LY24-5209-0
File No. S370/4300-30

Program Product

VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 1 General Information and Imperative Macros

Program Number 5746-XE9

Release 2

IBM

SUMMARY OF AMENDMENTS

This manual contains information previously published in DOS/VSE LIOCS Volume 1, General Information and Imperative Macros Logic, SY33-8559, and VSE/Advanced Functions Supplement, LD25-0012. Changes reflect support for:

- DASD independence for SAM/DAM files.
- Reduction of B-transient area contention.

The major impact of this support is in changes to OPEN and CLOSE B-transients and the inclusion in this volume of four B-transient phases previously described in DOS/VSE LIOCS Volume 2, SAM Logic. In addition, disk open and data security message writer B-transients are modified to cause A and D messages to be issued from the SVA so that the LTA is not held in the wait state for the responses. Several new B-transients are supplied to transfer control to and from the LTA and SVA for this purpose.

Additional changes include 3262 printer support, improved label processing, APAR corrections, and miscellaneous editorial corrections.

First Edition (October 1979)

This edition, LY24-5209-0, applies to Release 2 of VSE/Advanced Functions, (Program Number 5746-XE9) and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest System/370 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

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

This document has been provided subject to the terms and conditions of the License Agreement for IBM Program Products.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Laboratory, Programming Publications Department, Schoenaicher Str. 220, D-7030 Germany. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatsoever. You may, of course, continue to use the information you supply.

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

Volume 1: General Information and Imperative Macros, LY24-5209.

Volume 2: SAM, LY24-5210.

Volume 3: DAM and ISAM, LY24-5211.

Volume 4: SAM for DASD, LY24-5212.

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

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

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

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

PREREQUISITE PUBLICATIONS

- IBM System/370 Principles of Operation, GA22-7000, in conjunction with
- IBM System/360 Principles of Operation, GA22-6821.
- OS/VS - DOS/VSE - VM/370 Assembler Language, GC33-4010.
- VSE System Data Management Concepts, GC24-5209.
- VSE/Advanced Functions Macro User's Guide, SC24-5210.
- VSE/Advanced Functions Macro Reference, SC24-5211.
- VSE/Advanced Functions System Control Statements, SC33-6095.
- VSE/Advanced Functions Diagnosis Reference: Supervisor, LY33-9091.

RELATED PUBLICATIONS

- VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 2, SAM, LY24-5210.
- VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 3, DAM and ISAM, LY24-5211.
- VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 4, SAM for DASD, LY24-5212.
- VSE/Advanced Functions Tape Labels, SC24-5212.
- VSE/Advanced Functions DASD Labels, SC24-5213.
- VSE/Advanced Functions Messages, SC33-6098.
- System/370 and 4300 Processors Bibliography, GC20-0001.

CONTENTS

FIGURES

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

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

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

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

Full details on physical IOCS can be found in VSE/Advanced Functions Diagnosis Reference: Supervisor, LY33-9091.

LOGICAL IOCS

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

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

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

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

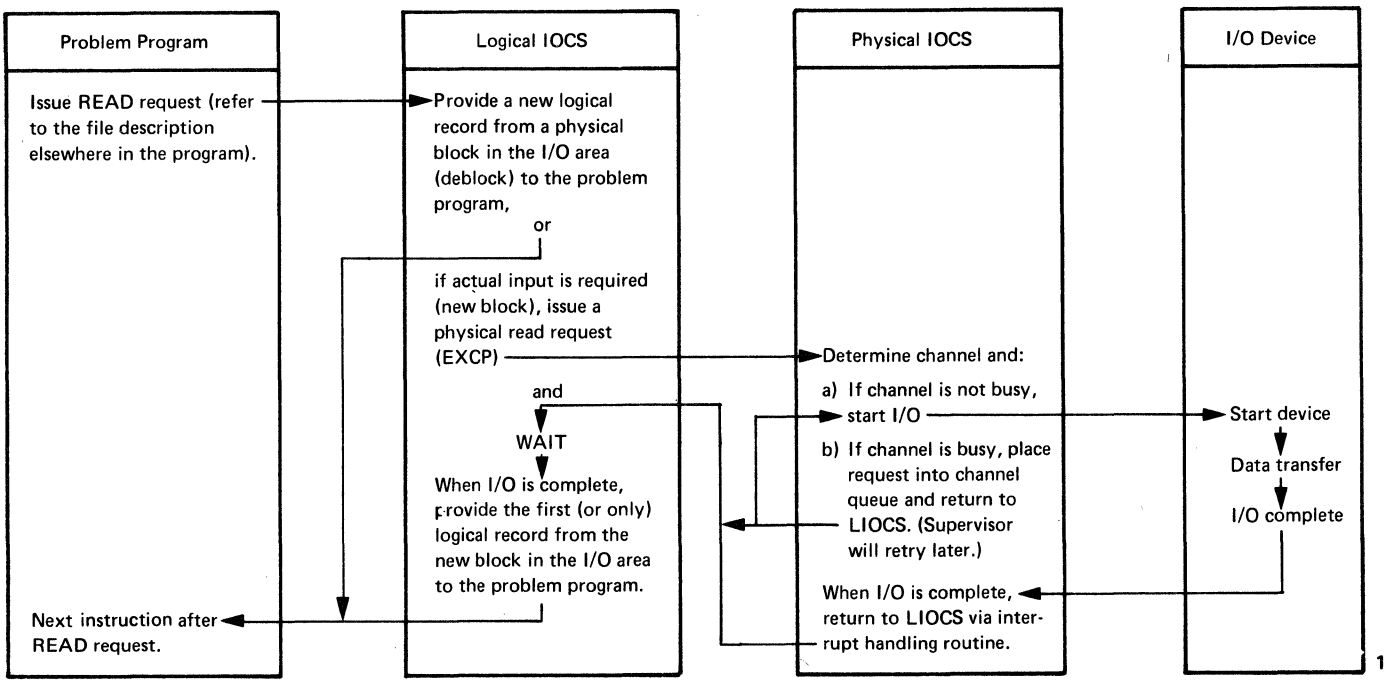


Figure 1. Example of LIOCS and PIOCS Interrelationship

LOGICAL IOCS PROCESSING METHODS

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

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

Sequential Access Method (SAM)

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

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

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

Direct Access Method (DAM)

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

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

Indexed Sequential Access Method (ISAM)

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

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

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

Virtual Storage Access Method (VSE/VSAM)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Storage Requirements

Some logical IOCS routines are generated as part of the problem program, others (supplied by IBM) reside in the System Virtual Area and are dynamically linked to the user program. Imperative macro expansions, which serve as linkage to the logical or physical IOCS routines, are generated inline at the point the macro is used in the problem program. The open,

close, EOF/EOV, and other special purpose routines are called into the B-transient (logical transient) area as required. The physical IOCS routines used by logical IOCS are generated as part of the supervisor program.

MODULAR-TABULAR SYSTEM

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

The modular-tabular system has the advantages of:

- Saving assembly time by allowing the data handling modules to be generated separately and to be stored in the relocatable library for subsequent use.
- Using one module with many files if the device types are the same and the files are similar.

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

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

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

DECLARATIVE MACROS

DTF (Define the File) Macros

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

---

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

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

Figure 2. DTF Table Types

Notes:

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

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

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

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

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

MOD (Module Generation) Macros

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

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

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

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

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

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

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

TRACK HOLD FUNCTION

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

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

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

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

1. Sequential DASD update files (data).

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

3. DAM files.

4. ISAM files.

REENTERABLE MODULES

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

The IBM-supplied logic modules used for DAM and sequential DASD (DTFSD, DTFCP, DTFDI) files are read-only and re-entrant, but do not require the user to provide a save area address in register 13 and will ignore such an address if provided.

Reenterable modules include: CDMOD, CPMOD, DAMOD, DIMOD, DUMOD, ISMOD, MTMOD, and SDMOD.

INTERRELATIONSHIPS OF THE DECLARATIVE MACRO INSTRUCTIONS

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

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

- Device CCB.
- A V-type statement used by the Linkage Editor to resolve the linkage to the logic module with this DTF. For DTFSD and DTFDA, zeros are generated, since OPEN will dynamically fill in this field with the address of the IBM-supplied logic module.
- Logic indicators; that is, one I/O area, two I/O areas, device type, and so on.
- Addresses of all (except work files) of the areas and controls used by this device.

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

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

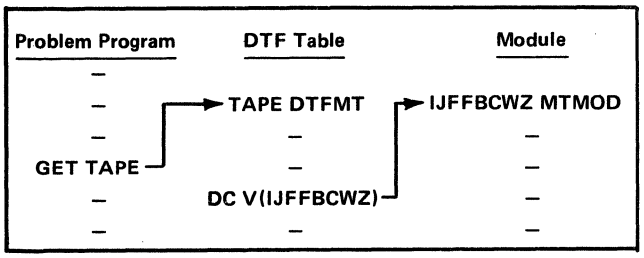


Figure 3. The Relationship Between Imperative and Declarative Macros

IMPERATIVE MACROS

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

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

1. Work files only.
2. Not for 2560 work files.

Figure 4. Logical IOCS Imperative Macros and DTFs

Macro Instruction		TYPE OF PROCESSING WITH LOGICAL IOCS																			
		Sequential															Indexed Sequential				
		3210/3215 Printer-Keyboards, Model 115, 125 Display Operator Console	1287 Optical Reader	1403/1443/3203/3211[13]/3800/3289-4, 5203 Printer, 2560, 5424/5425 with print feature	1419/1255/1259 Magnetic Ink Character Reader	1270/1275 Optical Reader/Sorter	3881 Optical Mark Reader	3886 Optical Character Reader	1442/2520/2540/3525 Punch 2560 MFCM, 5424/5425 MFCU	1442/2501/2520/2540/2596/3504/3505 Reader, 2560 MFCM, 5424/5425 MFCU	2311/2314/2319/3330/3340 3330-II/3350/FBA DASD	3540 Diskette Input/Output Unit	2400 and 3400 Magnetic Tape Units	2671/1017 Paper Tape Reader	1018 Paper Tape Punch	Direct Access	Load File	Add Records	Random Retrieve	Sequential Retrieve
Initialize	OPEN(R)		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	LBRET[1]										X		X								
Process	GET	X	X		X	X	X[2]			X[2]	X	X	X	X							X
	PUT	X		X					X	X[4]	X[3]	X	X		X						X
	PUTR[11]	X																			
	READ		X		X	X		X			X		X						X		
	WRITE										X		X			X	X	X	X		
	CHECK				X	X					X		X								
	RELSE[5]										X		X								
	TRUNK[6]										X		X								
	WAITF		X		X	X		X								X	X	X	X		
	RDLNE		X																		
	RESCN		X																		
	DSPLY		X																		
Set Mode	SETFL																X				
	ENDFL																X				
	SETL																				X
	ESETL																				X
	SETDEV							X													
Non Data Operations	CNTRL[7]		X	X			X	X	X	X	X[12]		X			X					
	CHNG												X								
	PRTOV			X[10]																	
	DISEN				X	X															
	LITE				X[9]	X[9]															
	ERET										X	X	X				X	X	X	X	X
Work Files for DASD and Magnetic Tape	READ										X		X								
	WRITE										X		X								
	CHECK										X		X								
	NOTE										X		X								
	POINTR										X		X								
	POINTW										X		X								
	POINTS										X		X								
Complete	CLOSE(R)		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	FEOV												X								
	FEOVD										X										
	FREE										X[8]				X						
	LBRET[1]										X		X								
	SEOV												X								

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

Figure 5. Logical IOCS Imperative Macros and Devices

IMPERATIVE MACRC EXPANSIONS

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

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

CHECK Macro

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

* Optional

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

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

CLOSE Macro

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

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

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

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

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

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

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

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

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

CLOSER Macro

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

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

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

&N = N of the last file +1.

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

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

CNTRL Macro

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

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

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

CNTRL expansion for the 3886 Optical Character Reader.

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

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

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

Procedure: The control routine waits for completion of any previous operation of the file. Then the device symbolic address is moved to the CCB. The command code is moved to the CCW, and the CCB address is loaded into register 1. Next an SVC 0 is issued to perform the control function indicated by the CNTRL macro instruction. Then control returns to the problem program. CNTRL is treated as a no-op for sequential (DTFSD) files.

DISEN Macro

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

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

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

DSPLY Macro

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

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

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

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

ENDFL Macro

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

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

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

ERET Macro

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

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

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

ESETL Macro

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

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

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

FEOV Macro

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

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

The FEOV macro fetches the EOF/EOV Monitor, $$BCEOV1, to close the current volume and open the new volume.

Procedure: This routine forces an end-of-volume in an output file by setting the EOV switch in the PUT routine. For an input file, the EOV switch in the GET routine is set. For PUT, a record is written as required, and control returns to the user.

FEOVD Macro

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

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

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

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

FREE Macro

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

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

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

GET Macro

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

* Optional

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

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

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

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

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

LBRET Macro

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

Label	LBRET	2	
	SVC	9	Return to logical IOCS.

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

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

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

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

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

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

Procedure for Tape and DASD Labels:

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

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

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

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

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

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

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

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

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

LITE Macro

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

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

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

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

NOTE Macro

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

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

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

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

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

BBB = physical Block Number of the Control interval and

n = the logical block number within the control interval.

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

OPEN Macro

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

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

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

&N = N of the last file +1.

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

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

OPENC Macro

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

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

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

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

OPENR Macro

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

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

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

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

POINTR Macro

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

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

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

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

POINTS Macro

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

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

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

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

POINTW Macro

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

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

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

For a DASD file, the DASD address of the record to be written is calculated. The POINTW routine determines if the record can be contained in the same extent used by the preceding record (the preceding record is the one identified by the NOTE macro). If not, the Sequential DASD Open routine is called to open the required extent. When the correct

extent is obtained, the CCW seek address is modified and the space remaining on the extent is updated in the DTF table. Control then returns to the problem program.

PRTOV Macro

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

* Optional
** Only if CHAN=9

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

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

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

PUT Macro

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

* Optional

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

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

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

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

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

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

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

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

PUTR Macro

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

* Optional

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

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

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

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

RDLNE Macro

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

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

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

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

READ Macro

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

* Portion of macro expansion determined by TYPE= parameter.

```
|Label    |READ |filename,DR,PARAM1,PARAM2
|---------+-----+-------------------------------------------------------------|
|         |L    |1,=A(filename)      |Loads address of DTF table.             |
|         |L    |0,=A(fldname)       |Loads the field name specified by PARAM1.|
|         |LR   |0,r                 |Loads the register specified by PARAM1. |
|         |BAL  |0,*+6               |                                        |
|         |DC   |AL1(PARAM1)         | Generated if PARAM1 and PARAM2 are numbers.|
|         |DC   |AL1(PARAM2)         |                                        |
|         |L    |15,16(1)            |Loads address of logic module.          |
|         |BAL  |14,8(15)            |Branch to read routine in logic module. |
|-------------------------------------------------------------------------------|
|READ macro expansion for the 3886 Optical Character Reader.                    |
```

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

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

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

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

RELEASE Macro -- Dynamic Device Release

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

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

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

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

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

RELSE Macro

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

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

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

RESCN Macro

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

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

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

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

SEOV Macro

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

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

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

SETDEV Macro

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

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

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

SETFL Macro

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

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

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

SETL Macro

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

*Optional

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

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

TRUNC Macro

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

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

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

WAITF Macro

Label	WAITF	filename[1],filename[2],...filenamen	
	ST	SYSLIST(n,1), IJJW&SYSNDX+n*4	Stores end of list code, n+1, in last entry in file list.
	L	1,=A(filenamen)	Loads address of DTF table.
	L	15,16(1)	Loads address of logic module.
IJJW&SYSNDX	BAL	14,4(15)	Branch to WAITF routine in logic module.
	DC	A(SYSLIST(n))	Address of file list.

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

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

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

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

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

WRITE Macro

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

* For RECFORM = FIXUNB.
** Optional

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

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

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

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

Example of a GET Macro

For this example of a GET macro (see Figure 6), assume that both the DTF table and the logic module were assembled separately, the GET macro expansion generated by the assembler, and symbolic addresses resolved by the linkage editor. IJFFZZZ (symbolic address) is generated as the name of the logic module and is included in the DTF table because of certain parameters specified in both the MTMOD and DTFMT macro instructions.

At object time, when the GET macro and its expansion are encountered, the starting address of the DTF table symbolized by the name OLDMSTR is loaded into general register 1, which serves as a base register for the DTF table. The next instruction in the GET macro expansion loads the contents of bytes 16-19 of the DTF table into general register 15 (bytes 17, 18, and 19 of any DTF table always contain the V-type ADCON for the logic module). This retrieves the address of the required logical IOCS data handling module. The third instruction in the GET macro expansion stores the address of the next sequential instruction (NSI) of the problem program in return register 14 and causes a branch to the logic module.

An assembly listing of the IJFFZZZZ logic module shows a branch instruction (B IJFFGET) eight bytes from the start of the module. When the BAL instruction in the GET macro expansion passes control to this point in the logic module, the branch instruction passes control to the GET routine.

The GET routine determines which of the I/O areas is available (in this case, IOAREA1), and inserts the address (IOAREA0) of that I/O area into the CCW. The GET routine, using the CCB contained in the DTF table (the address of the CCB is loaded into general register 1 by the first instruction in the GET macro expansion) issues an SVC 0 for the device assigned to SYS001. The resulting physical I/O operation makes the next record in the OLDMSTR file available to the problem program in location IOAREA0. After ensuring that the physical I/O operation and the transfer of data is complete, control returns to the next sequential instruction (NSI) in the problem program via a branch to the address stored in general register 14.

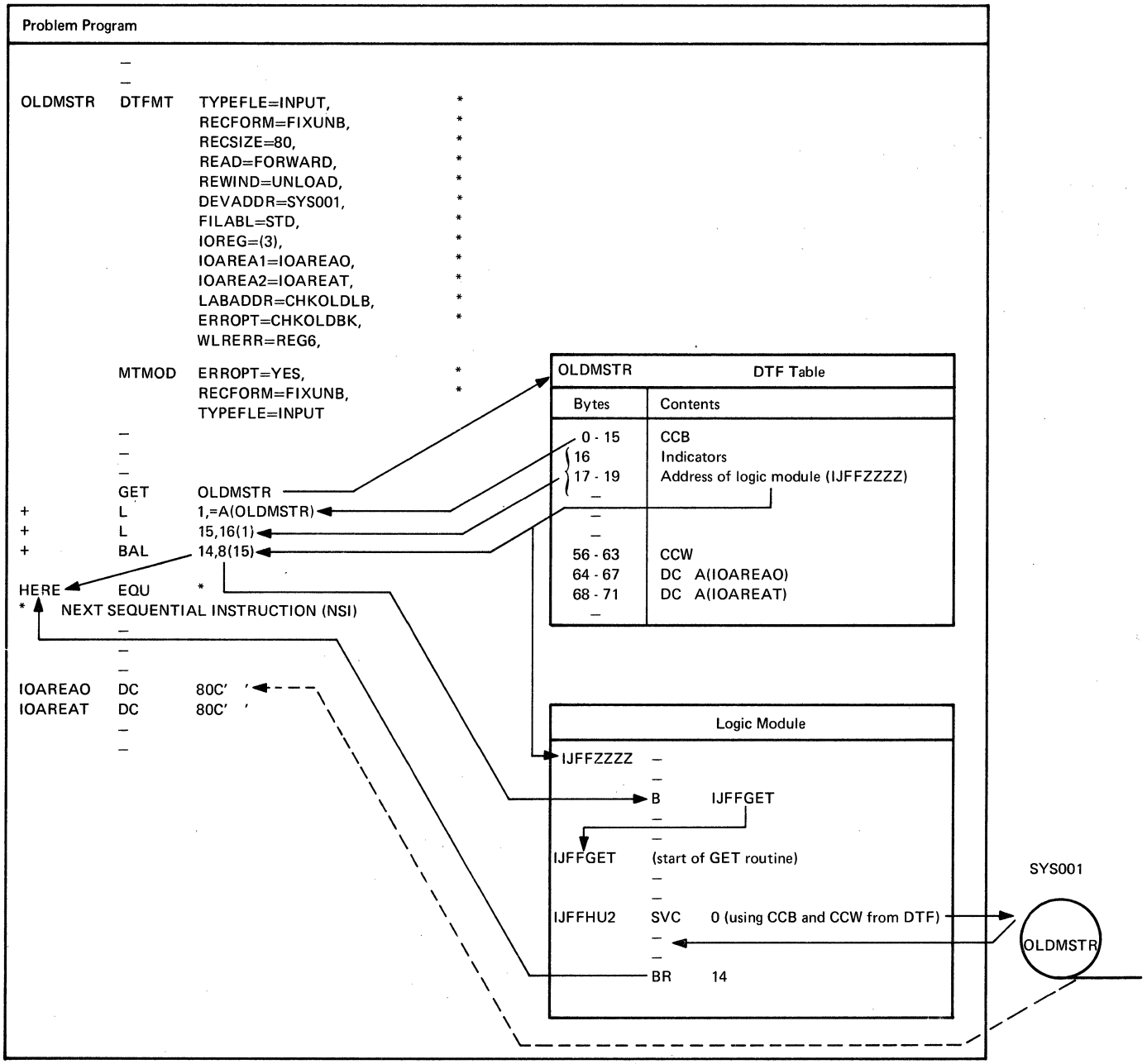


Figure 6. Example of a GET Macro

FILE INITIALIZATION AND TERMINATION

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

1. TES Processor ($$BOESTV).
2. Open Monitor ($$BOPEN, $$BOPEN1, $$BOPEN2, $$BOPEN4, and $$BOPLBL).
3. Close Monitor ($$BCLOSE, $$BCLOS2, $$BCLOS3, $$BCLOS4, $$BCLRPS, and $$BCLLBL).
4. EOF/EOV Monitor ($$BCEOV1).
5. Device or file-processing method oriented open and close transients.

OPEN ROUTINES CHARTS 01-04

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

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

Unit Record and 3881 Optical Mark Reader Files

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

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

Magnetic Ink Character Recognition Files

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

Optical Reader Files (Except 3881)

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

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

Magnetic Tape Files

When opening magnetic tape files, the Open Monitor checks the label set and determines which of the magnetic tape open transients

is needed. The required transient (see Chart 03) is then fetched to complete the open.

DASD Files

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

Diskette Files

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

CLOSE ROUTINES CHARTS 05, 06

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

Unit Record Files (Except MICR)

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

MICR (Magnetic Ink Character Recognition) Files

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

Magnetic Tape Files

For magnetic tape files, the close function is accomplished by logical transients called by either the Close Monitor ($$BCLOSE) or by the EOF/FOV Monitor ($$BCEOV1, Chart 06).

DASD Files

For DASD files processed by SAM the Close Monitor calls $$BOSFBL to link to the $IJJGTOP SVA phase to complete the close function. For DASD files processed by ISAM, the Close Monitor calls $$BCIS0A to update and rewrite the format-1 and format-2 standard file labels, and to set the close indicator in the DTF table. If an ISAM DTF is linked with a VSAM file, ISCCLOSE is called. For DASD files processed by DAM, $$BCLRPS is called to free storage that was obtained for the DTF extension.

Diskette Files

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

FILE LABELING

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

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

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

User labels are not supported for diskette files.

Detailed information about the labels

can be found in the DASD and Tape labels books listed in the Preface.

Creation of Tape Volume Labels

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

Standard Tape File Labels

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

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

Additional File Labels

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

LIOCS bypasses additional header labels on input files.

User Header and Trailer Labels on Tape

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

Tapemarks with Standard Tape Labels

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

1. No tapemark preceding the header label set.
2. Header label set:
   a. Standard volume label (required).
   b. Additional volume labels (0-7, optional: EBCDIC only).
   c. Additional user volume labels (0-9, optional: American National Standards Institute, Inc., only).
   d. Standard file header label (required).
   e. Additional file labels (0-7, EBCDIC: 0-8, American National Standards Institute, Inc., optional).
   f. User header labels (0-8, EBCDIC: or range 2/0-5/14 except 2/7, American National Standards Institute, Inc., optional).
3. Tapemark between header label set and first data record.
4. Physical data records for file.
5. Tapemark between last data record and trailer label set.
6. Trailer label set:
   a. Standard file trailer label (required at end-of-file and end-of-volume).

b. Additional file labels (0-7, EBCDIC: 0-8, American National Standards Institute, Inc., optional).

c. User trailer labels (0-8, EBCDIC: range 2/0-5/14 except 2/7 (quotation mark), American National Standards Institute, Inc., optional).

7. Tapemark after trailer label set.

8. If multifile reel (EOF label), next standard file header label follows here. If single-file reel (EOF label) or if last file of a multifile reel, another tapemark follows here. If multireel file (EOV label), one tapemark follows the EOV label on an EBCDIC file. Two tapemarks follow the EOV label on a multireel ASCII file.

Standard Tape Label Processing

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

The actual label processing consists of the following checks:

Tape Input File:

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

Tape Output File:

- The volume serial number in the standard volume label on the first or only reel is compared to the file serial number in the // TLBL card. All other volume labels on all reels are bypassed.
- The expiration date in the standard file header label is checked against the today's date in the communications region. If the expiration date has passed, the reel is backspaced to write the new standard file label. If not, the operator is notified of the condition. This check is performed on each reel of a multireel output file. If no file label is present, the tape is considered expired. For an expired 9-track tape, the user-specified density is compared to the VOL1 density of the mounted tape. If a discrepancy is found, and if the tape is at load point, the volume label(s) is rewritten according to the user-specified density.
- The new standard file label is written with the information supplied in the // TLBL card. For multireel files, the volume sequence number (EBCDIC file) or file section number (ASCII file) is increased by 1 for each successive reel.
- If user header labels are indicated, the user's label routine is entered to furnish the labels as each reel is opened. This can be done for as many as eight user header labels per EBCDIC file and for an unlimited number of user header labels per ASCII file.
- If end of reel is sensed before completing the file, an EOV trailer label is written with all fields presented in the // TLBL card plus a block count.
- When end of file is reached, an EOF trailer label is written identical to the EOV label previously mentioned.
- If user trailer labels are indicated, the user's label routine is entered to furnish the labels after each trailer (EOV or EOF) label is written. This can be done for as many as eight user trailer labels for EBCDIC files and an unlimited number of trailer labels for ASCII labels.

NONSTANDARD TAPE LABELS

Any tape labels that do not conform to the standard label specifications are

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

1. Nonstandard label(s), followed by a tapemark, to be checked.
2. Nonstandard label(s), not followed by a tapemark, to be checked.
3. Nonstandard label(s), followed by a tapemark, not to be checked.
4. Nonstandard label(s), not followed by a tapemark, not to be checked.

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

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

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

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

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

Unlabeled Tape Files

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

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

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

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

DASD Label Processing

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

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

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

DASD Input Files:

- The volume serial numbers in the volume labels are compared to the volume serial numbers in the DLBL/EXTENT cards.
- The file identification, format identifier, and the file serial number in the format-1 label are compared to the corresponding fields in the DLBL card. The volume sequence number, the creation and expiration dates are then

checked against their EBCDIC equivalents in the DLBL card.

- Each of the extent definitions in the format-1 and format-3 labels is checked against the limit fields supplied in the EXTENT cards.
- If user header labels are indicated (when DTFSD, DTFSR, DTFPH, or DTFDA are used), they are read as each volume is opened. After reading each label, the open routine branches to the user's label routine to perform any processing necessary.
- If user trailer labels are indicated (when DTFSD or DTFSR are used), they are read after reaching the end of the last extent on each volume or an end-of-file read by logical IOCS. As with the user header labels, the trailer labels are processed by the user's routine.

DASD Output Files:

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

Diskette Label Processing

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

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

Diskette Input Files

- The volume serial numbers in the labels are compared to the serial numbers in the DLBL/EXTENT cards.
- If 'file ID' is supplied on the DLBL card, then that file on the diskette is processed (if found). If 'file ID' is omitted, the DTF name is used.
- Both volume and file security label fields are examined and handled to ensure data integrity.
- All symbolic units specified in the EXTENT cards are checked to ensure that only one physical unit is being addressed. This is necessary to ensure that only one file is open on a diskette.
- The extent limits in the file label are checked for validity; if they are found to be correct, the DTF is initialized.
- For multivolume diskette input files using DTFDU, the extent cards and the multivolume indicator are used in conjunction by the OPEN transients to determine when end-of-file has occurred. If three extents were provided by the user, the following multivolume

indicator combination could occur:

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

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

Diskette Output Files

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

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

Included in this section are:

- TES Processor ($$BOESTV)
- Open Monitor ($$BOPEN, $$BOPEN1, $$BOPEN2, $$BOPEN4, and $$BOPLBL)
- Close Monitor ($$BCLOSE, $$BCLOS2, $$BCLOS3, $$BCLOS4, $$BCLLBL, and $$BCLRPS)
- Open for self-relocating programs ($$BOPENR, and $$BOPNR2, and $$BOPNR3)
- RPS SVA initialization routine ($$BOPENS) and RPS phase loading routine ($$VOPENT).
- DASD File Protect and VTOC Display and Dump routines.
- DASD RPS Common Close ($$BCLRPS)
- Check Duplicate Device Assignments for Logical Units ($$BOPENC)
- Enqueue and Dequeue for VSE/VSAM Routines ($$BENDQB)
- SD Close Input and Output ($$BOSDC1)
- Close, Free Track Function ($$BOSDC2)
- Forced End of Volume for Disk ($$BOSDEV)
- Dequeue Extent JIBs ($$BODQUE)
- Device Release ($$BRELSE)

The Charts section contains the detailed flowcharts for each of the routines discussed.

$$BOESTV: Error Statistics by Tape Volume Charts FN-FO

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

Entry:

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

Exit: To next transient.

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

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

$$BOPEN: Open Monitor Charts AA-AB

Objective:

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

2. Tape Error Recording Routine for Job Control open.

Entry:

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

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

Method:

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

$$BOPEN1: Open Monitor Phase 1 Charts AE-AJ

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

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

Exits:

- To $$BOSFBL for DTF type code X'20' and X'21'.
- To $$BOPLBL, and then to $$BOPEN2 for standard labeled magnetic tape files or ISAM files.
- To $$B35400 for diskette files.
- To an appropriate open routine if other files are to be opened.
- To a message writer routine if an error has occurred.
- To the problem program if no more files are to be opened.
- To $$BOESTV for tape DTFs to do recording.

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

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

- Console (DTFCN) files are ignored.
- Unit record (DTFCD, DTFPR, and DTFPT), optical reader (DTFOR), magnetic ink character recognition (DTFMR), compiler (DTFCP), and basic telecommunication access method - extended support (BTAM-ES) files are checked to validate the address limits of the respective DTF tables and the proper open phase is fetched.
- Magnetic tape (DTFMT or DTFPH-MT) files are tested to determine whether they are:
  1. work files,
  2. nonstandard label or unlabeled files, or
  3. standard label output or input forward or input backward.

From this information the name of the proper tape open routine is determined.

For output files with an IOREG specified by the user, $$BOPEN1 stores the address of IOAREA1 in the save area for that register. For the output and the standard label input files, $$BOPEN1 fetches $$BOPLBL, to GETVIS an area for the labels. $$BOPLBL then calls $$BOPEN2 of the Open Monitor. For nonstandard labeled tape files, $$BOPEN1 loads the user's IOREG, if specified, with the address of IOAREA1 and fetches the proper tape open phase directly.

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

$$BOPEN4: DASD DTF DEV Type Update OPEN Phase Charts HG-HH

Objective:

1. To locate the PUB for the DASD, using the corresponding LUB pointer.
2. To test the PUB to make sure it is used for a 3340.
3. To check the VOL ID to make sure that the corresponding 3340 is ready and the VOL ID is correct.

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

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

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

$$BOPIGN: Open Ignore Charts AL-AM

Objective: To check for the COBOL open ignore option.

Entry: From $$BOPEN1.

Exits:

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

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

$$BOPEN2: Open Monitor, Phase 2 Charts AN-AQ

Objective: To read label information from the label area for standard labeled magnetic tape and DASD files, and to fetch the required open phase for the file being opened.

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

Exits:

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

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

For magnetic tape files no checking is required, and the appropriate tape open phase (determined by phase $$BOPEN1) is set up to be fetched after recording has been done in $$BOESTV to provide the additional processing required to open the file.

IBM System/360 Assembler Coding Form

X28-6509-3 U. M050
Printed in U.S.A.

Name	Operation	Operand	Comments	Identification-Sequence
OLDMSTR	DTFMT	TYPEFLE=INPUT,RECFORM=FIXBLK,BLKSIZE=400,RECSIZE=80,		X
		READ=FORWARD,REWIND=UNLOAD,		X
		DEVADDR=SYS001,		X
		FILABL=STD,		X
		IOREG=(3),		X
		IOAREA1=AREAONE,		X
		LABADDR=CKOLDLAB,		X
		ERROPT=CKOLDBLK,		X
		WLRERR=REG6,		X
		EOFADDR=EOFMSTR		X
START	CNOP	0,4		
	BALR	12,0		
	USING	*,12		
	OPEN	OLDMSTR		
	.			
	.			
	.			
	.			
	CLOSE	OLDMSTR		
EOFCD	EOJ			

* A standard card form, IBM electro 6509, is available for punching source statements from this form.

Figure 7. Sample OPEN DTFMT Macro Instruction

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

Example of the Open Function

This example (see Figure 8) shows functions performed to open a magnetic tape data file with symbolic filename OLDMSTR.

Figure 7 shows a typical DTFMT macro instruction as it might appear in a problem program. The macro instruction causes the assembler to generate a corresponding DTF table, a portion of which is included in Figure 8, for the file OLDMSTR (refer to Volume 2 for the detailed entries in the DTF table for magnetic tape data files). Three of the DTFMT macro parameters determine which magnetic tape open phase is required to open the file. They are:

- TYPEFLE=INPUT
- READ=FORWARD
- FILABL=STD

In addition to performing other functions, these parameters generate a X'14' in the type byte (byte 20) of the DTF table. This type code controls the procedure followed by the Open Monitor in selecting the proper open procedure. In this case, X'14' indicates to the Open Monitor that phase $$BOMT01 (Open Input Standard Labels, Forward) is needed to complete the open function for the file OLDMSTR.

The assembler generates the following expansion for the OPEN OLDMSTR macro instruction:

```
                 |  CNOP  0,4
                 |  LA    1,=C'$$BOPEN'
                 |  BAL   0,IJJO&SYSNDX
                 |  DC    A (OLDMSTR)
IJJO&SYSNDX      |  SVC   2
```

- CNOP 0,4 is for boundary alignment.
- LA loads the address of the name of

phase 1 of the Open Monitor into register 1. This is used by the supervisor to call the Open Monitor.

- BAL causes a branch to the SVC 2 instruction and saves the address of the ADCON A(OLDMSTR), which provides linkage so the open routines can address the DTFMT table named OLDMSTR. It also serves as linkage to allow return to the problem program when the open routine issues a SVC 11.

- SVC 2 instructs the supervisor to call and branch to the TES processor, $$BOPEN. $$BOPEN creates the TES record for the previously processed magnetic tape unit and then writes the record on SYSREC.

- $$BOPEN then calls the Open Monitor, $$BOPEN1, that locates the address of:

  1. The user's register save area in storage.

  2. The label save area in storage.

  3. The DTF table for the file OLDMSTR.

From byte 20 of the DTF table, $$BOPEN determines the DTF type (X'14') and initializes to fetch $$BOMT01 after $$BOPEN2.

$$BOESTV is called and executed. $$BOESTV creates the TES record for the previously processed magnetic tape and writes the record on SYSREC. Then $$BOPEN2 is called and executed.

Prior to $$BOPEN2, $$BOPLBL has checked if $JOBACCT is active. If the job accounting interface is active, the label save area in the supervisor is used. $$BOPEN2 also searches the label area for the label information stored by job control for the file OLDMSTR. When this information is found, it is read into the label save area in main storage. $$BOPEN2 then fetches the tape open phase ($$BOMT01) determined by $$BOPEN1.

$$BOMT01 checks the actual tape labels against the label information in the label save area. If no errors are detected, this phase posts the file open in the DTF table (bit 5 of byte 36) and recalls $$BOPEN1 of the Open Monitor. The Open Monitor, in turn, returns control to the problem program. (In this example, only one file is specified in the operand of the OPEN macro. If the operand contains the names of more files, the Open Monitor opens the remaining files before returning control to the problem program.)

$$BOPLBL: Open Monitor Label Space Processor Chart AK

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

Entry: From $$BOPEN1.

Exit:

- To $$BOPEN2 for tape.
- To $$BOPEN4 for ISAM.
- To $$BOMSG1 if an error occurs.

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

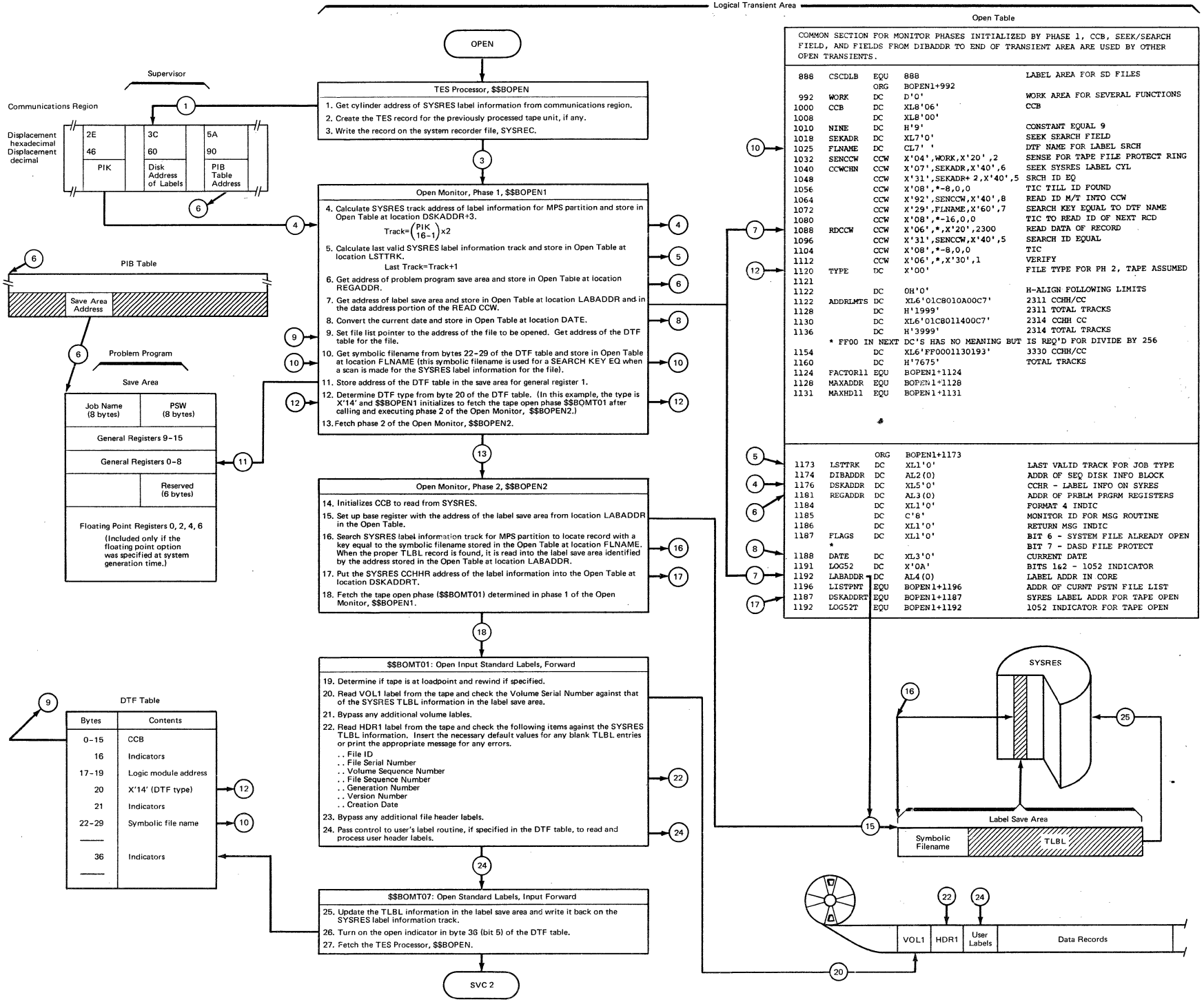


Figure 8. Example of the Open Function

$$BOPENR: Relocate DTF Address Constants Charts AS - AV

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

Entry: From the OPENR macro to the label START.

