.

DIRECT-ACCESS VOLUME SEARCHING: When a direct-access volume is mounted, the volume label for that volume points to the volume table of contents (VTOC) which contains a DSCB for each of the data sets on that volume. To locate the correct DSCB's for the data sets associated with the DCB's in the parameter list, open searches on key equal with DSNAME from the JFCB. When the search is equal, the correct DSCB for the data set is found. Open then reads the data portion of the DSCB (96 bytes). If the DSCB is not successfully read, the job step is terminated.

When the DCB specifies that the data set is not to be NEW, the expiration date in the DSCB is checked; if the date has not expired, a message is transmitted to the operator. If the operator indicates that the data set cannot be modified, the job step is terminated so current data sets will not be destroyed.

If the DSCB (format 1 block) specifies security, the open security module receives control. The security routine operates as for tape except that the specification of BPAM concatenation requires checking each member that is a security-protected data set.

Parallel Mounting

Parallel mounting is similar to general mounting but handles the specific requirements of processing programs using ISAM or BDAM. These access methods require that all volumes of a data set be mounted concurrently.

During job step initiation, the job scheduler will initiate mounting not only of the first volume of a data set but also of all the volumes required for ISAM or BDAM. However, the scheduler will only allocate units to handle all the volumes if the processing program specifies deferred mounting.

For parallel mounting, open checks all DCB's for those that specify use of ISAM or BDAM. When one is recognized, open examines the associated data set's task I/O table (TIOT) entry. Since the TIOT lists all the UCB's for devices which have been allocated to a data set, open can determine whether multiple volumes are required by a DCB's associated data set.

When the TIOT does not indicate more than one UCB, no parallel mounting takes place since the first volume of every data set has been checked for correct mounting by the general mount.

For those DCB's requiring parallel mounting, open uses the serial numbers given in the JFCB. Since a data set may reside on more than five volumes, open may require JFCB extension blocks to obtain the complete list of serial numbers. Open obtains the extension block from auxiliary storage.

To perform I/O, the parallel mount uses its own GETMAIN areas: one large enough to receive a JFCB or a JFCB extension block, the other large enough to receive a volume label.

In most respects, the parallel mount procedure is the same for BDAM and ISAM. The difference arises from the possibility of using more than one DD statement in defining an indexed sequential data set.

<u>BDAM Parallel Mounting:</u> When the examination of the TIOT indicates additional UCB's for a data set, open checks the associated JFCB for the next volume serial number. When five serial numbers have been obtained from a JFCB, it is necessary to read the extension block into the GETMAIN area.

When the serial number has been obtained from the JFCB or from the extension block, open ascertains that the number is nonblank. The recognition of a blank serial number field results in abnormal termination of the job step.

Open examines the UCB for the presence of the volume serial number. When the number is in the UCB, the volume has been mounted by the job scheduler. Open need only increment the data management count and set the mount switch off. When the volume is not mounted, open sets the mount switch on and issues a mount message. The open routine reads the volume label into the GETMAIN area for checking.

Volume verification is the same as for the general mounting. When the correct volume has been mounted, open gets the volume table of contents (VTOC) address from the label. The address is converted to the relative track form. The converted VTOC address and the volume serial number are placed into the UCB.

Open sets the mount switch off and increments the data management count as in the general mounting. Unlike the general mount when this would complete mounting for a DCB, the parallel mount must check the TIOT for specification of further UCB's for this data set. Only when the TIOT entry lists no other UCB's does parallel mounting for another DCB begin.

<u>ISAM Parallel Mounting:</u> The parallel mount procedure for ISAM is the same as for BDAM with one exception. When open examines the TIOT entry for a data set, the specification of no additional UCB's does not mean that all the volumes of the data set are mounted. Since the data set may require more than one DD statement to define it, there may be more than one TIOT entry for the data set.

When this occurs, the additional TIOT entries have their DDNAME fields blank, as for concatenation. Therefore, when the TIOT entry indicates no additional UCB's, open examines the next entry. If the next entry has a blank DDNAME field, there is another JFCB for the data set. Open reads the associated JFCB into the area. From this point, the procedure is the same as for BDAM.

When the examination of the next TIOT entry does not result in a blank DDNAME field, all the volumes associated with the data set have been mounted. Open frees any GETMAIN areas that were acquired and continues parallel mounting for any other DCB's that specify ISAM or BDAM.

<u>Reading Additional DSCB's:</u> After all volumes have been mounted for BDAM and ISAM, open reads in the associated DSCB's. Open uses the VTOC address from the UCB specified in the data set's TIOT entry to address the DSCB. In main storage, open sequentially chains a data set's DSCB's, beginning with the DSCB of the first volume to be processed. The open routine read the first DSCB following the general mounting.

For ISAM, the possibility exists of duplicating the reading of DSCB's since the same volume may be specified in more than one TIOT entry for a data set. To avoid this, open checks all format 1 blocks already read for one with a UCB pointer equal to the UCB pointer of the present DSCB. When the routine recognizes equal pointers, it proceeds to the next volume if one exists. The open routine places the UCB pointer in the suballocation field of format 1 blocks. (ISAM allocation does not use this field.) Open also checks for a format 2 block on the first volume of the data set. If this block is absent, the job step is abnormally terminated.

For BDAM, the open routine places the number of extents of the data set (excluding the first volume) into the DCB. The BDAM executor uses this count to determine when it has constructed all extents in the DEB.

Open determines whether the processing program has specified the same volume sequence as the sequence of the original allocation. The sequence is correct when successive sequence numbers in the TIOT are in ascending order. When the sequences do not agree, the job step is abnormally terminated. The open routine makes this check for both BDAM and ISAM.

MERGING OF CONTROL BLOCK INFORMATION

The number of completed fields in the DCB before it is opened varies with the type and number of parameters specified in the DCB macro-instruction. At execution time, additional attributes may be introduced into the DCB from the DD statement. Job management routines place these attributes into the JFCB, and the open routines transfer them to the DCB. The open routines use a merging process.

The user may add to or modify information in the DCB during opening by including an active DCB exit in his exit list. (The DCB exit is described in IBM System/360 Operating System: Data Management.) When requested, open takes this user exit after the information merge to the DCB is complete.

There are two types of merging with respect to the DCB:

- Forward; merging information from the DSCB or data set label to the JFCB to the DCB.
- Reverse; merging information from the DCB to the JFCB to the DSCB.

Forward Merge

The forward merge from the data set label to the JFCB completes zero fields in the JFCB.

The merge is the same for tape labels as for DSCB's except that different JFCB fields may be filled from the two label types. From DSCB's, the following JFCB fields may be completed:

- JFCRECFM - record format.
- JFCOPTCD - option codes.
- JFCKEYLE - direct access key length.
- JFCDSORG - data set organization.
- JFCBLKSI - block size.
- JFCLRECL - logical record length.

For tape, the merge may complete these fields:

- JFCRECFM - record format, carriage control character, and machine code.
- JFCLRECL - logical record length.
- JFCTRTCH - tape recording technique.

The forward merge from the JFCB to the DCB takes place for all DCB's to be opened. This merge places information from JFCB fields into corresponding DCB fields that are zero. The fields which may be merged are listed in the three merge tables. These are:

- DCB merge table, which contains displacements for the fields in the DCB that are to be merged.
- JFCB merge table, which contains displacements for the fields in the JFCB that are to be merged.
- Field length table, which contains the lengths of the fields to be merged.

Each table contains an access method dependent section, which contains the DCB fields that are present only for a particular access method and a section that contains DCB fields that are always present, regardless of the access method. The access method executor to receive control from open is determined at this time and indicated in the WTG table.

During the JFCB to DCB merge, modification of a field is noted by setting the field's associated bit. The set of DCB modification bits make up a mask that is placed in the DEB after its construction. The close routine uses the mask to reset the DCB to its pre-open status.

Reverse Merge

In the reverse merge from DCB to JFCB for output, the DCB fields override existing JFCB fields except the DSORG field. These JFCB fields specified by the merge tables (see JFCB to DCB merge) are made equal to corresponding DCB fields. For an output DSORG field and for input, the merge only occurs when the JFCB fields were previously zero.

The reverse merge from JFCB to DSCB takes place only for DCB's specifying direct access output. DSCB fields except the DSORG field (see DSCB to JFCB merge) are made equal to corresponding JFCB fields. Already existing fields are overridden. The DSORG field in the DSCB is made equal only if it has been previously zero. When the JFCB to DSCB merge occurs, an indicator is set to show that the DSCB has been modified.

ACCESS METHOD DETERMINATION

During the merge process, open uses the DSORG and MACR fields to determine the type of DCB being opened. From this, open ascertains which access method executors are required to process the DCB. Open finds the addresses of the executors in the XCTL table. Appendix B gives information about XCTL tables.

Open transfers the IDTTR and load length of the required executor from the XCTL table to the associated DCB's entry in the WTG table. The IDTTRs are placed in the table only for those DCB's being opened at this time. Otherwise, zeros are entered in place of the executor IDTTR.

To pass control to an access method executor, open moves the IDTTR field of the first nonzero entry in the variable section to the appropriate fields in the standard section of the WTG table. While the executor has control, it determines whether another executor is needed to continue processing. If another executor is required, the executor overlays the IDTTR in the associated entry in the variable section with the IDTTR of the required executor. When no other executor is needed for a DCB, the ID of its entry is set to zero.

When an executor has overlaid or zeroed its ID, it examines the immediately following DCB's entry to ascertain whether that entry's ID is equal to its own ID. If they are equal, the same executor must be entered for the next DCB. Therefore, the executor branches to its starting address. If the ID's are unequal, the executor determines whether the new ID is the last in the table by comparing it with the ID of the final open load module. If table end has not been reached, the executor continues its examination of following entries, until table end is recognized.

When the end of table has been reached, the table pointer is reinitialized to the first entry, a search is made for the first nonzero entry, and control is passed to the indicated module. Open progresses through all entries in the WTG table in this manner until the only nonzero entry is the final open module. Control is then passed to the final open module.

If the DSCB has been modified, as indicated by the bit set during the merging, it is written back onto its associated volume in modified form. If not modified, no rewrite is necessary. When the open functions are complete, the WTG table and the work area are no longer necessary, so the FREEMAIN macro-instruction is issued.

The DCB's are indicated as open; the OPEN bit is turned on and the BUSY bit is turned off. An SVC 03 (EXIT) is issued to return CPU control to the supervisor.

THE OPENJ ROUTINE

The openJ routine receives control from the SVC interruption handler after the OPEN macro-instruction with a TYPE specification of J (SVC 22) is issued.

The openJ routine, with one exception, operates and has the same functions as the open routine. (Refer to Charts 10 through 13.) The exception is that the JFCB is read from the job queue in the open routine, but must be in main storage when the openJ routine is entered. The openJ routine moves the JFCB into the openJ work area.

To locate the JFCB, the openJ routine checks for a DCB foundation extension that specifies the user's exit list. The exit list should contain an active exit (hexadecimal 07) that indicates the address of the JFCB in the dynamic area. If the extension, exit list, or JFCB address is not present, the ABEND routine is entered.

THE RDJFCB ROUTINE

The RDJFCB routine receives control from the SVC interruption handler when the RDJFCB macro-instruction (SVC 64) is issued. The RDJFCB routine reads the JFCB's associated with the DCB's in the parameter list into the dynamic area. (No JFCB extension blocks can be read.) Execution of the RDJFCB load module is repeated as required for each DCB in the parameter list before control is returned to the interruption handler. The RDJFCB routine, therefore, maintains the address of the current DCB being processed as well as the starting address of the parameter list.

The RDJFCB routine operates with opened and unopened DCB's. No processing is performed on an unopened DCB if the DDNAME in the TIOT does not match the DDNAME field in the DCB.

Chart 14 shows the functions and operation of the RDJFCB routine. A symbolic name is indicated above the blocks that describe each function. These names, and names of the RDJFCB load module are contained in a table in the listing of the RDJFCB routine. This table enables cross referencing between the charts and the listing. Since the RDJFCB routine is a type 3 SVC routine, it has only one load module.

The RDJFCB routine first inspects the LOCK bit in the DCB. The LOCK bit, if set to 0, indicates that another I/O support routine is currently processing that DCB and the DCB should not be altered. If the LOCK bit is zero, the user regains control with no processing on that DCB. The next DCB is inspected.

An error exit is taken to the ABEND routine if the user has not specified an active exit by means of the hexadecimal value 07 in the high order byte of an entry in his exit list. The exit list is addressed by the DCB to be processed. The entry indicates the address in problem program storage into which the JFCB is to be read. If the address is invalid, the ABEND routine is entered.

When the user has specified an active exit and the problem program address is valid, the RDJFCB routine obtains a work area (see Appendix A) by means of GETMAIN macro-instruction, sets up the work area to receive the JFCB for the DCB, constructs the control blocks necessary to read the JFCB by means of the EXCP macro-instruction, and constructs the channel program.

The convert routine is used to change the address of the JFCB from the relative track address (TTR) to the actual track address (MBBCCHHR). This routine is discussed in the publication IBM System/360 Operating System: Sequential Access Methods, Program Logic Manual.

The RDJFCB routine issues the EXCP macro-instruction to read a JFCB. The JFCB is read into the work area, and, if there are no I/O errors, the RDJFCB routine moves it to the dynamic storage area specified in the exit list. I/O errors cause the job step to be terminated.

The main storage into which the JFCB is read is released by means of the FREEMAIN macro-instruction, and the next DCB to be processed is obtained, and the processing is repeated. When all of the required JFCB's are read, the RDJFCB routine returns control to the supervisor.

CLOSING A DATA CONTROL BLOCK

The DCB associated with a data set must be closed after completion of all data transfer operations between main storage and auxiliary storage. The control program will pass control to the close routine if a DCB is still open when a task is terminated.

Closing a DCB includes restoring the DCB to its original condition, processing labels, determining volume disposition, removing the DEB from the DEB chain, and releasing access method subroutines.

Closing a DCB requires the execution of a CLOSE macro-instruction that does not have a type specification of T. Execution of the CLOSE macro-instruction causes an SVC interruption (SVC 20). The SVC interruption handler passes CPU control to the close routine.

A DCB can also be temporarily closed if the basic sequential access method is used. Temporary closing requires execution of a CLOSE macro-instruction that has a type specification of T. This macro-instruction causes an SVC 23 interruption. The SVC interruption handler passes control to the Tclose routine.

The Tclose routine differs from the close routine in that Tclose does not restore the DCB or release the DEB or subroutines. Tclose performs only label processing and repositioning. Therefore, when the Tclose routine closes a DCB, data set processing can be resumed without re-opening this DCB.

THE CLOSE ROUTINE

The close routine is a type 4 SVC routine. It is entered from the SVC interruption handler after the issuance of a CLOSE macro-instruction (SVC 20).

The first module to be executed gains control from program fetch after fetch has read it into the SVC transient area. All subsequent modules are loaded and gain control by means of the XCTL macro-instruction.

The close routine maintains the address of the parameter list of DCB's that are to be closed and the address of the DCB being closed. This facilitates the procedure of executing one load module for every DCB requiring it before passing control to another module.

In order for a DCB to be closed, it must have its LOCK bit off (off is 1 for the LOCK bit) and its OPEN bit on (on is 1 for the OPEN bit). These bits are within the DCBOFLGS field. The LOCK bit, when off, indicates that no other I/O support routine has begun operations upon that DCB. If the LOCK bit is on, the close routine does no processing upon that DCB and goes on to the next DCB. The OPEN bit, when on, indicates that the DCB has been opened and may be closed. If the OPEN bit is off, the close routine goes on to the next DCB.

Upon entry into the first module of close, the BUSY bit is set on (to 1) to indicate that the DCB is in the process of being closed. This bit remains on until the DCB is closed.

When the close routine encounters a condition that requires abnormal termination of a task, the routine first checks the ABEND bit in the TCB to determine whether the task is already undergoing termination. If the bit is on, the close routine does not issue the ABEND macro-instruction, but ignores the error condition and continues closing the DCB. If the ABEND bit is off, the close routine passes control to the ABEND routine for abnormal termination of the task.

This logic gives the ABEND routine the facility to close each DCB related to a TCB through a DEB, when the associated task is being abnormally terminated. The ABEND routine sets the ABEND bit to indicate to the close routine that the error condition should be ignored and that the DCB should be closed as normally as possible.

Charts 20 and 21 show the functions and operation of close. A symbolic name is indicated above the blocks that describe each function. These names and the names of the actual close load modules that perform the functions are contained in a table in the first load module of close. This table enables cross-referencing between the charts and the listings of the close load modules.

BASIC INITIALIZATION

The close routine, like the open routine, uses main storage for a work area and for a WTG table. A work area is obtained for each DCB in the parameter list. The size of the WTG table is determined (basic section and number of entries for the variable section). Storage for the work area and WTG table is acquired via a GETMAIN macro-instruction.

During initialization, the close routine:

* Ascertains task-data set relationship.
* Purges queued and active I/O requests associated with data sets that are being operated upon on the EXCP level.
* Constructs control blocks for reading JFCB's and DSCB's into the work areas.
* Determines and indicates on the WTG table (WTGPATH bytes) the access method executors and close modules necessary for closing each DCB.

Task-Data Set Relationship: To prevent a problem program from closing a data set that was opened under a different task from itself, the close routine checks family ID's. The family ID of the task being processed must be the same as the family ID in the DEB addressed by the DCB that is to be closed. If the family ID's do not agree, close does not process the DCB unless the ABEND routine has issued the CLOSE.

Purge: When a data set is closed, no I/O requests may be executed for it. Thus, when the close routine gets control, the queued I/O requests, and any that are actively being executed, are purged. The close routine issues the PURGE macro-instruction (SVC 16) for each DCB to remove I/O requests for data sets that are operated upon on the EXCP level. The purge routine is described in the publication IBM System/360 Operating System: Input/Output Supervisor, Program Logic Manual, Form Y28-6616.

Control Blocks: In the work area, the close routine constructs the control blocks necessary to perform I/O operations.

WTGPATH: The close routine determines the device type, whether a JFCB or a DSCB should be read for each DCB, the access method executors necessary for the DCB if the access methods are used, and if the device is tape, whether it has standard or nonstandard labels or is unlabeled.

The WTGPATH bytes in the WTG table are set, as in the open routine, to indicate the following:

* Tape or direct-access output.
* Output tape trailer label preparation.
* Tape positioning.
* Direct-access output and disposition.

The close routine determines from the XCTL table the ID and TTR of the access method executors required for closing each DCB. Close loads them into the variable section of the WTG table.

OUTPUT LABEL PROCESSING

When a write was the last operation that occurred before closing (for OUTPUT, OUTIN, or INOUT), the close routine processes data set labels. For tape, the trailer labels are constructed from information contained in the JFCB's. For direct-access devices, the DSCB's are read, and the last-block-written field and the track-balance field are updated.

Tape

The close routine indicated during previous initialization that a JFCB is necessary for the DCB's specifying tape output data sets. Close constructs the channel programs and reads the JFCB's.

The convert routine is used to convert the relative address of the required JFCB for each DCB to the actual track address of MBBCCHHR. The control blocks were previously constructed so the EXCP macro-instruction need only to be issued for reading the JFCB into main storage. If an I/O error occurs during the reading of a JFCB, the task is abnormally terminated unless the ABEND bit of the TCB is on. If operation is not on an EXCP level, the access method executors are executed. Upon return to close, trailer labels are constructed.

Before construction of a trailer label, the close routine inspects the WRITE bit within the DCBOFLGS field. If off, no output has been written and a trailer is unnecessary. When on, the close routine writes a tape mark after the data. Then close constructs trailer labels, EOF1 and EOF2, from the information in the JFCB, the DCB, TIOT, and UCB, and writes these labels on the tape. After labels are written for each DCB specifying standard output tapes, nonstandard label routines are entered if required by other DCB's.

Direct-Access

The close routine reads the DSCB by means of the EXCP macro-instruction using the DSCB address which is saved in the DEB prefix section by the open routine. When the DSCB is in the work area, the last-block-written field and track-balance field are updated.

For DCB's that do not have the WRITE bit set, the close routine need not update the

DSCB, ncr does it update the DSCB when the last block written is not in a sequential or partitioned organization data set.

The full device address is converted by means of the convert routine, and the relative device address is placed into the DSCB last block written field.

When the DEB (DEBOFIGS field) specifies that unused external storage is tc be released and the WRITE bit is on in the DCB (storage is not released when the ABEND bit is cn), control is passed to the DADSM release routine.

The release routine updates the DSCB's of both the data set and the available storage on the volume to show that the unused tracks are no longer assigned tc the data set. The updated DSCB's are then written back into the VTOC.

The close routine writes a file mark after each output data set that specifies physical sequential data set organization and READ/WRITE, GET/PUT, or EXCP macro-instruction reference. A data set that is not terminated by a file mark because of extent limits reaches end-of-data on input by coming to end-of-extent. Control is passed to the ABEND routine if a permanent I/O error is encountered while writing a file mark.

The file mark is written on the next track of the data set's extent. For fixed standard records, a file mark is also written in the present track (if another record will fit). A file mark is written on the first track if no WRITE was issued.

VOLUME DISPOSITION

For tape or direct-access volumes, parameters in the CLOSE macro-instruction may indicate the volume positioning required after closing. If LEAVE or REREAD is not specified in the macro-instruction, the close routine examines the TIOT for the KEEP or DELETE dispositions provided in the DD control statement. If no disposition is specified in either the macro-instruction or the DD statement, the close routine assumes the LEAVE disposition.

For an output tape data set, the WRITE (trailer switch) bit is on, indicating that a write has taken place and a tape mark must be written. The close routine writes two tape marks and then positions the tape according to the disposition specified.

If the LEAVE disposition is specified, the close routine positions the current volumes after the file mark if unlabeled, or after the file mark following the trailer label if labeled. The close routine then increments the logical and sequential data set sequence numbers in the UCB by 1. If the REREAD disposition is specified, the close routine positions the current volume tc process the data set again. No tape or volume positioning is performed for SYSIN, SYSOUT or for null data sets.

If the KEEP disposition is specified and the volume:

- is private,
- is nct permanently resident,
- is nct used for SYSRES, and
- has its user and data management counts equal to one,

close issues a message to the operator to dismount and keep the volume. To effect the KEEP disposition, a rewind unload command is issued for a tape volume; for direct-access, the not-ready bit is set in the UCB. If the DELETE disposition is specified, a rewind command is issued for a tape device. If DELETE is specified for direct-access, no action is taken.

DATA CONTROL BLOCK RESTORATION

To restore the DCB, the close routine uses two tables: a table of DCB displacements and a table of DCB field lengths. By using a mask in the DEB which is set by the open routine, close zeros the fields that were merged to the DCB from the JFCB.

Termination

After restoring the DCB, the close routine releases the main storage used for subroutines, appendages, the DEB, and the work area. The DEB is removed from the DEB chain; remaining DEB's are rechained. The data management count in the UCB is decremented by one for each DCB which is being closed for tape or direct-access. Close then checks for concatenation of data sets with unlike attributes. In this case, the close routine transfers control to the open routine. Otherwise, the close routine returns control to the supervisor.

THE TCLOSE ROUTINE

Chart 22 shows the operation and functions performed by the Tclose routine. A symbolic name is indicated above the blocks that describe each function. These names and the names of the actual Tclose load modules that perform the functions are contained in a table in the first load module of Tclose. This table enables cross-referencing between the charts and the listings of the Tclose load modules.

The Tclose routine provides volume positioning. It assumes the LEAVE disposition if no disposition is specified in the macro-instruction (without checking the TIOT).

For direct access, the Tclose routine resets pointers in the DCB either after the last data record or before the first.

Tclose also processes labels if required.

The Tclose routine differs from the close since it does not restore the DCB or release any main storage other than that acquired for its work area.

END-OF-VOLUME PROCESSING

The end-of-volume (EOV) routine processes end-of-volume and end-of-data set conditions for data sets having sequential organization. This routine is entered when one of the following conditions occurs:

- Tape mark read on tape.
- File mark read on direct-access device.
- End of last extent recognized on direct-access volume.
- End of file indicated after last record on unit record equipment.
- End of reel encountered.
- FEOV macro-instruction issued.

The EOV routine receives control via an SVC instruction. EOV performs final processing on the data set labels on the volume and specifies additional volumes needed to continue processing the data set. This specification of additional volumes consists of verifying the mounting of the proper volume, and either checking the data set label if the data set is input, or building the data set label if the data set is output.

The EOV routine is invoked either from a processing program when the user wishes to force an end-of-volume condition, or from the control program when an end-of-volume or end-of-data set condition is encountered by a sequential access method routine. The user causes entry to the EOV routine by issuing an FEOV macro-instruction in his processing program. The expansion of this macro-instruction includes an SVC 31 instruction. When either the CHECK routine of the basic sequential access method (BSAM) or a synchronizing routine of the queued sequential access method (QSAM) finds that a channel program encountered either a permanent error or an end-of-volume condition, the routine issues an SVC 55 instruction.

When either an SVC 31 or SVC 55 instruction is executed, the resulting interruption causes control to be given to the SVC interruption handler. This routine analyzes the interruption, brings the first load module of the EOV routine into the SVC transient area, and passes control to it.

The first module loaded into the SVC transient area for an SVC 31 instruction is the FEOV executor; for an SVC 55 instruction, the first module loaded is the SYNAD/EOV executor.