Exits:

- To $$BOPNR3.
- To the Open Monitor, $$BOPEN, when the last DTF table is processed.

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

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

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

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

$$BOPENC: Check Duplicate Device Assignments for Logical Units Chart AW

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

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

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

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

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

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

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

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

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

Method: When a VSE/VSAM routine issues the ENQB macro, $$BENDQB is fetched (via SVC 2) from the core image library and put into the B-transient area. Control is transferred to $$BENDQB, which temporarily returns control (via SVC 8) to the routine that issued the ENQB macro. (The B-transient area is not released.) When the DEQB macro is issued, control is returned (via SVC 9) to the B-transient routine $$BENDQB, which has been previously loaded into the transient area by the ENQB macro. $$BENDQB now executes an SVC 11 to release the B-transient area and to return to the highest-priority program ready to run. (Note: The ENQB and DEQB macros destroy the original contents of registers 0 and 1.

$$BOPNR2: Relocate DTF Address Constants, Phase 2 Charts FA-BC

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

Entry: From $$BOPNR3.

Exit: To $$BOPEN.

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

$$BOPNR3: Relocate DTF Address Constants, Phase 3 Charts BE-BF

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

Entry: From $$BOPENR.

Exits:

- To $$BOPNR2 if other than unit record files still have to be relocated.
- To the Open Monitor, $$BOPEN, if no more files have to be relocated.

MODLOOP (Address Modification) Subroutine Chart BD

The MODLOOP subroutine performs the actual address modification using an address modification table. The following example of the relocation of a work file DTFMT table (see Figure 9) illustrates the operation of the MODLOOP subroutine and the use of the address modification table.

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

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

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

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

Numbers in parentheses are displacements in hexadecimal notation.

Figure 9. DTFMT Work File Format

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

count now goes to 0, an exit is made from the modification loop.

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

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

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

$$BOPENS: RPS SVA Initialization Routine Chart BS

Objective: To initialize the fetch and page management channel programs for RPS, if this routine was called by IPL. To load the RPS local directory list and phase loading routine into the SVA, if this routine was called by $$BOPEN during the first DASD open.

Entry: From IPL and $$BOPEN.

Exits: To IPL and $$BOPEN.

Method: When called by IPL, the SVA initialization routine determines the device type of SYSRES. If SYSRES is an RPS device, the device type is set in the fetch table for the core image library.

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

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

$$VOPENT: RPS Phase Loading Routine Charts BT-BU

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

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

Exit: To the calling transient.

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

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

$$BCLOSE: Close Monitor, Phase 1 Charts BG-BI

Objective: To determine the DTF file type and to fetch the proper close phase for sequential DASD, DAM, and ISAM files (Phase 2 of the Close Monitor, $$BCLOS2 is fetched to handle other file types. For magnetic tape files, $$BCLOS3 is fetched).

Entry: From a problem program CLOSE macro expansion, or from a successful CLOSE if more than one file is specified by the same CLOSE macro instruction. $$BCLOSE is also entered from phase 2 of the Close Monitor, $$BCLOS2. In addition, $$BPCLOS enters $$BCLOSE at EOJ to close any unclosed 3800 printer extended buffering DTFs.

Exits:

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

- To $$BPCLOS when $$BCLOSE was originally invoked by $$BPCLOS.

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

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

For all sequential DASD files, $$BCLOSE fetches the SVA link phase $$BOSFB1 to link to the $IJJGTOP SVA phase to complete the close processing. For ISAM DTFs, $$BCLOS4 is called. For DAM DTFs, $$BCLRTS is called.

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

$$BCLOS2: Close Monitor, Phase 2 Charts BK-BM

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

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

Exits:

- To phase 1 of the Close Monitor, $$BCLOSE, for DTFCP DASD and ISAM-ADD files.
- To $$BCLOSP for punch and paper tape files.
- To $$BCTC01 for BTAM-ES telecommunication files.
- To $$BCLOS3 for magnetic tape (DTFMT) files.
- To $$BCMR01 for magnetic ink character recognition (MICR) type files.
- To the message writer phase, $$BOMSG1, if an invalid file type is detected.
- To $$BPCLOS (the 3800 printer automatic close module) when CLOSE was issued by $$BPCLOS.

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

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

$$BCLOS3: Close Monitor, Phase 3 Chart BN

Objective: To initiate the proper close procedure for magnetic tape files.

Entry: From $$BCLOS2.

Exit: To the appropriate close phase for magnetic tape (DTFMT) files.

Method: The function performed by the third phase of the Close Monitor depends on the file type.

- For DTFMT work files, $$BCLOS3 saves the console indicator in the open table for use by the next phase, and fetches phase $$BCMT06 to close the file.
- For all DTFMT input files, $$BCLOS3 initializes the deblocker areas in the DTF table. For standard labeled input and output files, SYSRES label information is read into the open table for use by the next phase. For blocked output files, the address of the logic module is stored in the save area for register 11. The address of the TRUNC routine is substituted for the logic module address in the DTF and the TRUNC routine is executed. $$BCLOS3 then saves the console indicator in the open table, and fetches phase $$BCMT05 to close the file.
- $$BCLOS3 calculates the PUB2 address of a tape DTF to be closed and saves it in the open table.

$$BCLOS4: Close Monitor, Phase 4 Chart BO

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

Entry: From $$BCLOSE.

Exits:

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

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

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

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

$$BCLLBL: Close Monitor Label Space Processor Chart BJ

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

Entry: From $$BCLOSE.

Exit:

- To $$BCLOS2 for tape.
- To $$BCLOS4 for an ISAM file.
- To $$BOMSG1 if an error occurs.

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

$$BCLRPS: DASD RPS Common Close Charts BQ-BR

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

Entry:

- From $$BCLOSE or $$BCLOS2 for DAM or ISAM DTFs.

Exits:

- To $$BCLOSE for direct access or IOCS type DTFs.
- To $$BODACL for direct access type DTFs

with user trailer labels.

- To $$BOISOA for indexed sequential access type DTFs.

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

All access methods use this routine. Therefore, it is necessary to first determine the DTF type, since the displacements are different in each case. Refer to Figures 10 and 11.

DTF	Type Code	Byte	Dev Type Bit[1]	DTF Type Bit[2]	Exit to
DTFDA No trailer labels and DTFPH	22,23	32(20)	1	7	$$BCLOSE
DTFDA with trailer labels	22	32(20)	1	7	$$BODACL
DTFIS (all)	24,25,26,27	65(41)	4,7[3]	5	$$BCISOA

[1] If this bit is set on, the device supports RPS.

[2] If this bit is set on, the DTF extends into the partition virtual area.

[3] Bit 4 on indicates prime data. Bit 7 on indicates index.

Figure 10. Use of Different DTF Types by $$BCLRPS

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

$$BOSDC1: SD Close Input and Output, Charts CA-CB

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

Entry: From $$BCLOSE.

Exits:

- To the CLOSE Monitor, $$BCLOSE.

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

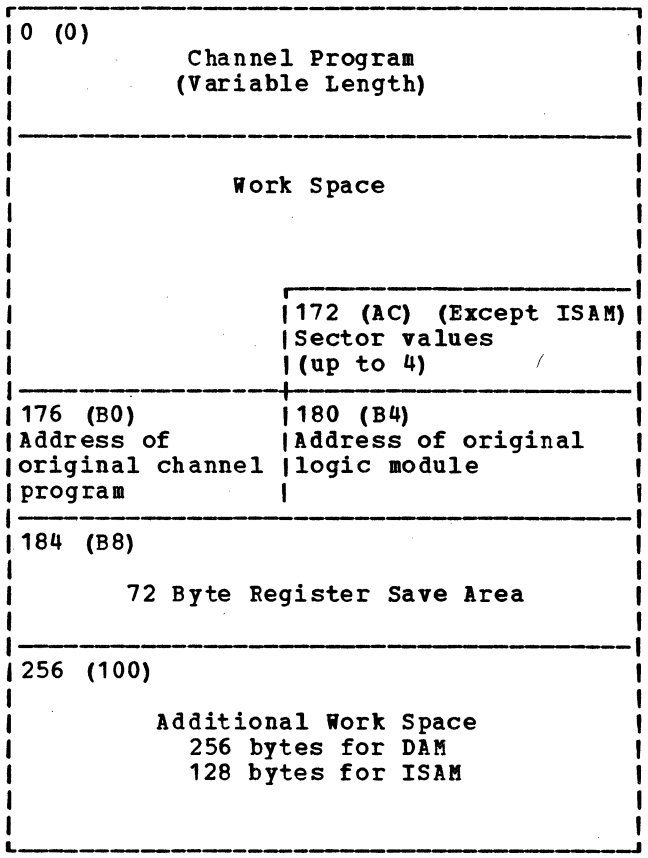


Figure 11. ISAM RPS or DAM DASD Device Independent Extension Work Area

$$BOSDC2: Close: Free Track Function, Chart CC

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

Entry: From ISAM CLOSE.

Exits:

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

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

$$BOSDEV: SD Clcse, Chart CD

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

Entry:

- From the FEOVD macro.

Exits:

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

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

$$BODQUE: Dequeue Extent JIBs, Chart CE

Objective: To find the JIB (Job Information Block) chain for a particular logical unit; and to clear any extent type JIBs associated with the logical unit, and release them to the available JIB chain.

Entry: From the ISAM DASD open phase.

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

Method: After storing the contents of registers 3 through 8 and the name of the phase that is tc be returned to, if specified, phase $$BODQUE locates the proper 2-byte entry in the LUB table for the logical unit specified and examines the second byte of the LUB entry to determine if any JIBs are chained to the LUB. If JIBs are chained to the LUB; that is, if the second byte of the LUB is not hex 'FF', the address of the first JIB in the chain is calculated by adding the pointer (byte 2 of the LUB) multiplied by 4 (the length of a JIB entry) to the starting address of the JIB table.

Byte 2 of the JIB entry is then examined to determine if the JIB contains an extent. If the JIB contains an extent, the extent is cleared. Once the extent is cleared, the pointer to the next JIB in the chain is obtained from the fourth byte of the current JIB. The current JIB is then placed in the available JIB chain and the pointer to the first available JIB (FAVP) is modified accordingly. When the JIB has been placed in the available chain, or if the JIB does not contain an extent, the address of the next JIB in the chain is calculated using the pointer obtained from the fourth byte of the current JIB. The procedure is repeated for the next JIB.

Note: All of the JIB processing described is handled through the supervisor extent interface.

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

$$BRELSE: Device Release Charts EE-EG

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

Entry: Fetched by RELEASE from SYSRES to the transient area.

Exits:

- To the problem program via SVC 11.
- To RELEASE via LINKREG.

Method: To perform the actual device release, the transient sets the unit to the permanent assignment, if one exists. Otherwise, the device is unassigned. If the device is at permanent assignment level, the transient takes no action on the unit. Before any release is attempted, a check is made for ownership of the unit. If the requesting partition does not own the unit, or if the unit is already unassigned, the transient ignores the request.

COMMONLY USED LOGICAL TRANSIENTS

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

$$BOFLPT: DASD File-Protect Charts FA-FC

Objective: To place the upper and lower extent limits into Job Information Blocks (JIBs) to provide file protection for DASD files.

Entry:

- From phase $$BOIS07 for ISAM files.

Exits:

- To the open monitor, $$BOPEN, if more

files are to be opened and a specific phase name is not supplied.

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

Method: The $$BOFLPT phase provides file protection for DASD files by storing extent limit information in the JIB table. For the IBM 2311 Disk Storage Drive, the IBM 2314 Direct Access Storage Facility, and the IBM 2319 Disk Storage Facility, the lower and upper cylinder limits are stored in a single JIB. For the IBM 3330, 3340, and 3350, the extent limit information is stored in two chained JIBs, the first containing the lower extent limit and the second the upper extent limit. The extent JIBs are chained to the Logical Unit Block (LUB) entry to which the device is assigned. Further information pertaining to the JIBs and LUBs is found in VSE/Advanced Functions Diagnosis Reference: Supervisor.

The $$BOFLPT phase begins by determining:

- The number of extents to be processed.
- The addresses of the DLBL-EXTENT card image, FAVP (the pointer to the first available JIB), and the JIB table.
- The file type.
- The device type.

When these factors are known, the phase determines the address of the LUB entry for the logical unit used by the file. The contents of the LUB are then loaded into a pair of registers, LUBADRLL (lower limit) and LUBADRUL (upper limit), that are used to insert the extent information into extent type JIBs.

The second byte of the LUB contains a pointer to the first JIB in the chain for the LUB (if the byte does not contain X'FF', indicating that no JIBs are chained to the LUB). This pointer calculates the address of the JIB. The JIB, in turn, contains a similar pointer that calculates the address of the next JIB in the chain. A pointer of X'FF' indicates the end of the chain.

If extents for the file remain to be processed and one of the following conditions is reached, phase $$BOFLPT obtains and builds a new JIB entry:

- No JIBs are chained to the LUB.
- No extent type JIBs remain in the chain.
- The end of the JIB chain is reached and more JIBs are required.

The address of the new JIB is calculated by using the pointer to the first unused JIB in the JIBs available chain, found in location FAVP in the supervisor. As in the case of JIBs chained to the LUB, this new JIB contains a pointer to the next available JIB that will be used if needed.

After the extent information is stored in the JIB(s), the pointers are modified (as required), to complete the chain and the registers are restored. From information passed by the calling phase, $$BOFLPT determines the next action required and issues either an SVC 2 to fetch the proper transient phase, or an SVC 11 to return to the problem program.

$$BODSPV: VTOC Display, Phase 1 Chart FD

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

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

Exit:

- To the second phase of VTOC display, $$BODSPW. (If a diskette is being displayed, exit is to phase $$BODSPO.)
- To job control via an SVC 11 if the operator's response to message 4V95A is END or CANCEL and the open was for job control.
- To phase $$BCNCL via an SVC 6 to cancel the job if the operator's response to message 4V96A is END or CANCEL and the open was not for job control.

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

$$BODSPW: VTOC Display, Phase 2 Charts FE-FF

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

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

Exit: To $$BOMSG1 or $$BODSMW.

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

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

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

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

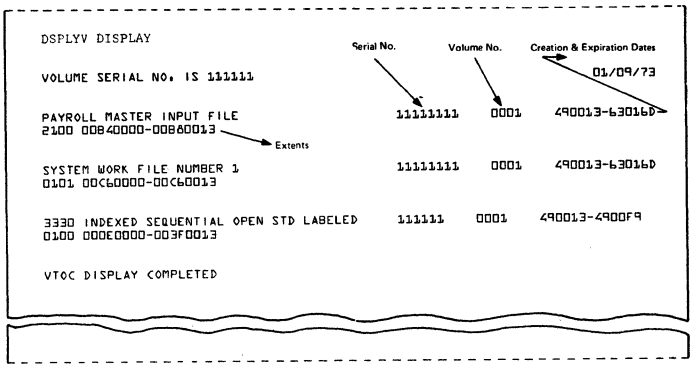


Figure 12. VTOC Display of Disk Pack (DSPLYV Response)

When all format-1 labels in the VTOC have been processed, the message 'VTOC DISPLAY COMPLETED' is printed and control is passed to $$BOVDMP. Figure 12 is a sample of the VTOC display printed by this phase.

$$BODSPO: Diskette VTOC Display, Chart JA

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

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

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

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

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

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

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

When all HDR1 labels in the VTOC have been processed, control is returned to the calling transient. Figure 13 is a sample of the VTOC display printed by this phase.

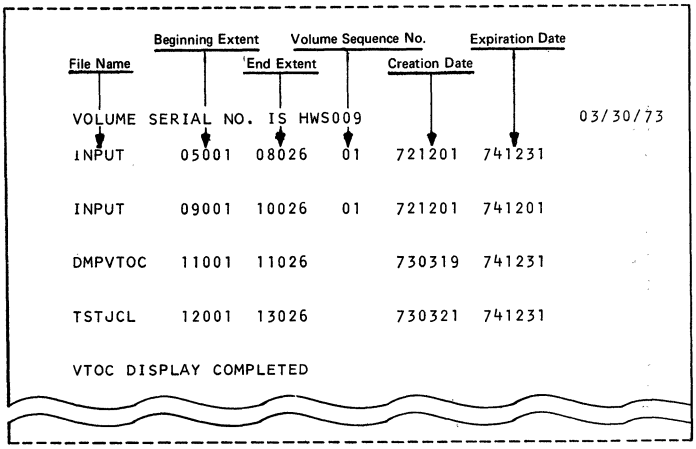


Figure 13. VTOC Display of Diskette (DSPLYV Response)

$$BOVDMO: Diskette VTOC Dump Chart JB

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

Entry: From phase 2 of the Diskette Open Message Writer, $$BODMS2, or $$BODI08, when the operator's response is CANCELV, or from the problem program.

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

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

$$BOWDMO: Diskette List VTOC Chart JC

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

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

Exits: Control returns to job control or to the user's program.
Figure 14 is a sample of the VTOC Dump printed by this phase.

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

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

$$BODMSG: Diskette Open Error Message Writer Phase 1 Chart GA

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

Entry:

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

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

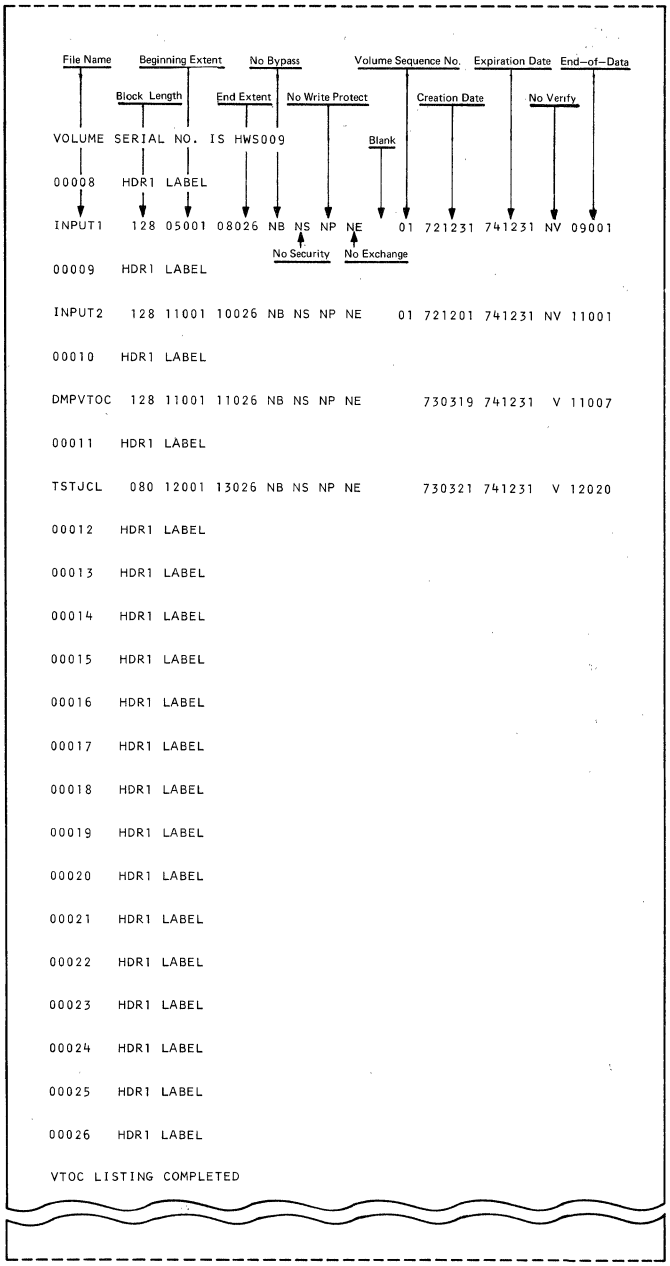


Figure 14. VTOC Dump of Diskette (CANCELV Response)

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

- Register 0 contains the last four characters in the name of the phase requesting the message. On cancel messages, register 0 need not be initialized. $$BO is assumed for the first four characters of the phase name.
- Register 2 contains the address of the DTF table for the current file.
- Register 3 contains the message code (in binary) for the message to be printed. This code is converted to the last two digits of the message number (XX in the example 4nXXI).

- Transient region + 1185 contains the numeric decimal value assigned to the various open/close phases for message numbering (X in the example 4XnnI).

- Transient region + 1000 contains the start of the CCB.

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

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

$$BODMS2: Diskette Open Error Message Writer, Phase 2 Charts GF-GC

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

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

Exit:

- To the VTOC dump phase, $$BOVDMO.

- To phase 1 of the VTOC display routine, $$BODSPV.

- To the diskette open/close organization phase requesting the message (if a cancel was not encountered).

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

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

- Register 3 contains the address of the message to be written on SYSLOG.

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

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

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

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

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

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

1. Operator reply of IGNORE for a D-type message.

2. Equal file ID message.

3. No EXTENT to be bypassed.

4. Next pack not mounted.

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

Figure 16 shows the message code (passed via register 3) together with the last two digits and action indicator of the associated number. For reference purposes, the text of the message is also included.

$$BODSMO: Diskette Data Security Message Writer Chart GD

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

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

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

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

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

$$BOVDMP: VTOC Dump Charts FG-FH

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

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

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

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

$$BOWDMP: List VTOC Chart FI

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

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

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

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

Figure 15 is a sample of the VTOC Dump printed by this phase.

```
CANCELV DISPLAY

VOLUME SERIAL NO. IS  111111

0000000004   FORMAT 4 LABEL

04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  F4000000
000001EF  00C80002  003A8001  000000C8  00141C7E  922D2D01  02161611  00000000  00000000  00000000  00000000  00000000
00000000  00000000  00010000  00000000  00001300  00000000  00000000  00000000  00000000  00000000  00000000

0000000005   FORMAT 5 LABEL

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

0000000006   FORMAT 1 LABEL

PAYROLL MASTER INPUT FILE                    SERIAL NO. 111111  VOL NO. 0001  490013-630160  014040
                                                                                SYS. CODE IS DOS/370 VER 4
40404040  40404000  00000008  00080000  00008040  40404000  00000000  4040
2100  00840000-00880013  0000  00000000-00000000  0000  00000000-00000000                  POINTER IS  0000000000

0000000007   FORMAT 1 LABEL

SYSTEM WORK FILE NUMBER 1                    SERIAL NO. 111111  VOL NO. 0001  490013-630160  010700
                                                                                SYS. CODE IS DOS VERSION 5
00000000  00000040  00000000  00000000  00000000  00000000  00000000  0000
0101  00C60000-00C60013  0000  00000000-00000000  0000  00000000-00000000                  POINTER IS  0000000000

0000000008   FORMAT 1 LABEL

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

VTOC LISTING COMPLETED
```

Figure 15. VTOC Dump of Disk Pack (CANCELV Response)

$$BOMSG1 Disk Open Error Message Writer, Phase 1 Chart FJ

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

Entry:

- From a DASD open or close phase.
- From the DTFCP open phases, $$BOCP01, $$BOCP02, $$EOCP11, or $$BOCP12.

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

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

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

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

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

$$BOMSG2: Disk Open Error Message Writer, Phase 2 Charts FK-FL

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

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

Exit:

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

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

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

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

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

Figure 16 shows the message code (passed via register 3) together with the last two digits and action indicator of the associated message number. For reference purposes, the text of the message is also included.

$$BODSMW Data Security Message Writer Chart FM

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

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

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

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

Message Code	Message Number	Message
0	44A	OVERLAP CN UNEXPRD FILE
1	55A	WRONG PACK, MOUNT nnnnnn
2	40A	EXTENT OVERLAPS ANOTHER
3	41A	EXTENT OVERLAP ON VTOC
4	42A	NO MATCHING EXTENT
5	33A	EQUAL FILE ID IN VTOC
6	66A	1 TRACK USER LBL EXTENT
7	59A	INVALID EXTENT
15	84D	NEED FILE PROTECT RNG
16	31D	VOLUME SEQUENCE ERROR
17	38D	USER HDR LBL IS NOT STD
18	39D	USER TRL LBL IS NOT STD
19	08D	NO UTL0 FILE MARK FOUND
20	47A	EXTENTS NOT ON SAME UNIT
21	86D	TAPE UNIT NOT READY
22	00I	NO RECORD FOUND
23	01I	NO RECORD FOUND
24	02I	NO RECORD FOUND
25	03I	NO RECORD FOUND
26	04I	NO RECORD FOUND
27	05I	NO RECORD FOUND
28	06I	NO RECORD FOUND
29	07I	NO RECORD FOUND
31	09I	NO RECORD FOUND
32	00I	NO LABEL SPACE IN VTOC
33	01I	NO FORMAT 1 LABEL FOUND
34	02I	NO FORMAT 2 LABEL FOUND
35	03I	NO FORMAT 3 LABEL FOUND
36	04I	NO FORMAT 4 LBL IN VTOC

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

Figure 16. Message Code for Disk Open Error Message Writer (Part 1 of 3)

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

Figure 16. Message Code for Disk Open Error Message Writer (Part 2 of 3)

Message Code	Message Number	Message
59	48I	SYSIN/SYSOUT UNSUPPORTED
60	70I	1ST XTNT CD NOT INDX VOL
61	71I	EXTENT INFO NEEDED
62	72I	MOD AND DTF INCOMPATIBLE
63	58I	NO EXTENT FOR OUTPUT FILE
64	88I	EOF ON SYSTEM FILE
67	91I	NO ASCII SUPPORTED SUPVR.
68	98I	OVLAP UNEXPRD SECRD FILE
69	69I	FILE IS OPEN FOR ADD
70	97I	OVLAP EXPIRED SECRD FILE
71	85I	INVALID FORMAT RECORD
74	30I	INVALID HDR1 LABEL
75	33I	EQUAL FILE ID VTOC
76	37I	CHAINING TO SYSTEM UNIT
77	31I	VOLUME SEQUENCE ERROR
80	82I	ISAM NULL FILE
82	74I	BLKSIZE OPEN FAILURE
83	75I	BLKSZ NOT MULT OF RECSZ
85	78I	NO LOGIC MODULE ...
86	79I	GETVIS FAILED
87	05I	UNRECOVERABLE I/O ERROR

Figure 16. Message Code for Disk Open Error Message Writer (Part 3 of 3)

CHARTS

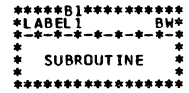
Explanation of Flowchart Symbols

DESCRIPTION

EXAMPLE

A group of program instructions that perform a processing function of the program. The label, if any, is shown above the block.

*B2
IF ANY ADDITIONAL EXPLANATION IS REQUIRED, ITS LOCATION ON THE CHART IS IDENTIFIED BY AN ASTERISK AND THE BLOCK ID.



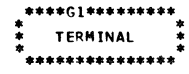
Description or title of a routine that is detailed on another flowchart. The starting label of the routine and the flowchart ID appear above the stripe.

An instruction, or group of instructions, that changes portion of a routine or initializes a routine for a given condition.

A group of operations not detailed in the flowcharts in this manual, such as user routines.

Any function of an input/output device or program, usually branching to an I/O routine to perform the function stated in the block.

Points where the program branches to alternate processing, based upon variable conditions such as program switch settings and test results.



The beginning or end of a program or routine.

On-page connector. An entry from or an exit to another function on the same flowchart. The location in the connector identifies the block to which entry on a chart is made.

Off-page connector. An entry from, or exit to, a given point on another flowchart. The characters in the connector identify the chart and block to which or from which control is passed. The corresponding label, if any, is placed outside the outgoing connector. For multiple entries, an asterisk is placed in the connector and the locations from which control is passed are listed nearby.

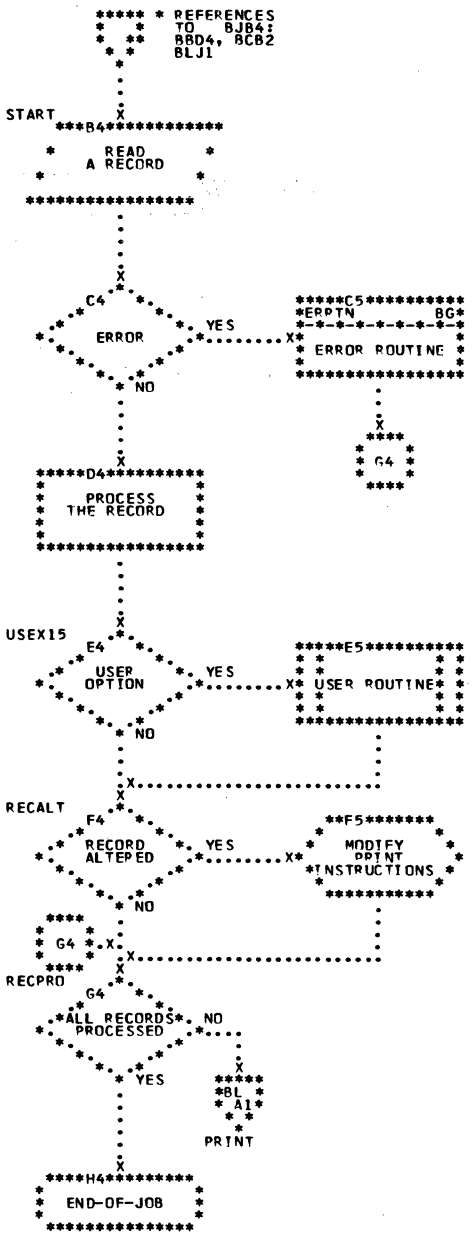


Chart 01. Open Monitor

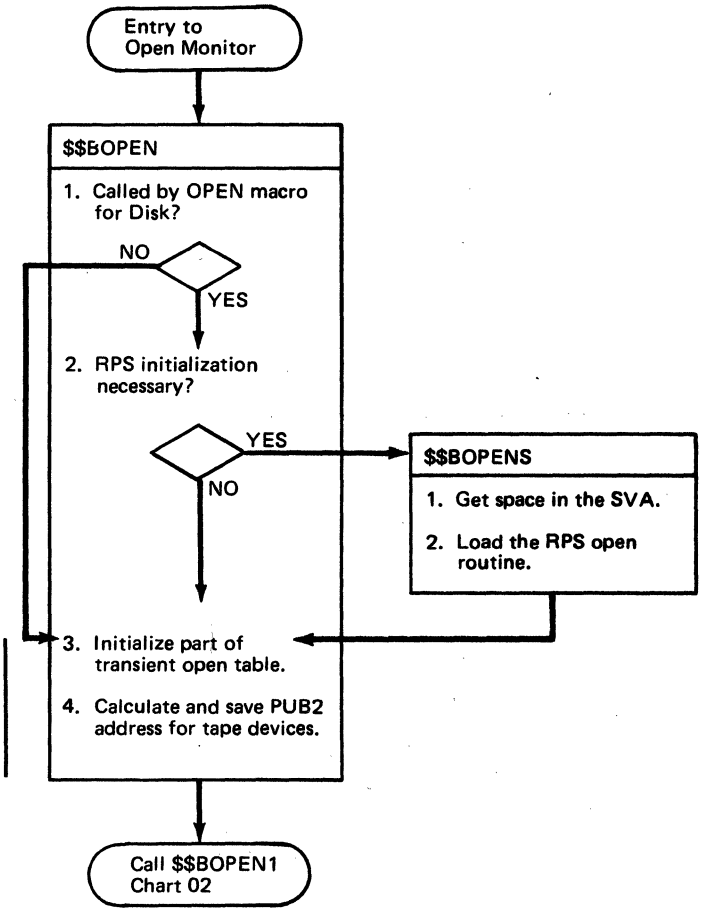
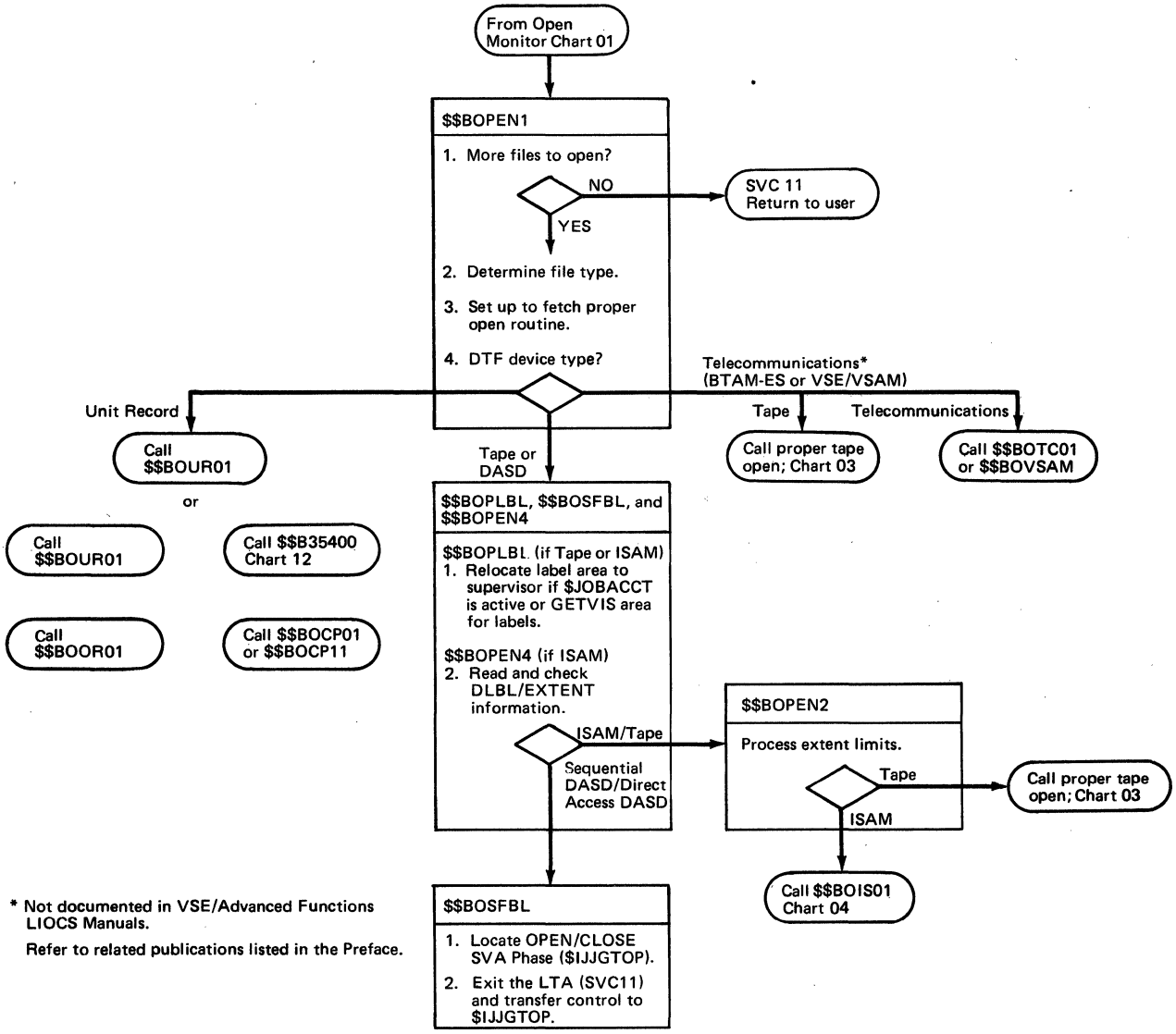


Chart 02. Open Monitor



* Not documented in VSE/Advanced Functions LIOCS Manuals.
Refer to related publications listed in the Preface.