Upon completion of the EOV routine, control is given to the EOV/new volume executor. These executors perform functions for the sequential access methods and are described in the publication _IBM System/360 Operating System: Sequential Access Methods, Program Logic Manual_.

Charts 30 through 33 show the functions and operation of EOV. A symbolic name is indicated above the blocks that describe each function. These names and the names of the actual EOV load modules that perform the functions are contained in a table in EOV module IGG0550Z. This table enables cross-referencing between the charts and the listings of the EOV load modules.

INITIAL PROCESSING

After the SYNAD/EOV executor has completed its processing, the first module of the I/O support portion of the EOV routine is brought into the SVC transient area. After building a data extent block (DEB), data control block (DCB), input/output block (IOB), and event control block (ECB) for its own I/O processing, the EOV routine reads a JFCB into a work area that was acquired in the dynamic storage by the SYNAD/EOV executor. This is the JFCB of the data set being read or written when the EOV condition occurred (EOV data set); the location on auxiliary storage of the JFCB was found in the task input/output table (TIOT).

The unit control block (UCB) is checked to determine the type of device on which the EOV occurred. For magnetic tape, the tape processing portion of the EOV routine is entered. For a direct-access device, the direct-access portion of the EOV routine is entered. When the device is neither tape nor direct-access, it is assumed to be unit record.

CONCATENATION

If the EOV condition occurred because of an end-of-data set on an input data set, the EOV routine determines whether this data set is concatenated to another data set. A data set is concatenated if, in the TIOT, the next DDNAME field entry is blank. If the data set is not concatenated, the EOV routine passes control through the supervisor to a user-written end-of-data set routine that is in the dynamic area.

When data sets are concatenated but do not have the same attributes, the EOV routine terminates by having the first module of the close routine brought into the SVC transient area to close the EOV data set. After close completes its processing, it passes control to the open routine. When the concatenated data sets have the same attributes, the volume type on which the new data set resides is determined, and the EOV routine performs the processing for that volume before returning control to the supervisor.

EOV ON MAGNETIC TAPE

Processing EOV conditions on magnetic tape consists primarily of verifying and constructing labels. Nonstandard label processing is performed by installation routines that are brought into the SVC transient area from the SVC library (SYS1.SVCLIB).

EOV ON OUTPUT DATA SETS

When an output data set has standard labels, the EOV routine generates trailer labels and writes them on the tape. If the data set has nonstandard labels, the installation programmer must provide a routine to generate and write trailer labels. This routine is incorporated into SYS1.SVCLIB.

If necessary, the EOV routine issues mounting instructions for either a specified or a scratch volume to continue writing the data set. The label characteristics and density (for dual density devices only) of the volume are compared with user specifications. If either does not agree, the volume editor routine receives control as for open. When a volume with correct label characteristics is mounted, the EOV routine determines by reading the sense bytes whether the tape volume is file-protected. If the mode is OUTPUT or OUTIN and the tape is file-protected, a message is issued to the operator to insert a file protect ring. If the mode is INOUT and the volume is file protected, the message will require the operator to determine whether a file protect ring is necessary. This is the case when no writing has yet been done on the data set. Otherwise, the message is the same as for the OUTPUT or OUTIN modes.

When standard labels are present, the EOV routine checks the header label of the first data set on the tape. If the expiration date of the first data set on the new volume has not passed, the EOV routine transmits a message to the operator. If the operator replies that the tape is still to be used, header labels for the EOV data set are written; if the operator replies that the tape is not to be used, a new tape may be mounted.

The EOV routine also checks the label for the password protection indication. When the label indicates password protection, the routine checks the JFCB. If the JFCB also specifies password protection, the label ID must be the same as the DSNAME or the volume is dismounted and a scratch volume is mounted.

When the EOV routine has verified that the tape may be used to continue writing a data set, the routine overlays the labels already on the tape with new header labels.

If an I/O device is available, the EOV routine requests mounting of an additional volume that is to receive portions of the data set.

When a sequential access method is being used, the EOV routine has itself replaced by a sequential access method executor in the SVC transient area. If no access method is being used, the EOV routine releases its work area and returns CPU control to the supervisor.

EOV ON INPUT DATA SETS

When an input data set has standard labels, the EOV routine checks the block count in the trailer label to determine whether all the records have been read; if not the job step is terminated. If this data set does not continue on some other volume, the TIOT is checked to see whether the data set is concatenated. When a concatenated data set is security-protected, control is passed to the EOV security routine to obtain the password. When the data set continues on another volume that is not yet mounted, the EOV routine issues mounting instructions.

Label processing is performed to verify that the proper volume has been mounted. If the devices are available, the EOV routine requests that other volumes containing unprocessed portions of this data set be mounted. If a sequential access method is being used, CPU control is passed to a sequential access method executor that is brought into the SVC transient area. Otherwise, the EOV routine releases its work area and returns control to the supervisor.

EOV ON DIRECT-ACCESS DEVICES

Chart 33 shows the flow of CPU control through the EOV routine for an end-of-volume condition on a direct-access device.

EOV FOR OUTPUT DATA SETS

An EOV condition for an output data set being written on a direct-access volume indicates that the auxiliary storage space assigned to that data set, when the job step was initiated, has been filled. The EOV routine uses a DADSM (direct access device space management) routine to allocate more space on the same volume on which the EOV occurred. The space requested is that amount specified in the secondary quantity field of the JFCB. If no secondary quantity was specified, the job step is terminated.

If the additional space cannot be allocated on the same volume, the EOV routine requests that another volume be mounted on an available device, and the DADSM routine is requested to allocate space on that volume.

The volume-mounting request is for a scratch volume unless the user has specified volumes for this data set. If no devices are available for this volume the job step is terminated.

Once additional space has been allocated, the EOV routine builds a new DEB that reflects the newly acquired storage. Part of the old DEB is saved and the storage that it occupies is released. The data set control block (DSCB) of the EOV data set is read from the device where the EOV occurred. Information from this DSCB and from the old DEB is used to build the new DEB. The DSCB and JFCB are checked for security violations before processing is continued.

If a sequential access method is being used to operate on the data set, the EOV module replaces itself in the SVC transient area with an executor routine of the sequential access method. If an access method has not been used, the EOV routine releases its work area and returns CPU control to the supervisor.

EOV FOR INPUT DATA SETS

When an EOV condition occurs for an input data set, the EOV routine determines whether the data set continues on any other volumes. If not, the TIOT is checked to see whether the data set is concatenated to other data sets. The correct password for a security-protected data set must be obtained before the concatenated data set may be read. When the data set continues on another volume, the EOV routine determines whether this next volume was previously mounted; if not, mounting instructions are issued. The DSCB of the data set is read from this new volume, and portions of the old DEB are saved. The old DEB is replaced with a new DEB built from the saved portion of the old DEB and the DSCB.

If a device is available for an additional volume that contains unprocessed portions of this data set, mounting instructions are issued. If a sequential access method is being used, CPU control is passed to an executor module of that access method. Otherwise, the work area is released and control passed to the supervisor.

FORCE END-OF-VOLUME

Issuing of a force end-of-volume during the generation of a data set causes the EOV routine to read the DSCB, to change its extent indications to reflect the auxiliary storage occupied by the data set, and to rewrite the DSCB on the volume.

When FEOV is issued for an input data set, the EOV routine issues mounting messages and checks for concatenation and security. No DSCB processing is performed.

The following charts are designed to be used with the listings of the I/O support routines. Each chart functional block has been given a name of the form PLMRTXXX, where RT is symbolic of the routine name and XXX is a decimal number identifying the function.

The listings provide tables which associate these names with the actual module names, enabling cross-referencing between the listings and the PLM charts. These tables reside in the following modules:

IGC00019	- open
IGC00022	- openJ
IGC00064	- RDJFCB
IGC00020	- close
IGC00023	- Tclose
IGG0550Z	- EOV
IGG0550Z	- FEOV

Chart 10. The OPEN Routine, Input -- Tape and Direct-Access

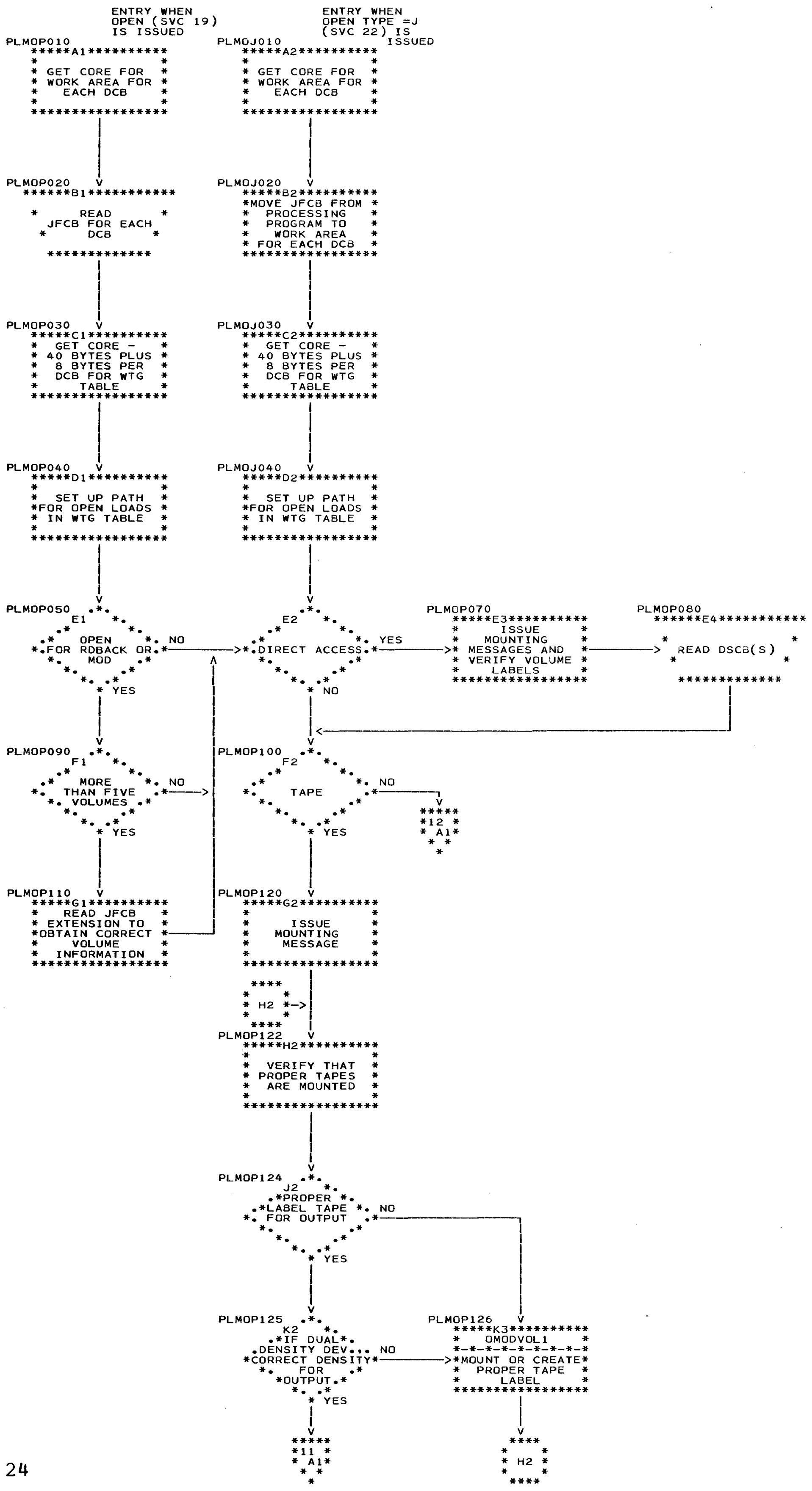
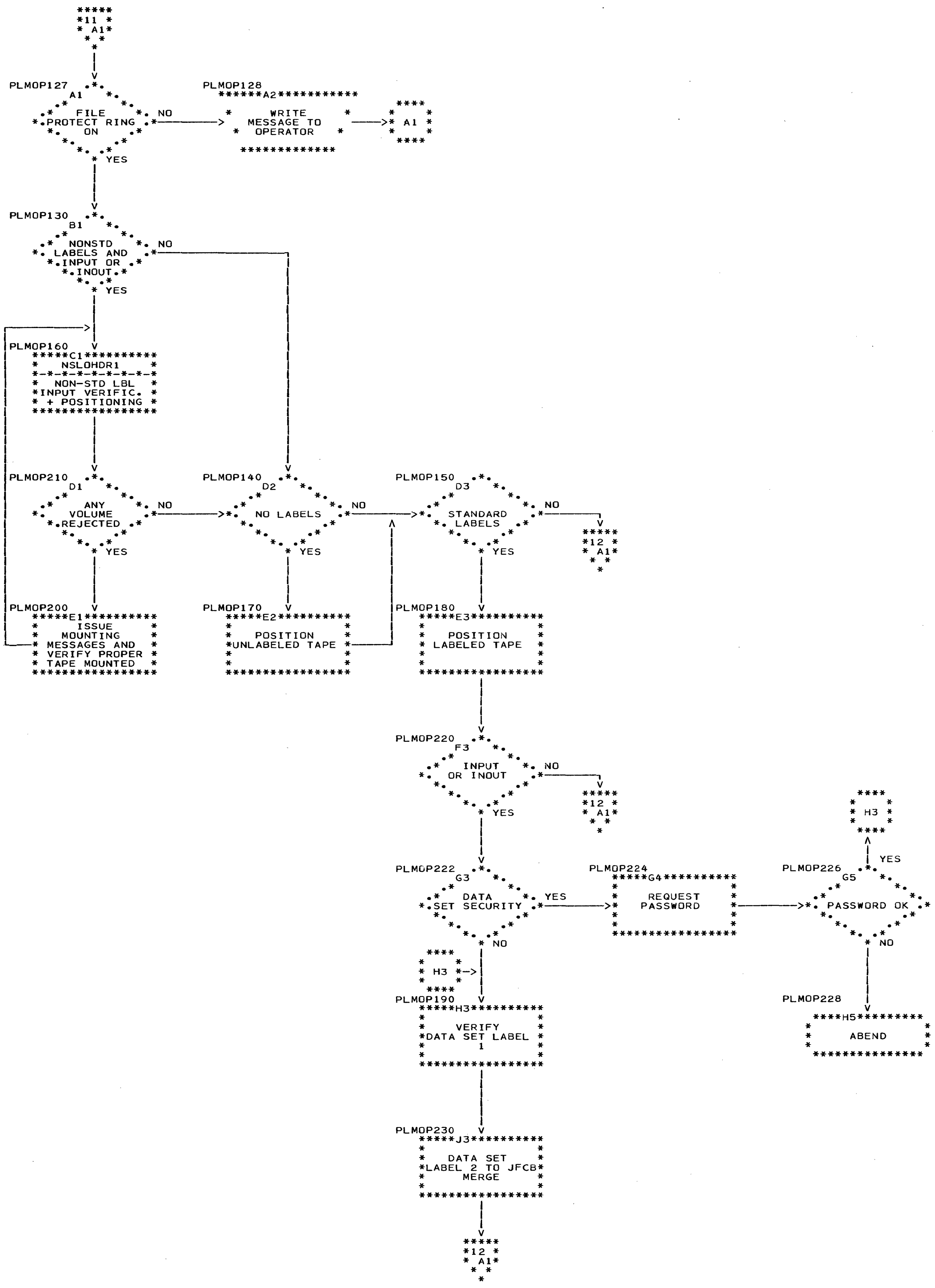
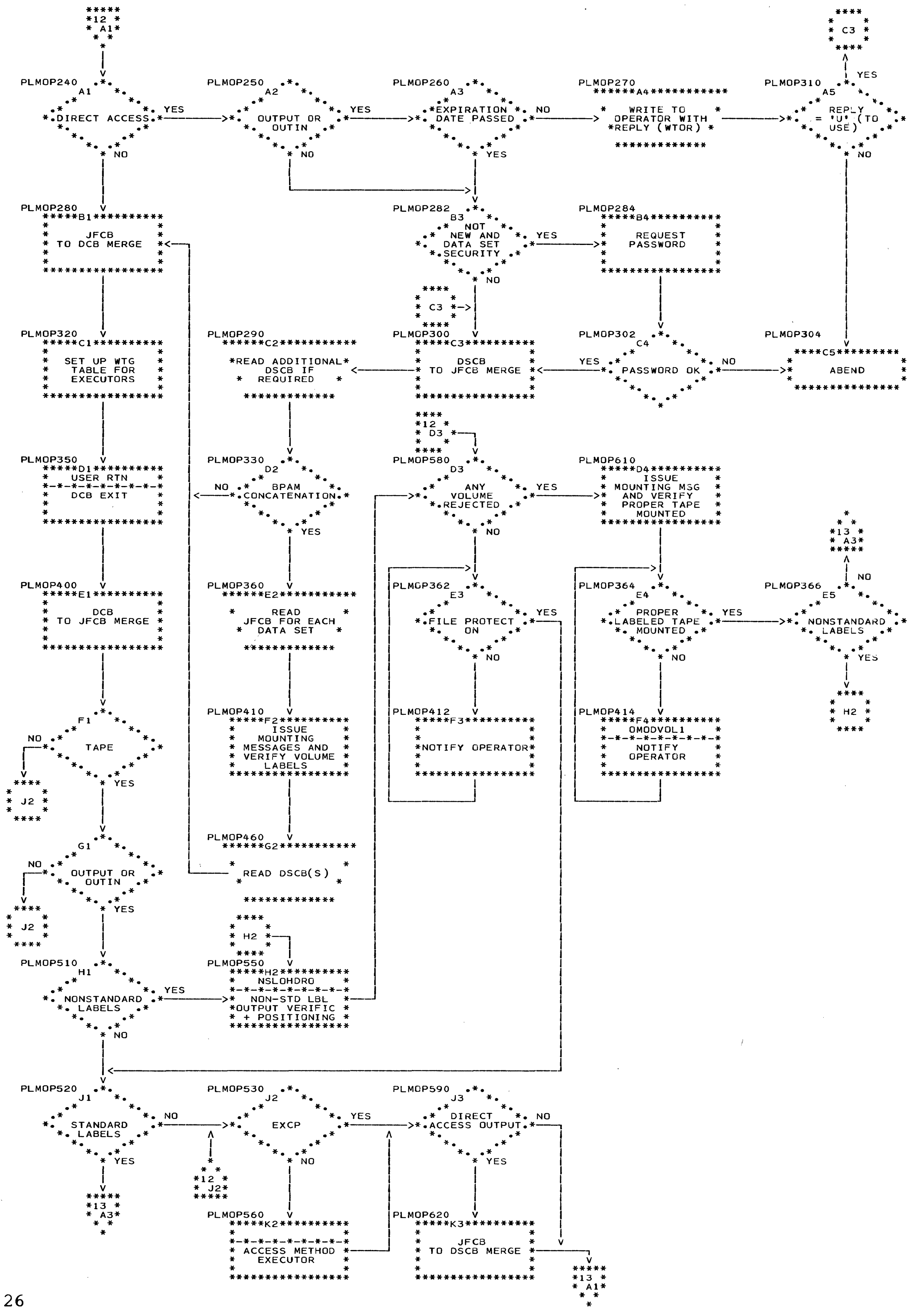

ENTRY WHEN OPEN (SVC 19) IS ISSUED
ENTRY WHEN OPEN TYPE =J (SVC 22) IS ISSUED
PLMOP010 A1 GET CORE FOR WORK AREA FOR EACH DCB
PLMOJ010 A2 GET CORE FOR WORK AREA FOR EACH DCB
PLMOP020 B1 READ JFCB FOR EACH DCB
PLMOJ020 B2 MOVE JFCB FROM PROCESSING PROGRAM TO WORK AREA FOR EACH DCB
PLMOP030 C1 GET CORE - 40 BYTES PLUS 8 BYTES PER DCB FOR WTG TABLE
PLMOJ030 C2 GET CORE - 40 BYTES PLUS 8 BYTES PER DCB FOR WTG TABLE
PLMOP040 D1 SET UP PATH FOR OPEN LOADS IN WTG TABLE
PLMOJ040 D2 SET UP PATH FOR OPEN LOADS IN WTG TABLE
PLMOP050 E1 OPEN FOR RDBACK OR MOD
NO
YES
E2 DIRECT ACCESS
YES
NO
PLMOP070 E3 ISSUE MOUNTING MESSAGES AND VERIFY VOLUME LABELS
PLMOP080 E4 READ DSCB(S)
PLMOP090 F1 MORE THAN FIVE VOLUMES
NO
YES
PLMOP100 F2 TAPE
NO
YES
12 A1
PLMOP110 G1 READ JFCB EXTENSION TO OBTAIN CORRECT VOLUME INFORMATION
PLMOP120 G2 ISSUE MOUNTING MESSAGE
H2
PLMOP122 H2 VERIFY THAT PROPER TAPES ARE MOUNTED
PLMOP124 J2 PROPER LABEL TAPE FOR OUTPUT
NO
YES
PLMOP125 K2 IF DUAL DENSITY DEV. CORRECT DENSITY FOR OUTPUT
NO
YES
PLMOP126 K3 OMODVOL1 MOUNT OR CREATE PROPER TAPE LABEL
11 A1
H2

Chart 11. The OPEN Routine, Input -- Tape and Direct-Access

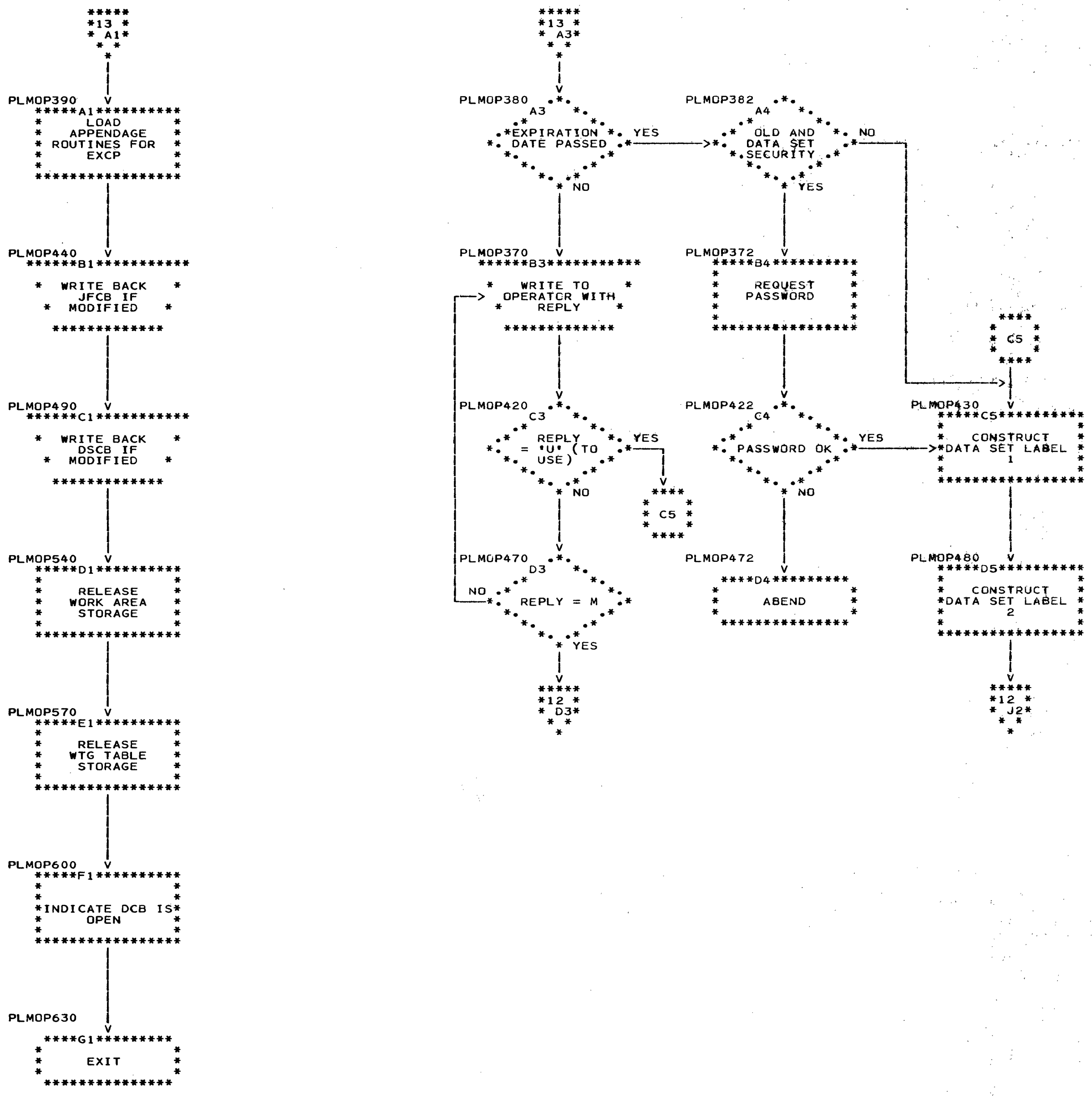


11 A1
PLMOP127 A1 FILE PROTECT RING ON
NO
YES
PLMOP128 A2 WRITE MESSAGE TO OPERATOR
A1
PLMOP130 B1 NONSTD LABELS AND INPUT OR INOUT
NO
YES
PLMOP160 C1 NSLOHDR1 NON-STD LBL INPUT VERIFIC. + POSITIONING
PLMOP210 D1 ANY VOLUME REJECTED
NO
YES
PLMOP140 D2 NO LABELS
NO
YES
PLMOP150 D3 STANDARD LABELS
NO
YES
12 A1
PLMOP200 E1 ISSUE MOUNTING MESSAGES AND VERIFY PROPER TAPE MOUNTED
PLMOP170 E2 POSITION UNLABELED TAPE
PLMOP180 E3 POSITION LABELED TAPE
PLMOP220 F3 INPUT OR INOUT
NO
YES
12 A1
H3
YES
PLMOP222 G3 DATA SET SECURITY
YES
NO
PLMOP224 G4 REQUEST PASSWORD
PLMOP226 G5 PASSWORD OK
NO
H3
PLMOP190 H3 VERIFY DATA SET LABEL 1
PLMOP228 H5 ABEND
PLMOP230 J3 DATA SET LABEL 2 TO JFCB MERGE
12 A1