Chart 03. Open Magnetic Tape

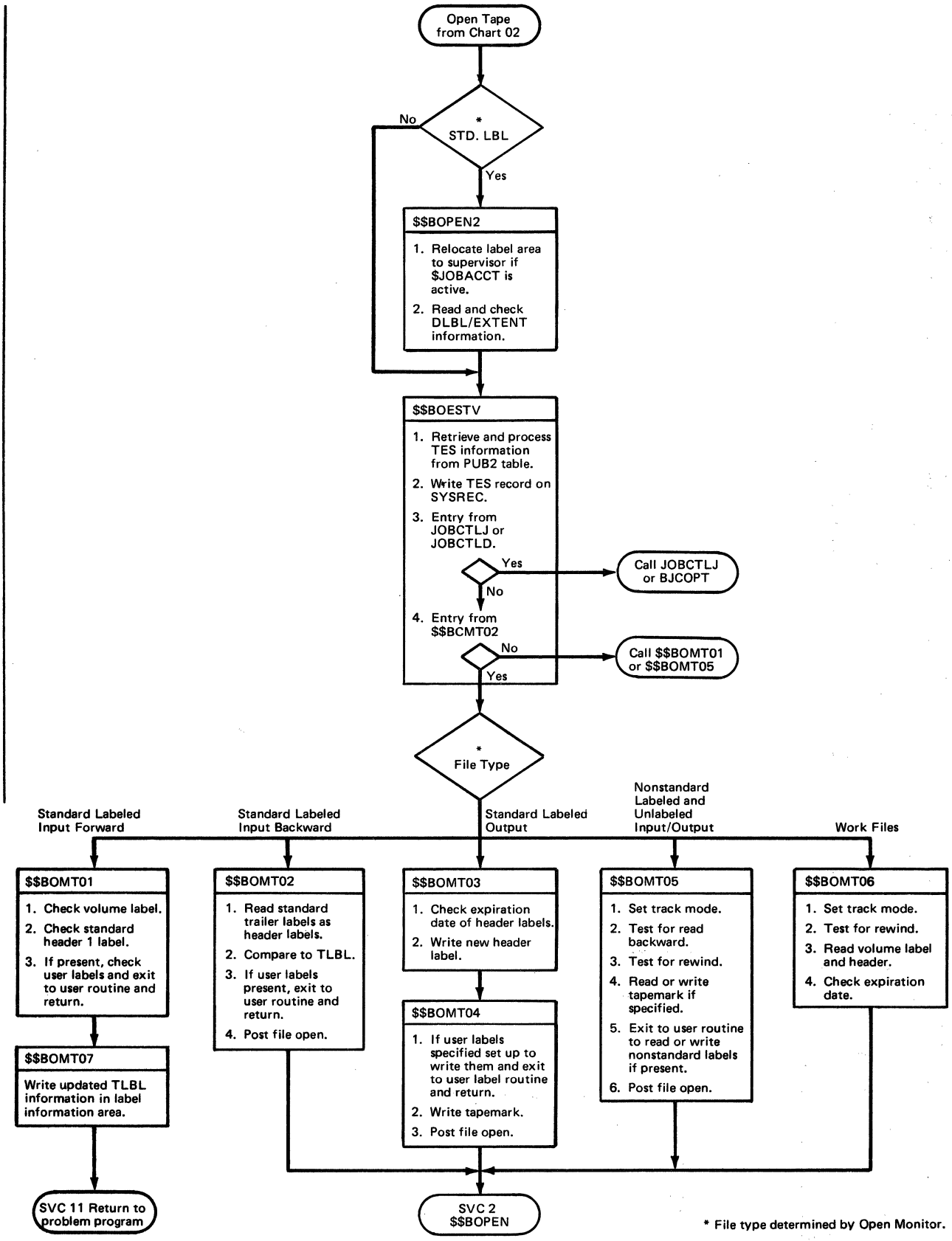


Chart 04. Open ISAM

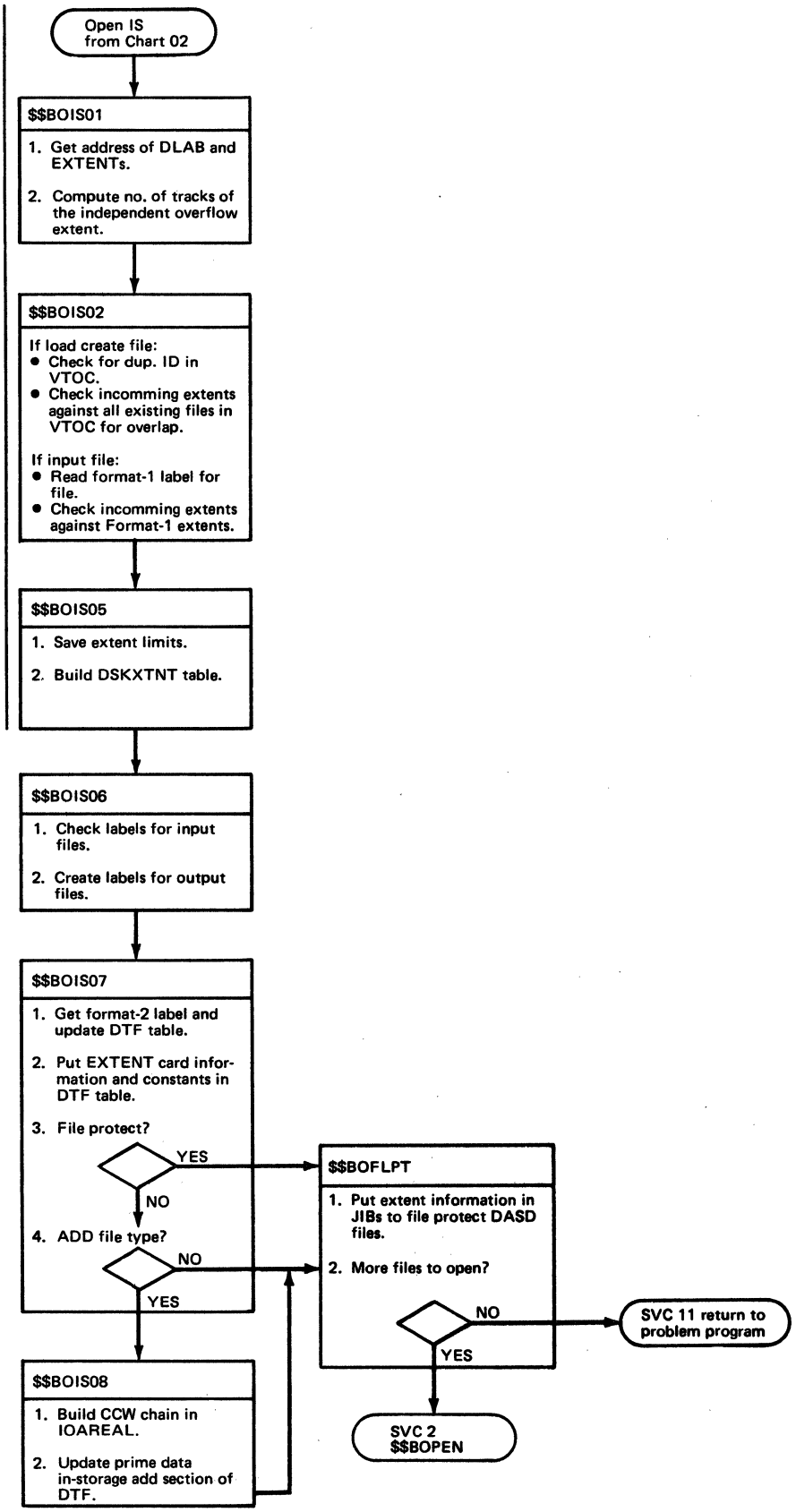
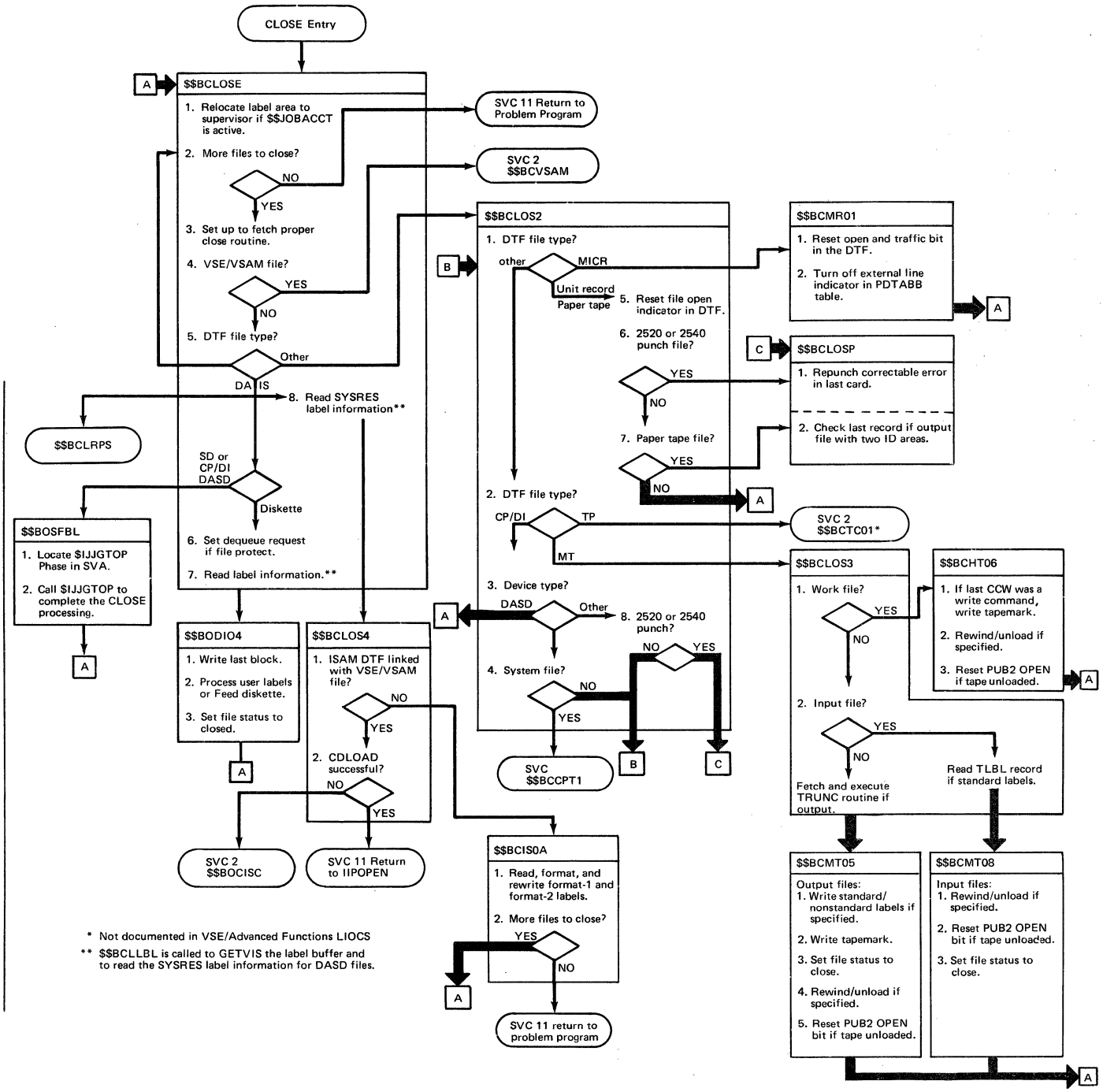


Chart 05. Close Monitor



* Not documented in VSE/Advanced Functions LIOCS

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

Chart 06. EOF/EOV Routine

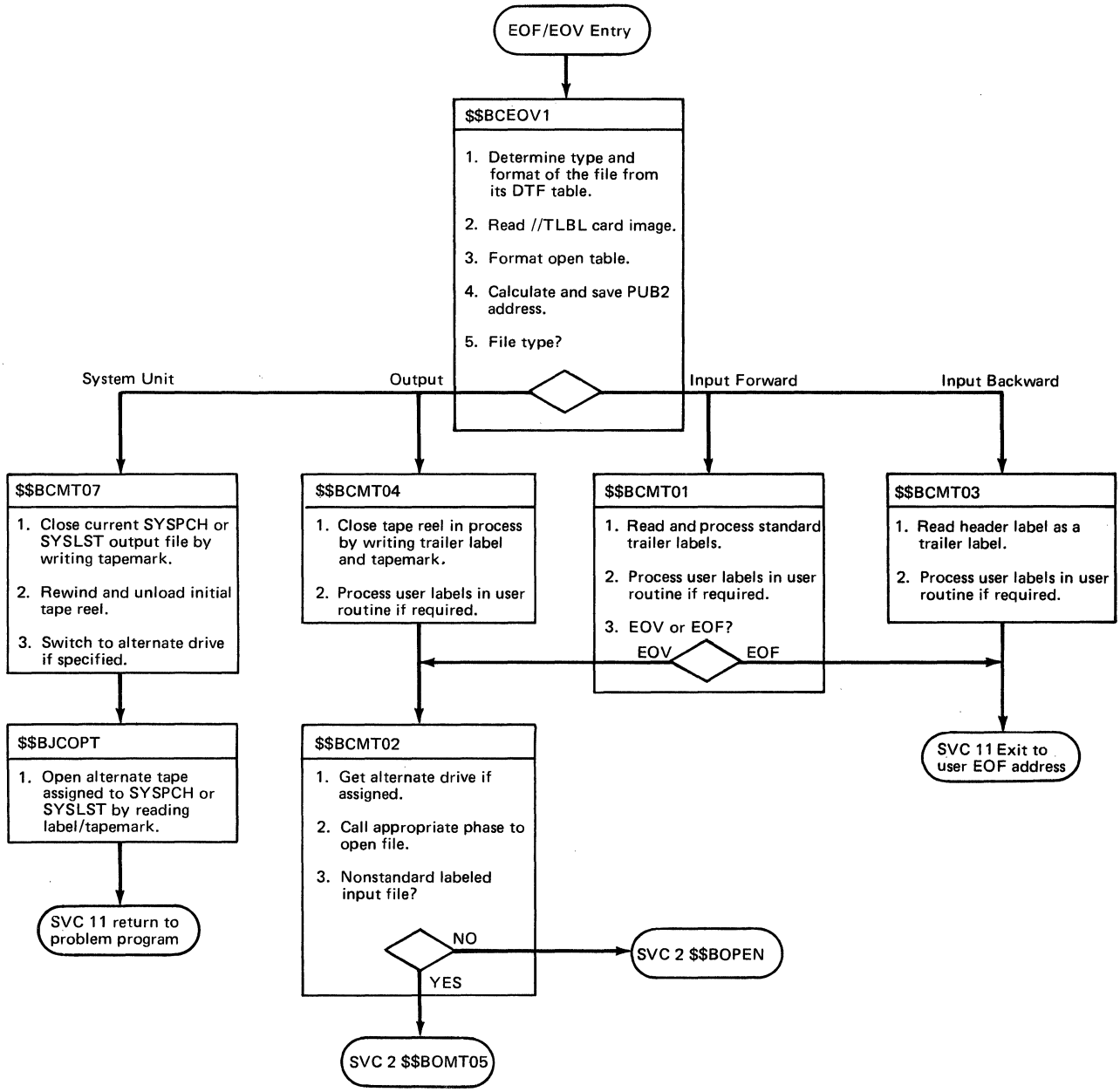


Chart 07. Open Diskette, Input

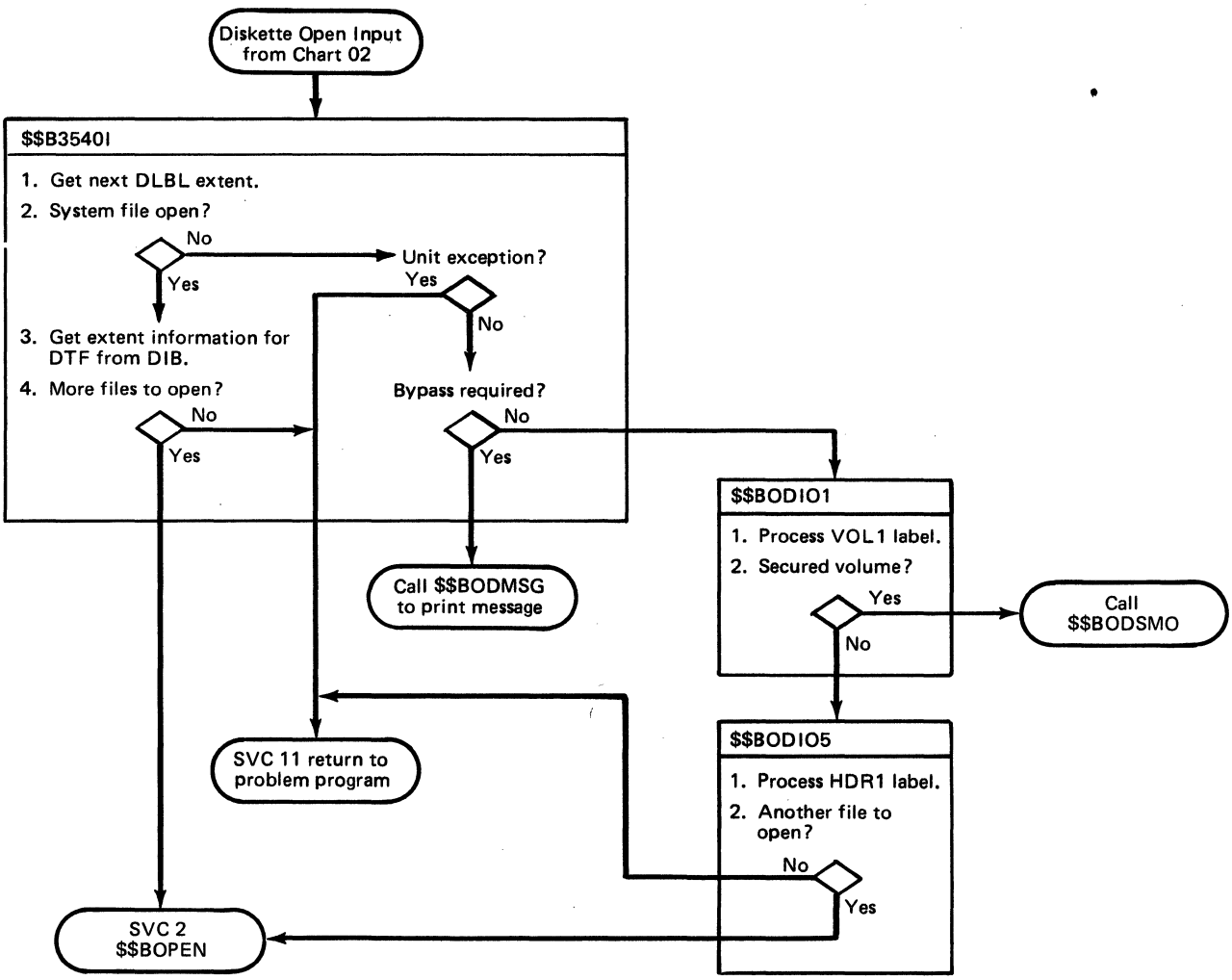


Chart 08. Open Diskette, Output

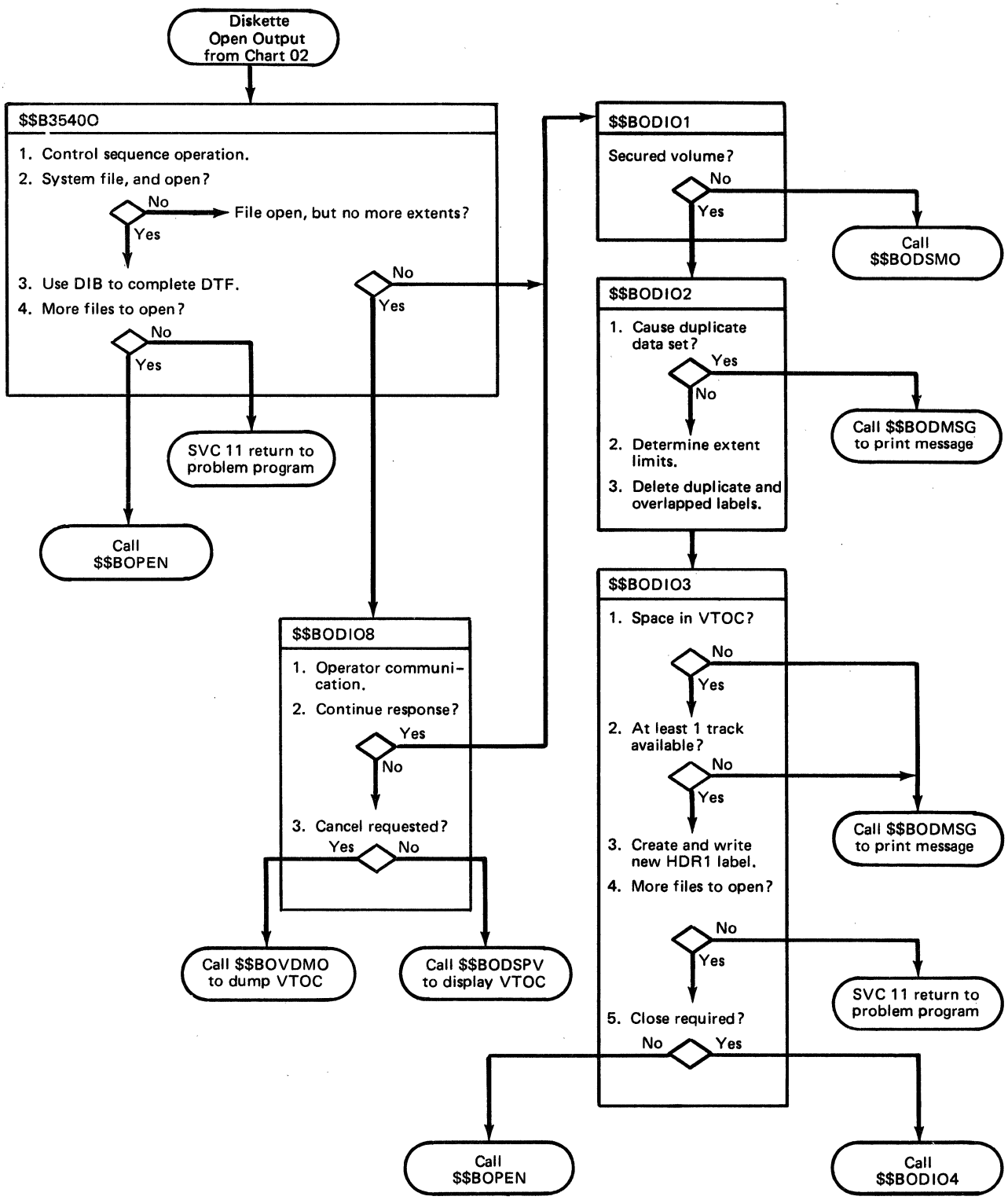


Chart AA. $$BOPEN: Open Monitor (Part 1 of 2)

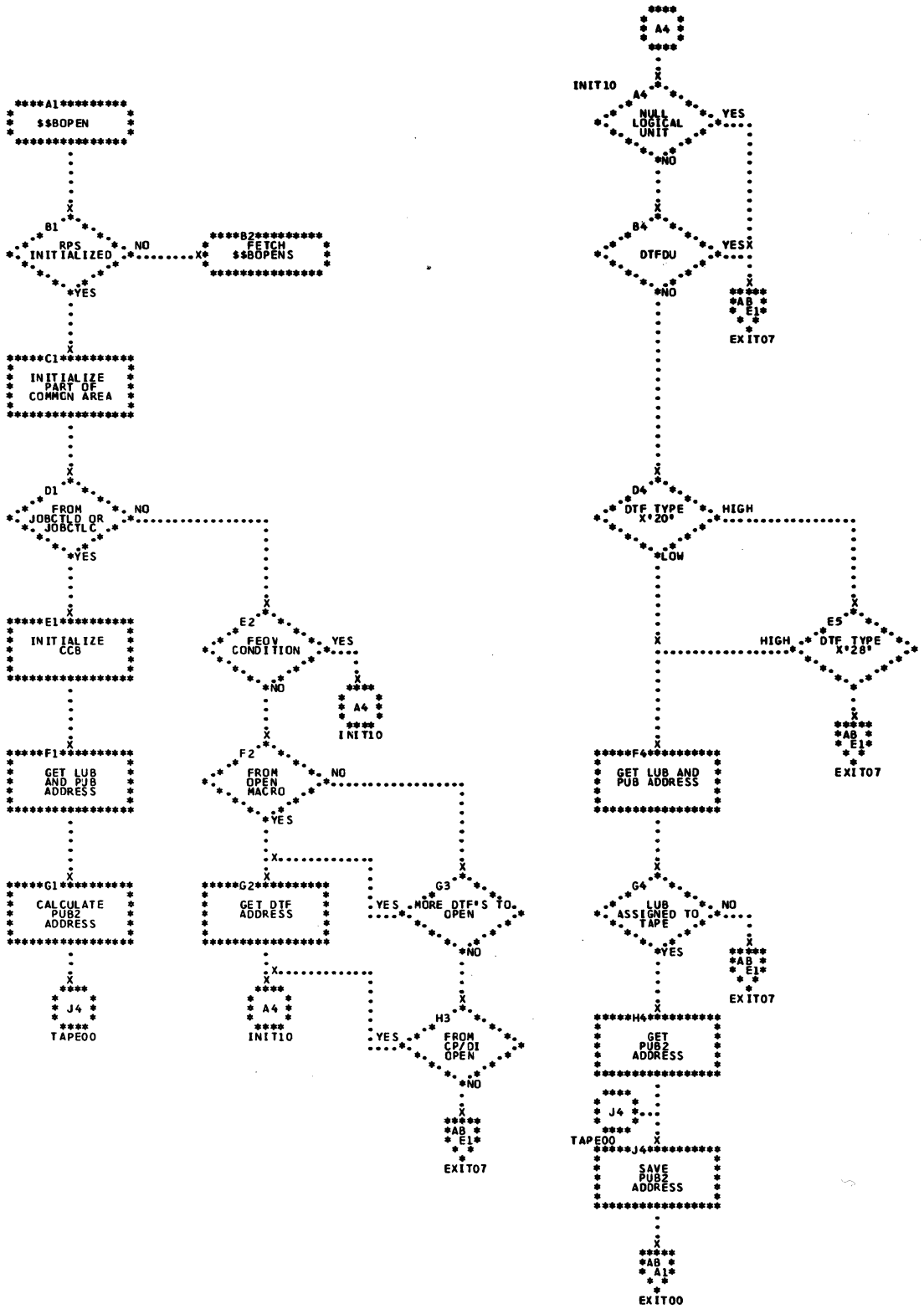


Chart AB. $$BOPEN: Open Monitor (Part 2 of 2)

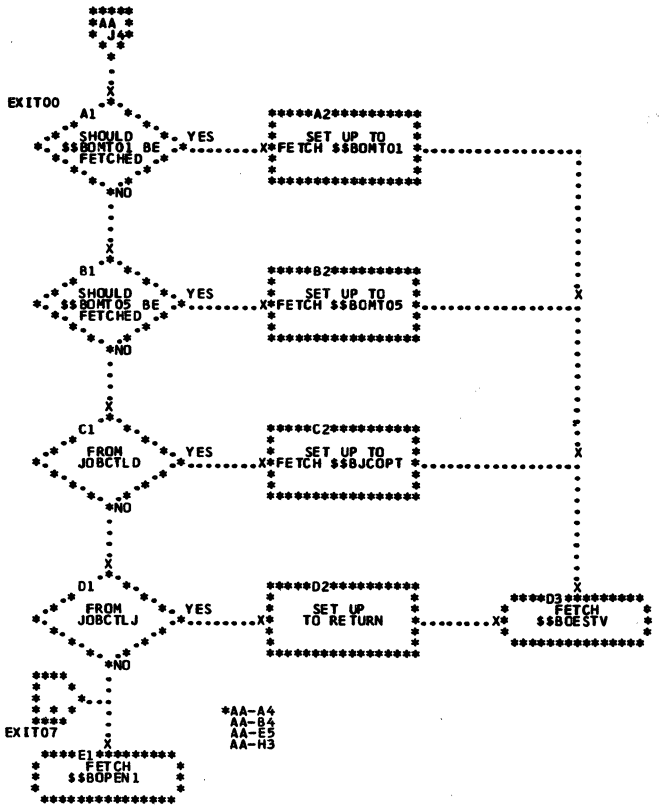


Chart AE. $$BOPEN1 Monitor, Phase 1 (Part 1 of 6)

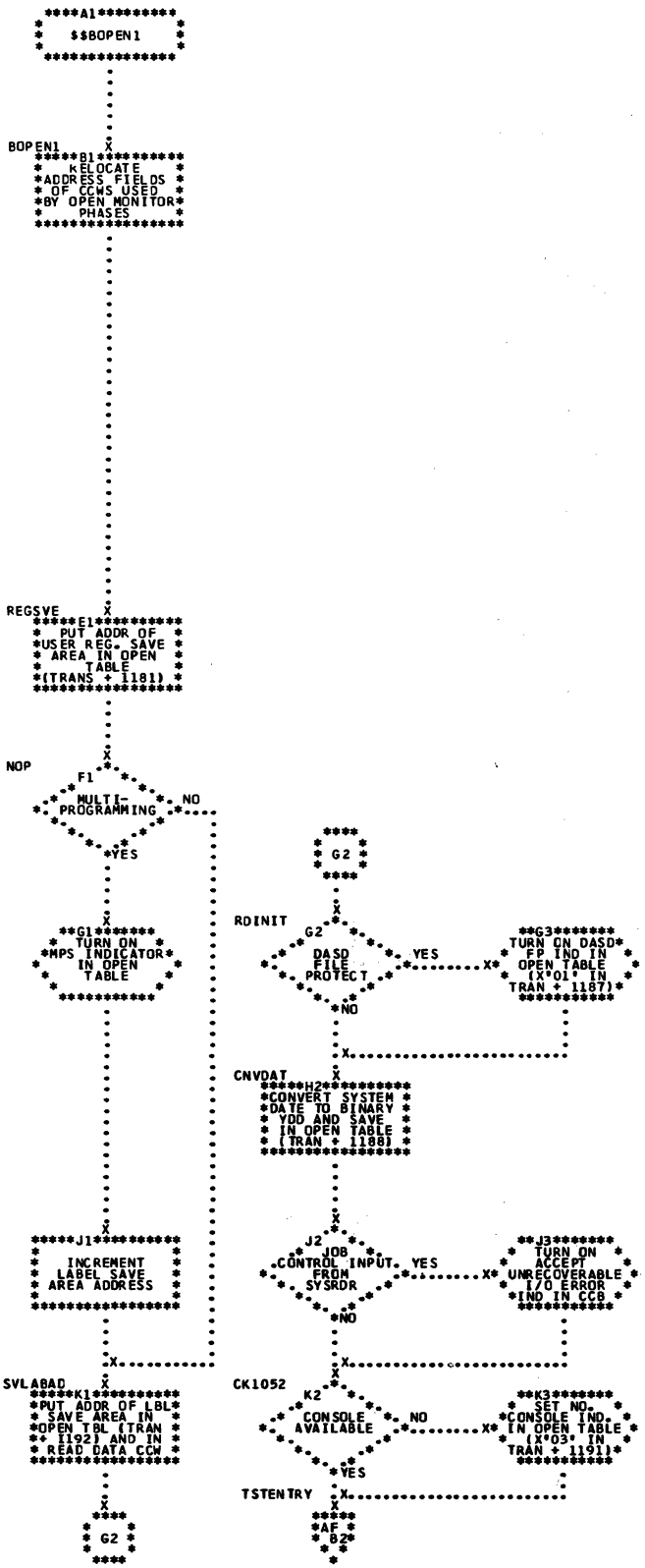


Chart AF. $$BOPEN1 Monitor, Phase 1 (Part 2 of 6)

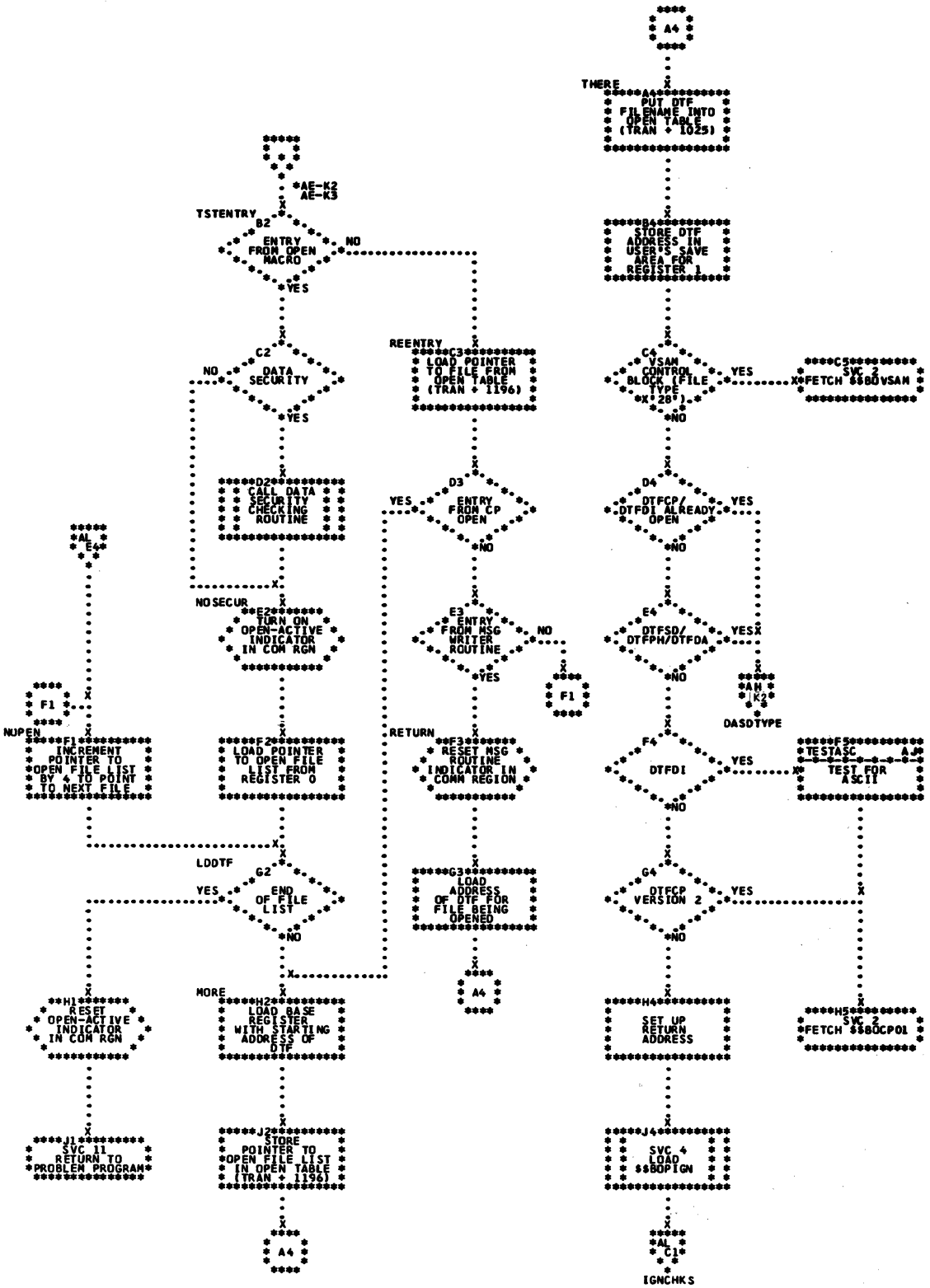


Chart AG. $$BOPEN1 Monitor, Phase 1 (Part 3 of 6)

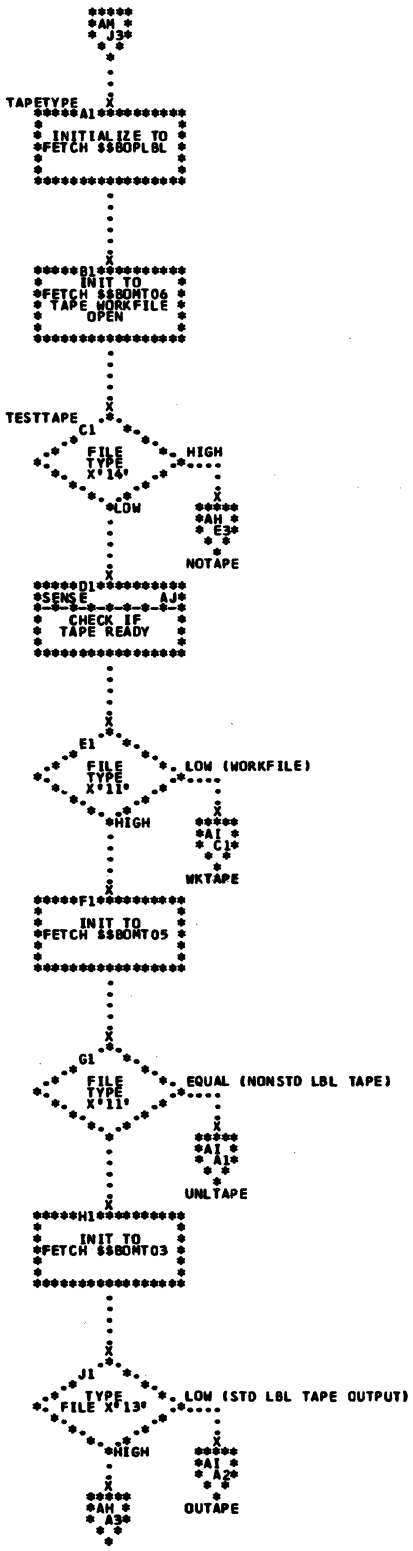


Chart AH. $$BOPEN1 Monitor, Phase 1 (Part 4 of 6)

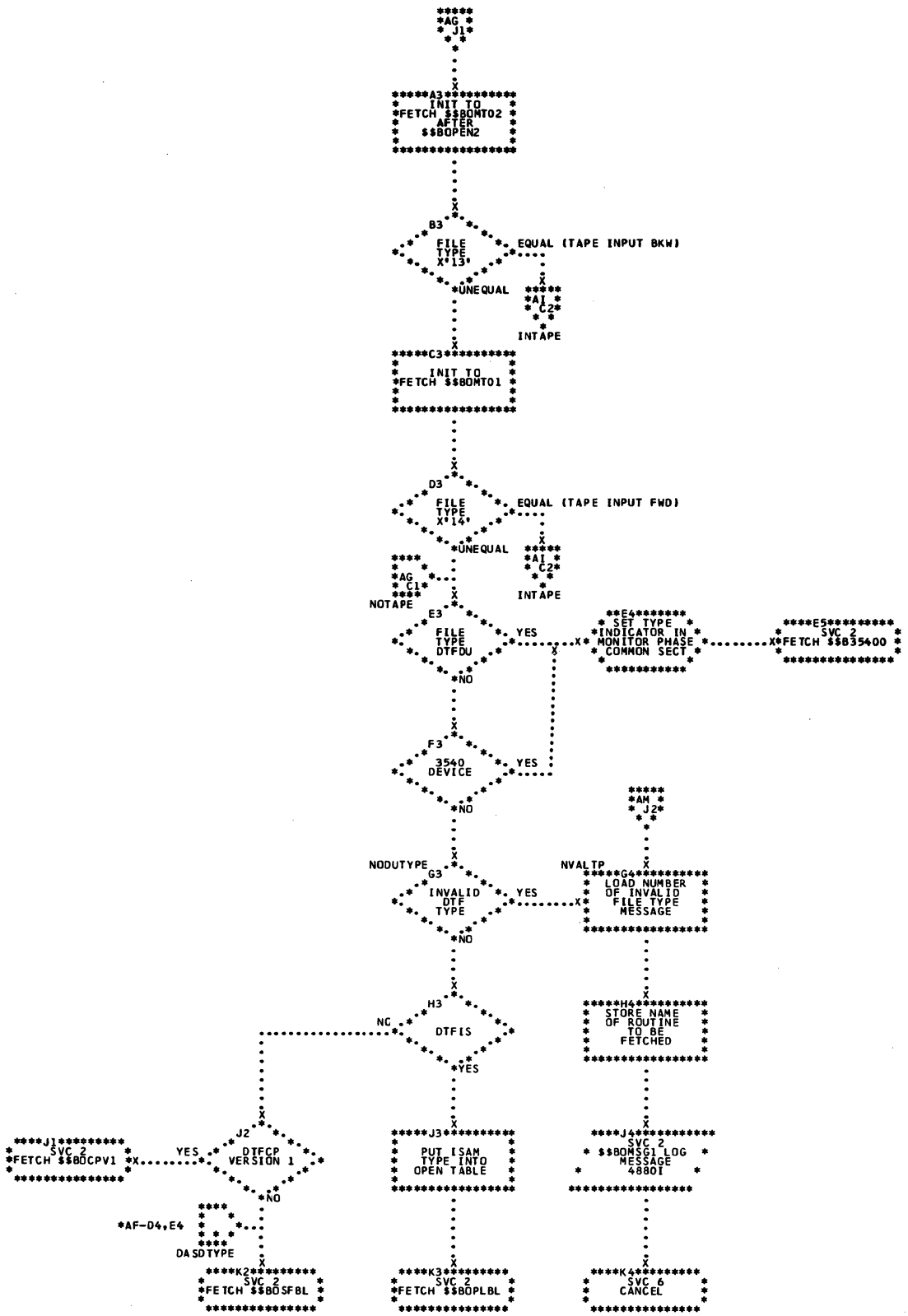


Chart AI. $$BOPEN1 Monitor, Phase 1 (Part 5 of 6)

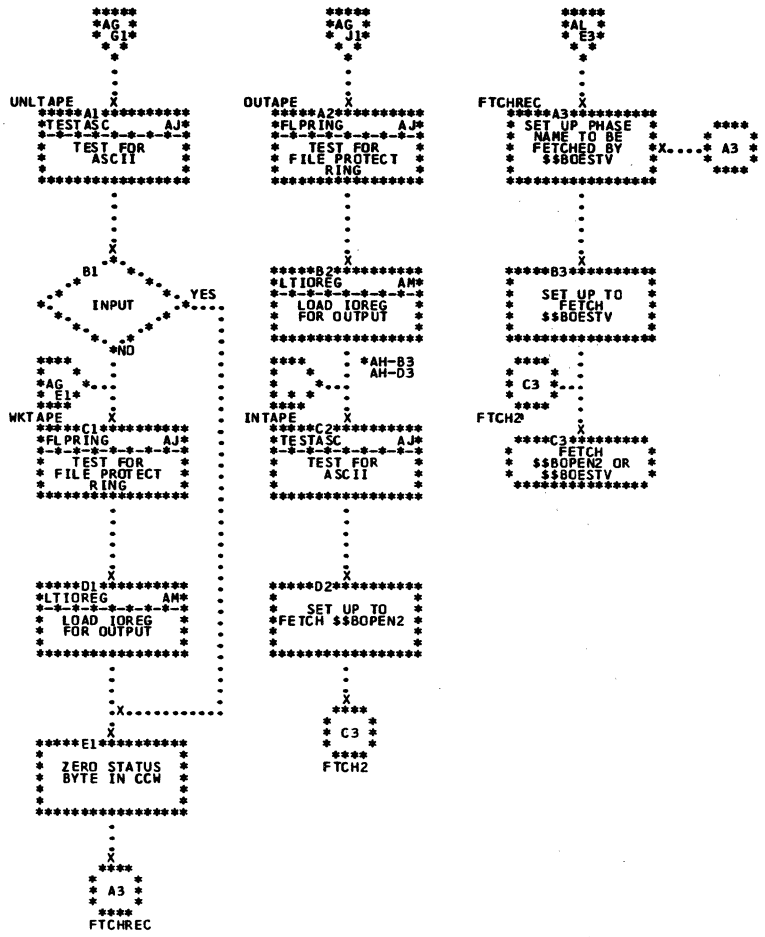


Chart AJ. $$BOPEN1 Monitor, Phase 1 (Part 6 of 6)

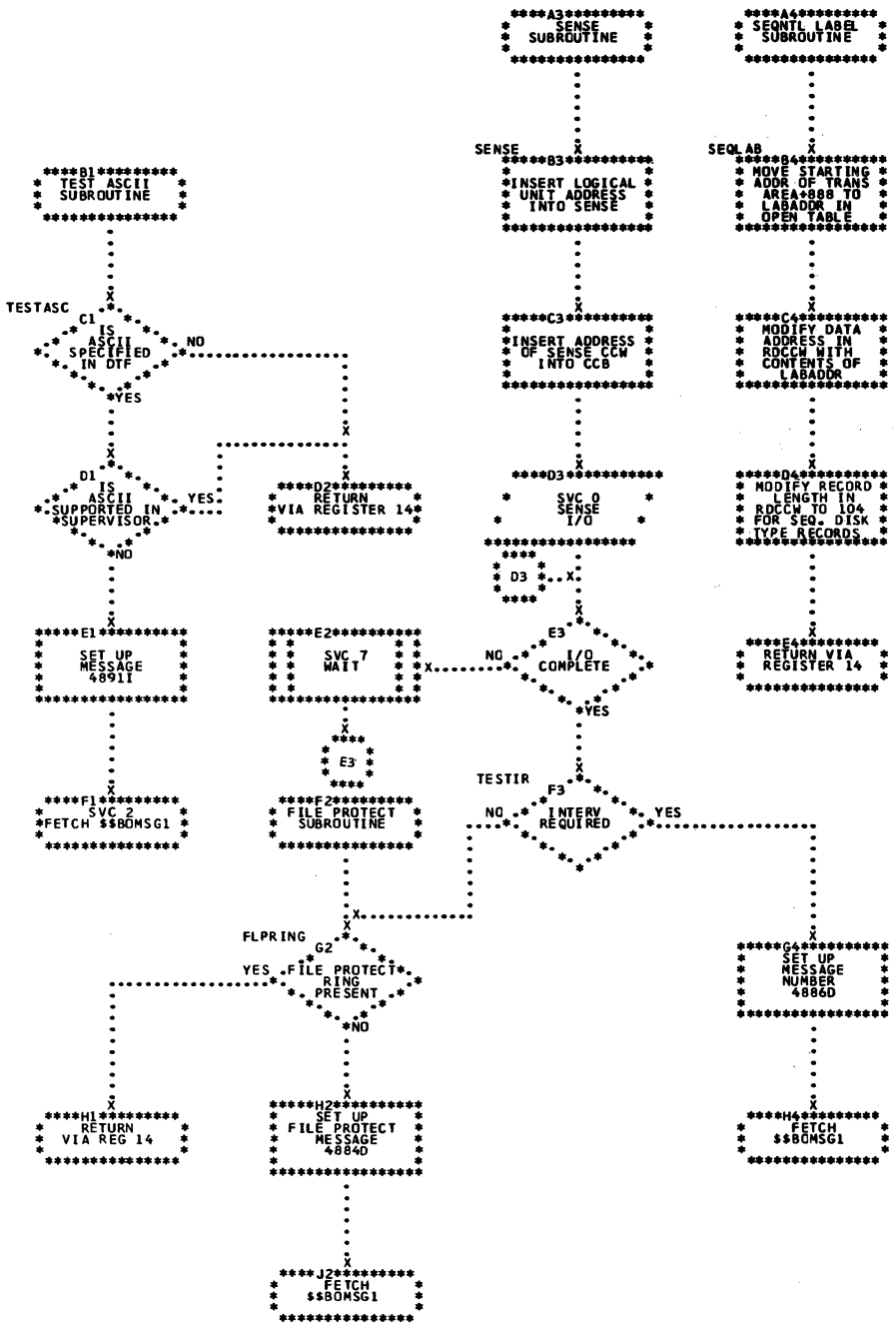


Chart AK. $$BOPLBL: Open Monitor Label Space Processing

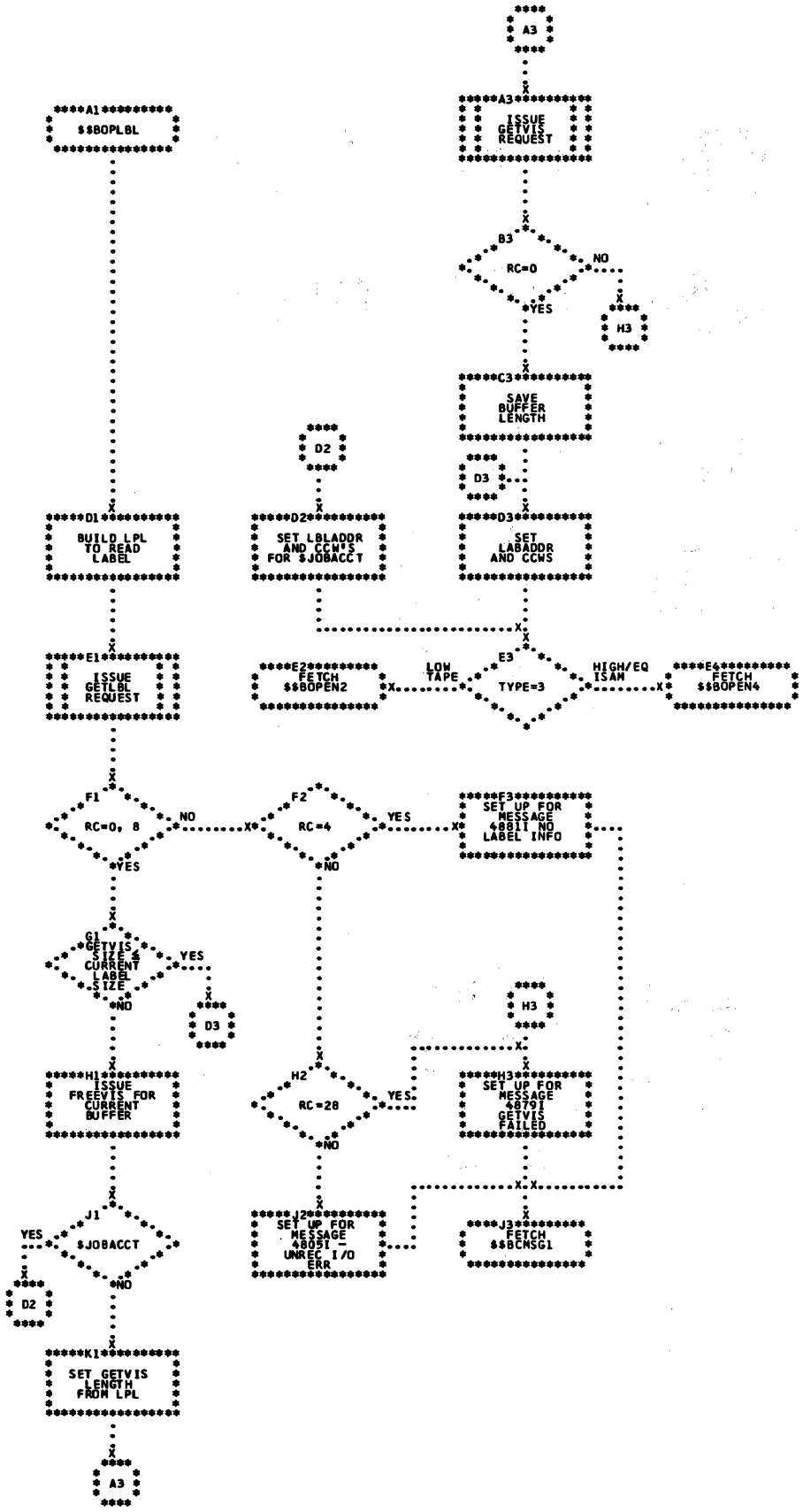


Chart AL. $$BOPIGN: Open Ignore (Part 1 of 2)

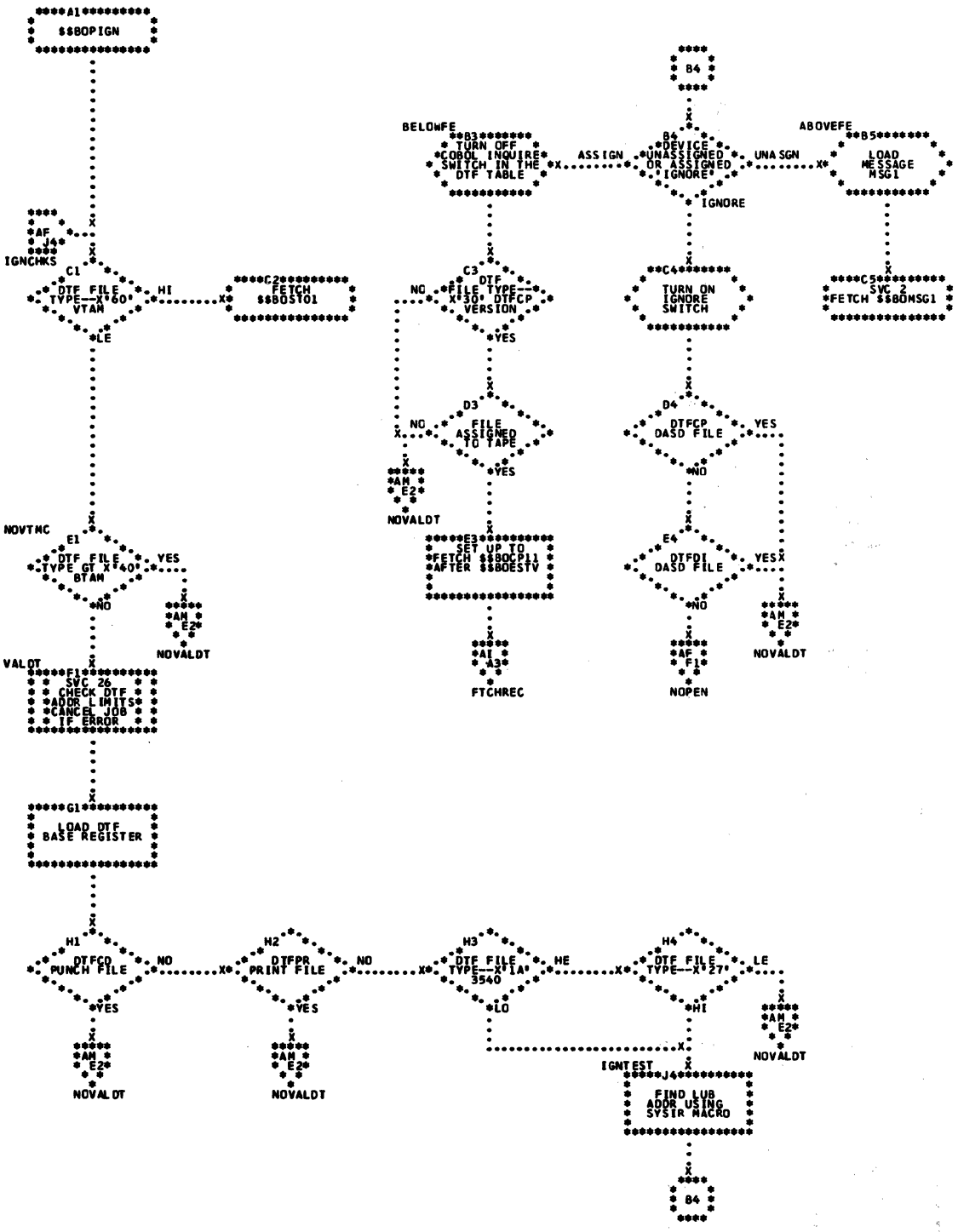


Chart AM. $$BOPIGN: Open Ignore (Part 2 of 2)

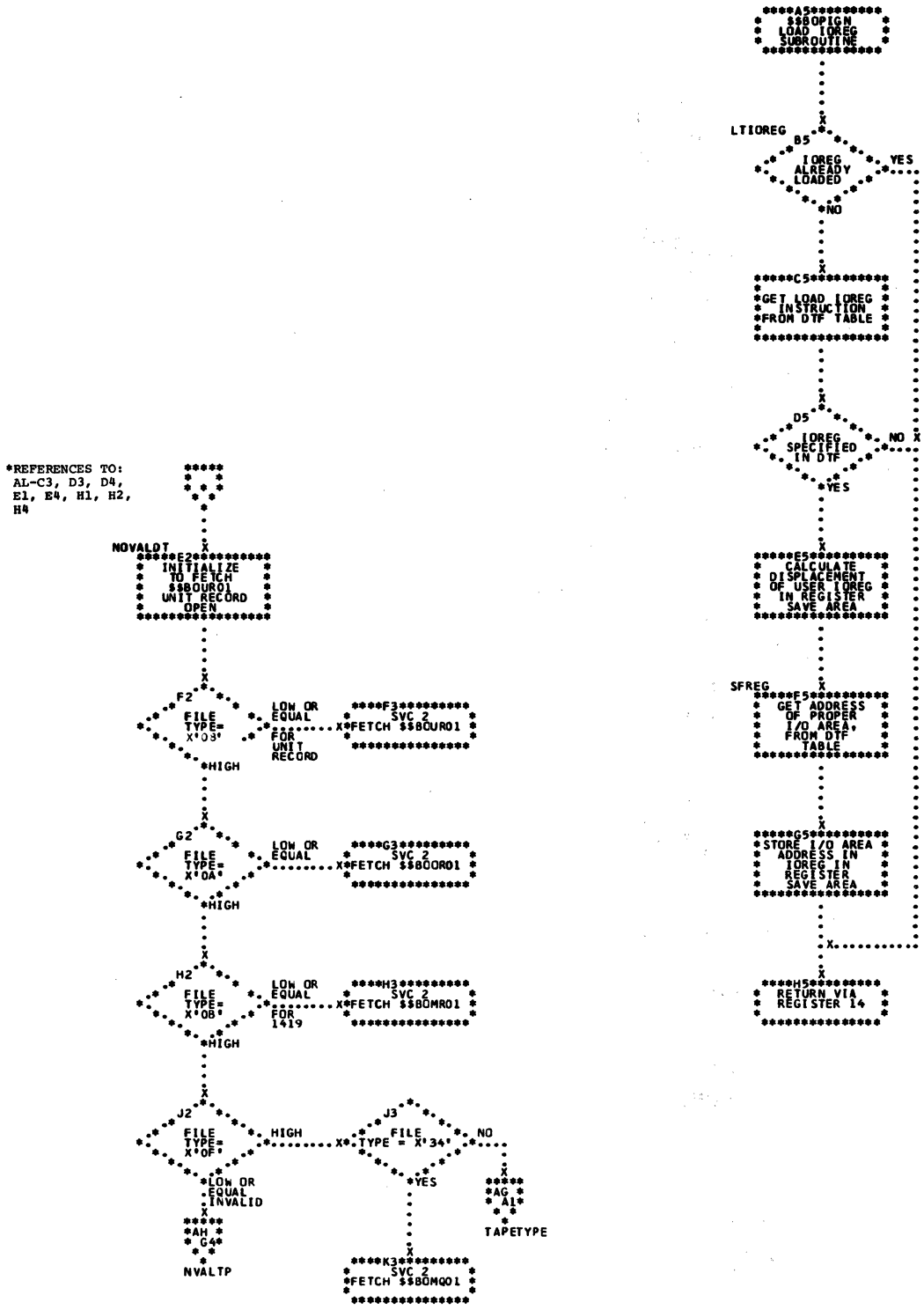


Chart AN. $$BOPEN2: Open Monitor, Phase 2 (Part 1 of 3)

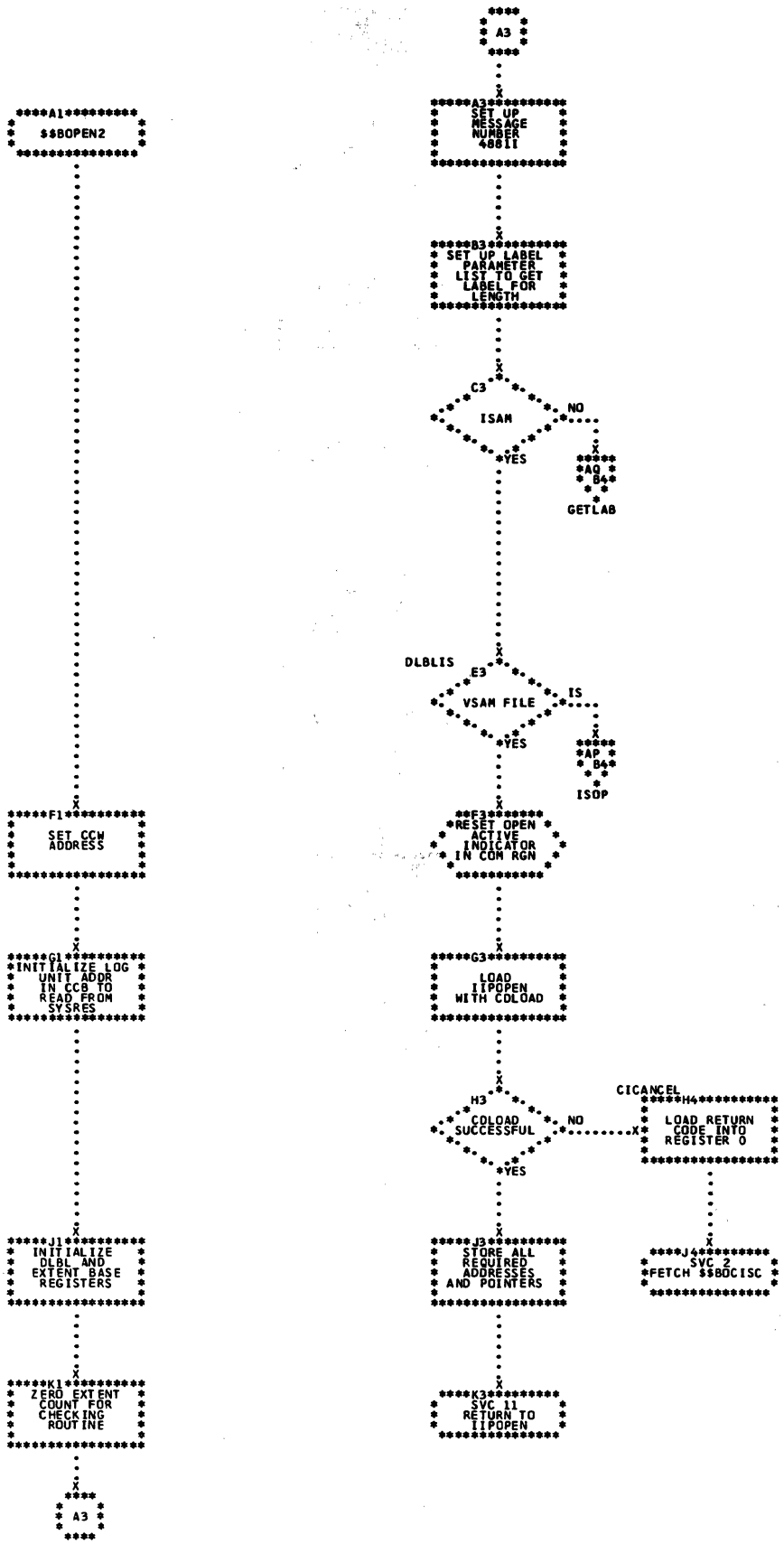


Chart AP. $$BOPEN2: Open Monitor, Phase 2 (Part 2 of 3)

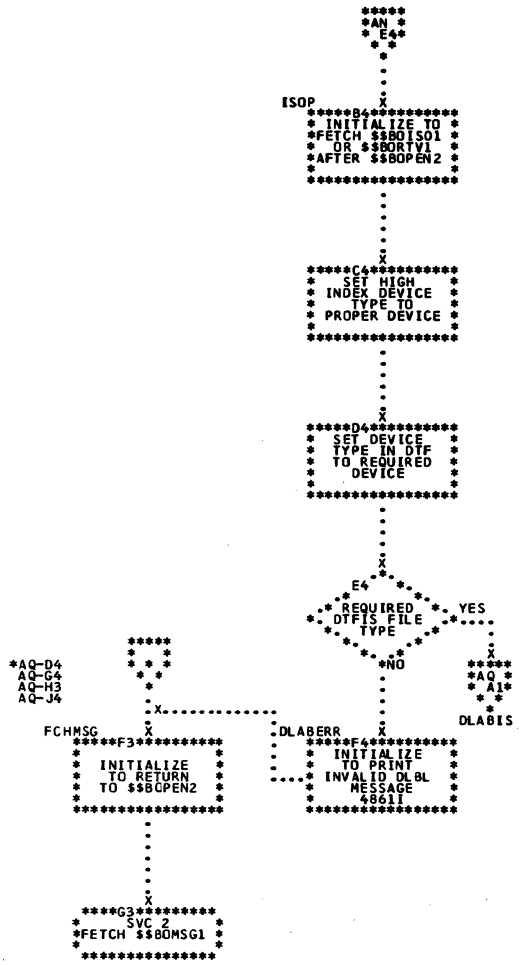


Chart AQ. $$BOPEN2: Open Monitor, Phase 2 (Part 3 of 3)

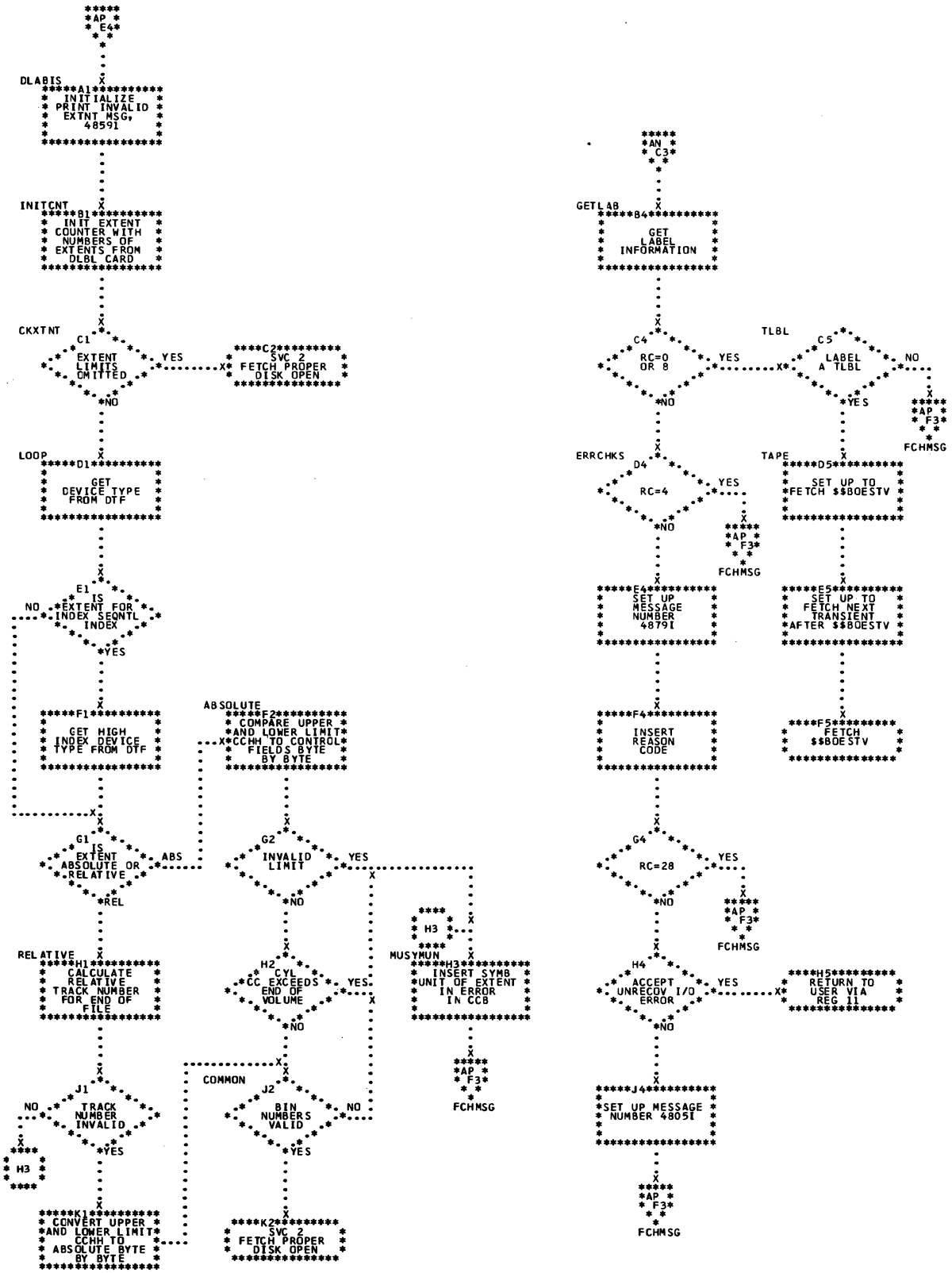


Chart AS. $$BOPENR: Relocate DTF Address Constants (Part 1 of 4)

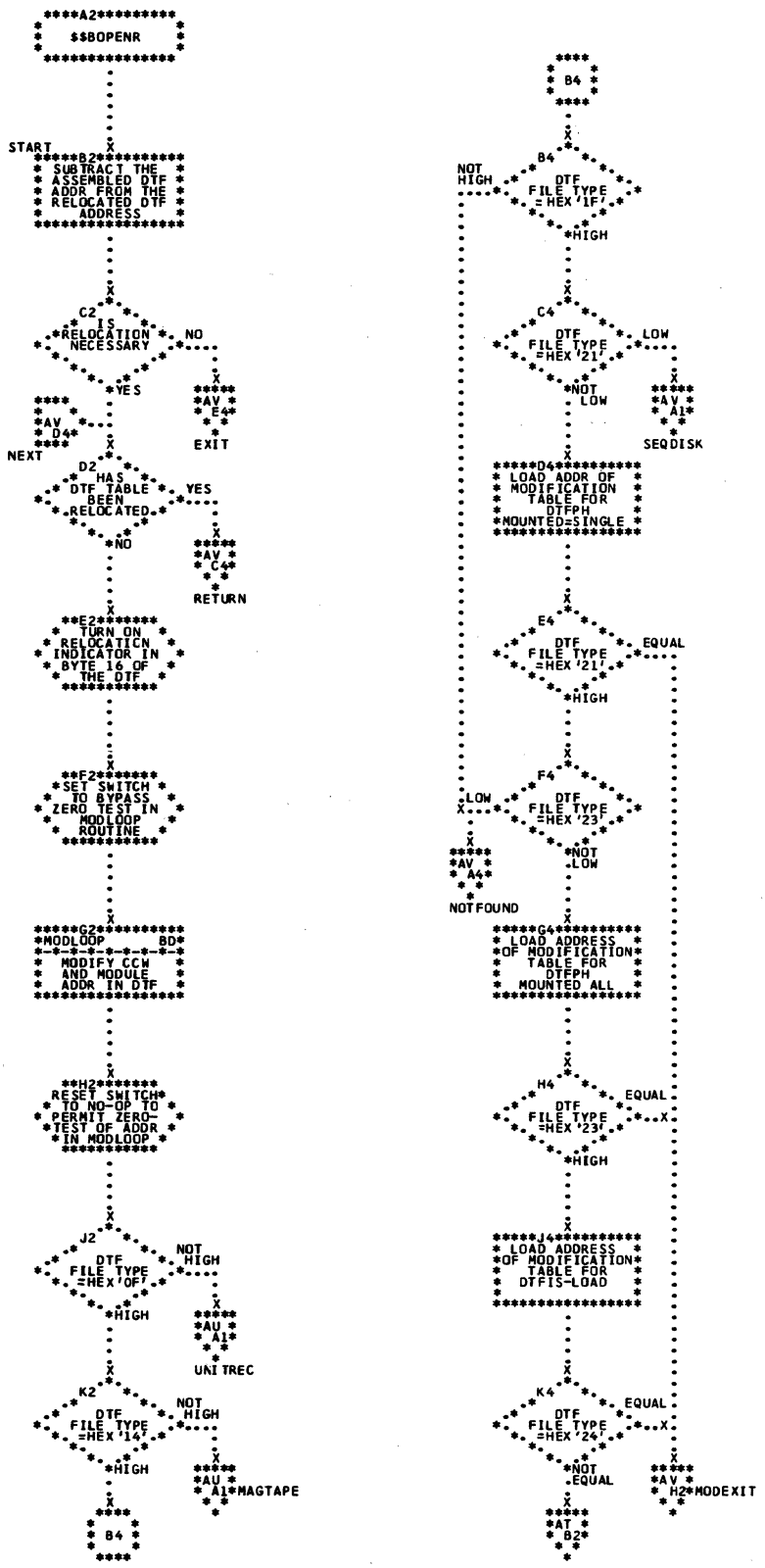


Chart AT. $$BOPENR: Relocate DTF Address Constants (Part 2 of 4)

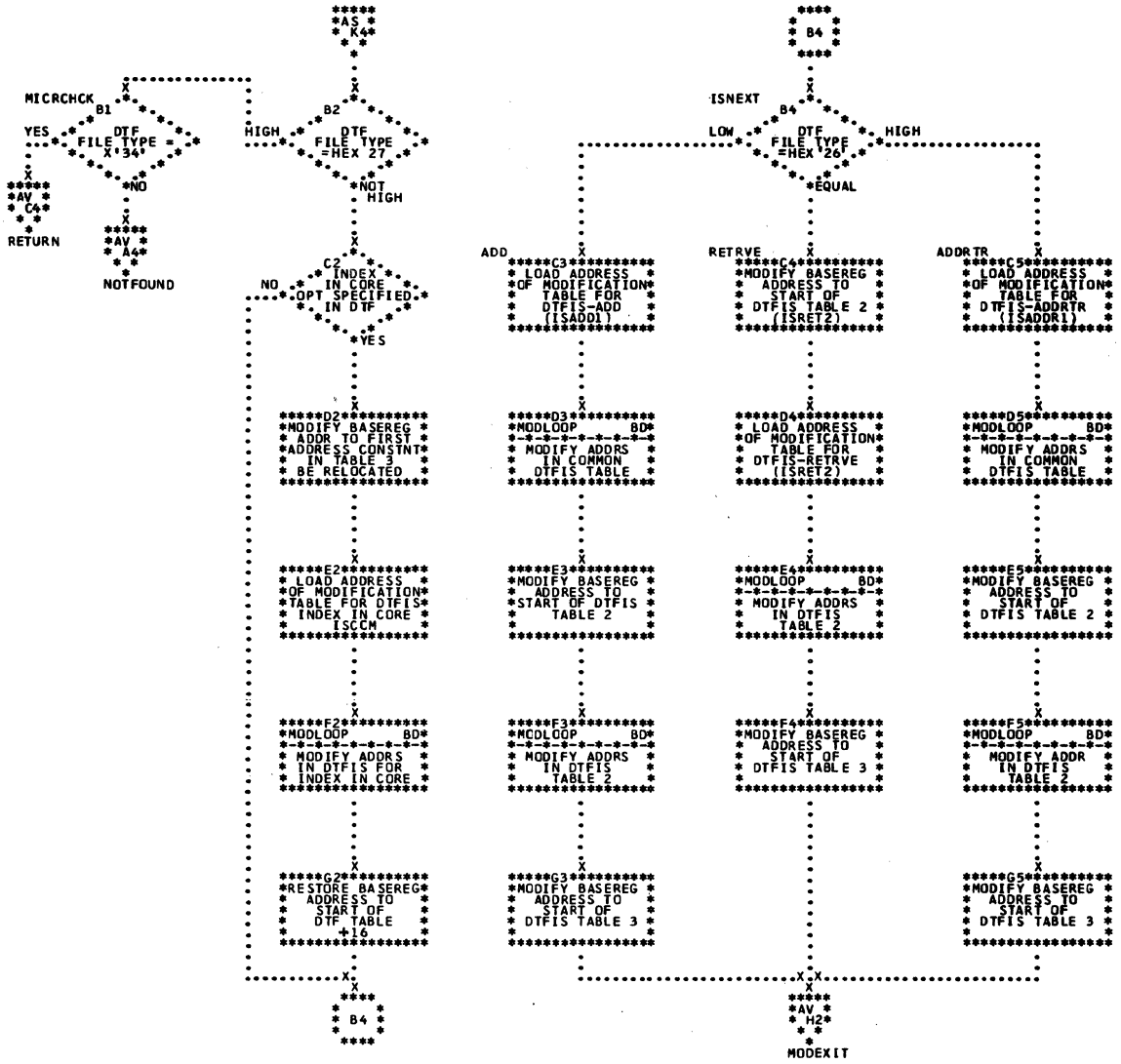


Chart AU. $$BOPENR: Relocate DTF Address Constants (Part 3 of 4)

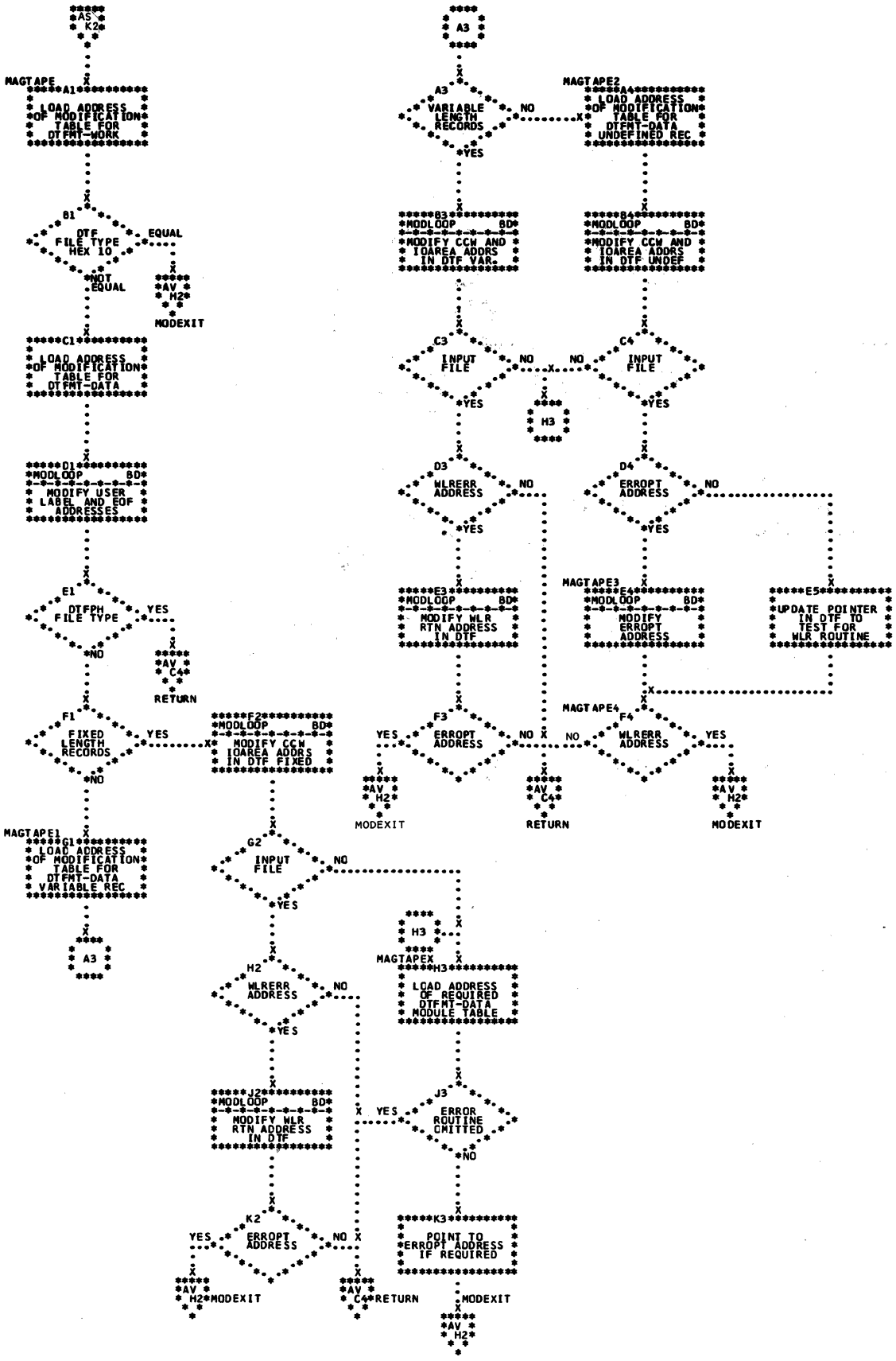


Chart AV. $$BOPENR: Relocate DTF Address Constants (Part 4 of 4)