• Chart 12. The OPEN Routine, Output -- Tape and Direct-Access


12 A1
PLMOP240 A1 DIRECT ACCESS YES NO
PLMOP250 A2 OUTPUT OR OUTIN YES NO
PLMOP260 A3 EXPIRATION DATE PASSED NO YES
PLMOP270 A4 WRITE TO OPERATOR WITH REPLY (WTOR)
PLMOP310 A5 REPLY = 'U' (TO USE) YES NO
C3
PLMOP280 B1 JFCB TO DCB MERGE
PLMOP282 B3 NOT NEW AND DATA SET SECURITY YES NO
PLMOP284 B4 REQUEST PASSWORD
C3
PLMOP320 C1 SET UP WTG TABLE FOR EXECUTORS
PLMOP290 C2 READ ADDITIONAL DSCB IF REQUIRED
PLMOP300 C3 DSCB TO JFCB MERGE
PLMOP302 C4 PASSWORD OK YES NO
PLMOP304 C5 ABEND
12 D3
PLMOP350 D1 USER RTN DCB EXIT
PLMOP330 D2 BPAM CONCATENATION NO YES
PLMOP580 D3 ANY VOLUME REJECTED YES NO
PLMOP610 D4 ISSUE MOUNTING MSG AND VERIFY PROPER TAPE MOUNTED
13 A3
PLMOP400 E1 DCB TO JFCB MERGE
PLMOP360 E2 READ JFCB FOR EACH DATA SET
PLMOP362 E3 FILE PROTECT ON YES NO
PLMOP364 E4 PROPER LABELED TAPE MOUNTED YES NO
PLMOP366 E5 NONSTANDARD LABELS NO YES
H2
F1 TAPE NO YES
J2
PLMOP410 F2 ISSUE MOUNTING MESSAGES AND VERIFY VOLUME LABELS
PLMOP412 F3 NOTIFY OPERATOR
PLMOP414 F4 OMODVOL1 NOTIFY OPERATOR
G1 OUTPUT OR OUTIN NO YES
J2
PLMOP460 G2 READ DSCB(S)
H2
PLMOP510 H1 NONSTANDARD LABELS YES NO
PLMOP550 H2 NSLOHDRO NON-STD LBL OUTPUT VERIFIC + POSITIONING
PLMOP520 J1 STANDARD LABELS NO YES
PLMOP530 J2 EXCP YES NO
PLMOP590 J3 DIRECT ACCESS OUTPUT NO YES
12 J2
13 A3
PLMOP560 K2 ACCESS METHOD EXECUTOR
PLMOP620 K3 JFCB TO DSCB MERGE
13 A1

• Chart 13. The OPEN Routine, Output -- Tape and Direct-Access


13 A1
PLMOP390 A1 LOAD APPENDAGE ROUTINES FOR EXCP
PLMOP440 B1 WRITE BACK JFCB IF MODIFIED
PLMOP490 C1 WRITE BACK DSCB IF MODIFIED
PLMOP540 D1 RELEASE WORK AREA STORAGE
PLMOP570 E1 RELEASE WTG TABLE STORAGE
PLMOP600 F1 INDICATE DCB IS OPEN
PLMOP630 G1 EXIT
13 A3
PLMOP380 A3 EXPIRATION DATE PASSED
YES
NO
PLMOP382 A4 OLD AND DATA SET SECURITY
NO
YES
PLMOP370 B3 WRITE TO OPERATOR WITH REPLY
PLMOP372 B4 REQUEST PASSWORD
C5
PLMOP420 C3 REPLY = 'U' (TO USE)
YES
NO
C5
PLMOP422 C4 PASSWORD OK
YES
NO
PLMOP430 C5 CONSTRUCT DATA SET LABEL 1
PLMOP470 D3 REPLY = M
NO
YES
12 D3
PLMOP472 D4 ABEND
PLMOP480 D5 CONSTRUCT DATA SET LABEL 2
12 J2

]

Chart 14. The RDJFCB Routine

```
                 PLMRJ010
                   *****A2**********
                   *               *
ENTRY WHEN         *  GET STORAGE  *
RDJFCB (SVC 64)    *      FOR      *
IS  ISSUED         *   WORK AREA   *
                   *               *
                   *****************
                           |
                           |
                           |
                           |
                           |
                 PLMRJ020  V
                  ******B2***********

                   *      READ      *
                          JFCB
                    *              *

                     *************
                           |
                           |
                           |
                           |
                           |
                 PLMRJ030  V
                   *****C2**********
                   *  GET ADDR OF  *
                   * DYNAMIC AREA  *
                   *   FROM DCB    *
                   *   EXIT LIST   *
                   *               *
                   *****************
                           |
                           |
                           |
                           |
                           |
                 PLMRJ040  V
                   *****D2**********
                   *               *
                   *   MOVE JFCB   *
                   *      TO       *
                   * DYNAMIC AREA  *
                   *               *
                   *****************
                           |
                           |
                           |
                           |
                           |
                 PLMRJ050  V
                   *****E2**********
                   *               *
                   * FREE STORAGE  *
                   *    OF WORK    *
                   *     AREA      *
                   *               *
                   *****************
                           |
                           |
                           |
                           |
                           |
                 PLMRJ060  |
                           V
                    ****F2*********
                   *               *
                   *     EXIT      *
                   *               *
                    ***************
```

Chart 20. The CLOSE Routine, Tape

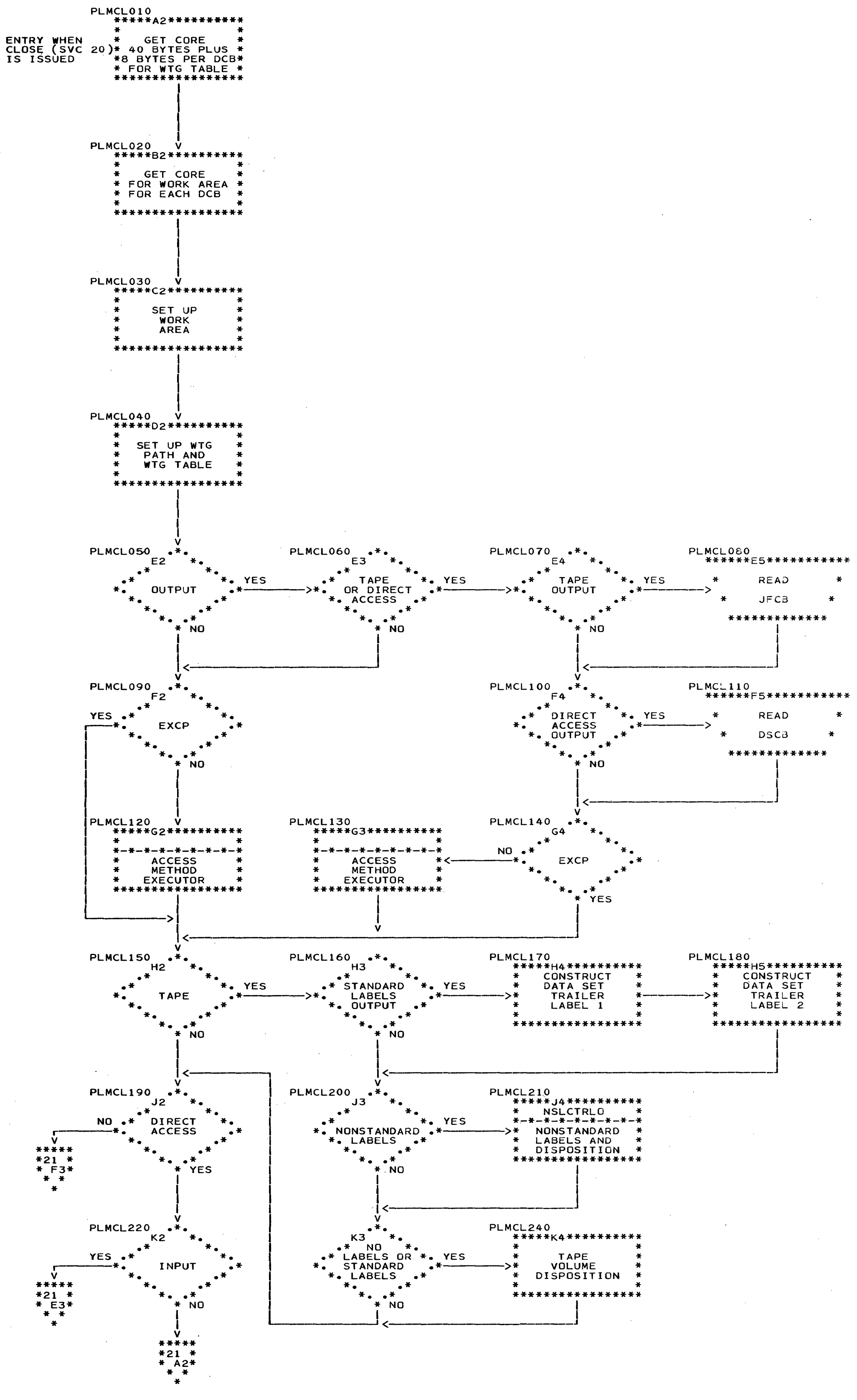
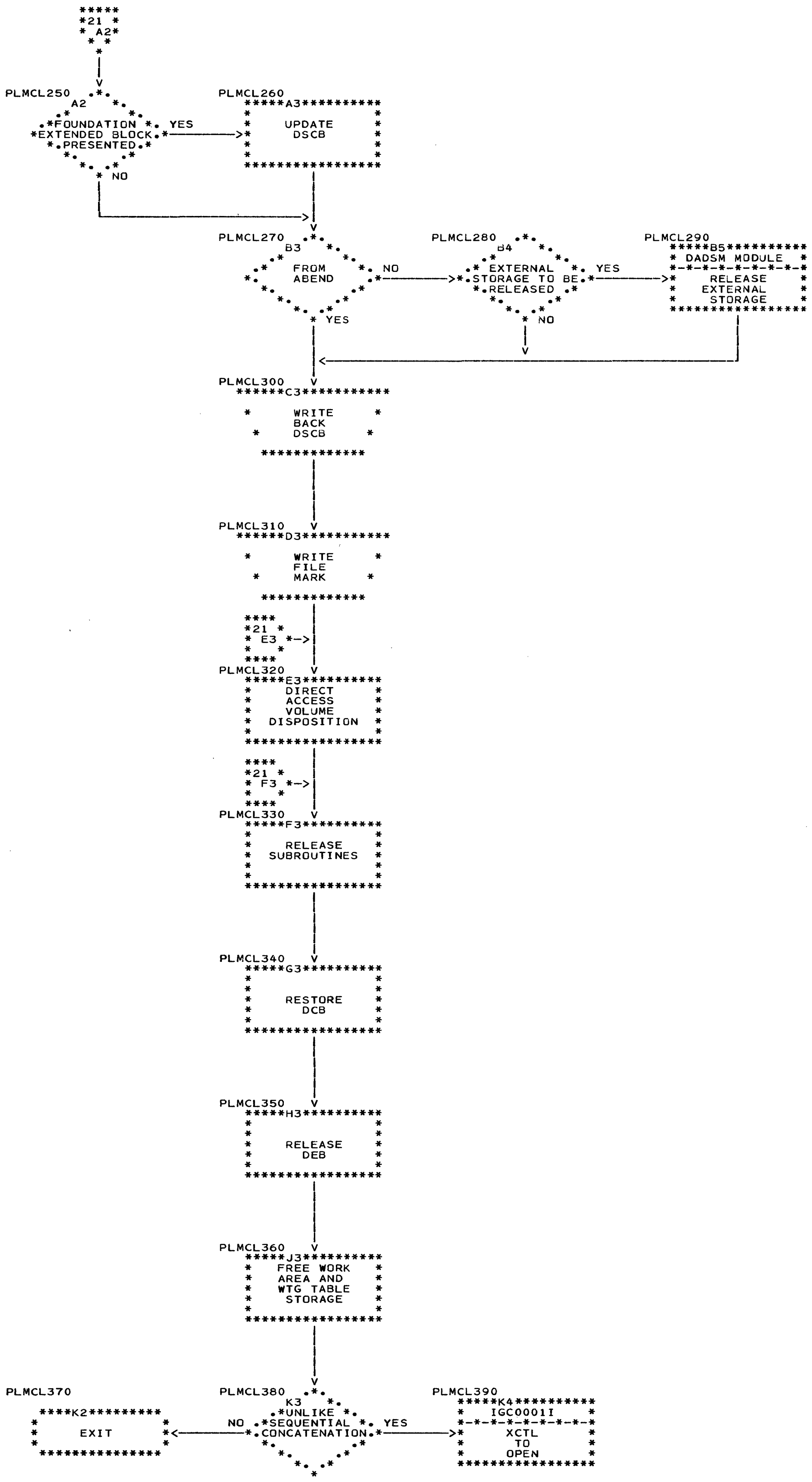
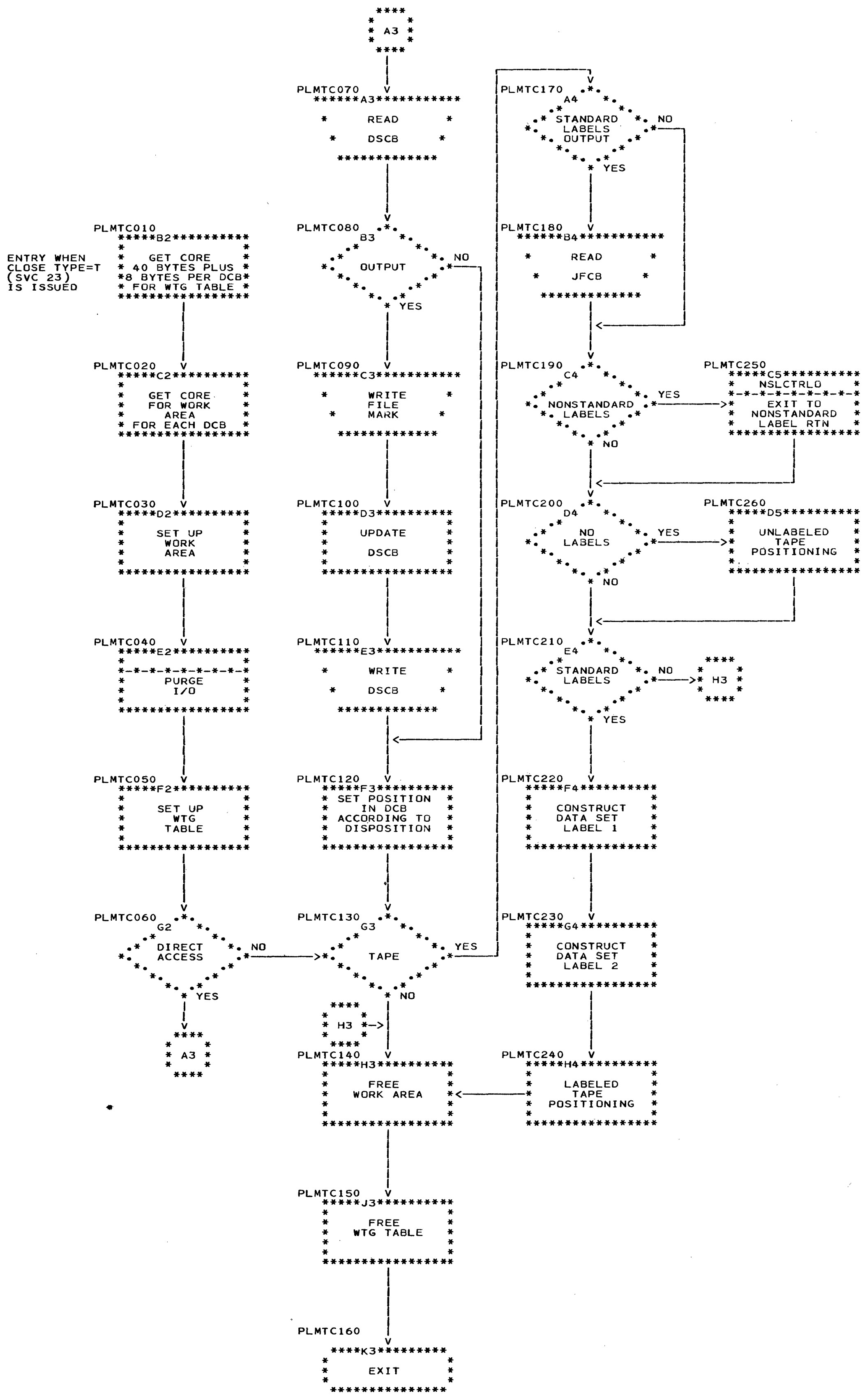