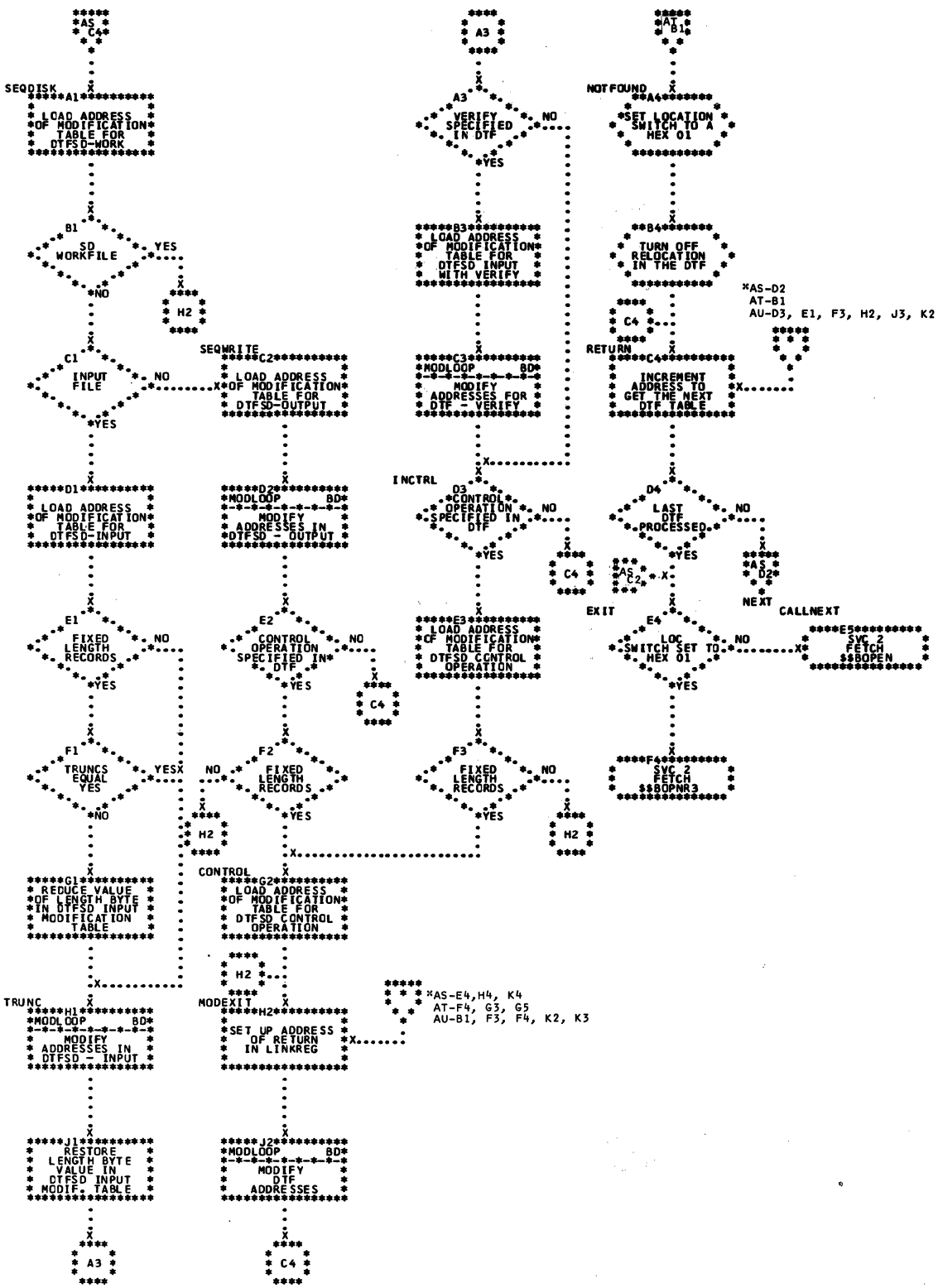


Chart AW. $$BOPENC: Check Duplication Device Assignment for Logical Units

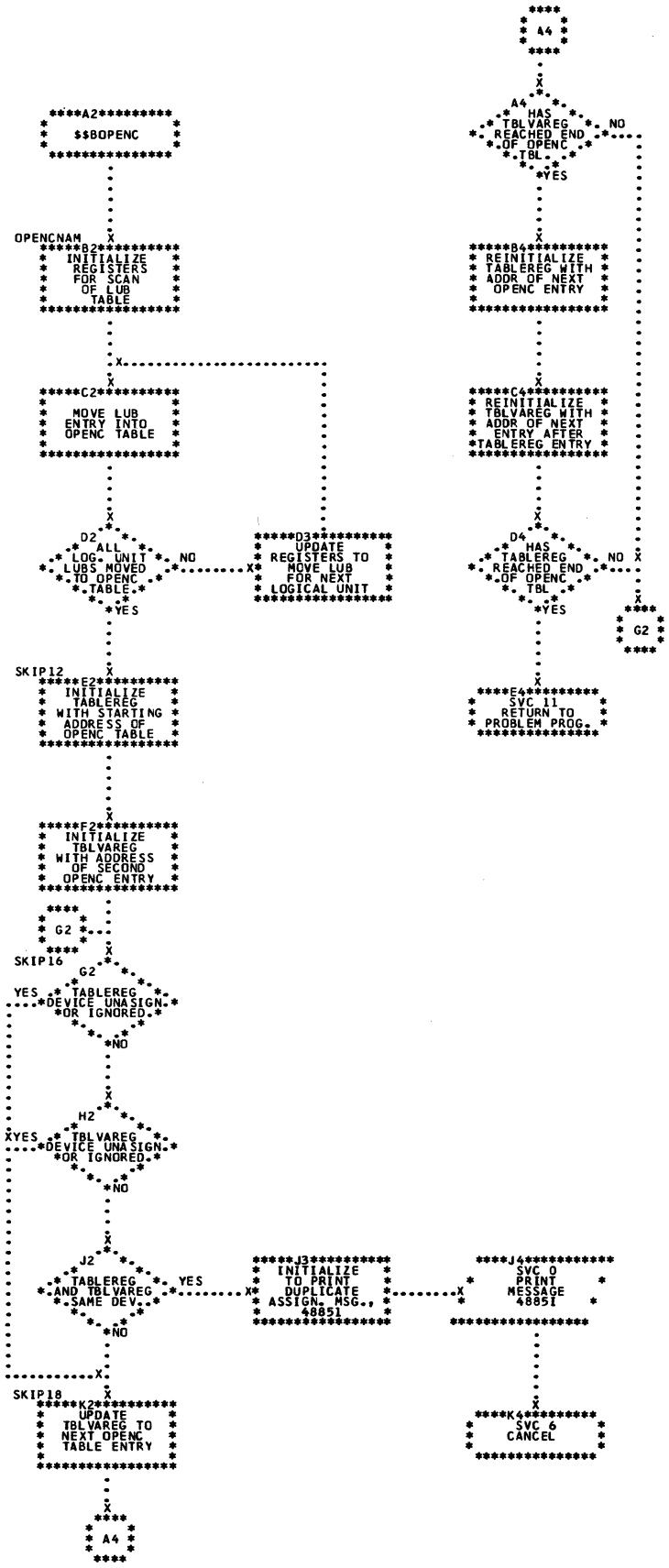


Chart AX. $$BENDQB: Enqueue and Dequeue for VSE/VSAM Routines

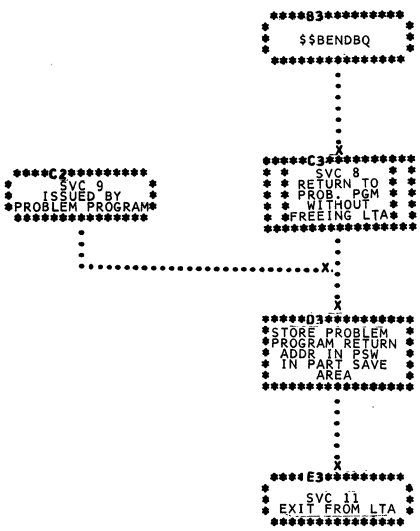


Chart BA. $$BOPNR2: Relocate DTF Address Constants, Phase 2 (Part 1 of 3)

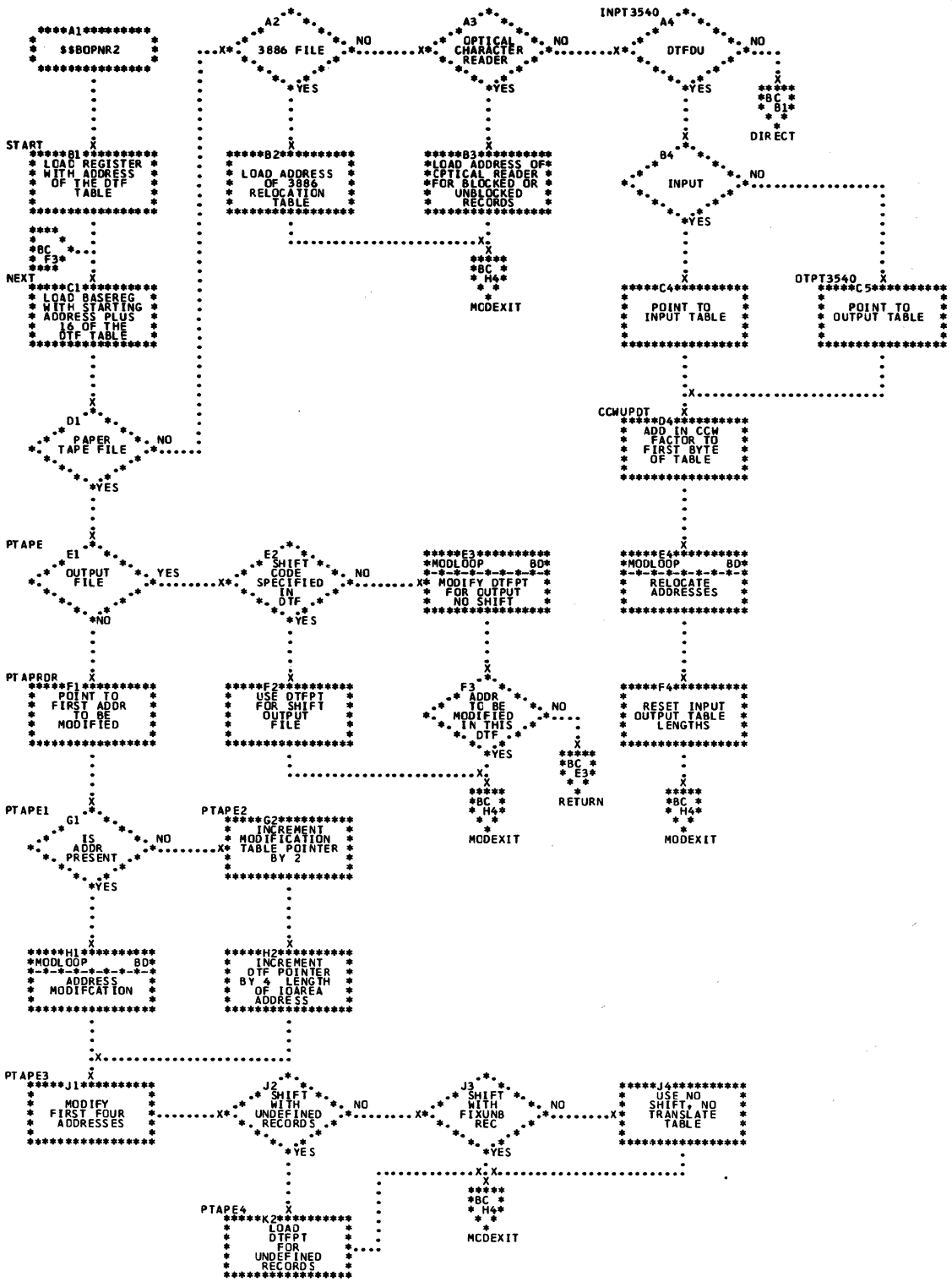


Chart BB. $$BOPNR2: Relocate DTF Address Constants, Phase 2 (Part 2 of 3)

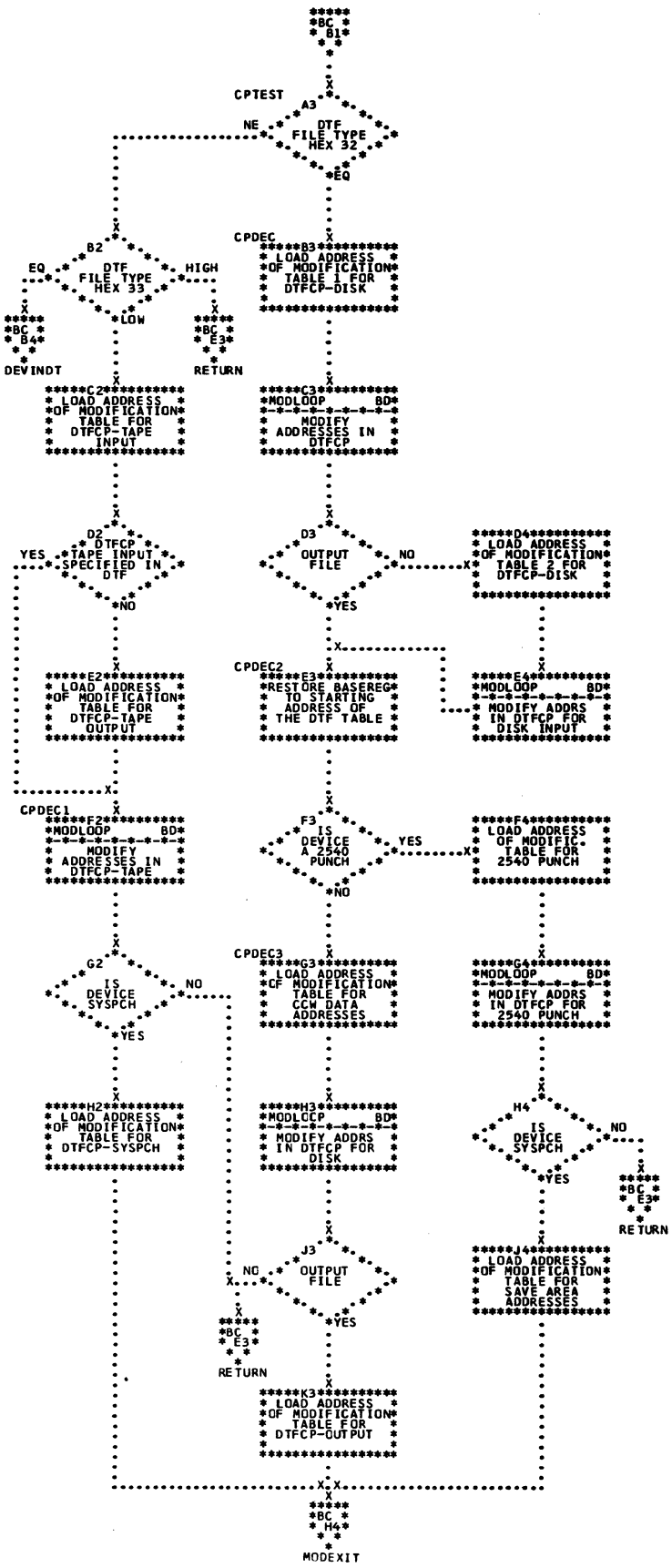


Chart BC. $$BOPNR2: Relocate DTF Address Constants, Phase 2 (Part 3 of 3)

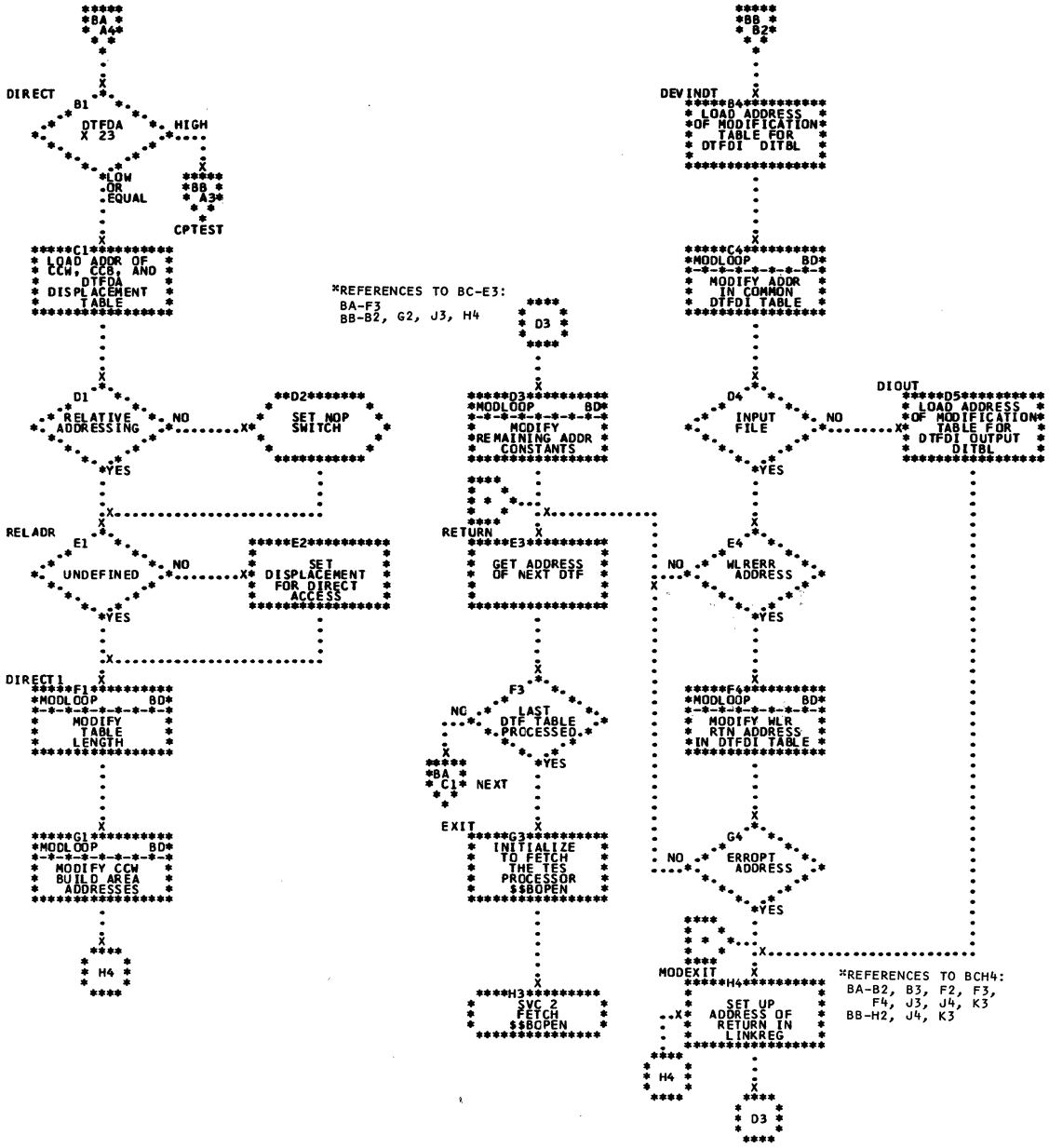


Chart BD. MODLOOP Subroutine to $$BOPENR and $$BOPNR3

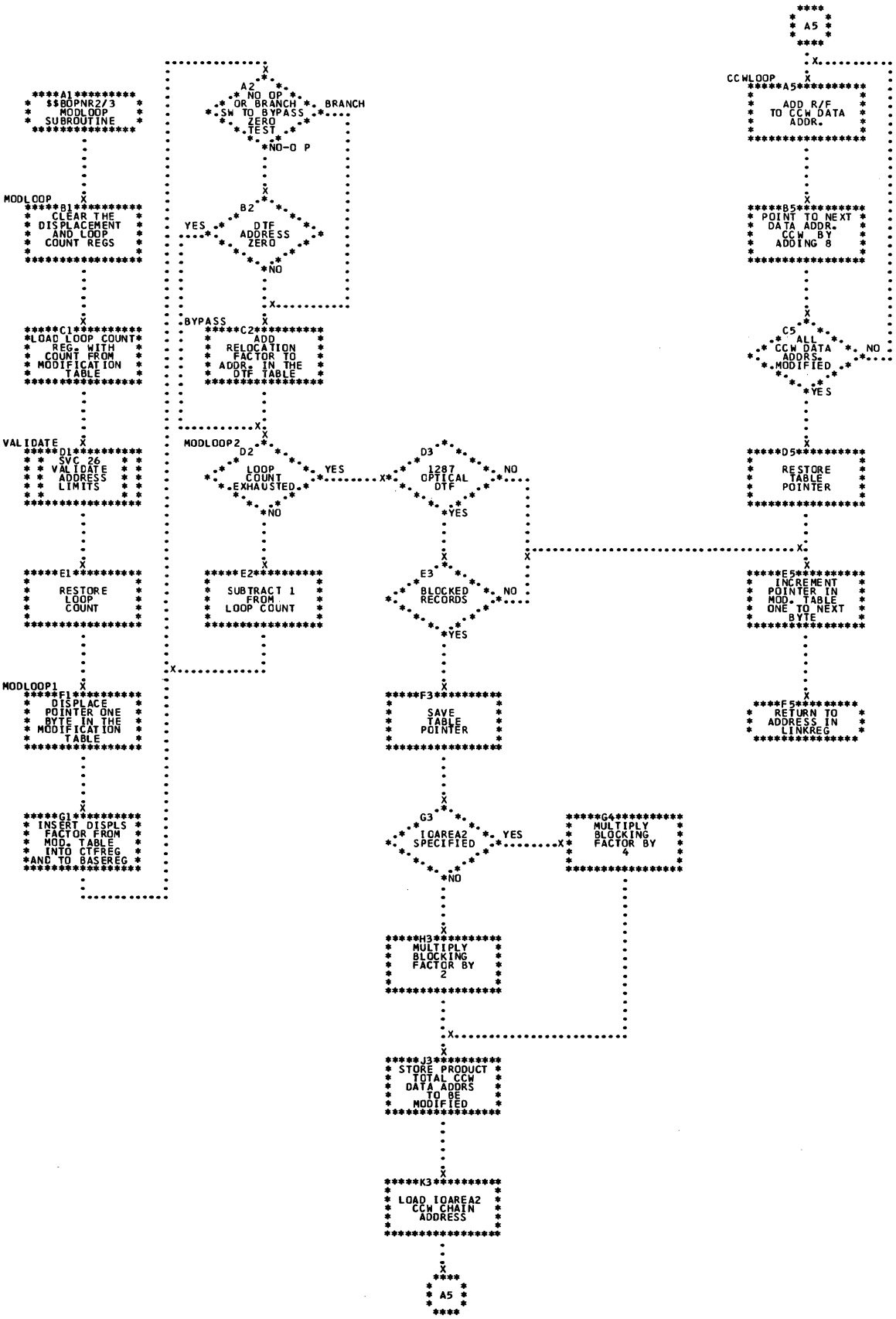


Chart BE. $$BOPNR3: Relocate DTF Address Constants, Phase 3 (Part 1 of 2)

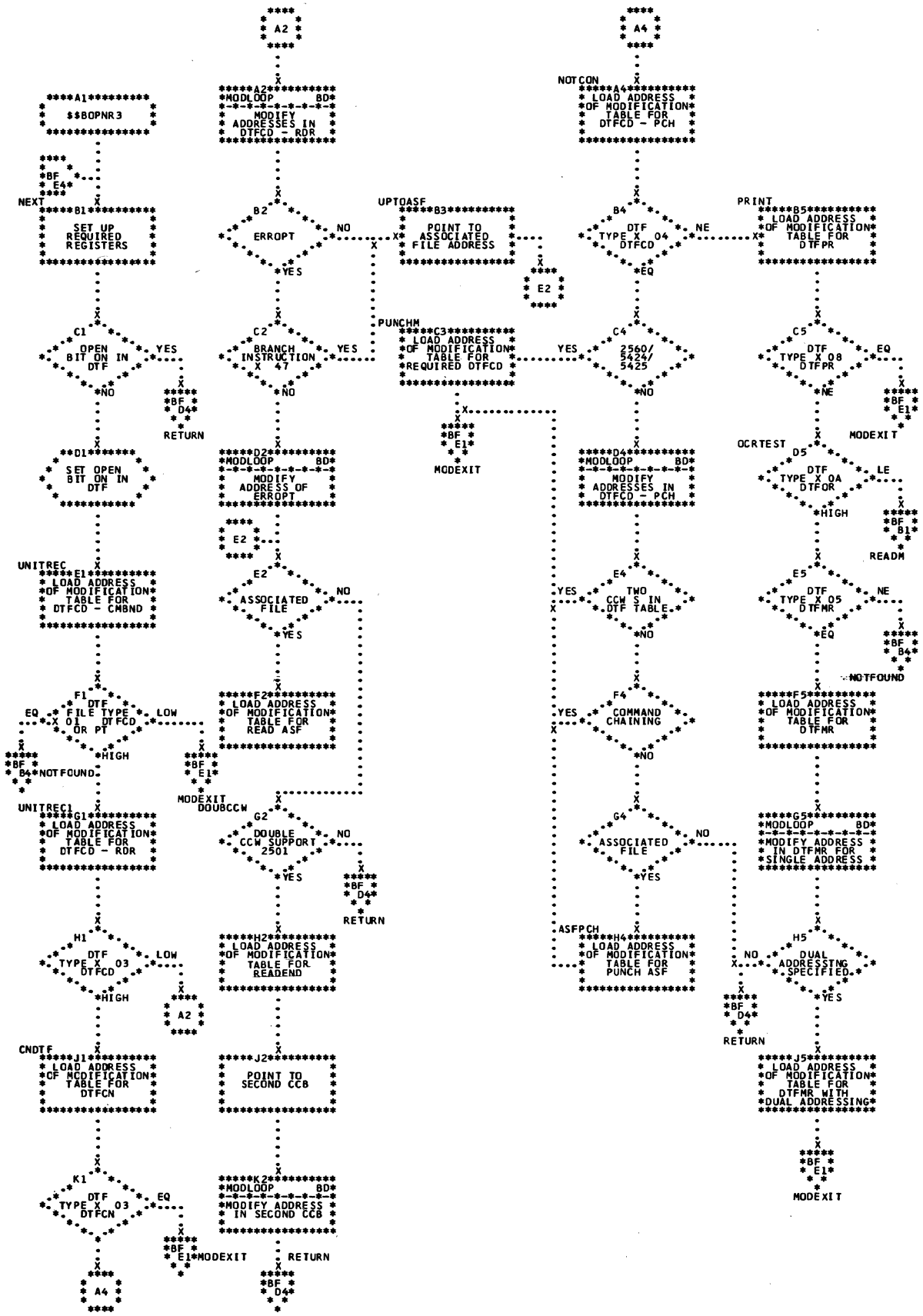


Chart BF. $$BOPNR3: Relocate DTF Address Constants, Phase 3 (Part 2 of 2)

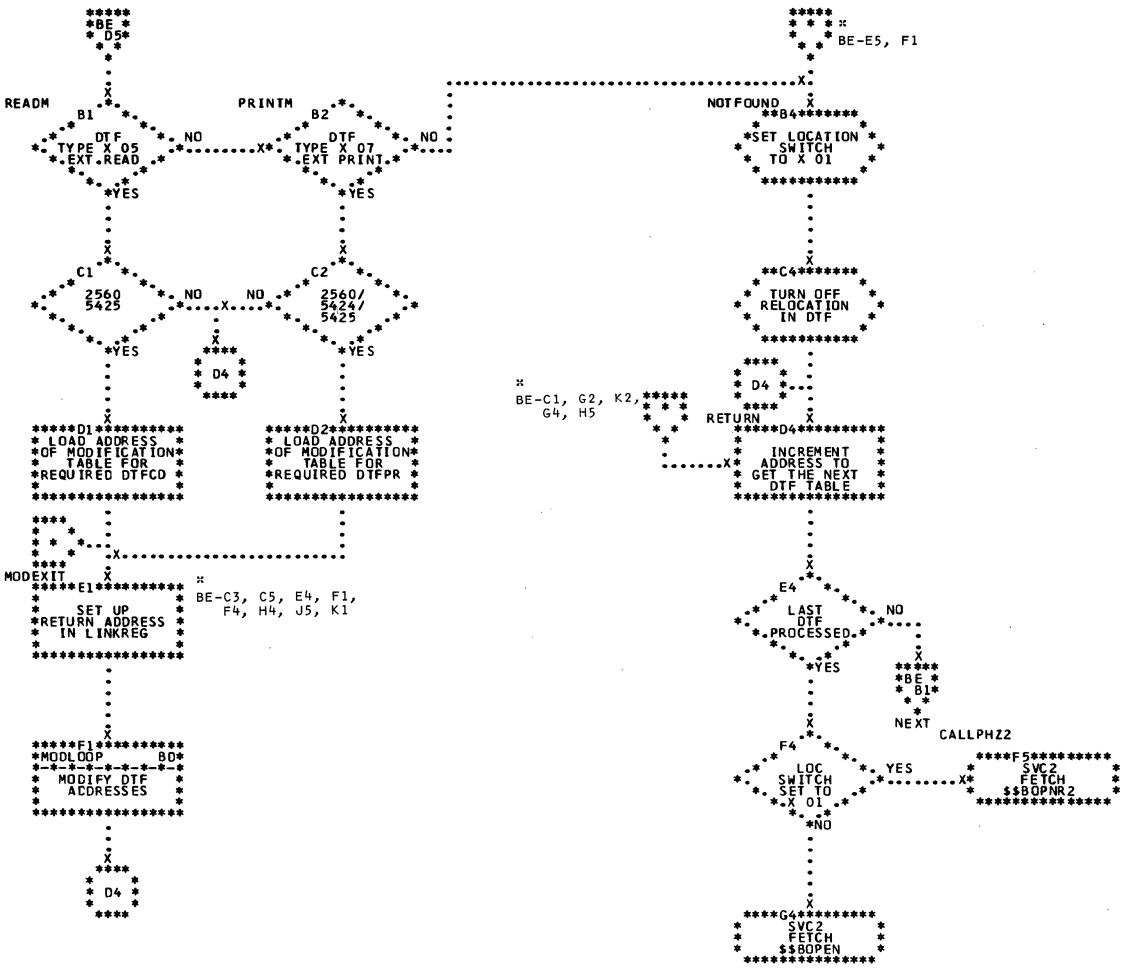


Chart BG. $$BCLOSE: Close Monitor, Phase 1 (Part 1 of 3)

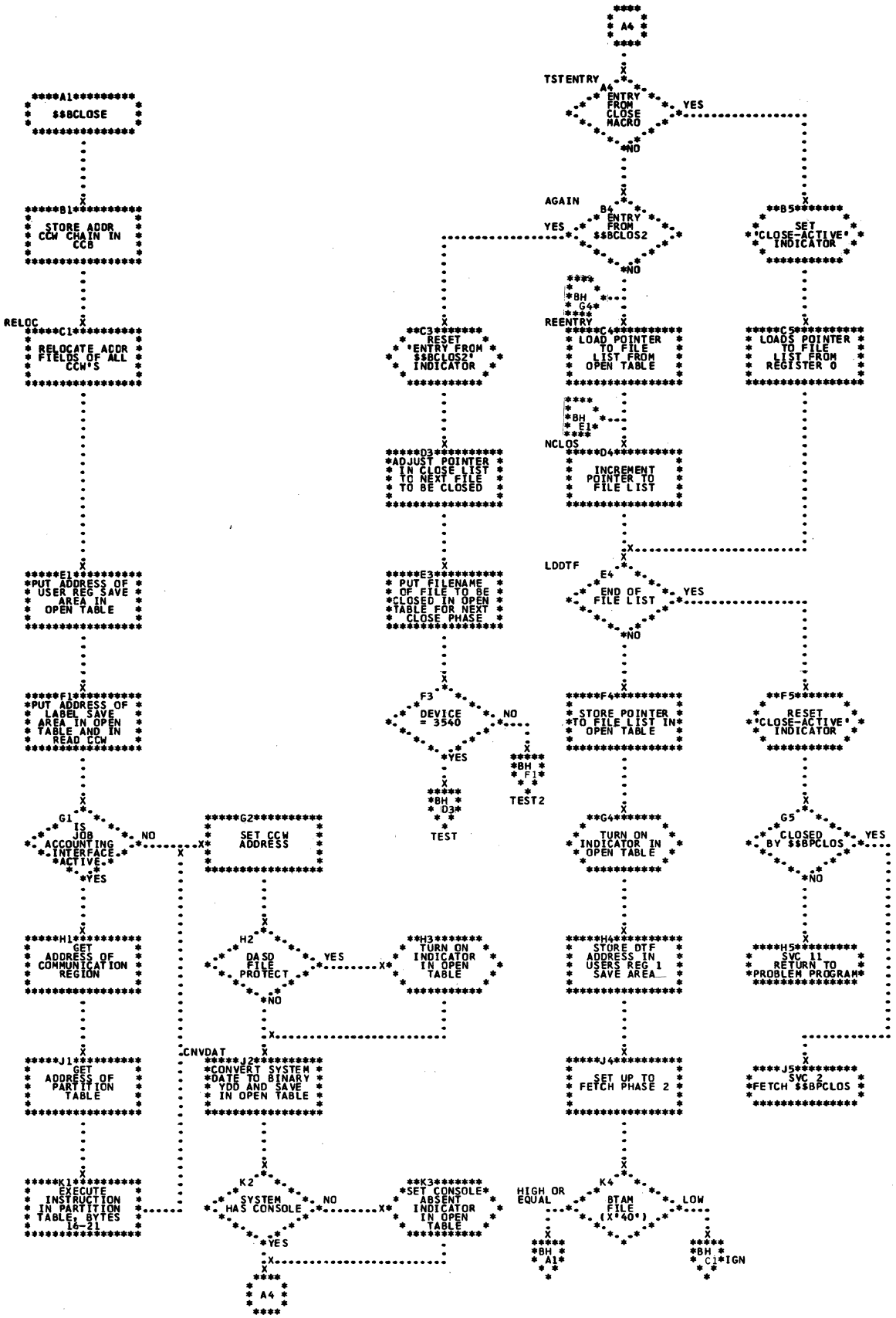


Chart BH. $$BCLOSE: Close Monitor, Phase 1 (Part 2 of 3)

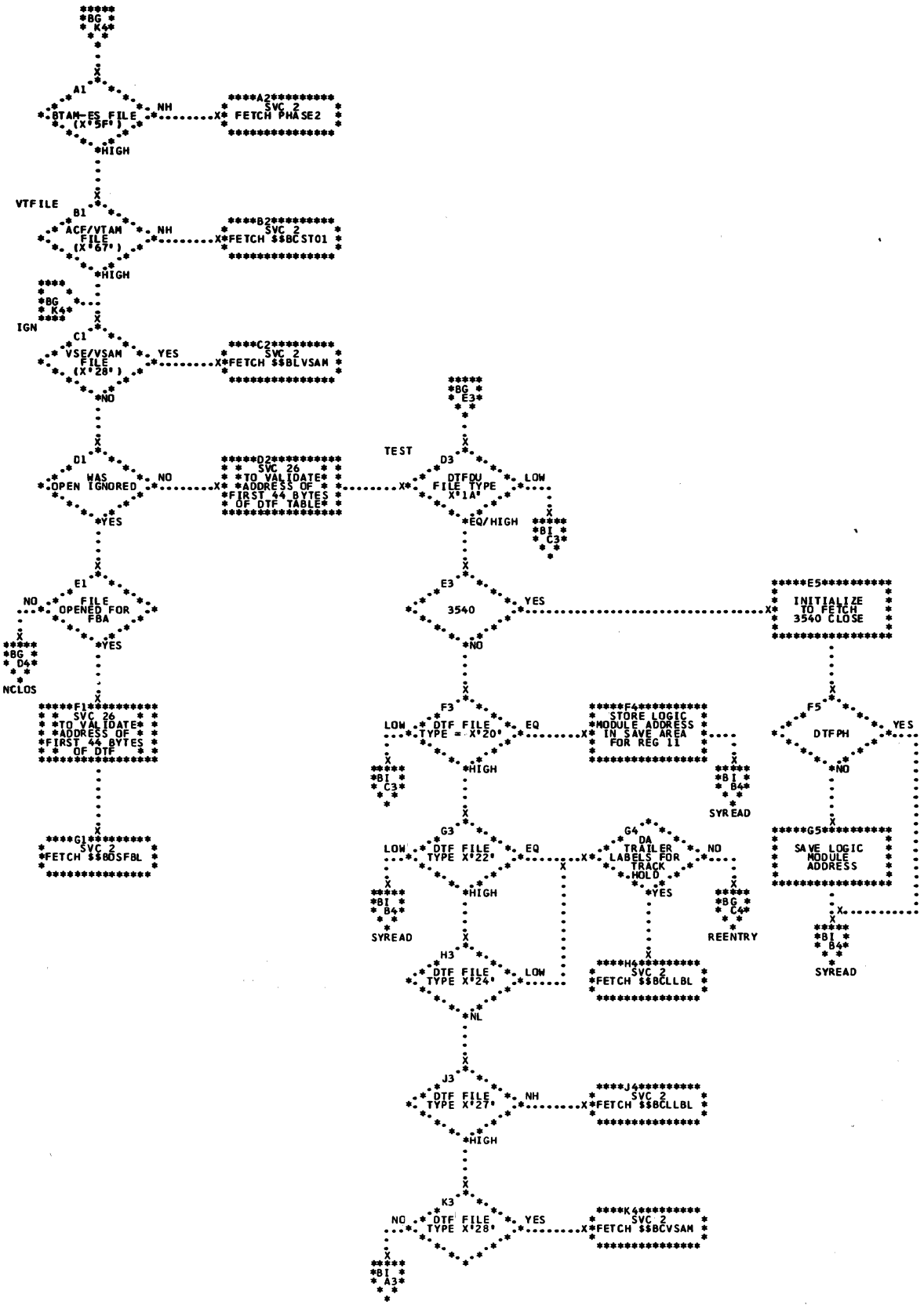


Chart BI. $$BCLOSE: Close Monitor, Phase 1 (Part 3 of 3)

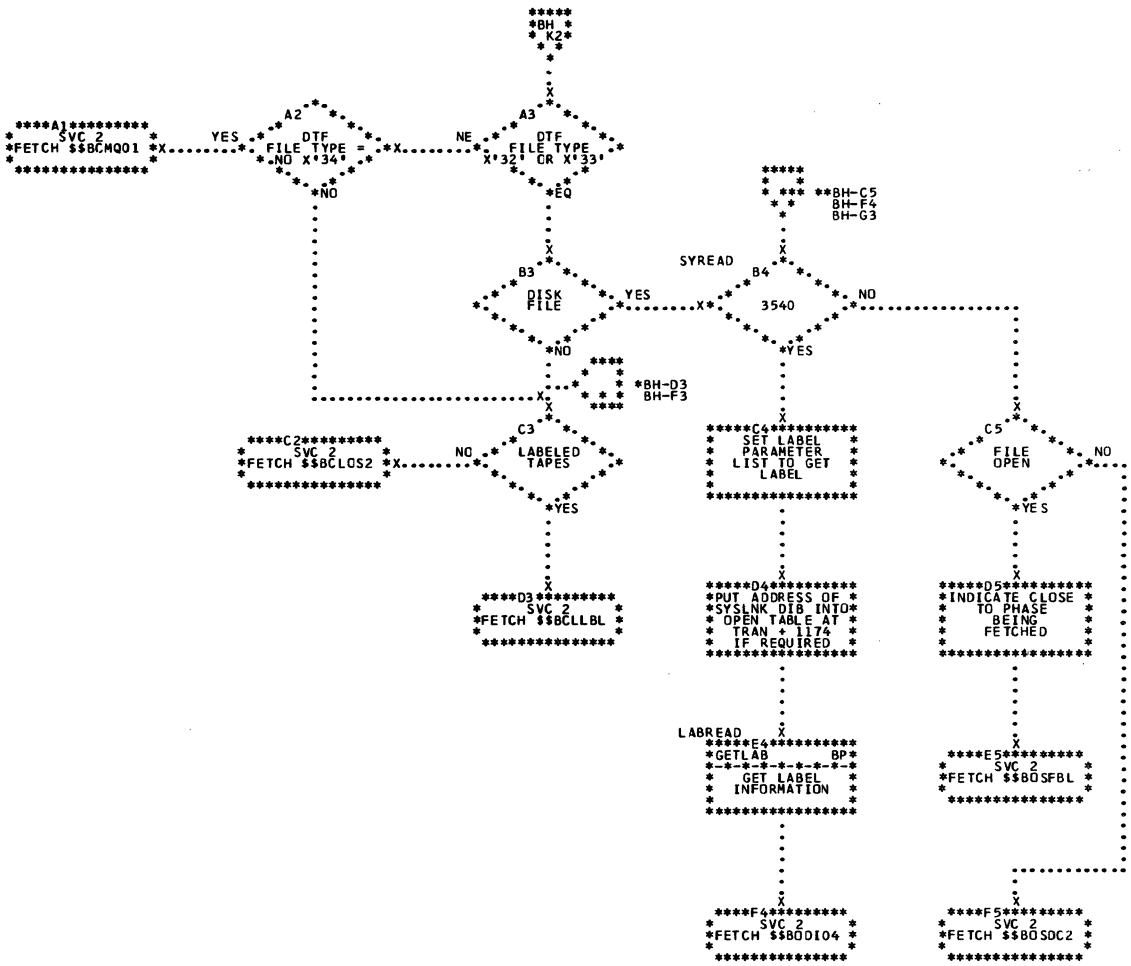


Chart BJ. $$BCILBL: Close Monitor Label Space Processing

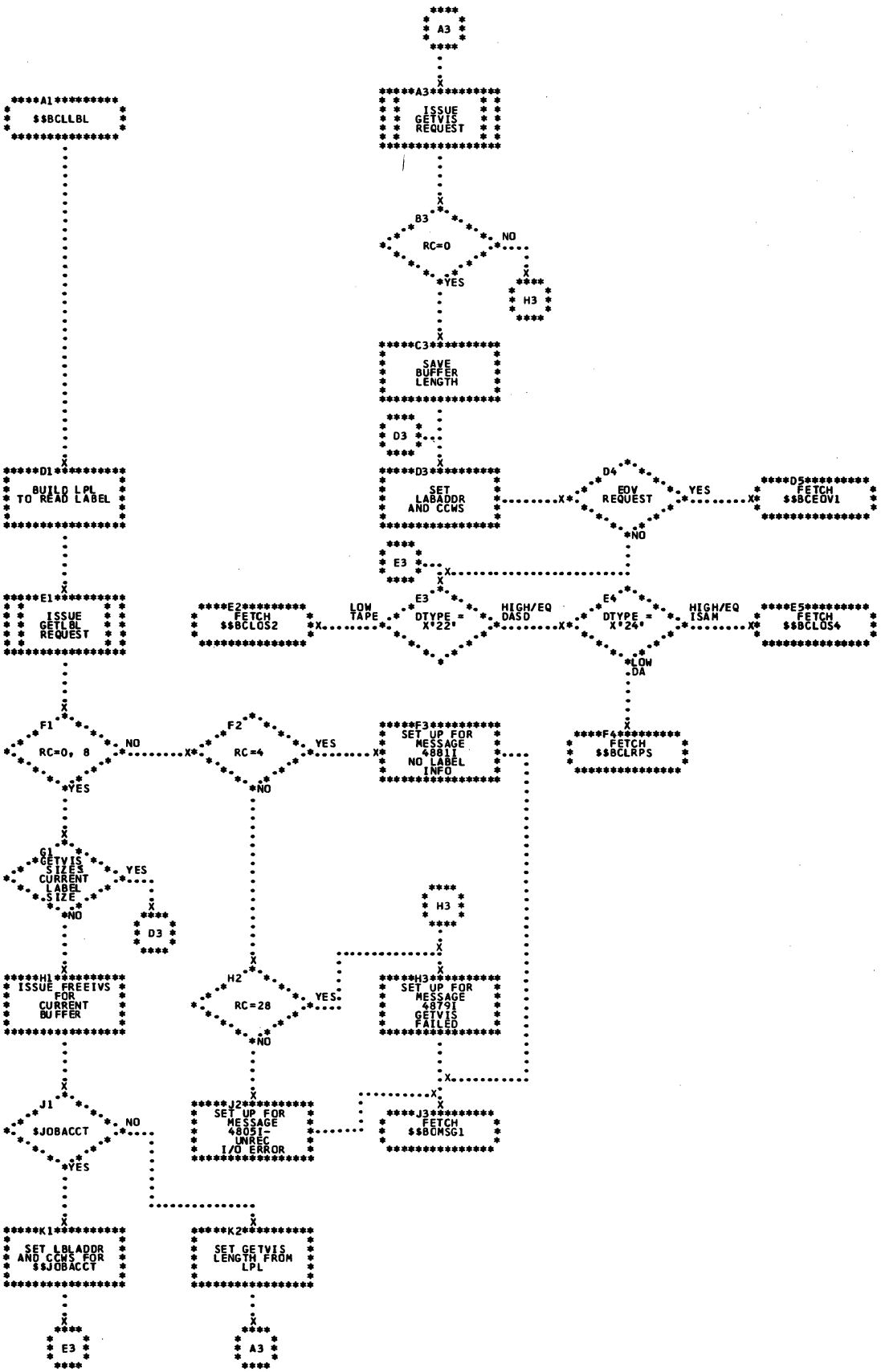


Chart BK. $$BCLOS2: Close Monitor, Phase 2 (Part 1 of 3)

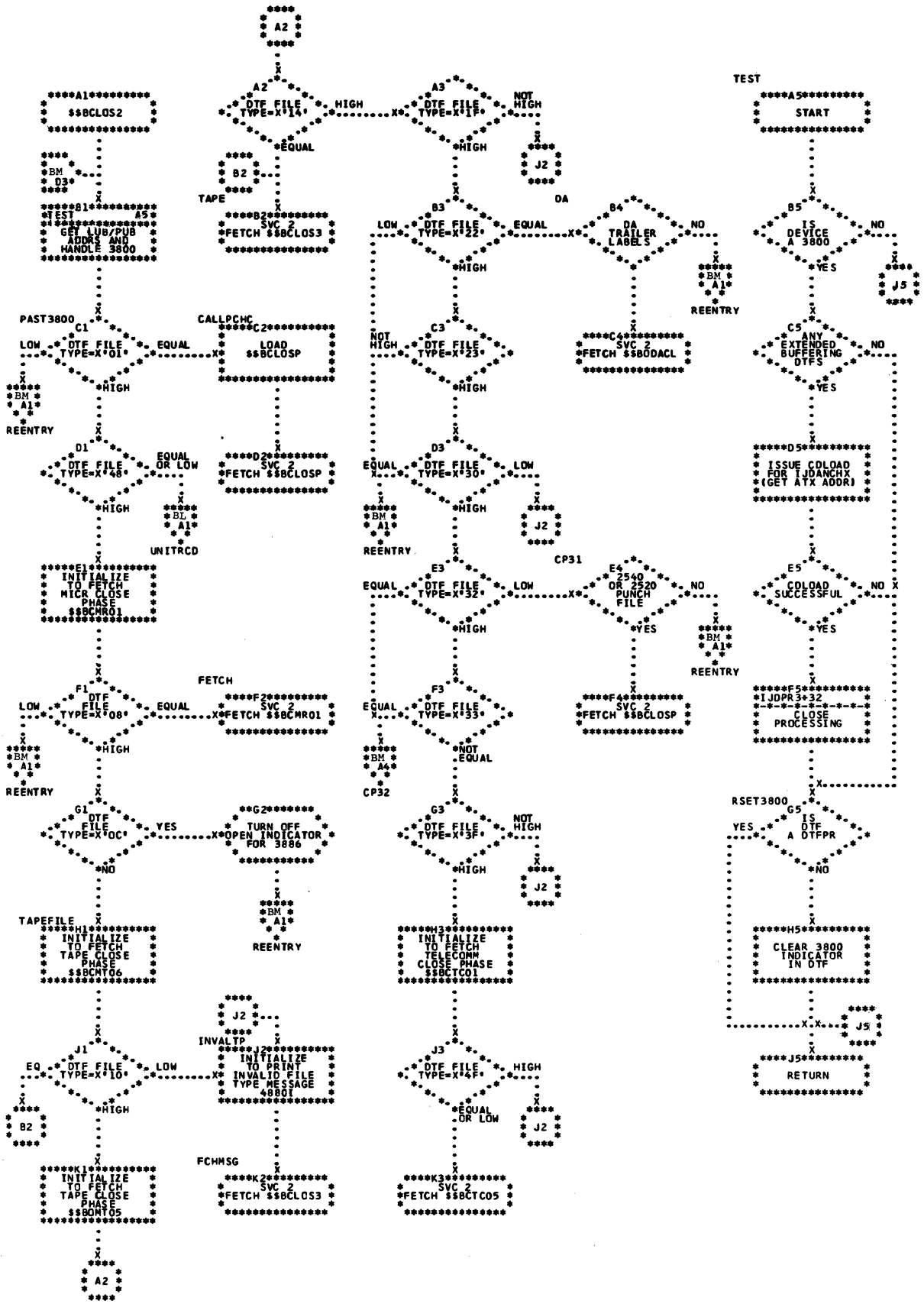


Chart BL. $$BCLOS2: Close Monitor, Phase 2 (Part 2 of 3)

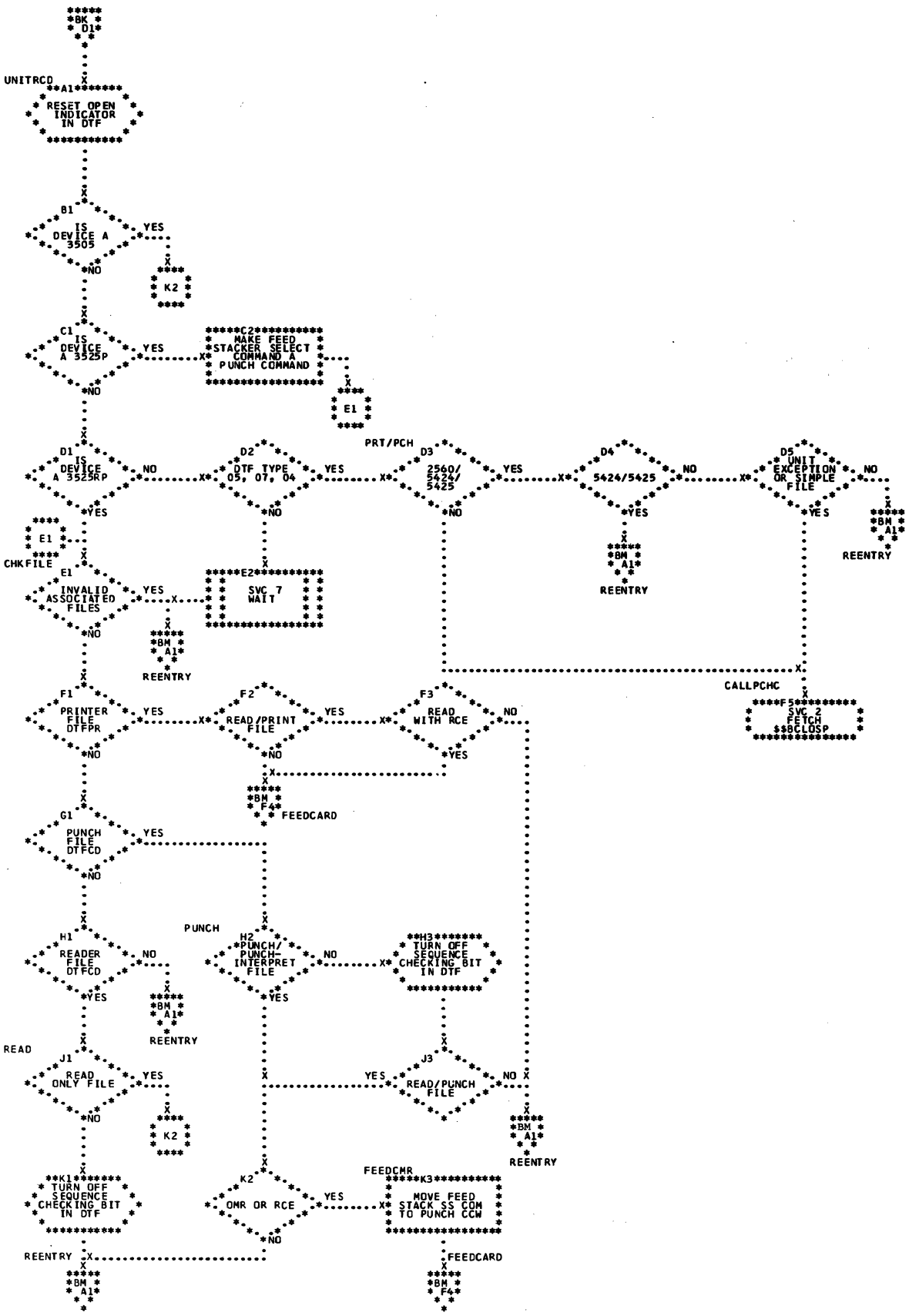


Chart BM. $$BCLOS2: Close Monitor, Phase 2 (Part 3 of 3)

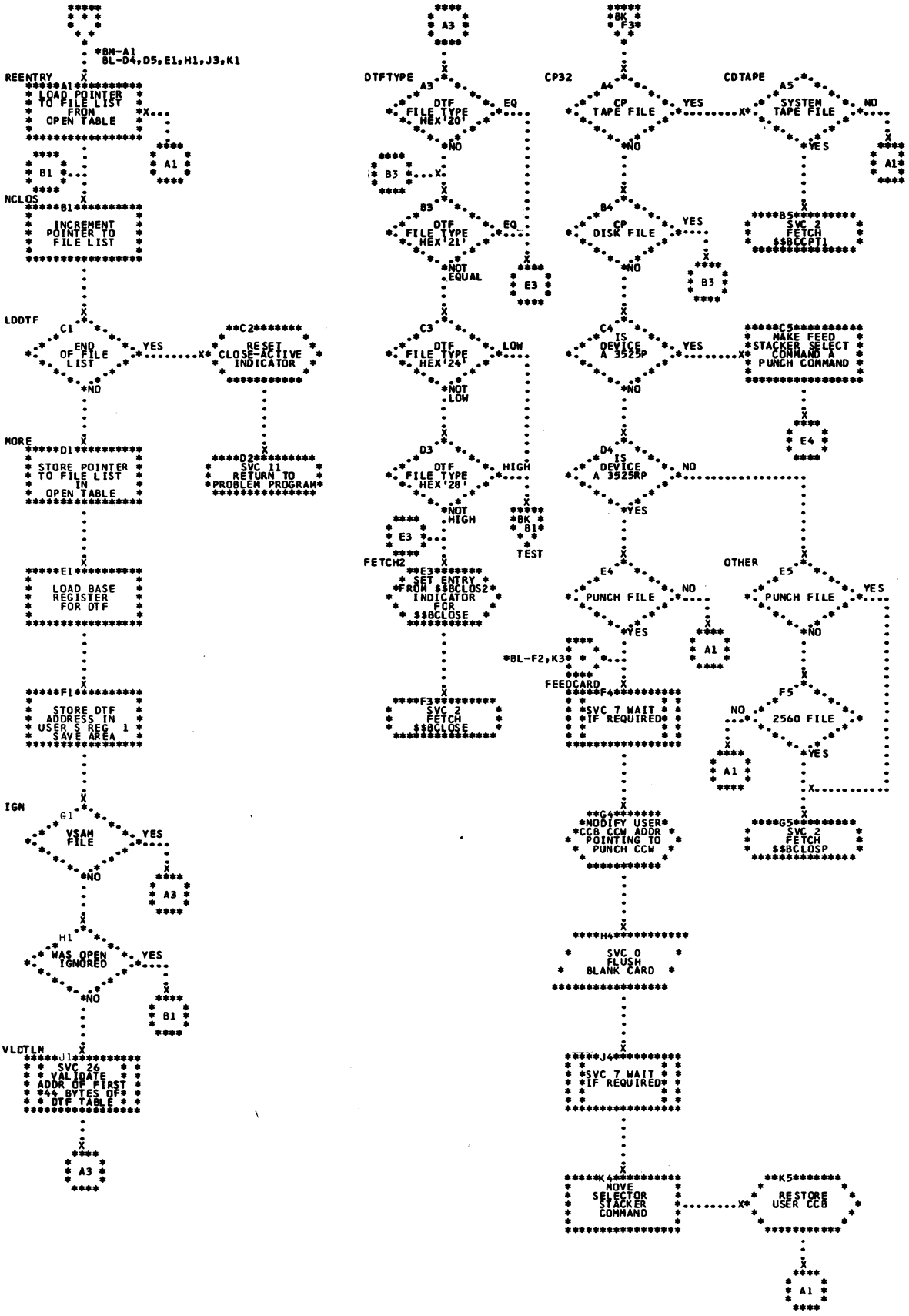


Chart BN. $$BCLOS3: Close Monitor, Phase 3

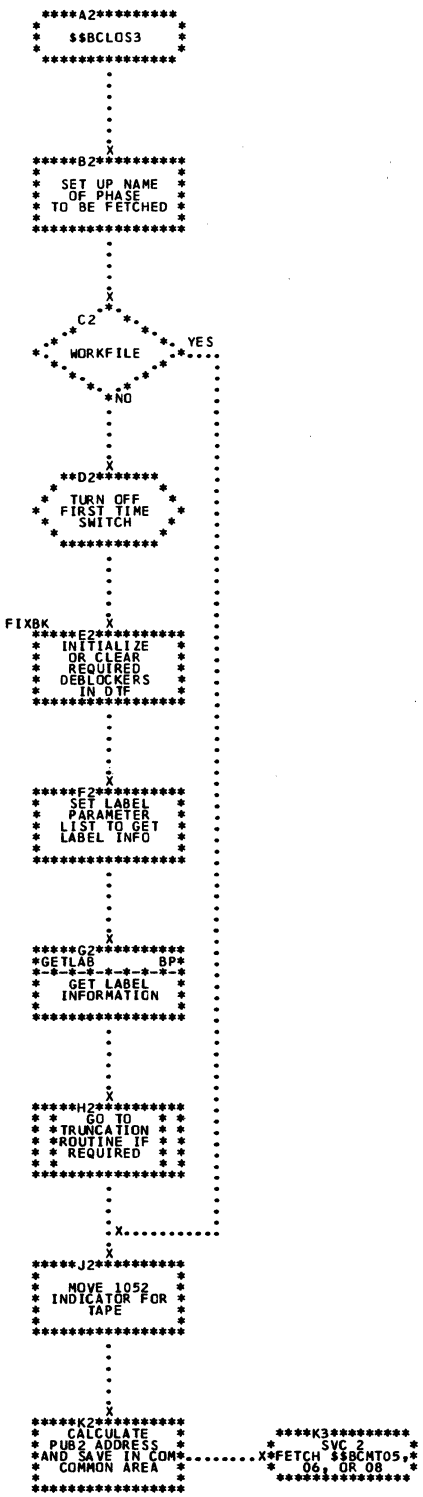


Chart BO. $$BCLOS4: Close Monitor, Phase 4

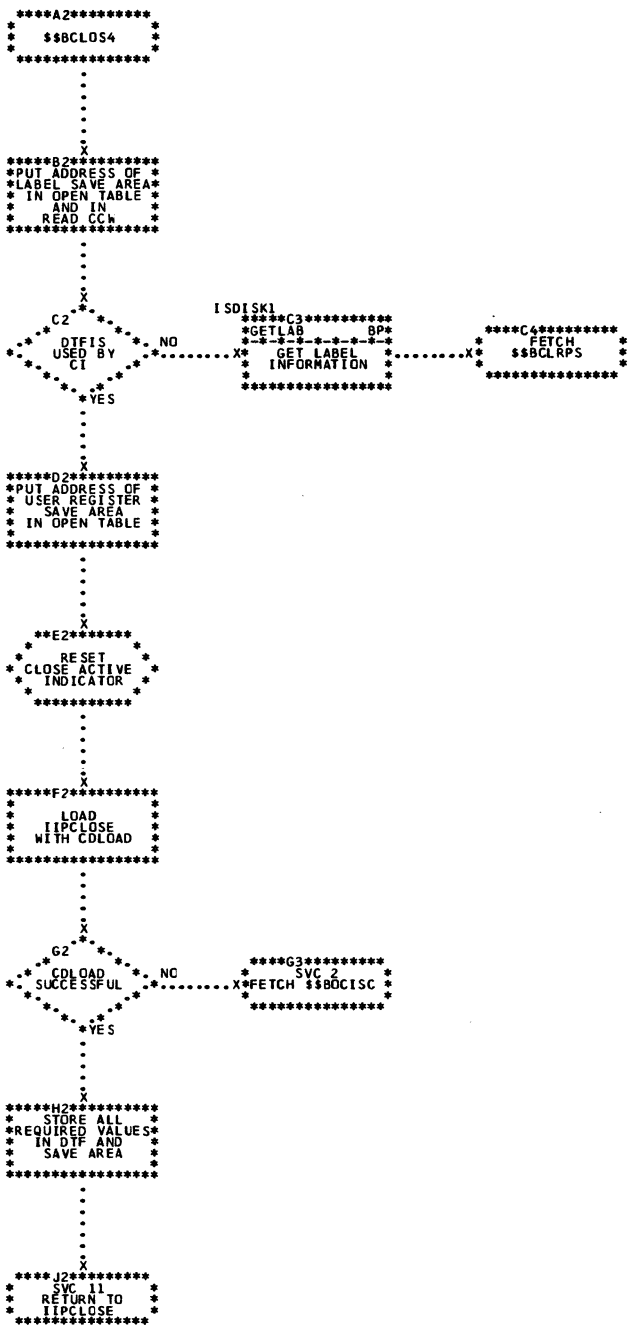


Chart BP. Close Monitor Subroutines

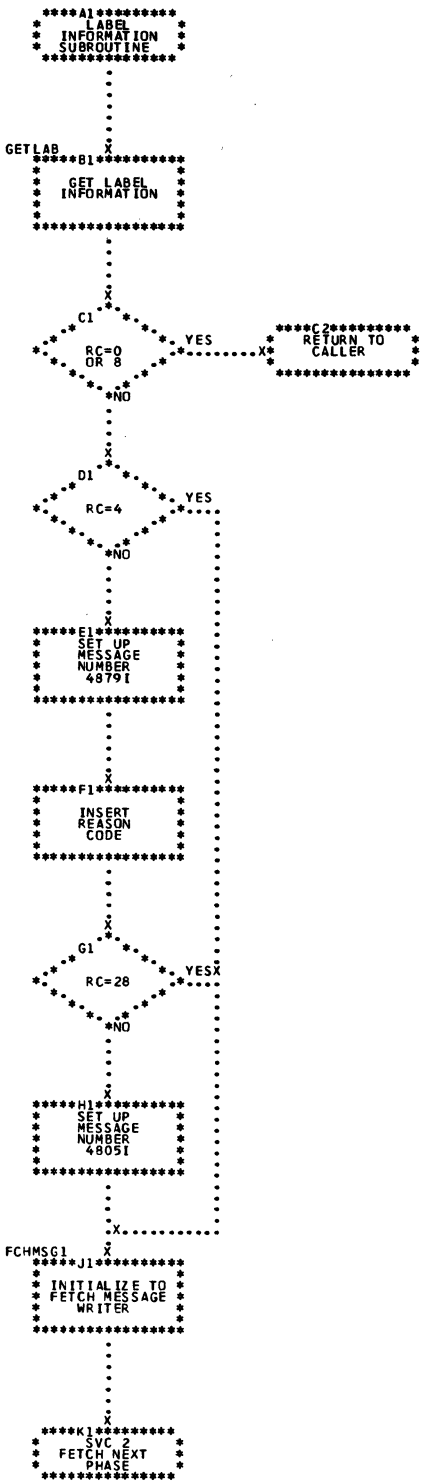


Chart BQ. $$BCIRPS: DASD RPS Common Close

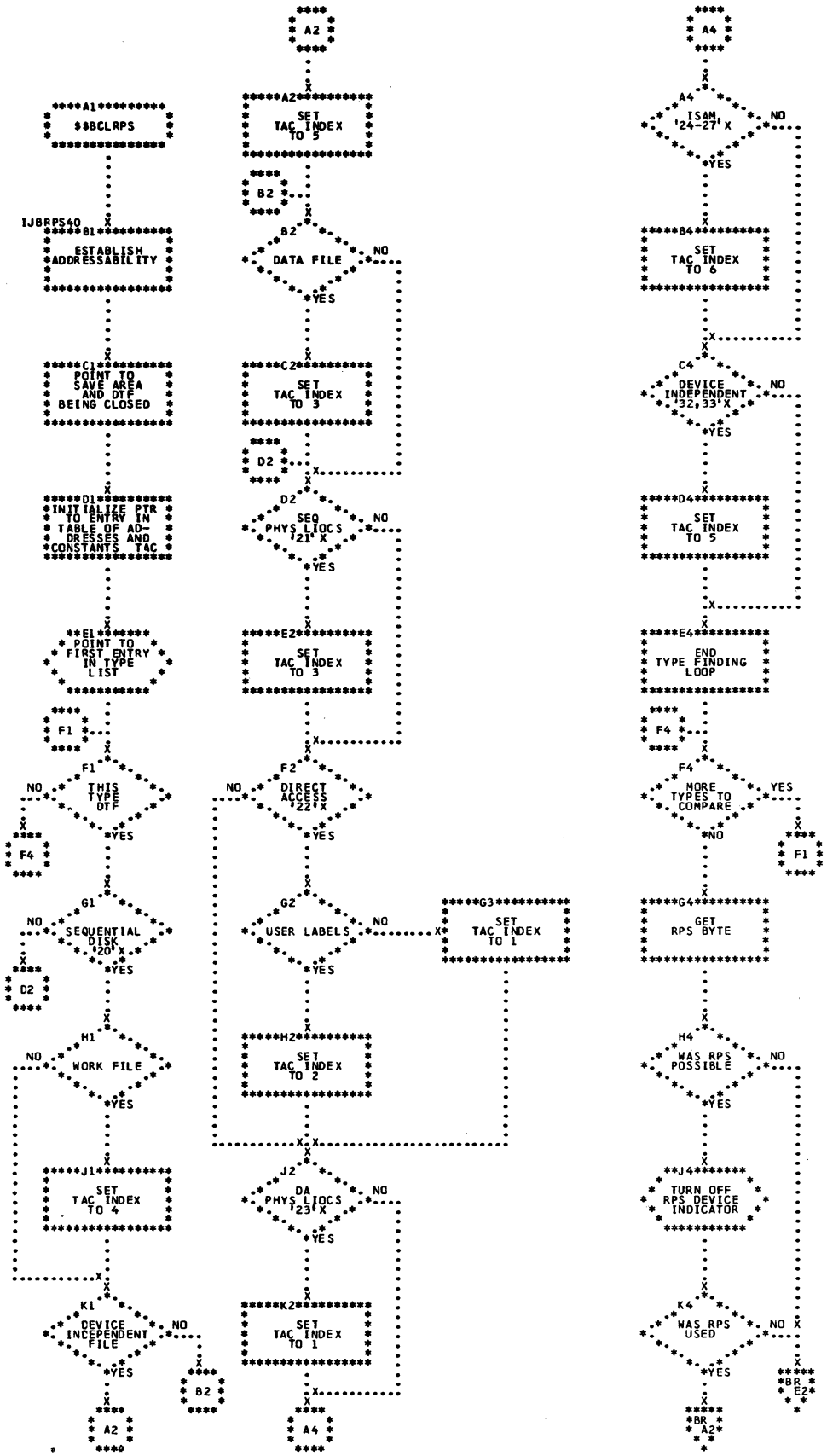


Chart BR. $$BCIRPS: DASD RPS Common Close, Restore User's DTF

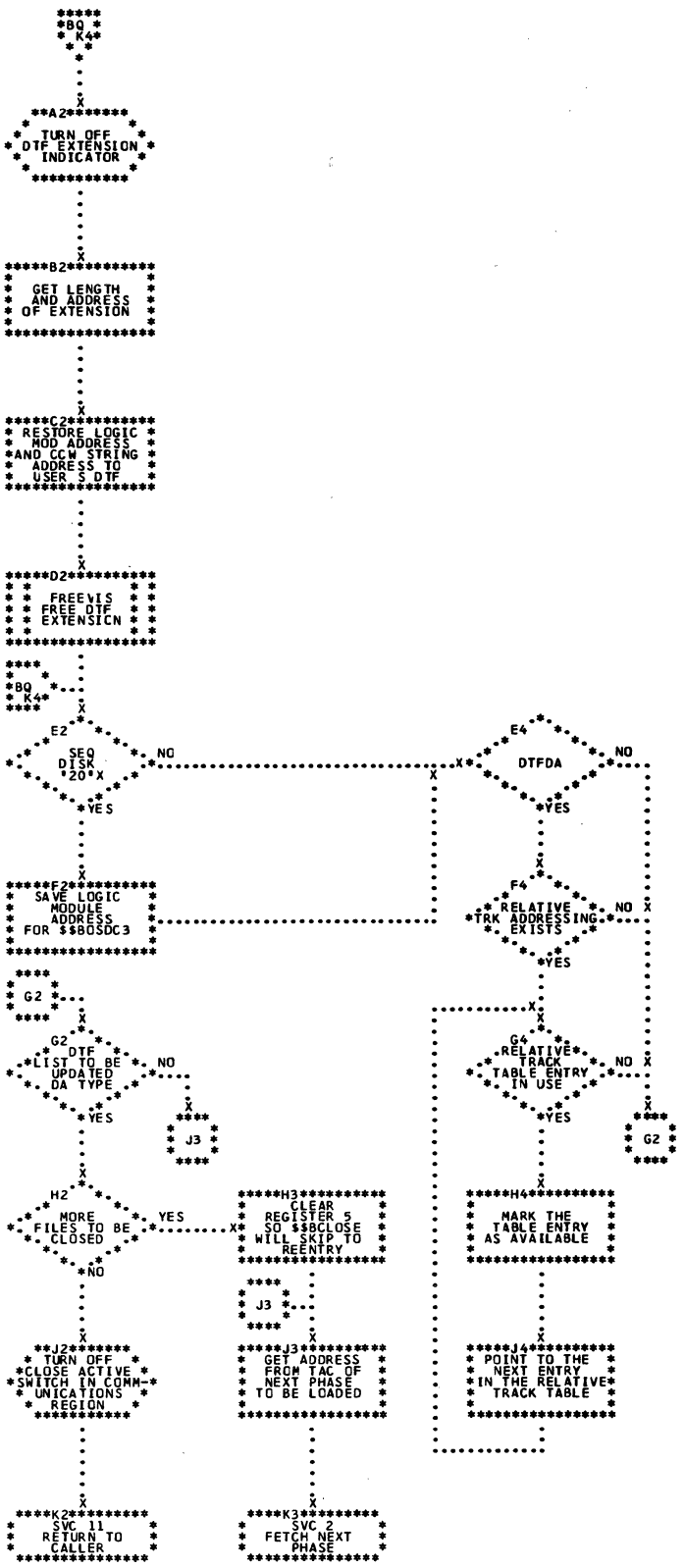


Chart BS. $$BOPENS: IOCS and Device Independent I/O Initialization

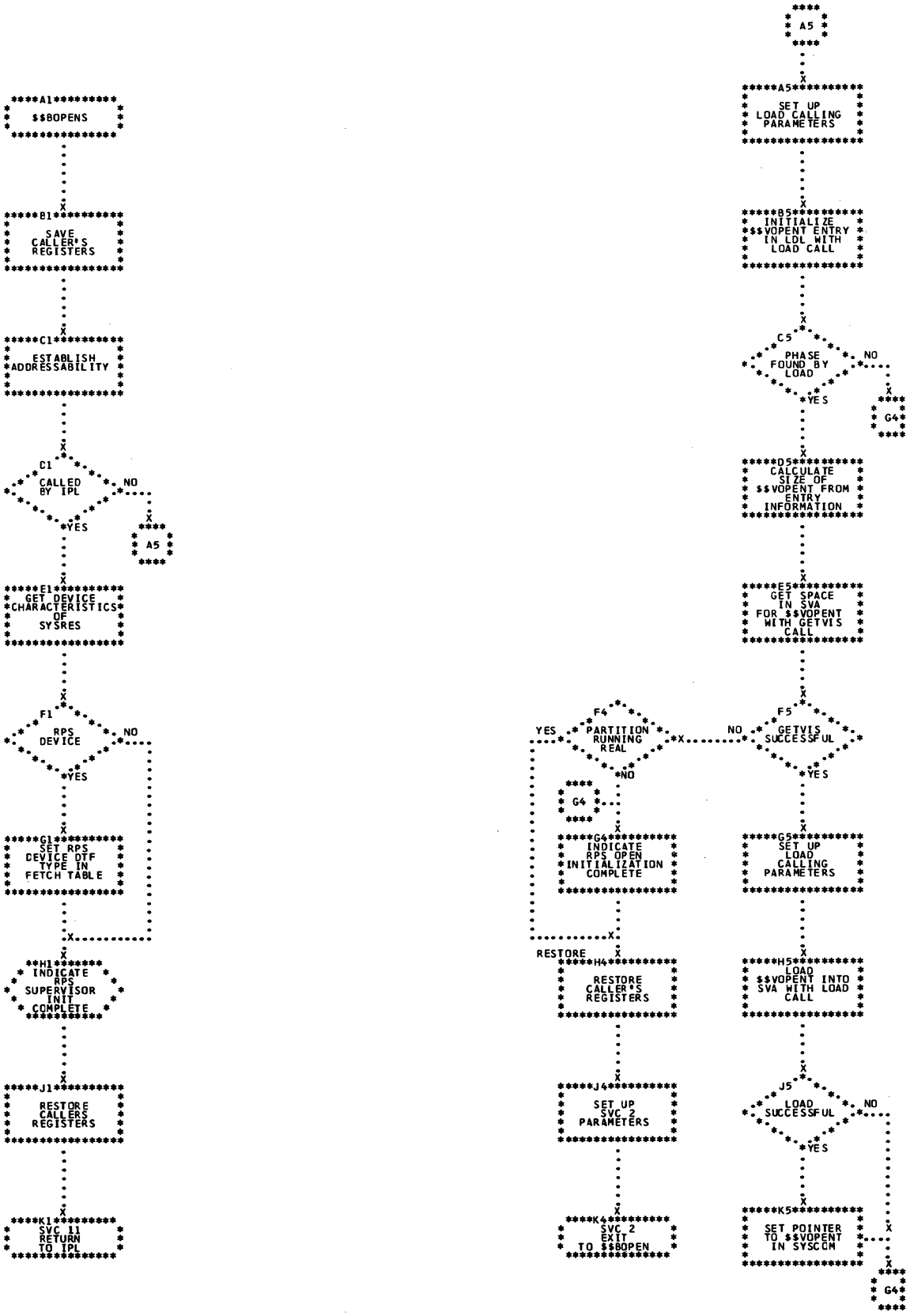


Chart BT. $$VOPENT: IOCS and Device Independent I/O Initialization (Part 1 of 2)

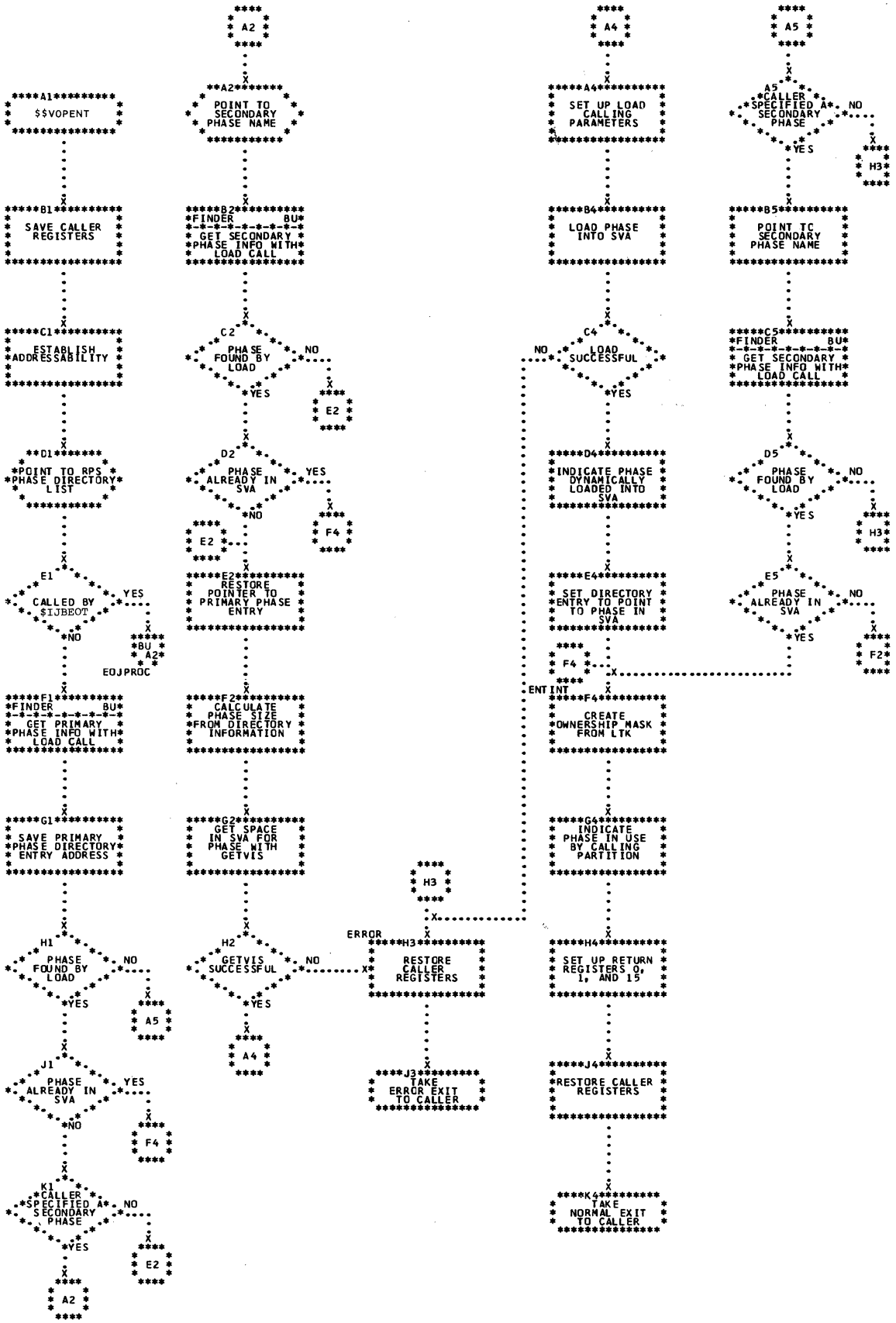


Chart BU. $$VOPENT: IOCS and Device Independent I/O Initialization (Part 2 of 2)

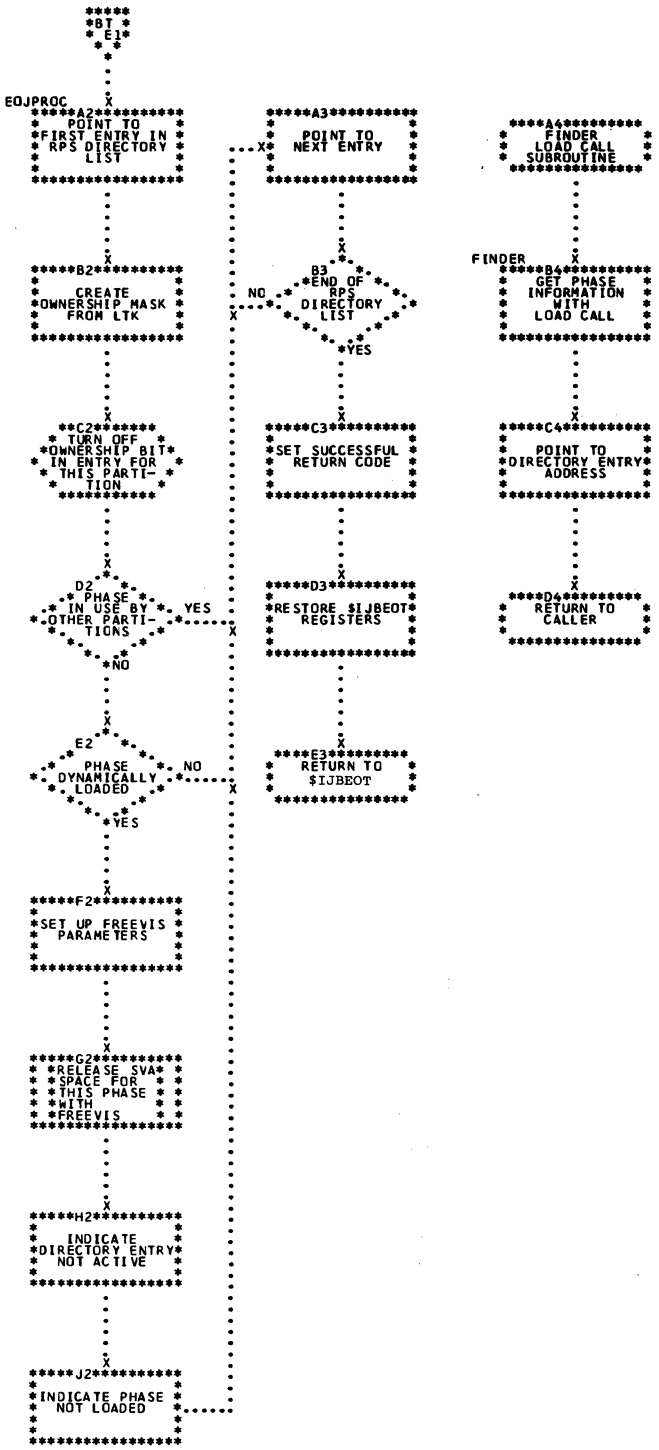


Chart CA. $$BOSDC1: SD Close Input and Output (Part 1 of 2)