Chart 21. The CLOSE Routine, Direct-Access



21 A2
PLMCL250 A2 FOUNDATION EXTENDED BLOCK PRESENTED
YES
NO
PLMCL260 A3 UPDATE DSCB
PLMCL270 B3 FROM ABEND
NO
YES
PLMCL280 B4 EXTERNAL STORAGE TO BE RELEASED
YES
NO
PLMCL290 B5 DADSM MODULE RELEASE EXTERNAL STORAGE
PLMCL300 C3 WRITE BACK DSCB
PLMCL310 D3 WRITE FILE MARK
21 E3
PLMCL320 E3 DIRECT ACCESS VOLUME DISPOSITION
21 F3
PLMCL330 F3 RELEASE SUBROUTINES
PLMCL340 G3 RESTORE DCB
PLMCL350 H3 RELEASE DEB
PLMCL360 J3 FREE WORK AREA AND WTG TABLE STORAGE
PLMCL370 K2 EXIT
PLMCL380 K3 UNLIKE SEQUENTIAL CONCATENATION
NO
YES
PLMCL390 K4 IGC0001I XCTL TO OPEN

Chart 22. The CLOSE (TYPE=T) Routine



A3
PLMTC070 A3 READ DSCB
PLMTC080 B3 OUTPUT NO YES
PLMTC090 C3 WRITE FILE MARK
PLMTC100 D3 UPDATE DSCB
PLMTC110 E3 WRITE DSCB
PLMTC120 F3 SET POSITION IN DCB ACCORDING TO DISPOSITION
PLMTC130 G3 TAPE YES NO
H3
PLMTC140 H3 FREE WORK AREA
PLMTC150 J3 FREE WTG TABLE
PLMTC160 K3 EXIT
ENTRY WHEN CLOSE TYPE=T (SVC 23) IS ISSUED
PLMTC010 B2 GET CORE 40 BYTES PLUS 8 BYTES PER DCB FOR WTG TABLE
PLMTC020 C2 GET CORE FOR WORK AREA FOR EACH DCB
PLMTC030 D2 SET UP WORK AREA
PLMTC040 E2 PURGE I/O
PLMTC050 F2 SET UP WTG TABLE
PLMTC060 G2 DIRECT ACCESS NO YES
A3
PLMTC170 A4 STANDARD LABELS OUTPUT NO YES
PLMTC180 B4 READ JFCB
PLMTC190 C4 NONSTANDARD LABELS YES NO
PLMTC250 C5 NSLCTRLO EXIT TO NONSTANDARD LABEL RTN
PLMTC200 D4 NO LABELS YES NO
PLMTC260 D5 UNLABELED TAPE POSITIONING
PLMTC210 E4 STANDARD LABELS NO YES
H3
PLMTC220 F4 CONSTRUCT DATA SET LABEL 1
PLMTC230 G4 CONSTRUCT DATA SET LABEL 2
PLMTC240 H4 LABELED TAPE POSITIONING

Chart 30. The EOV Routine, Initialization

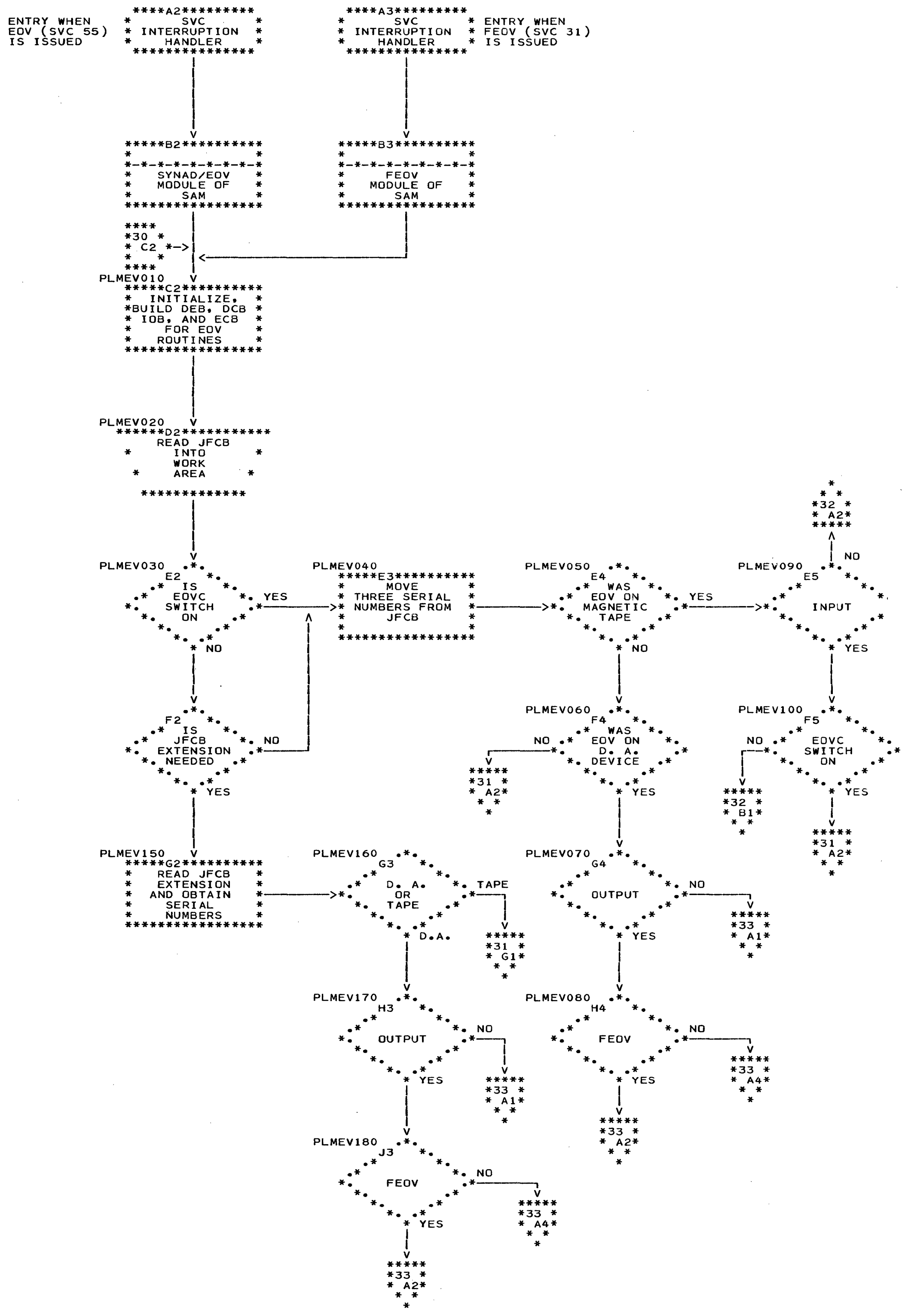

ENTRY WHEN EOV (SVC 55) IS ISSUED
A2 SVC INTERRUPTION HANDLER
A3 SVC INTERRUPTION HANDLER
ENTRY WHEN FEOV (SVC 31) IS ISSUED
B2 SYNAD/EOV MODULE OF SAM
B3 FEOV MODULE OF SAM
30 C2
PLMEV010 C2 INITIALIZE, BUILD DEB, DCB IOB, AND ECB FOR EOV ROUTINES
PLMEV020 D2 READ JFCB INTO WORK AREA
PLMEV030 E2 IS EOVC SWITCH ON
YES
NO
PLMEV040 E3 MOVE THREE SERIAL NUMBERS FROM JFCB
PLMEV050 E4 WAS EOV ON MAGNETIC TAPE
YES
NO
PLMEV090 E5 INPUT
NO
32 A2
YES
F2 IS JFCB EXTENSION NEEDED
NO
YES
PLMEV060 F4 WAS EOV ON D. A. DEVICE
NO
31 A2
YES
PLMEV100 F5 EOVC SWITCH ON
NO
32 B1
YES
31 A2
PLMEV150 G2 READ JFCB EXTENSION AND OBTAIN SERIAL NUMBERS
PLMEV160 G3 D. A. OR TAPE
TAPE
31 G1
D.A.
PLMEV070 G4 OUTPUT
NO
33 A1
YES
PLMEV170 H3 OUTPUT
NO
33 A1
YES
PLMEV080 H4 FEOV
NO
33 A4
YES
33 A2
PLMEV180 J3 FEOV
NO
33 A4
YES
33 A2