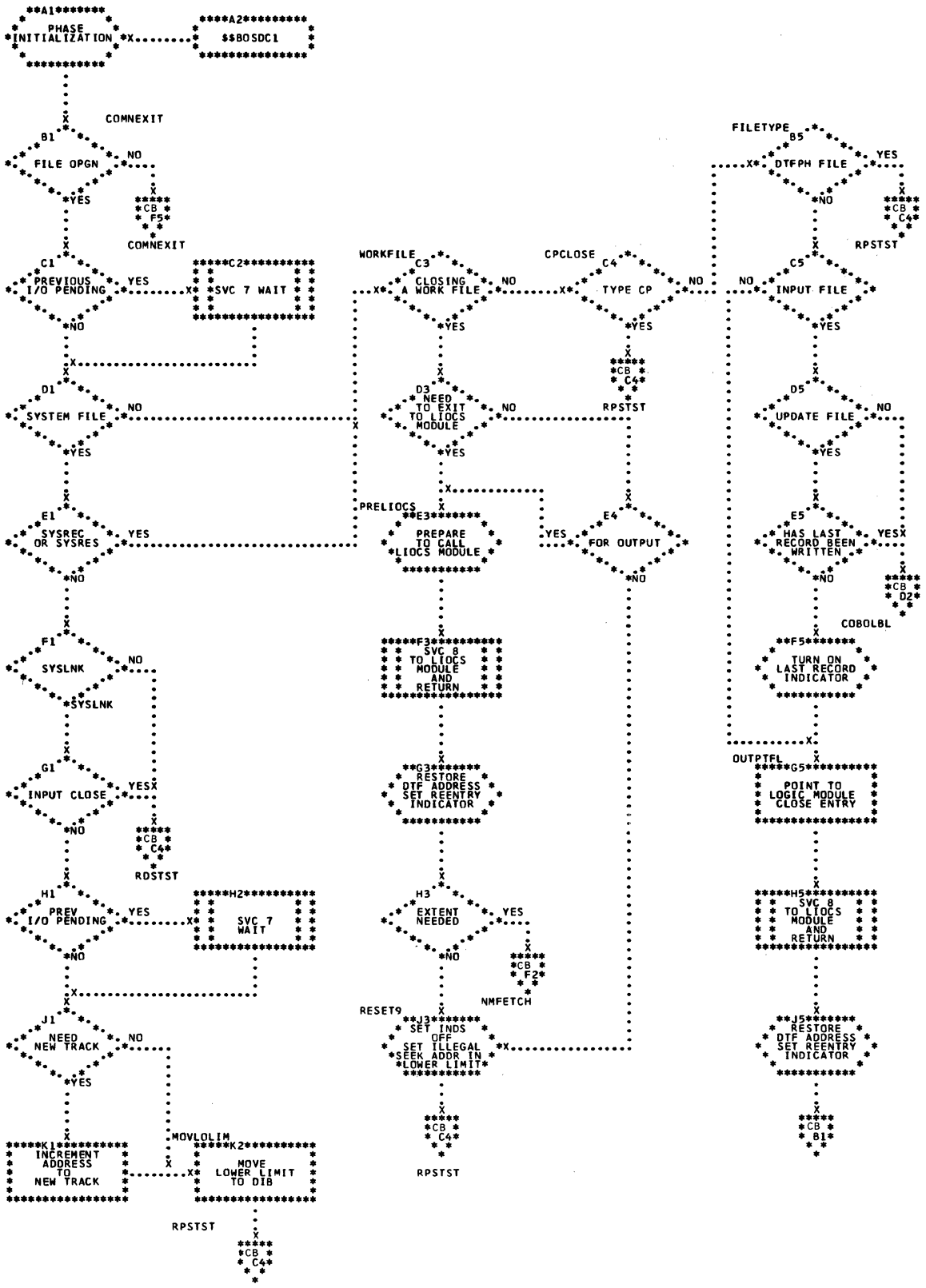


Chart CB. $$BOSDC1: SD Close Input and Output (Part 2 of 2)

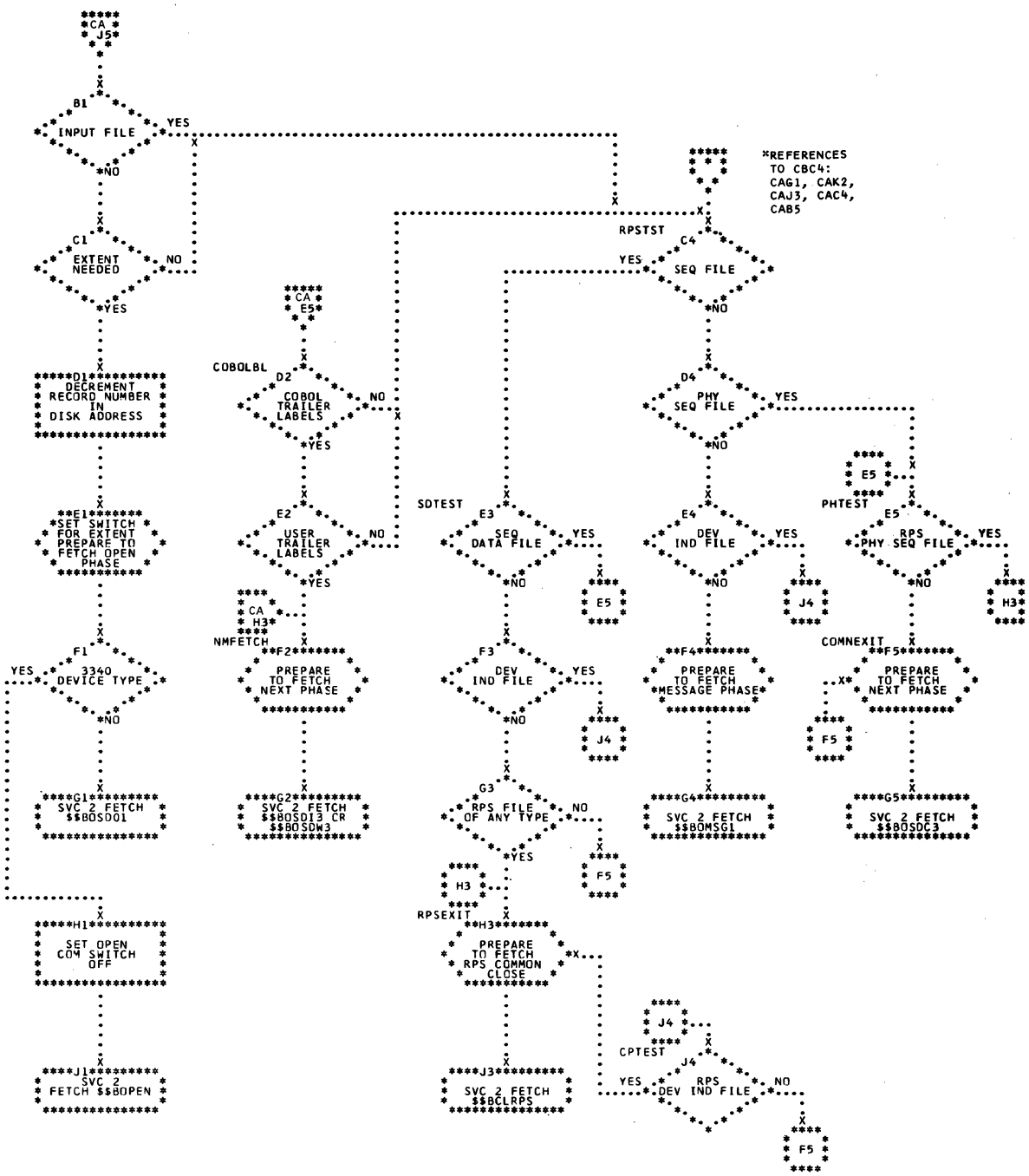


Chart CC. $$BOSDC2: Close, Free Track Function

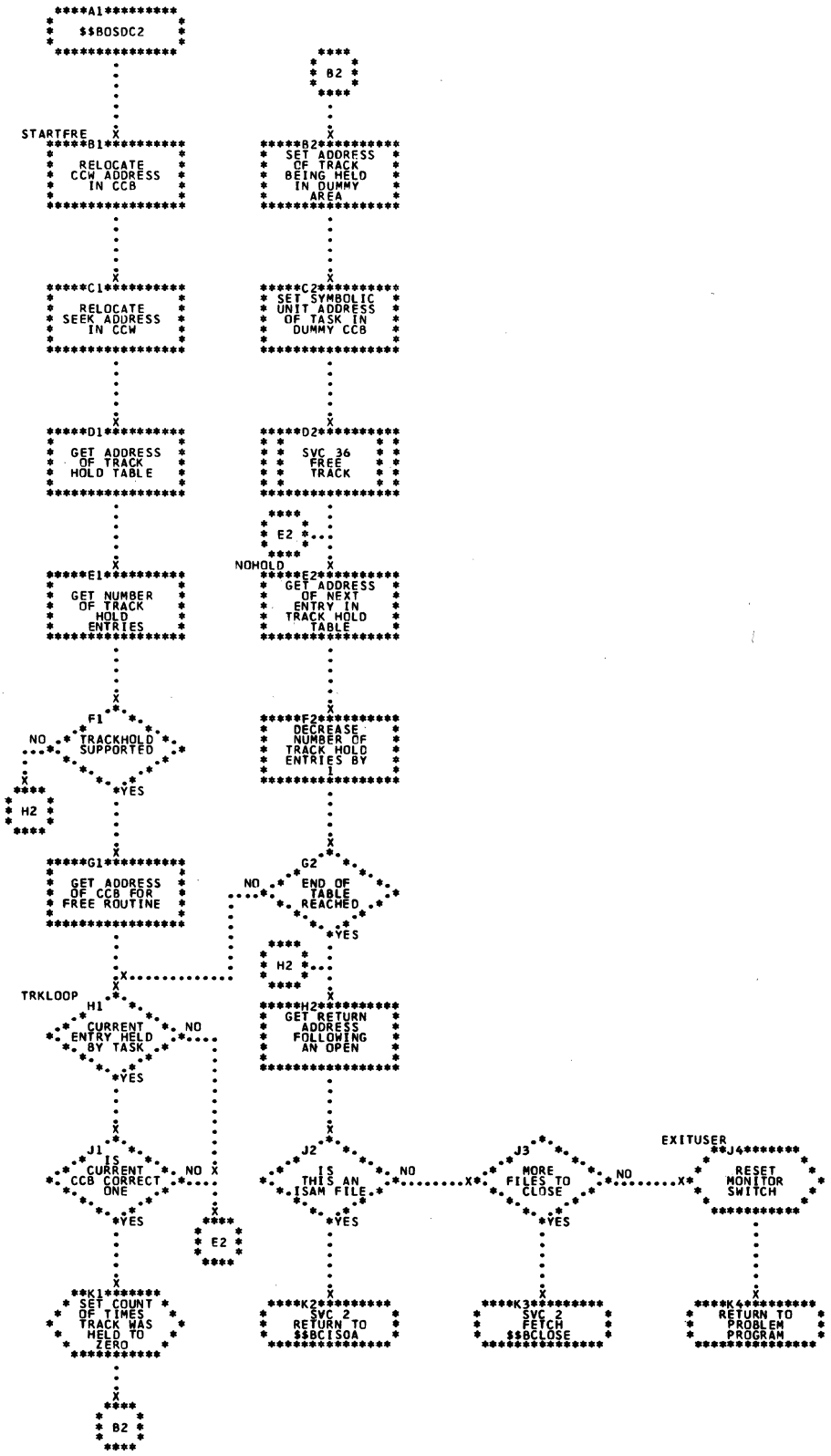


Chart CD. $$BOSDEV: Forced End of Volume for Disk

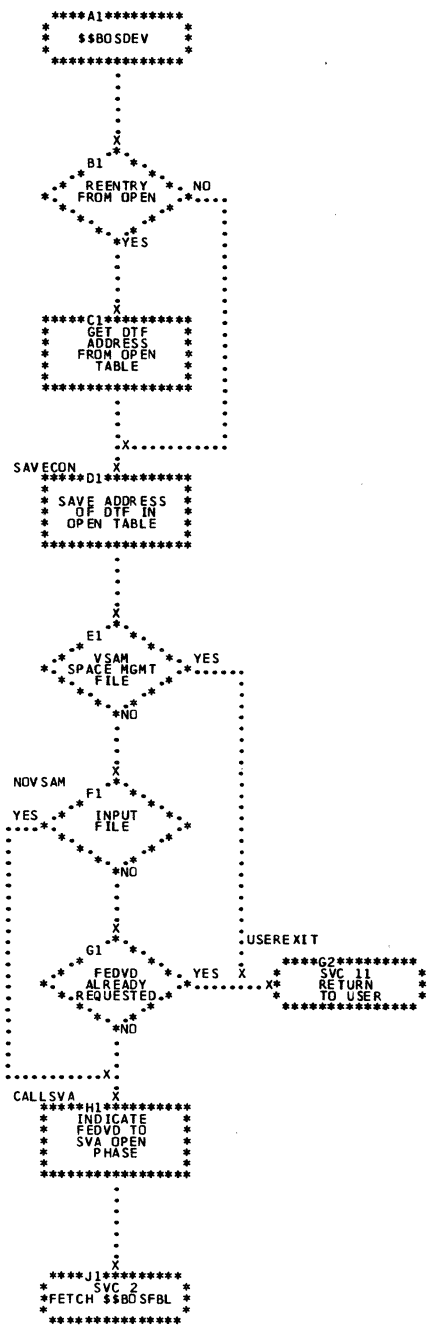


Chart CE. $$BODQUE: Dequeue Extent JIBs

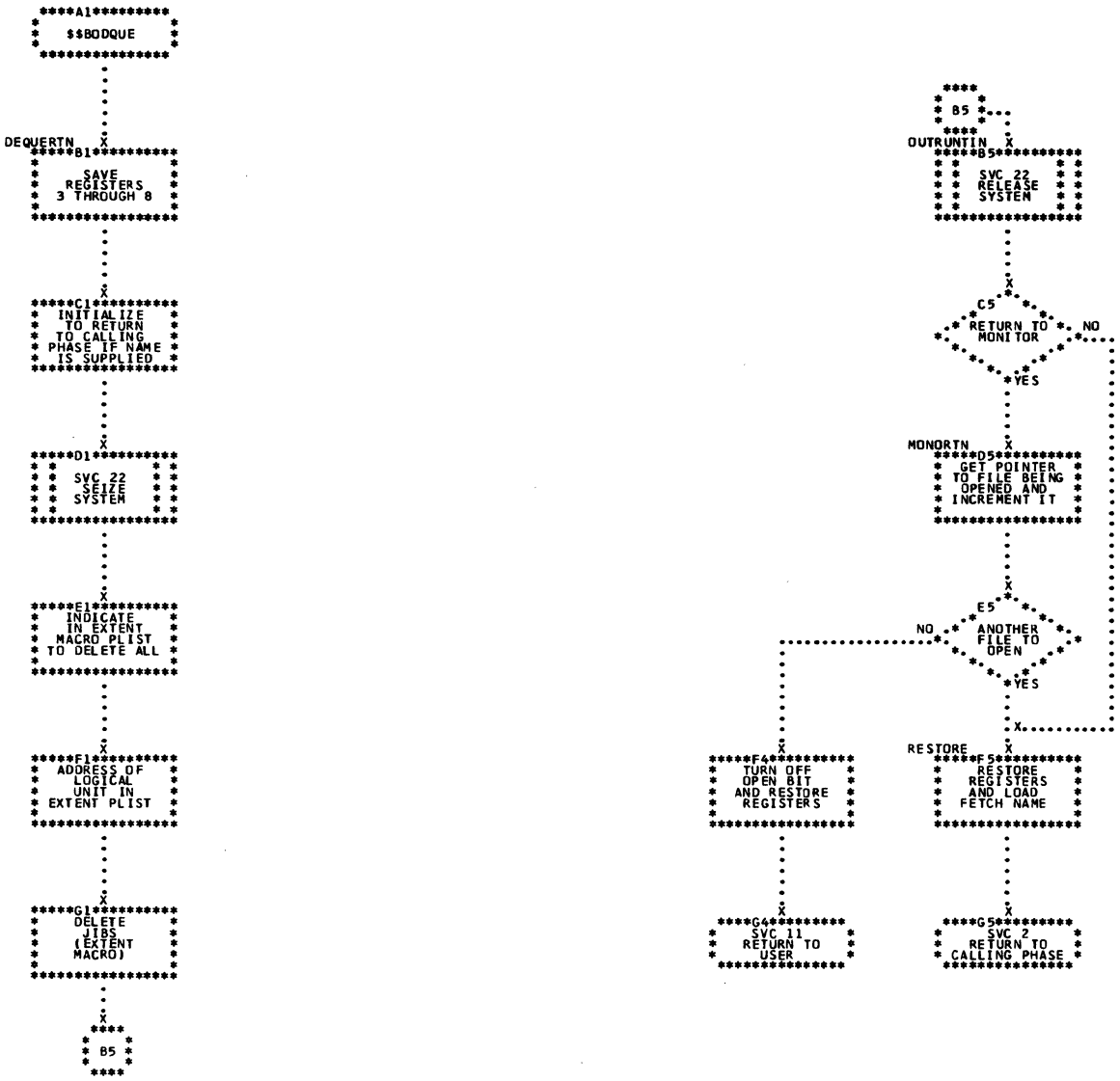


Chart EE. $$BRELSE: Dynamic Device Release Transient (Part 1 of 3)

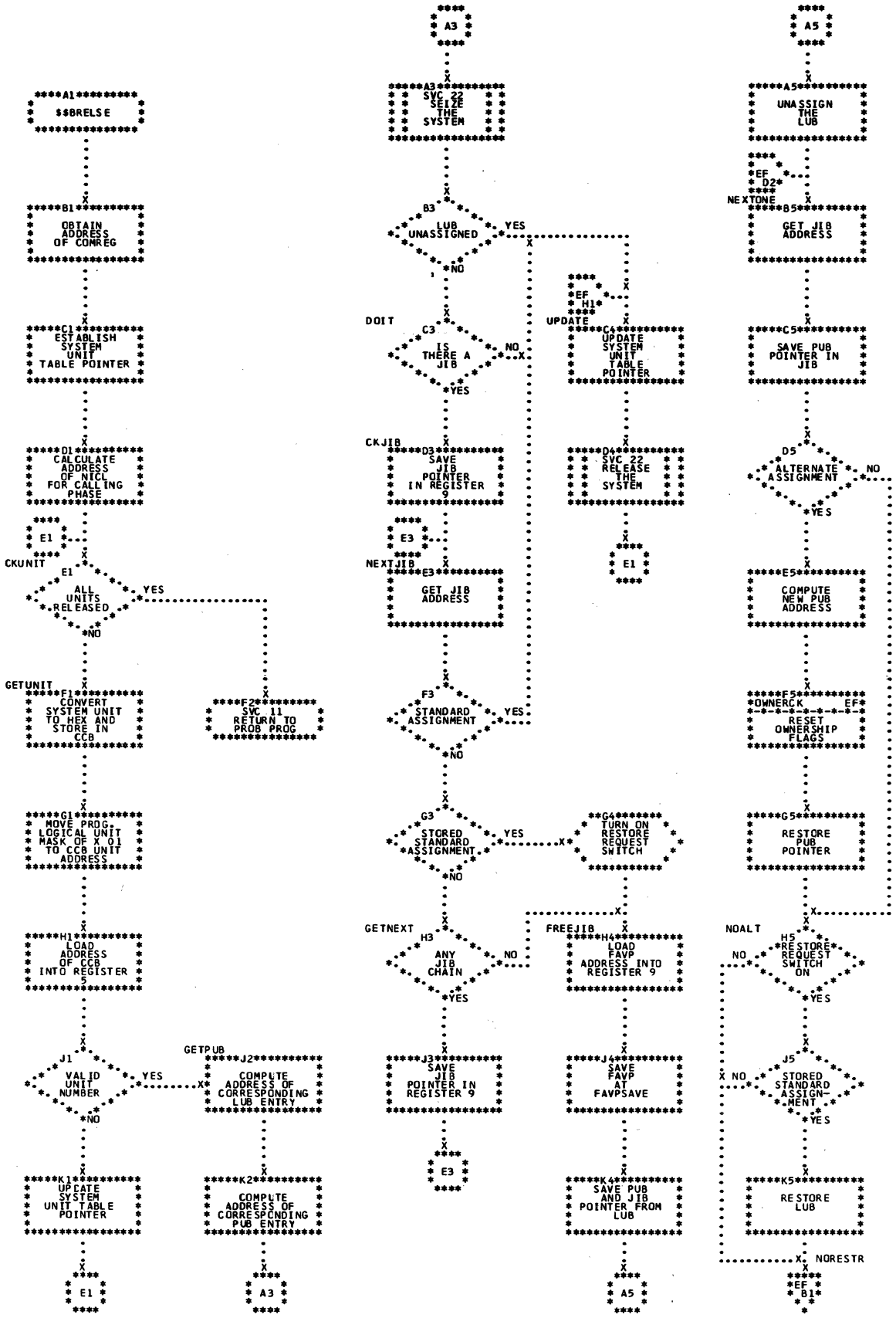


Chart EF. $$BRELSE: Dynamic Device Release Transient (Part 2 of 3)

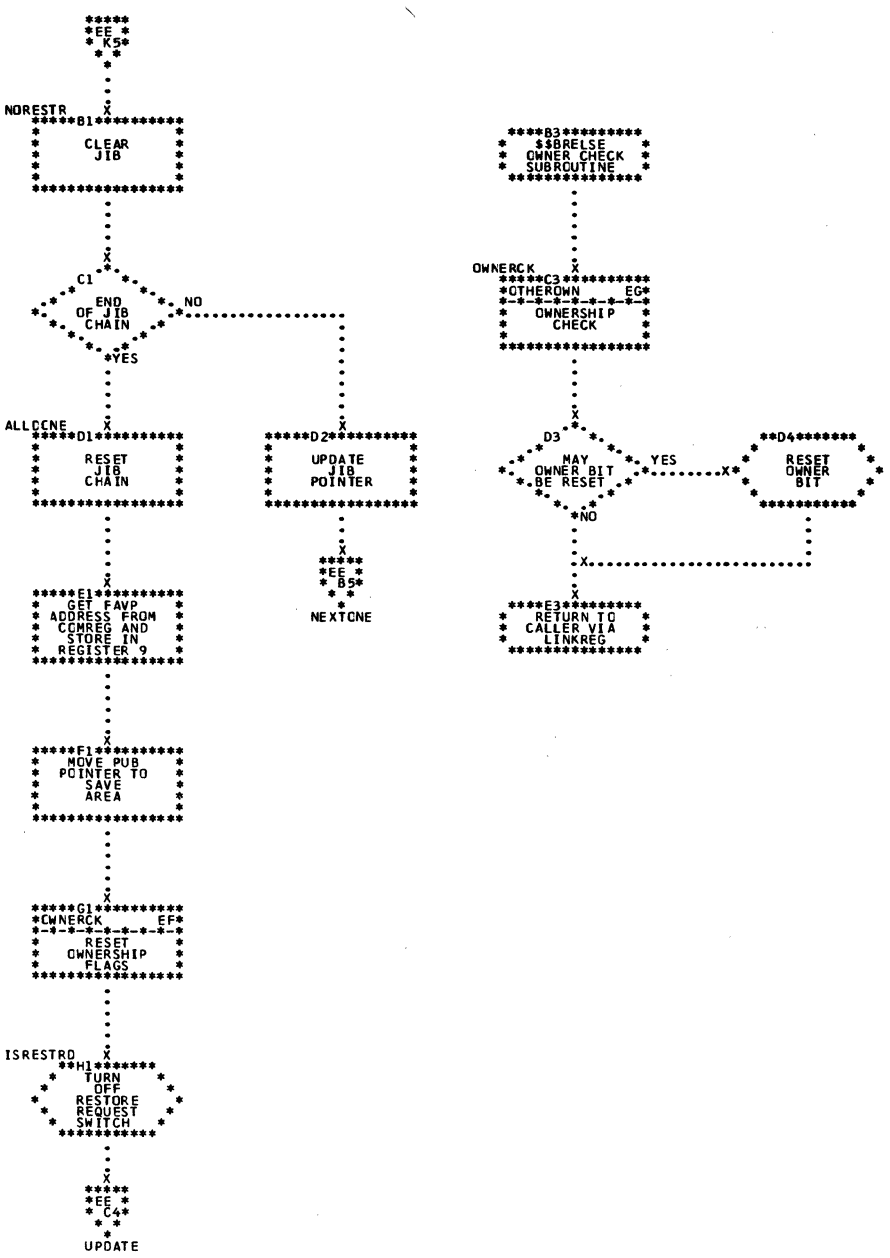


Chart EG. $$BRELSE: Dynamic Device Release Transient (Part 3 of 3)

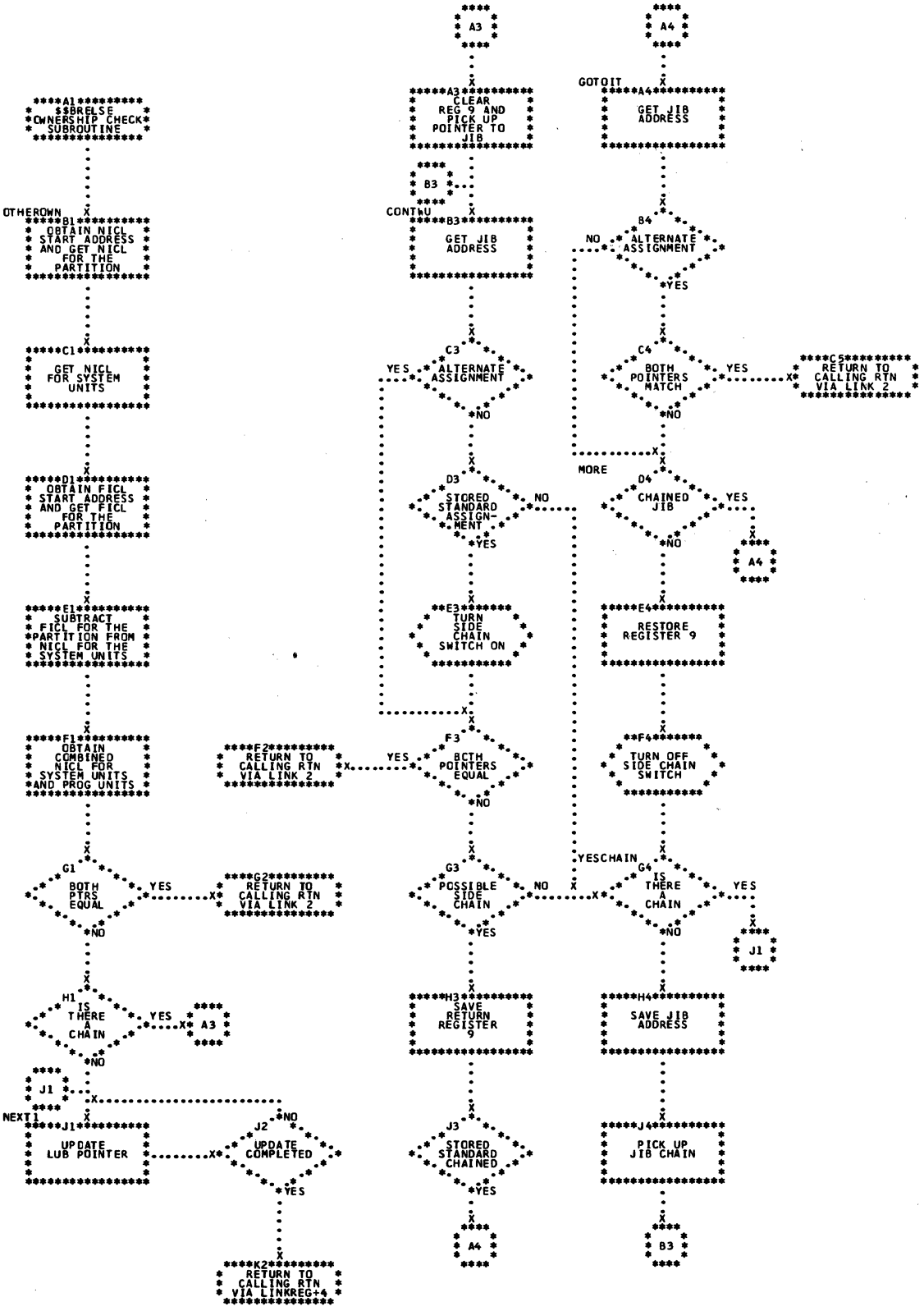


Chart FA. $$BOFLPT: DASD File-Protect (Part 1 of 3)

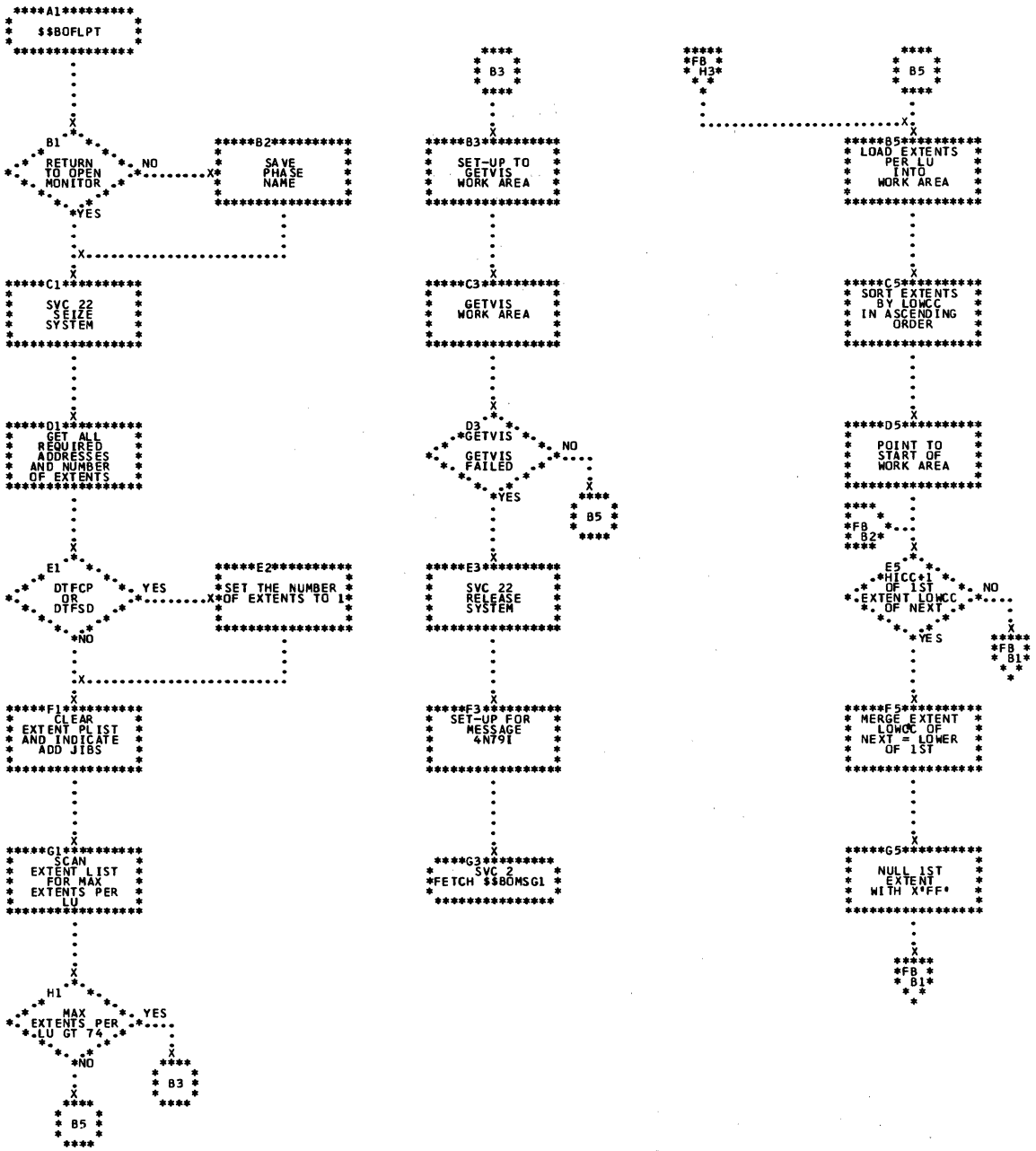


Chart FB. $$BOFLPT: DASD File-Protect (Part 2 of 3)

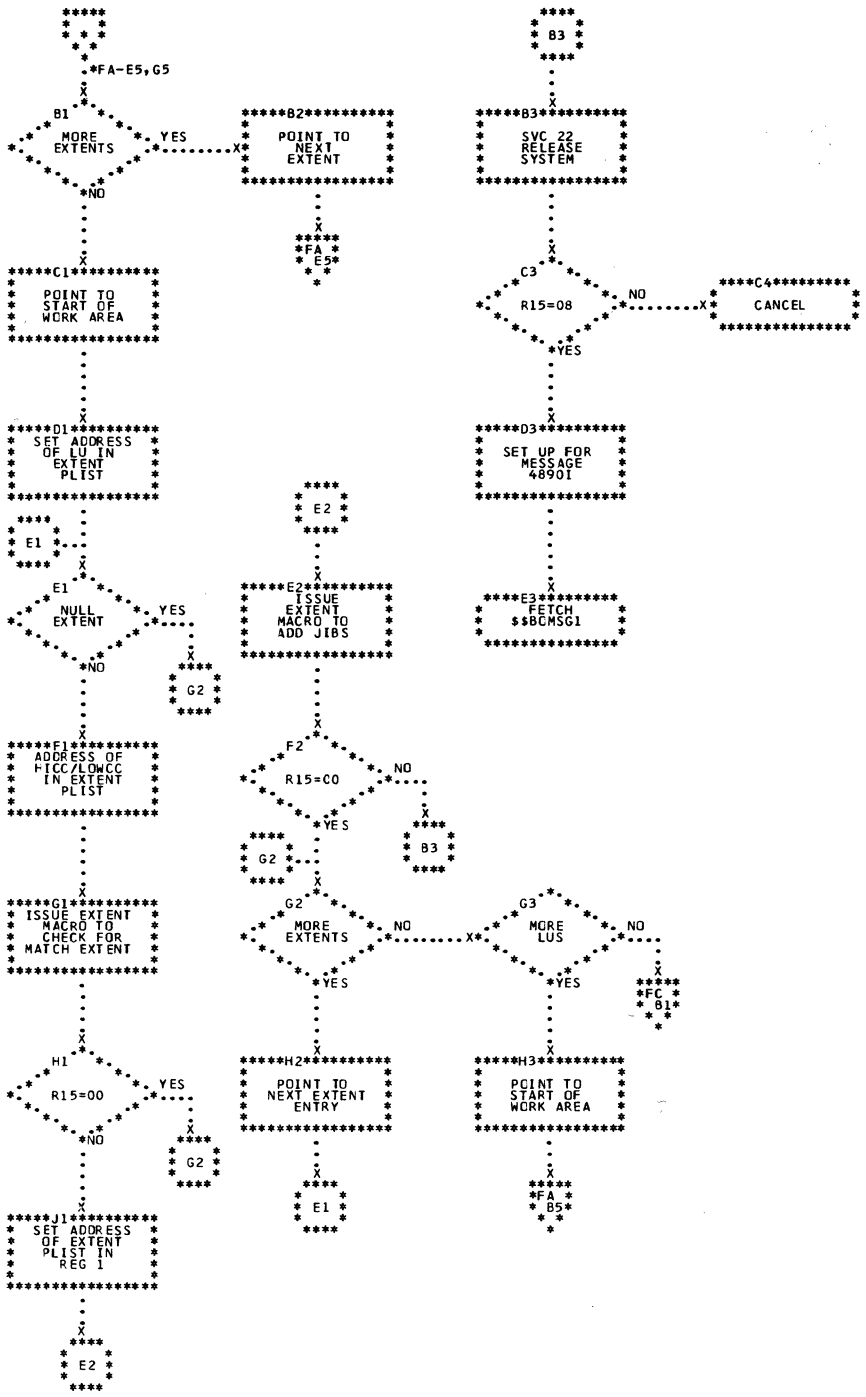


Chart FC. $$BOFLPT: DASD File Protect (Part 3 of 3)