Chart 31. The EOV Routine, Initialization

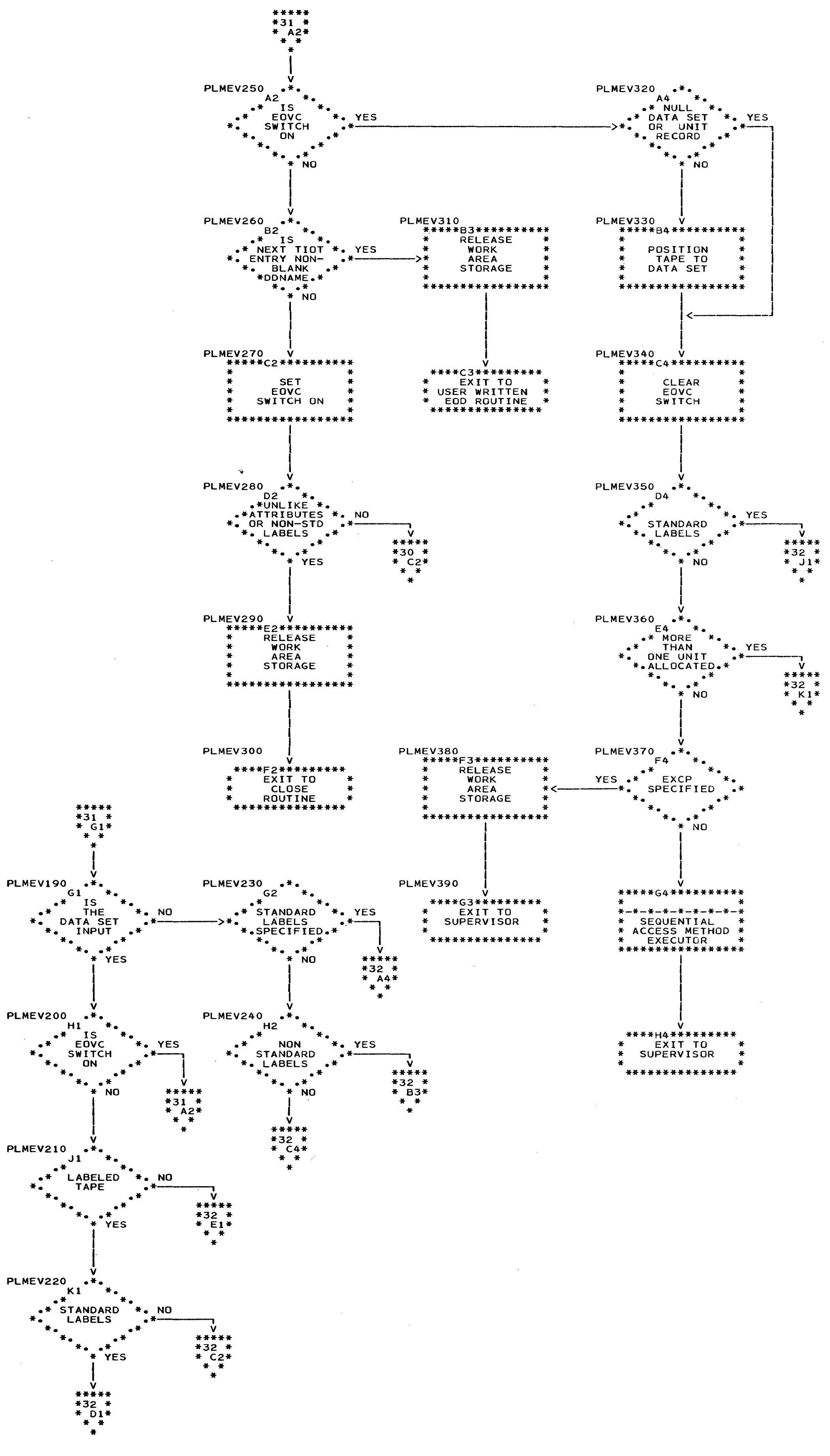

31 A2
PLMEV250 A2 IS EOVC SWITCH ON
YES
NO
PLMEV260 B2 IS NEXT TIOT ENTRY NON-BLANK DDNAME
YES
NO
PLMEV310 B3 RELEASE WORK AREA STORAGE
C3 EXIT TO USER WRITTEN EOD ROUTINE
PLMEV270 C2 SET EOVC SWITCH ON
PLMEV280 D2 UNLIKE ATTRIBUTES OR NON-STD LABELS
NO
30 C2
YES
PLMEV290 E2 RELEASE WORK AREA STORAGE
PLMEV300 F2 EXIT TO CLOSE ROUTINE
PLMEV320 A4 NULL DATA SET OR UNIT RECORD
YES
NO
PLMEV330 B4 POSITION TAPE TO DATA SET
PLMEV340 C4 CLEAR EOVC SWITCH
PLMEV350 D4 STANDARD LABELS
YES
32 J1
NO
PLMEV360 E4 MORE THAN ONE UNIT ALLOCATED
YES
32 K1
NO
PLMEV370 F4 EXCP SPECIFIED
YES
NO
PLMEV380 F3 RELEASE WORK AREA STORAGE
PLMEV390 G3 EXIT TO SUPERVISOR
G4 SEQUENTIAL ACCESS METHOD EXECUTOR
H4 EXIT TO SUPERVISOR
31 G1
PLMEV190 G1 IS THE DATA SET INPUT
NO
YES
PLMEV230 G2 STANDARD LABELS SPECIFIED
YES
32 A4
NO
PLMEV200 H1 IS EOVC SWITCH ON
YES
31 A2
NO
PLMEV240 H2 NON STANDARD LABELS
YES
32 B3
NO
32 C4
PLMEV210 J1 LABELED TAPE
NO
32 E1
YES
PLMEV220 K1 STANDARD LABELS
NO
32 C2
YES
32 D1

Chart 32. The EOV Routine, Tape

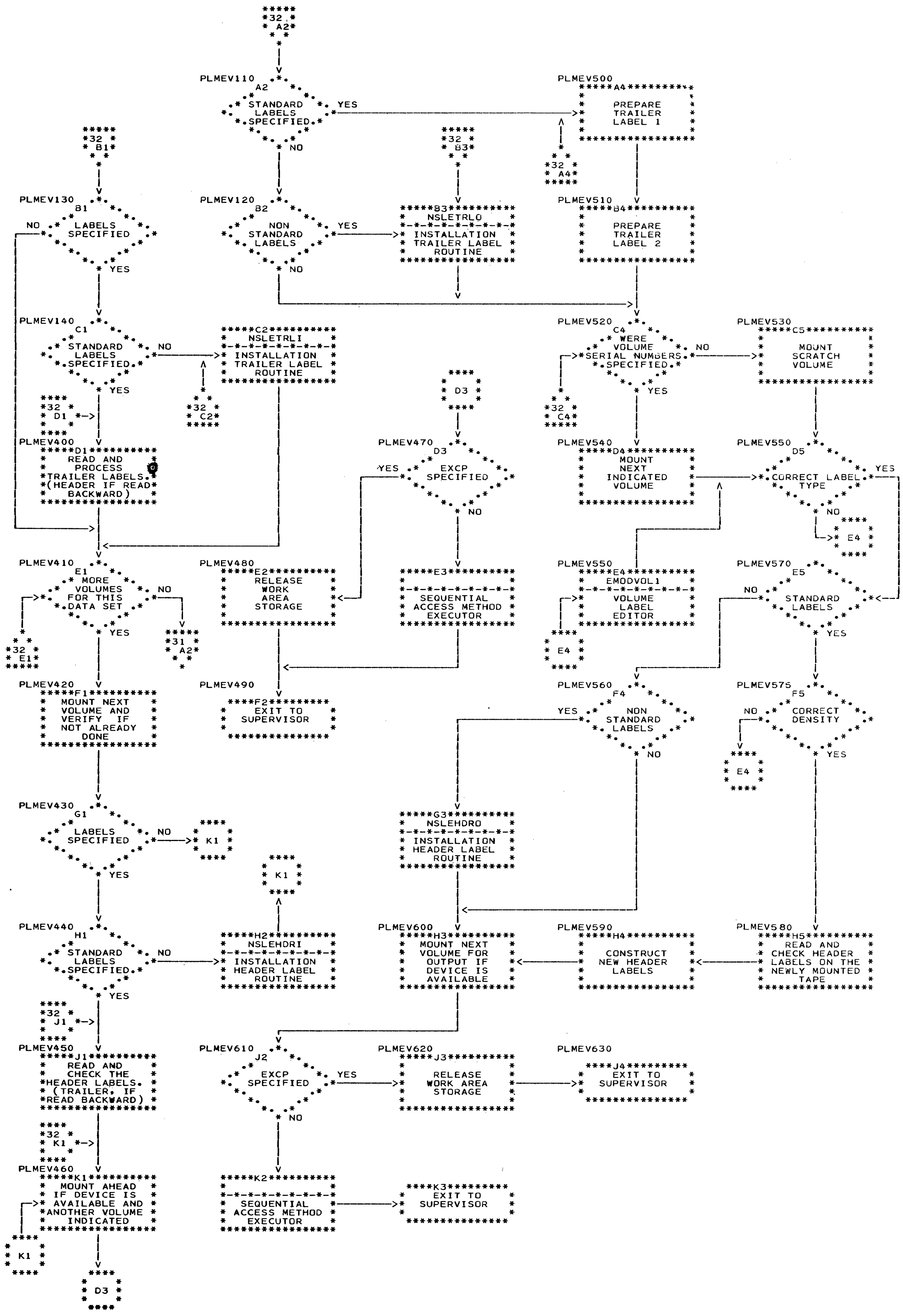
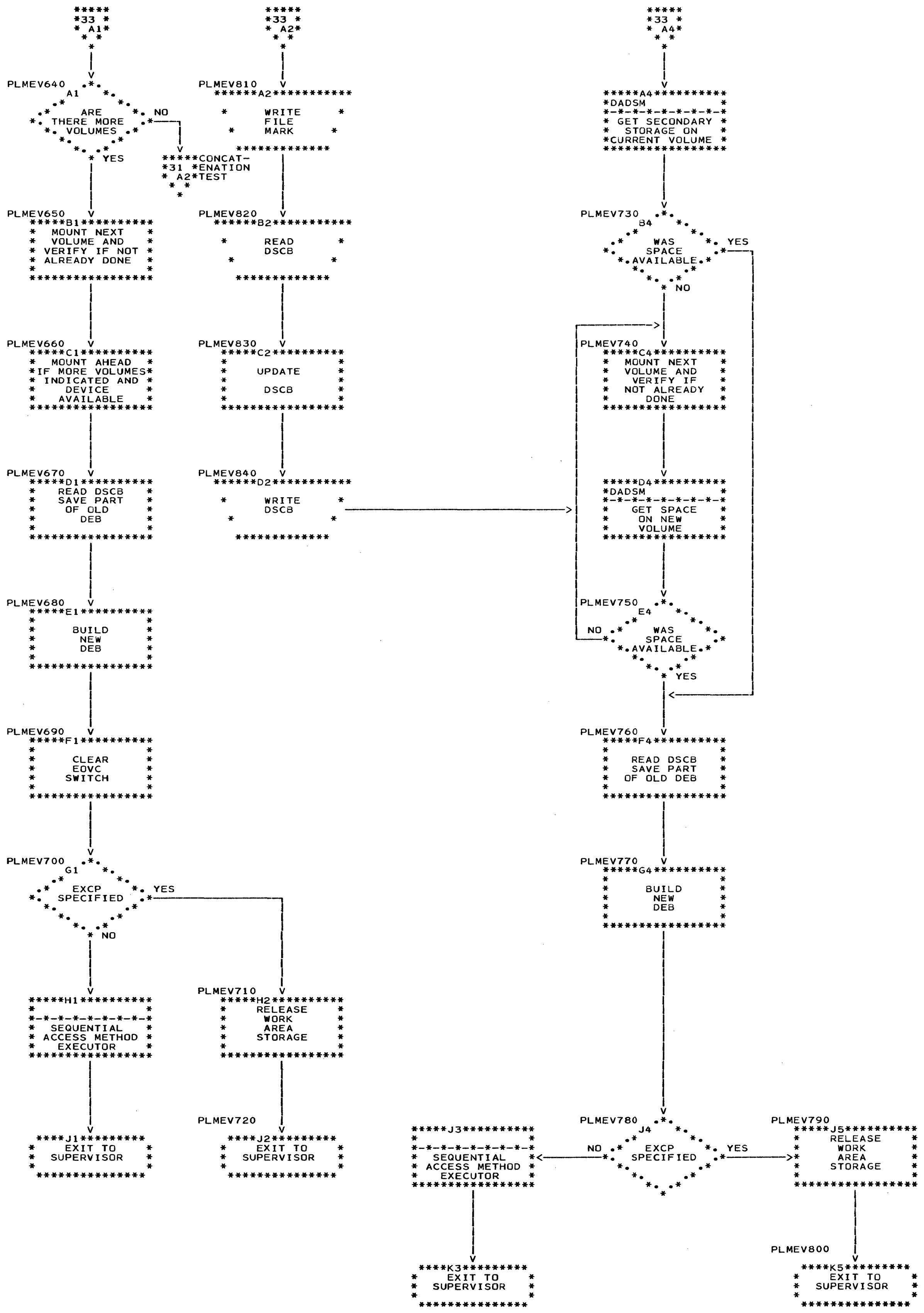

32 A2
PLMEV110 A2 STANDARD LABELS SPECIFIED
YES
NO
PLMEV500 A4 PREPARE TRAILER LABEL 1
32 A4
32 B1
PLMEV130 B1 LABELS SPECIFIED
NO
YES
PLMEV120 B2 NON STANDARD LABELS
32 B3
B3 NSLETRLO INSTALLATION TRAILER LABEL ROUTINE
PLMEV510 B4 PREPARE TRAILER LABEL 2
PLMEV140 C1 STANDARD LABELS SPECIFIED
C2 NSLETRLI INSTALLATION TRAILER LABEL ROUTINE
32 C2
32 D1
PLMEV400 D1 READ AND PROCESS TRAILER LABELS. (HEADER IF READ BACKWARD)
D3
PLMEV470 D3 EXCP SPECIFIED
PLMEV520 C4 WERE VOLUME SERIAL NUMBERS SPECIFIED
32 C4
PLMEV530 C5 MOUNT SCRATCH VOLUME
PLMEV540 D4 MOUNT NEXT INDICATED VOLUME
PLMEV550 D5 CORRECT LABEL TYPE
E4
PLMEV410 E1 MORE VOLUMES FOR THIS DATA SET
32 E1
31 A2
PLMEV480 E2 RELEASE WORK AREA STORAGE
E3 SEQUENTIAL ACCESS METHOD EXECUTOR
PLMEV550 E4 EMODVOL1 VOLUME LABEL EDITOR
PLMEV570 E5 STANDARD LABELS
PLMEV420 F1 MOUNT NEXT VOLUME AND VERIFY IF NOT ALREADY DONE
PLMEV490 F2 EXIT TO SUPERVISOR
PLMEV560 F4 NON STANDARD LABELS
PLMEV575 F5 CORRECT DENSITY
PLMEV430 G1 LABELS SPECIFIED
K1
G3 NSLEHDRO INSTALLATION HEADER LABEL ROUTINE
PLMEV440 H1 STANDARD LABELS SPECIFIED
H2 NSLEHDRI INSTALLATION HEADER LABEL ROUTINE
PLMEV600 H3 MOUNT NEXT VOLUME FOR OUTPUT IF DEVICE IS AVAILABLE
PLMEV590 H4 CONSTRUCT NEW HEADER LABELS
PLMEV580 H5 READ AND CHECK HEADER LABELS ON THE NEWLY MOUNTED TAPE
32 J1
PLMEV450 J1 READ AND CHECK THE HEADER LABELS. (TRAILER, IF READ BACKWARD)
PLMEV610 J2 EXCP SPECIFIED
PLMEV620 J3 RELEASE WORK AREA STORAGE
PLMEV630 J4 EXIT TO SUPERVISOR
32 K1
PLMEV460 K1 MOUNT AHEAD IF DEVICE IS AVAILABLE AND ANOTHER VOLUME INDICATED
K2 SEQUENTIAL ACCESS METHOD EXECUTOR
K3 EXIT TO SUPERVISOR

Chart 33. The EOV Routine, Direct-Access


33 A1
33 A2
33 A4
PLMEV640 A1 ARE THERE MORE VOLUMES
NO
YES
CONCAT- ENATION TEST 31 A2
PLMEV650 B1 MOUNT NEXT VOLUME AND VERIFY IF NOT ALREADY DONE
PLMEV660 C1 MOUNT AHEAD IF MORE VOLUMES INDICATED AND DEVICE AVAILABLE
PLMEV670 D1 READ DSCB SAVE PART OF OLD DEB
PLMEV680 E1 BUILD NEW DEB
PLMEV690 F1 CLEAR EOVC SWITCH
PLMEV700 G1 EXCP SPECIFIED
YES
NO
H1 SEQUENTIAL ACCESS METHOD EXECUTOR
J1 EXIT TO SUPERVISOR
PLMEV810 A2 WRITE FILE MARK
PLMEV820 B2 READ DSCB
PLMEV830 C2 UPDATE DSCB
PLMEV840 D2 WRITE DSCB
PLMEV710 H2 RELEASE WORK AREA STORAGE
PLMEV720 J2 EXIT TO SUPERVISOR
A4 DADSM GET SECONDARY STORAGE ON CURRENT VOLUME
PLMEV730 B4 WAS SPACE AVAILABLE
YES
NO
PLMEV740 C4 MOUNT NEXT VOLUME AND VERIFY IF NOT ALREADY DONE
D4 DADSM GET SPACE ON NEW VOLUME
PLMEV750 E4 WAS SPACE AVAILABLE
NO
YES
PLMEV760 F4 READ DSCB SAVE PART OF OLD DEB
PLMEV770 G4 BUILD NEW DEB
PLMEV780 J4 EXCP SPECIFIED
NO
YES
J3 SEQUENTIAL ACCESS METHOD EXECUTOR
K3 EXIT TO SUPERVISOR
PLMEV790 J5 RELEASE WORK AREA STORAGE
PLMEV800 K5 EXIT TO SUPERVISOR

APPENDIX A: I/O SUPPORT WORK AREA

The contents of the work area shown in Figure 3 are used for the OPEN, OPEN (TYPE=J), CLOSE, CLOSE (TYPE=T), and EOV routines. The numbers in parentheses are the number of bytes of storage for a particular section.

The first 464 bytes are used by the OPEN, CLOSE, and EOV routines. The next 38 bytes are used only by the EOV routine. The next 24 bytes are used by the OPEN and EOV routines. EOV uses an additional 10 bytes along with the preceding section.

Bytes	Contents						
OPEN,CLOSE,EOV	one of the following:						
100	volume label bytes (80)	file label #1 bytes (80)	file label #2 bytes (80)	DSCB data portion format 1 bytes (96)	DSCB key portion format 3 bytes (44)	DSCB data portion format 3 bytes (96)	message area (variable size)
176	Job File Control Block (JFCB)						
4	Event Control Block (ECB)						
40	Input/Output Block (IOB)						
44	Data Extent Block (DEB)						
4	Data Control Block (DCB)						
96	Channel Command Words (CCWs)						
37	XCTL Work Area (EOV only)						
1	Switch (EOV only)						
24	Work Area for Volume Serial Numbers (OPEN and EOV only)						
10	Additional Work Area for Volume Serial Numbers (EOV only)						

Figure 3. OPEN, CLOSE, and EOV Work Area

The XCTL table is used to transfer control between loads and between a load and an access method executor. The format of the XCTL table is shown in Figure 4. There is a table starting on a double-word boundary in each load. The table consists of the other load ID's to which this load can transfer control. Each entry consists of the load ID, its relative disk address (TTR) and the length of the load. The TTR's are inserted by the IEHIOSUP utility program when the system is generated. The last four bytes of the load consist of a supervisor-call (SVC) pointing to the beginning of the XCTL table. The pointer is expressed in double words from the beginning of the load.

```
┌──────────────────────┬─────────────────────────────────────────────────────────────┐
│                      │                                                             │
│    Load              │                     Program                                 │
│                      │                                                             │
├──────────────────────┼─────────────────────────┬───────────────────────────────────┤
│                      │  Relative disk          │  Length of                        │
│  Load ID1            │  address (TTR)          │  load expressed in double words   │
│  (2 bytes)           │  (3 bytes)              │  (1 byte)                         │
├──────────────────────┼─────────────────────────┼───────────────────────────────────┤
│  Load ID2            │  TTR of load            │  Length                           │
├──────────────────────┼─────────────────────────┼───────────────────────────────────┤
│  Load ID3            │  TTR of load            │  Length                           │
├──────────────────────┼─────────────────────────┼───────────────────────────────────┤
│  Load ID's           │                         │                                   │
│  continue            │                         │                                   │
├──────────────────────┼─────────────────────────┼───────────────────────────────────┤
│  Load IDn            │  TTR of load            │  Length                           │
├──────────────────────┼─────────────────────────┴───────────────────────────────────┤
│  00                  │                                                             │
│  (end of table)      │                                                             │
├──────────────────────┘                                                             │
│                            (Program continues)                                     │
│                                                                                    │
│                      ┌─────────────────────────┬───────────────────────────────────┤
│                      │  SVC code               │  Pointer to the first ID1 relative│
│                      │                         │  to 0 and expressed in double words│
└──────────────────────┴─────────────────────────┴───────────────────────────────────┘
                       <------------------- last 4 bytes of load ------------------->
```

Figure 4. XCTL Table

APPENDIX C: THE WHERE-TO-GO (WTG) TABLE

Both the open and close routines set up a WTG table to indicate which load modules and routines are necessary for processing. Figure 5 shows the WTG table format. The first twenty-nine bytes (0-28) form a parameter for the directory entry portion of the XCTL and LOAD macro-instructions. Bytes 30 and 31 indicate by bit setting the required path through the open or close load modules. Bit assignments in the first byte are for:

- Direct-access device.
- Standard label tape positioning.
- Unlabeled tape positioning.
- Input label processing (verification).
- Nonstandard input tape.
- Nonstandard output tape.
- Volume label editing.
- Data set security.

Bit assignments in the second byte are for:

- Output label processing (verification).
- More than five volumes specified by the JFCB.
- Volume label editor processing.

The rest of the WTG table indicates the access method executors required to process each DCB. Open and close effect this indication by providing each DCB with an associated entry when the table is built. Open and close transfer, from the XCTL table to the DCB's entry, the identification (ID) and relative disk address (TTR) of the first access method executor required to process the DCB. The first five bytes of each eight-byte entry is used for this purpose. The last three bytes of each entry contain the address of the work area assigned to the DCB. The last entry in the variable section is the IDTTR of the open or close load module to which the access method routines return control.

Byte				
0	Name			
8	Relative Disk Address (TTR) of First Record			Concatenation Number
12	Zero	(see note)[1]	TTR of First Text	
16	Record	Zero	TTR of NOTE List or Scatter List	
20	Translation Table	Zero or Number of Entries NOTE List	Attributes	
24	Total Contiguous Main Storage Required for Module			Length of First
28	Text Record	Length WTG Table (in double-words)	Path Through Loads of Routine	
32	IDTTR of Executor for First DCB			Work Area Address for First DCB
	Table of IDTTR's (8 bytes)			
	IDTTR of Executor for nth DCB			Work Area Address for nth DCB
32+n(8)	IDTTR of Load to Which Control Returns			Not Used

[1]Alias indicator and user data field descriptor. An alias indicator is one bit, the number of TTR's in the user data field is two bits.

Figure 5. Where-To-Go (WTG) Table for OPEN and CLOSE Routines

Y28-6609-1

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENTS

Title: IBM System/360 Operating System
Input/Output Support (OPEN/CLOSE/EOV)
Program Logic Manual

Form: Y28-6609-1

Is the material:	Yes	No
Easy to Read?	___	___
Well organized?	___	___
Complete?	___	___
Well illustrated?	___	___
Accurate?	___	___
Suitable for its intended audience?	___	___

How did you use this publication?
___ As an introduction to the subject ___ For additional knowledge
Other ______________________________

fold

Please check the items that describe your position:

___ Customer personnel	___ Operator	___ Sales Representative
___ IBM personnel	___ Programmer	___ Systems Engineer
___ Manager	___ Customer Engineer	___ Trainee
___ Systems Analyst	___ Instructor	Other _____________

Please check specific criticism(s), give page number(s), and explain below:
___ Clarification on page(s)
___ Addition on page(s)
___ Deletion on page(s)
___ Error on page(s)

Explanation:

fold

CUT ALONG LINE

FOLD ON TWO LINES, STAPLE AND MAIL
No Postage Necessary if Mailed in U.S.A.

Y28-6609-1

st

fold

FIRST CLASS
PERMIT NO. 81

POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N. Y. 12602

ATTN: PROGRAMMING SYSTEMS PUBLICATIONS
DEPT. D58

fold

f

Printed in USA Y28-6609-1

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

sta

102664834