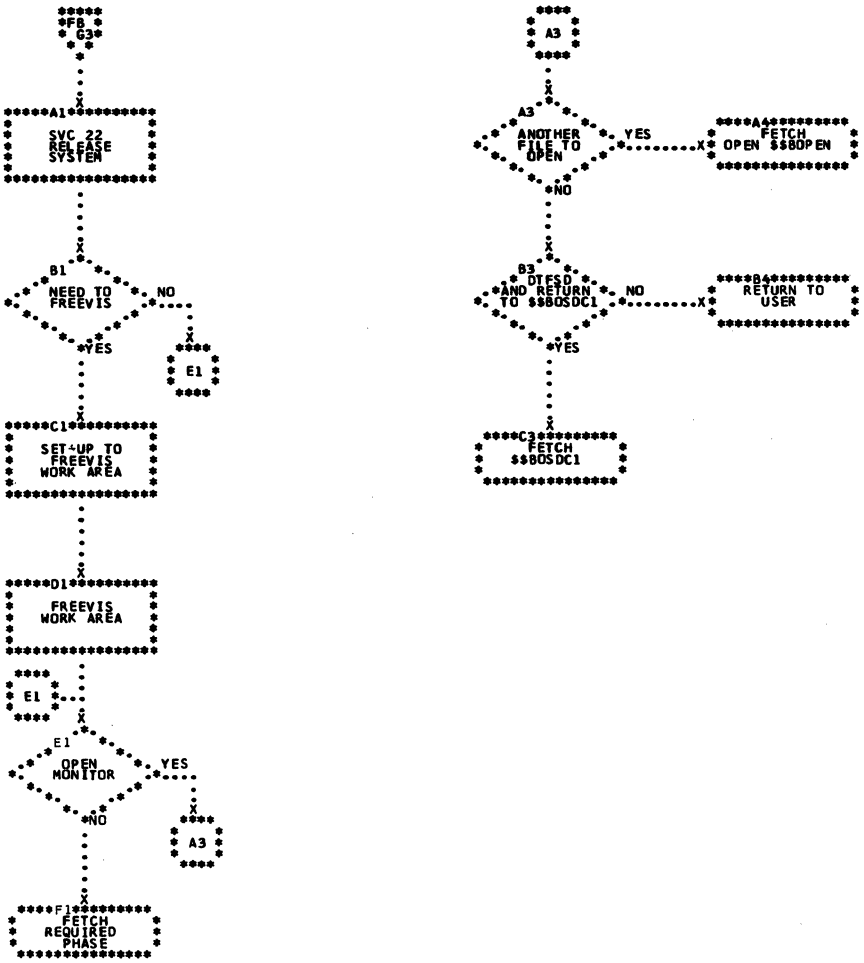


Chart FD. $$BODSPV: VTOC Display, Phase 1

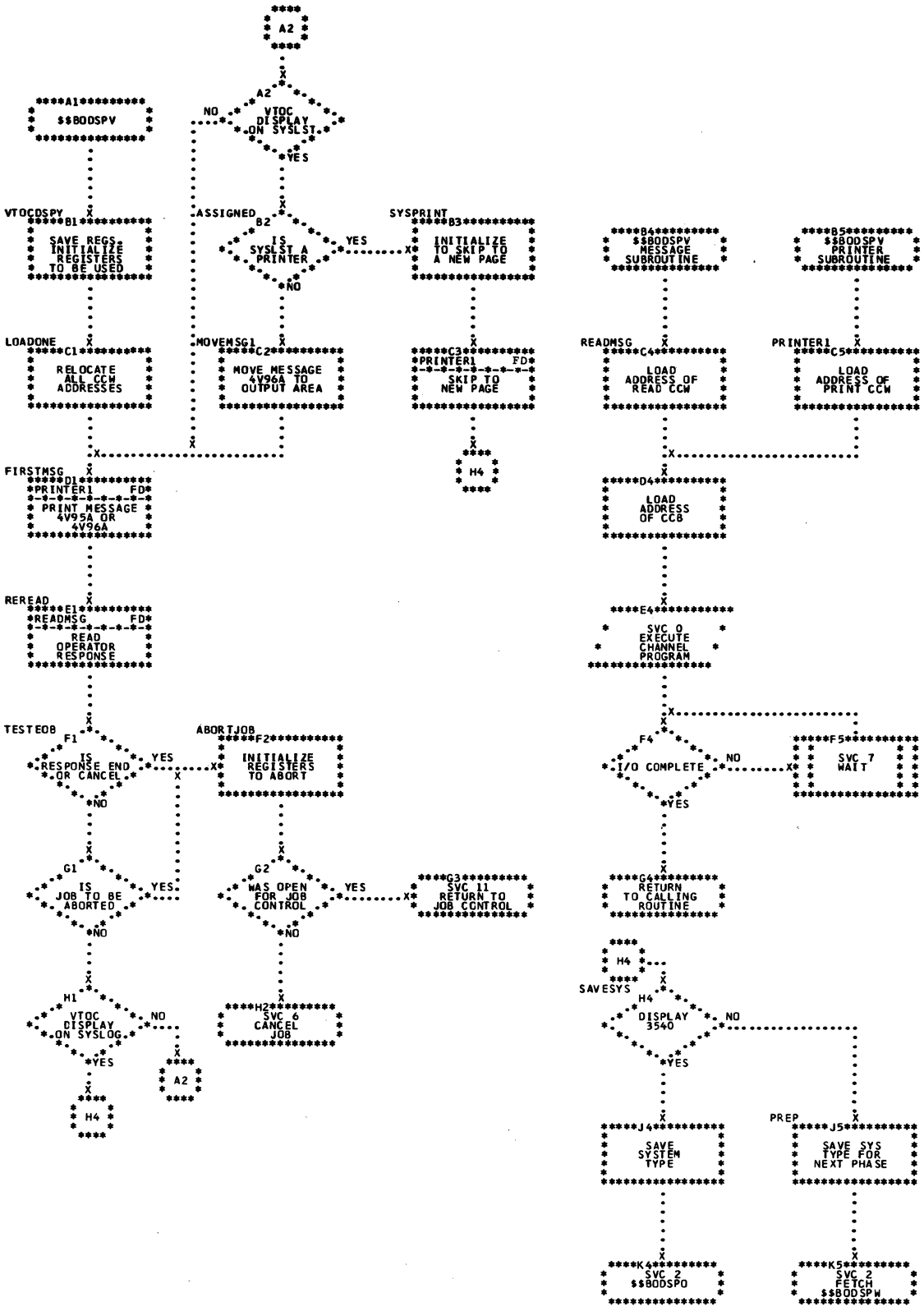


Chart FE. $$BODSPW: VTOC Display, Phase 2 (Part 1 of 2)

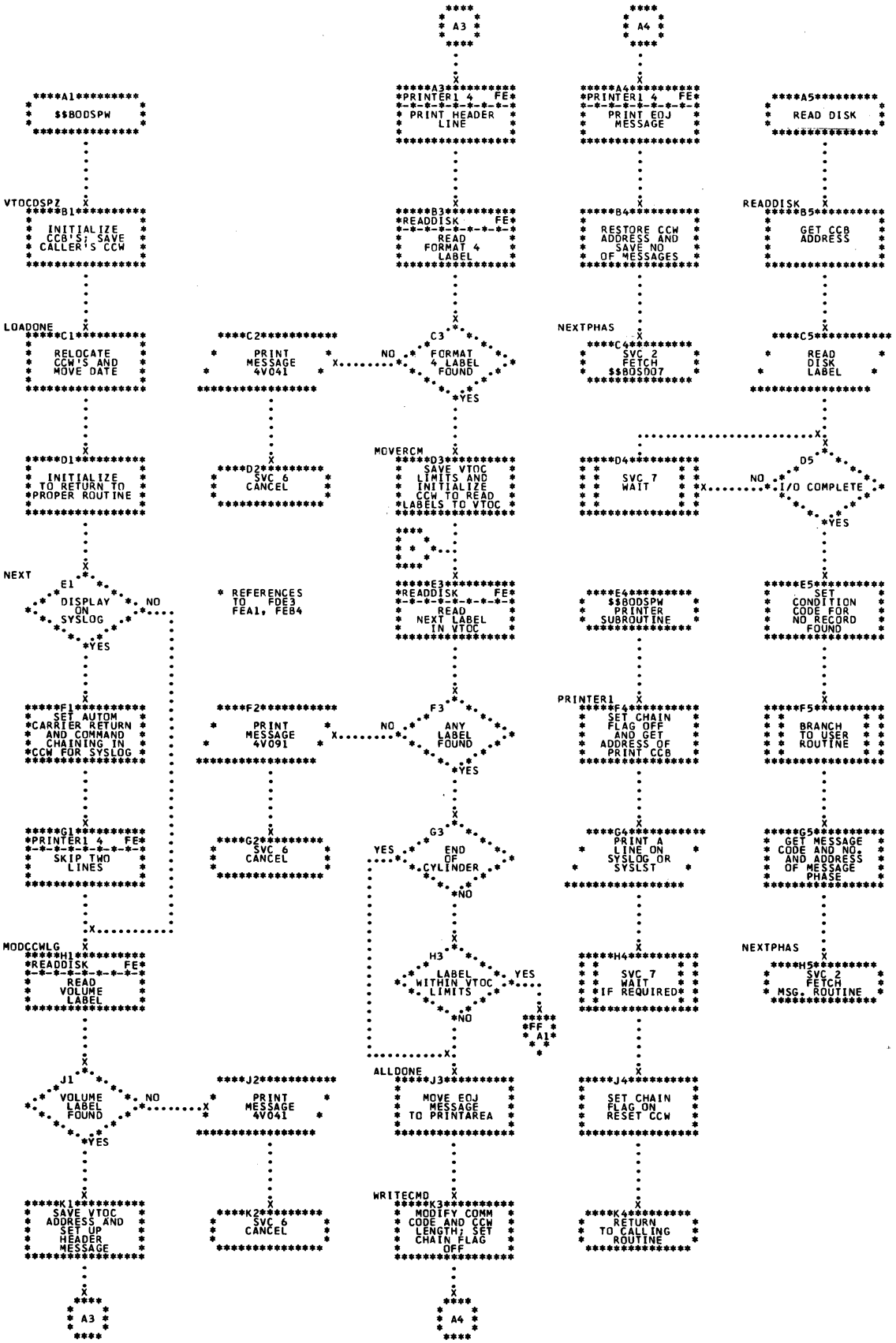


Chart FF. $$BODSPW: VTOC Display, Phase 2 (Part 2 of 2)

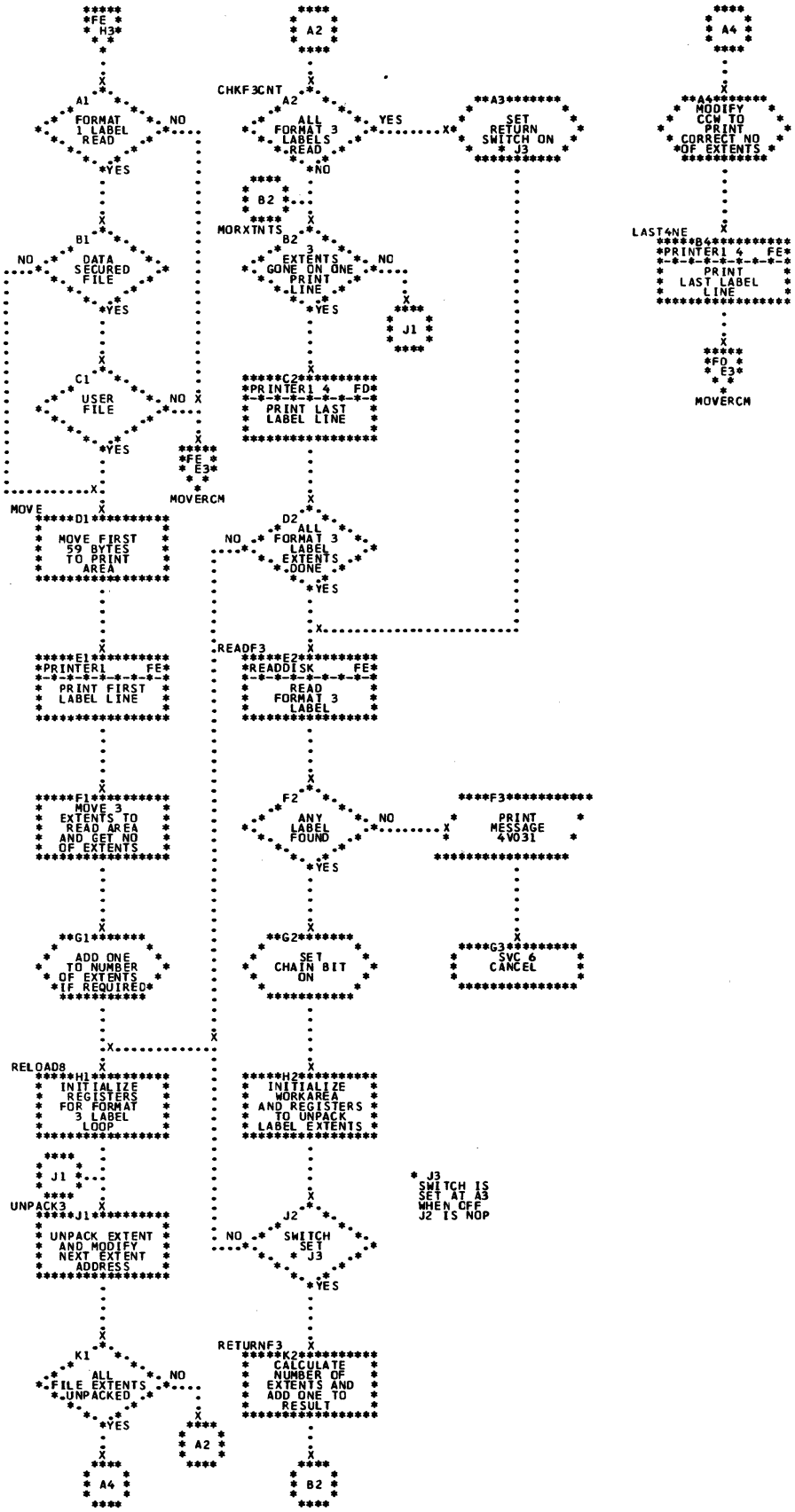


Chart FG. $$BOVDMP: VTOC Dump (Part 1 of 2)

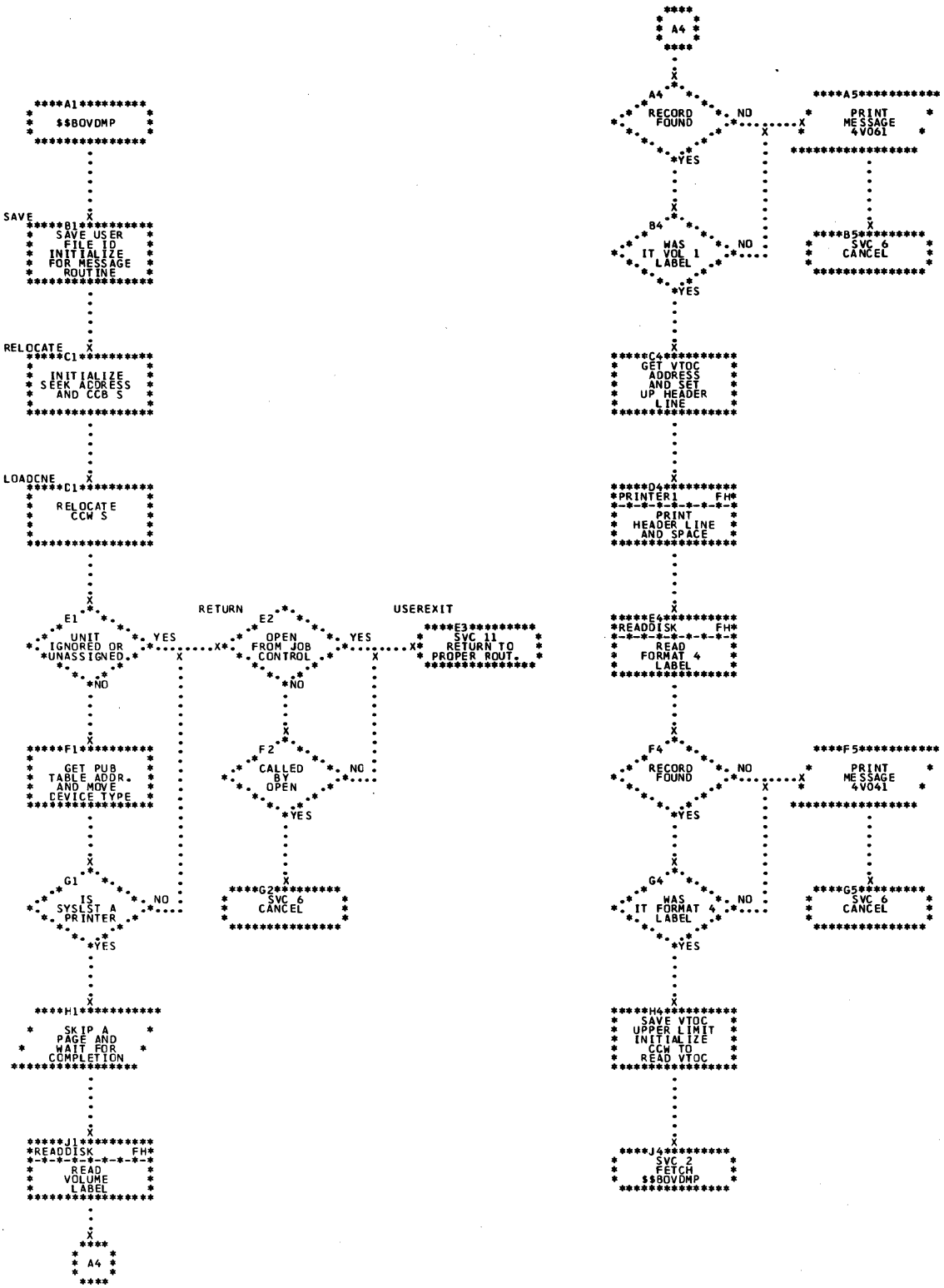


Chart FH. $$BOVDMP: VTOC Dump (Part 2 of 2)

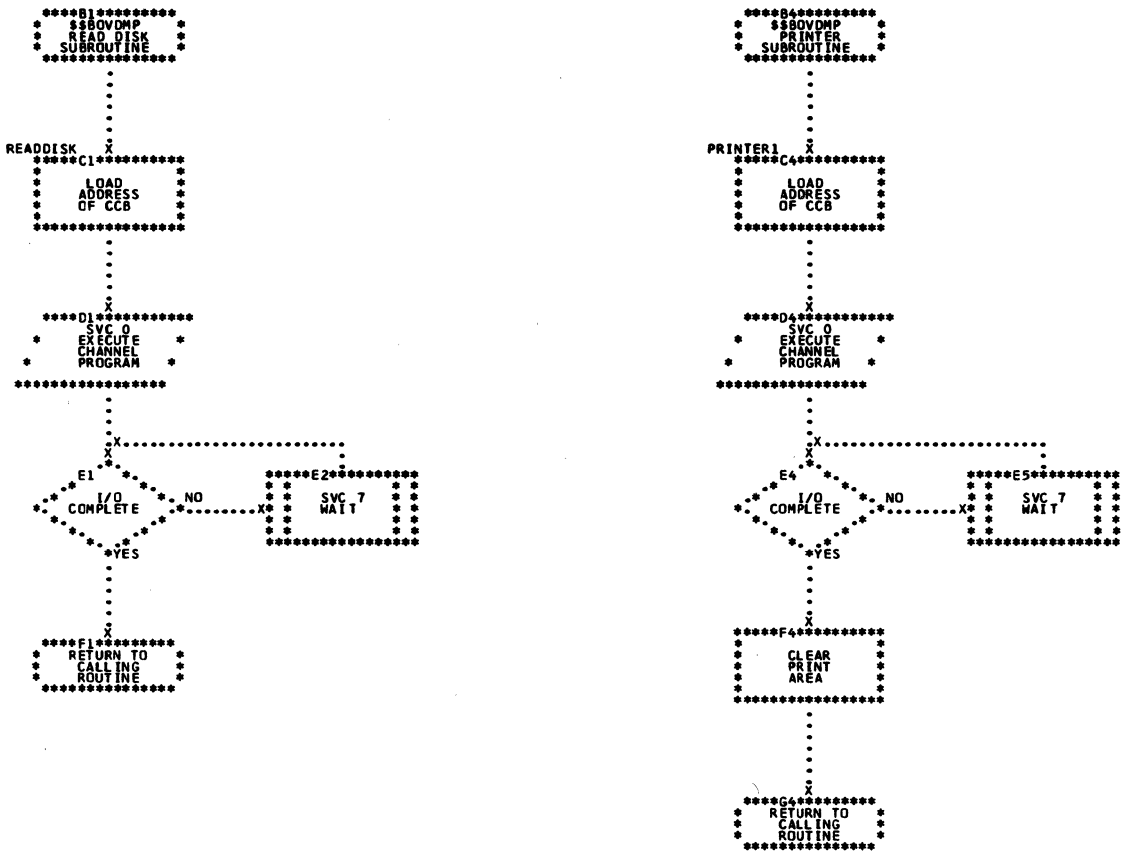


Chart FI. $$BOWDMP: List VTOC

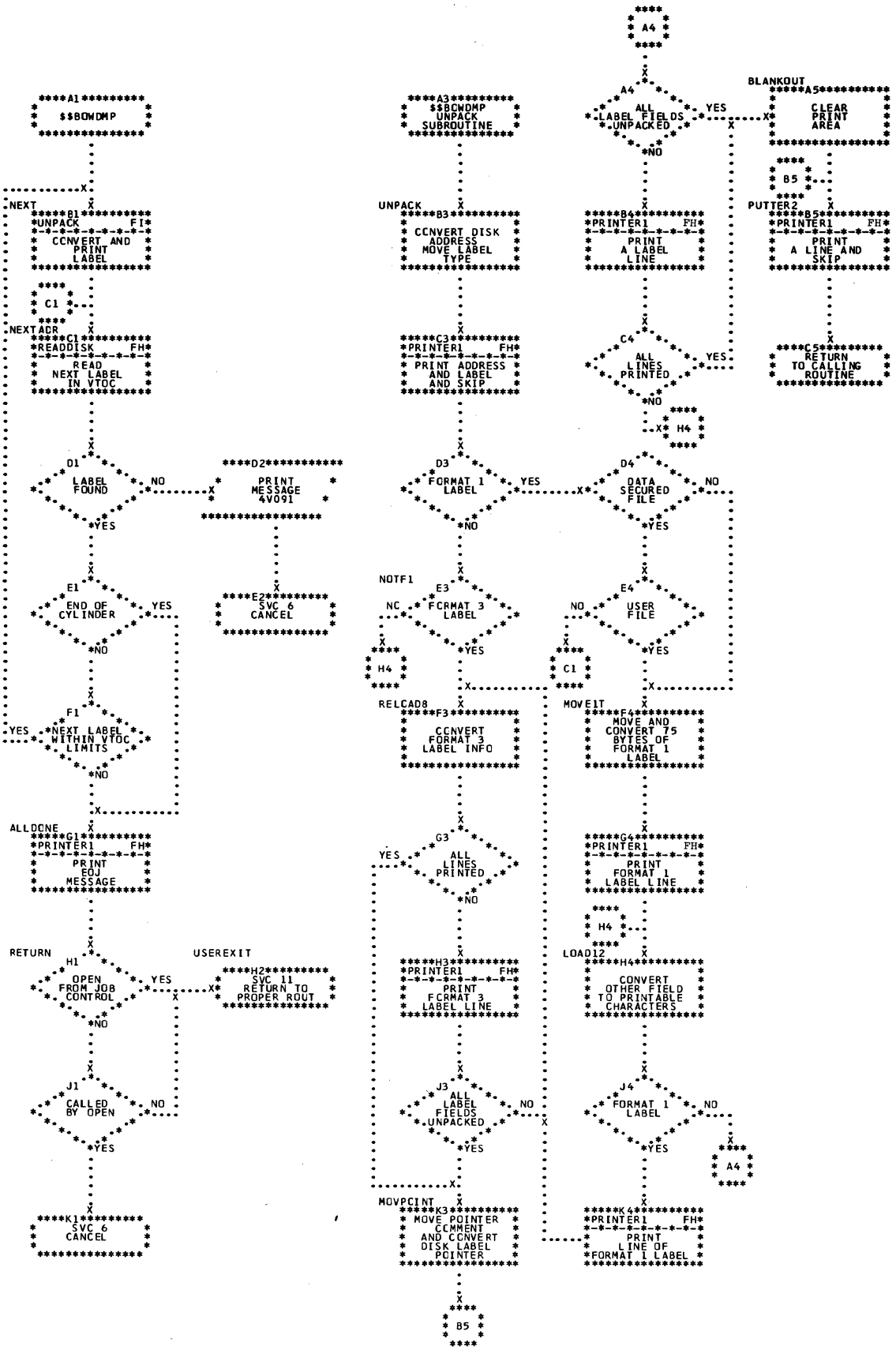


Chart FJ. $$BOMSG1: Disk Open Error Message Writer, Phase 1

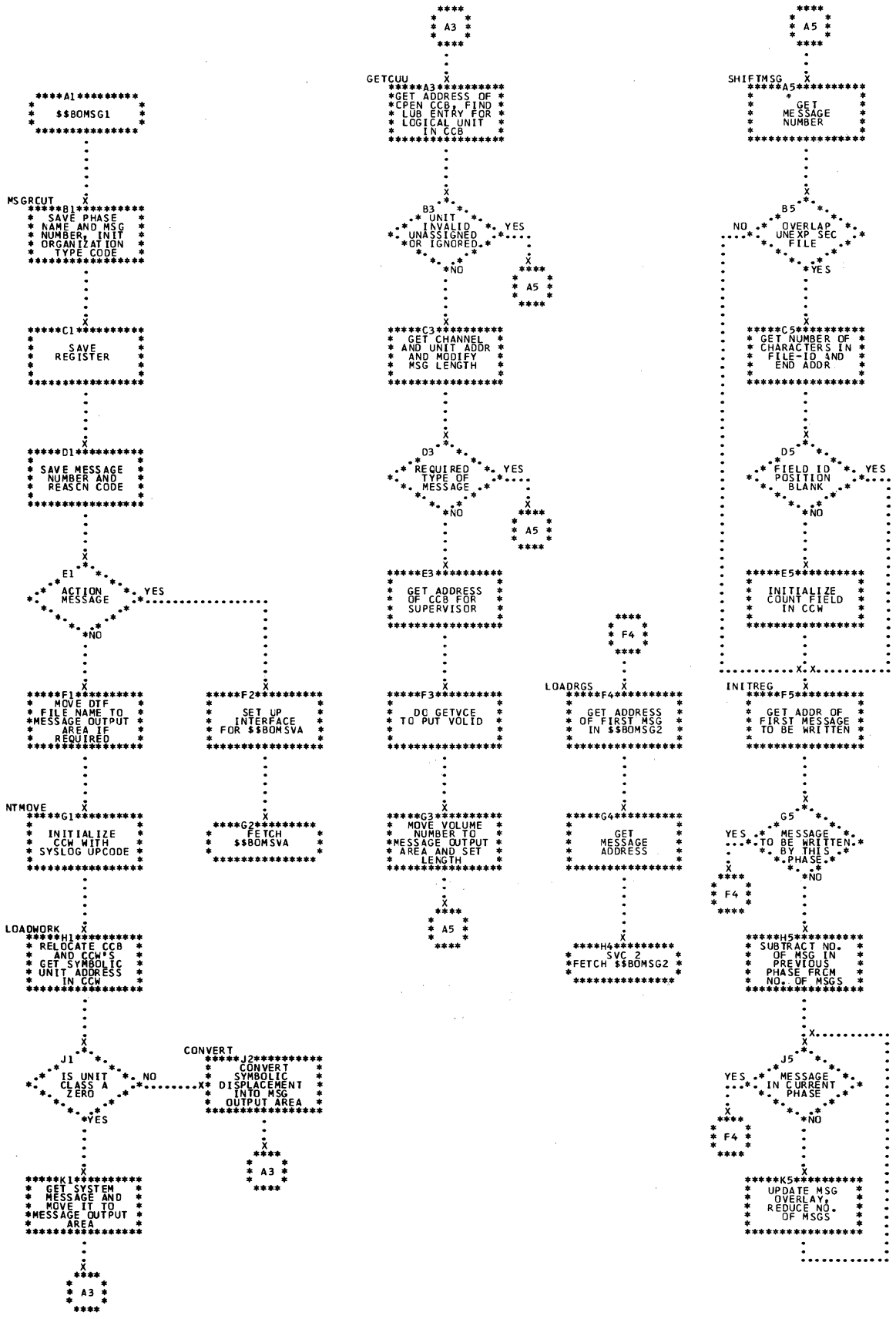


Chart FK. $$BOMSG2: Disk Open Error Message Writer, Phase 2 (Part 1 of 2)

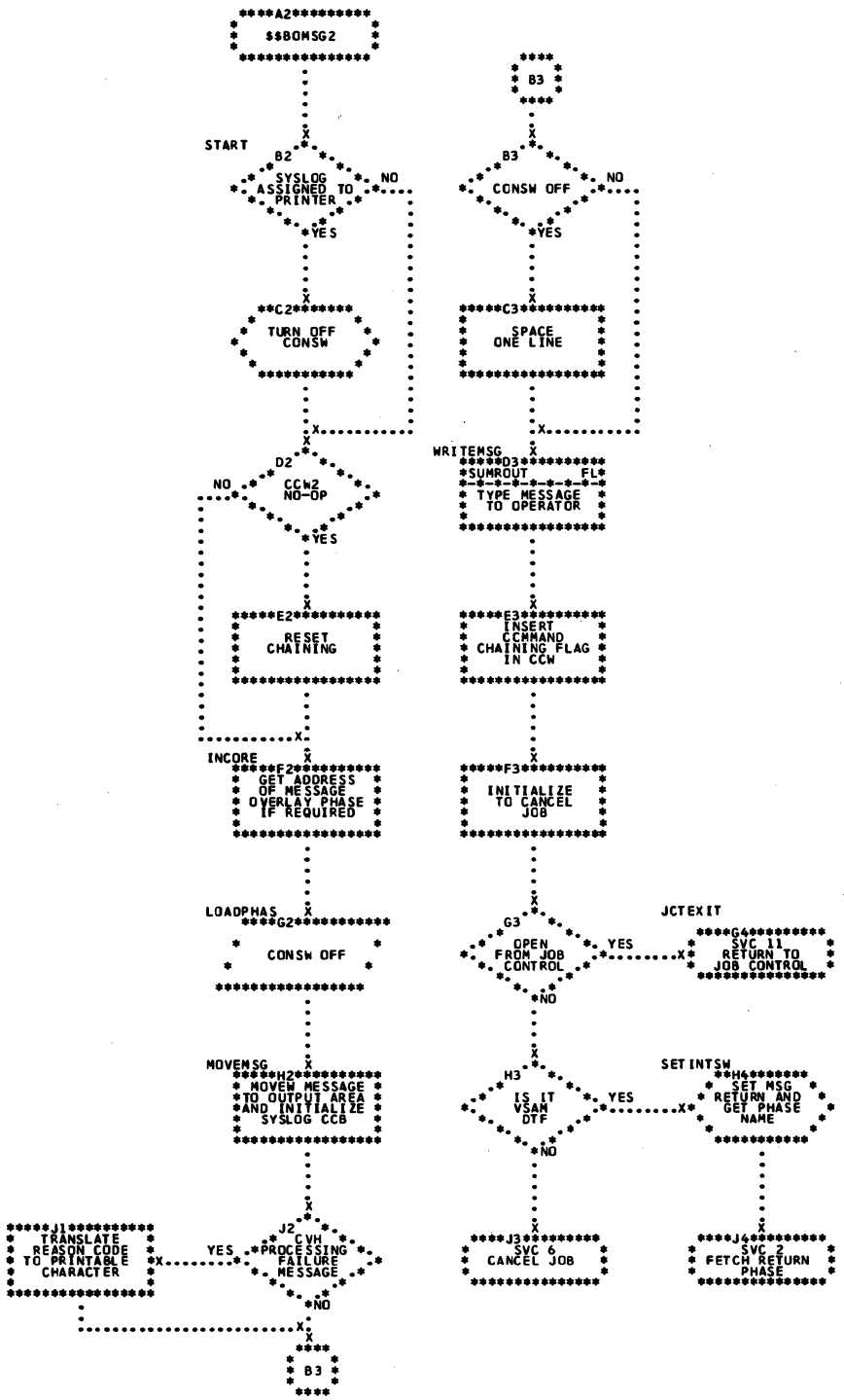


Chart FL. $$BOMSG2: Disk Open Error Message Writer, Phase 2 (Part 2 of 2)

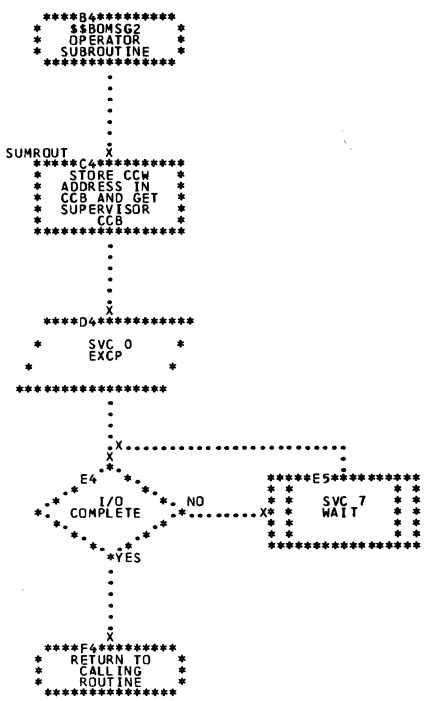


Chart FM. $$BODSMW: Data Security Message Writer

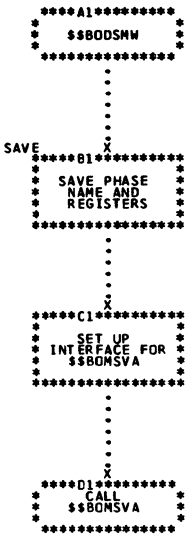


Chart FN. $$BOESTV: Error Statistics by Tape Volume (Part 1 of 2)

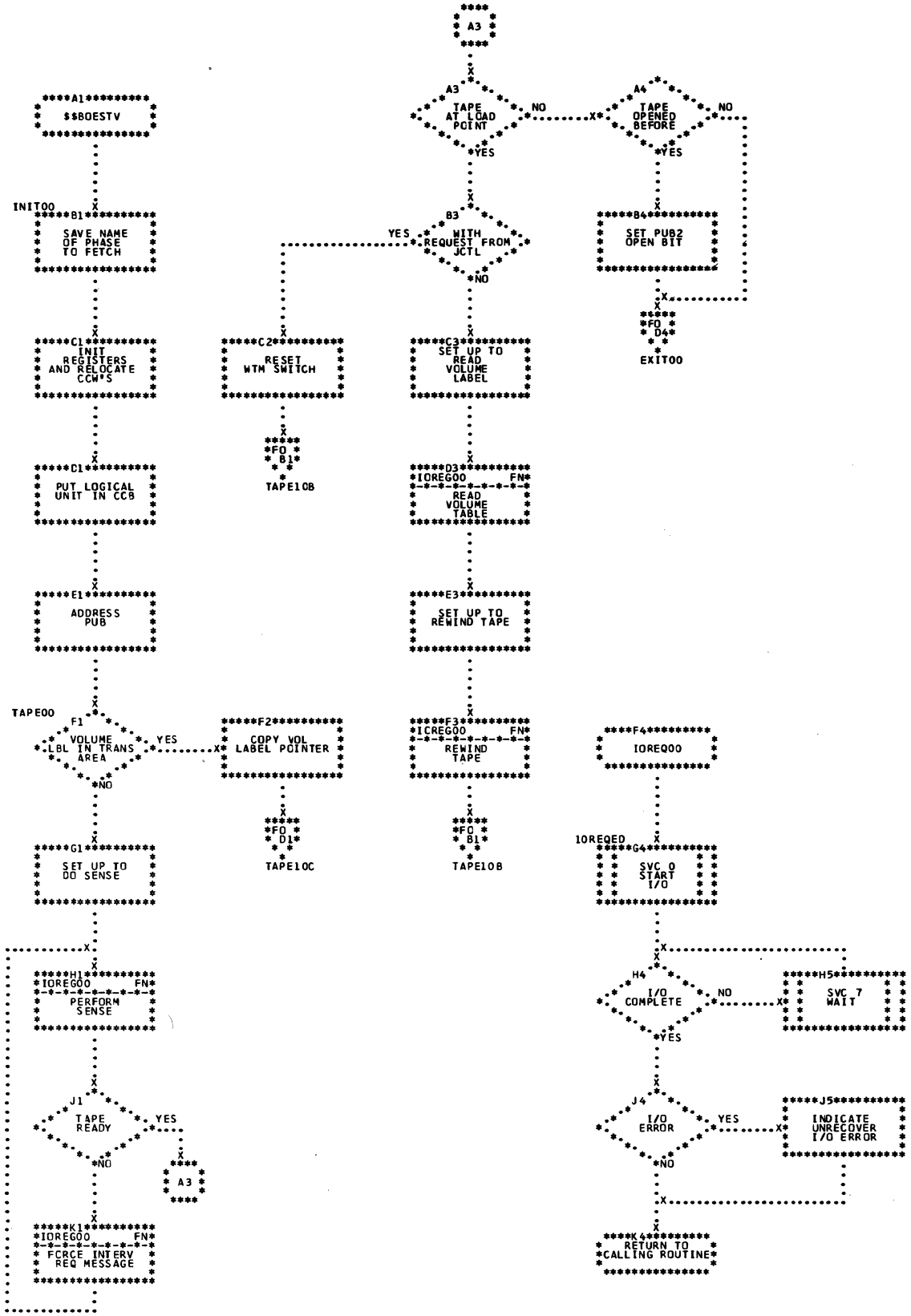


Chart FO. $$BOESTV: Error Statistics by Tape Volume (Part 2 of 2)

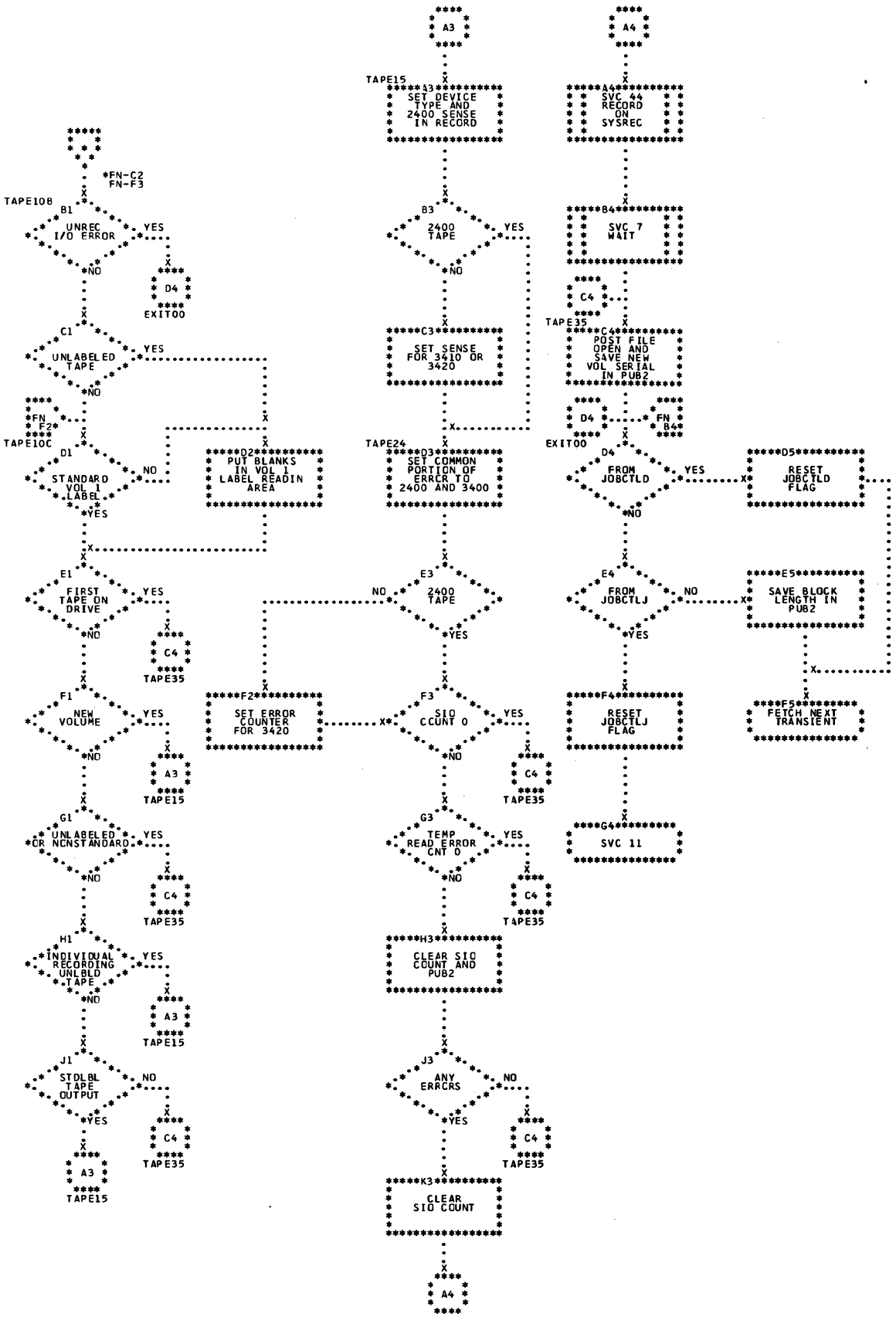


Chart GA. $$BODMSG: Diskette Open Message Writer, Phase 1

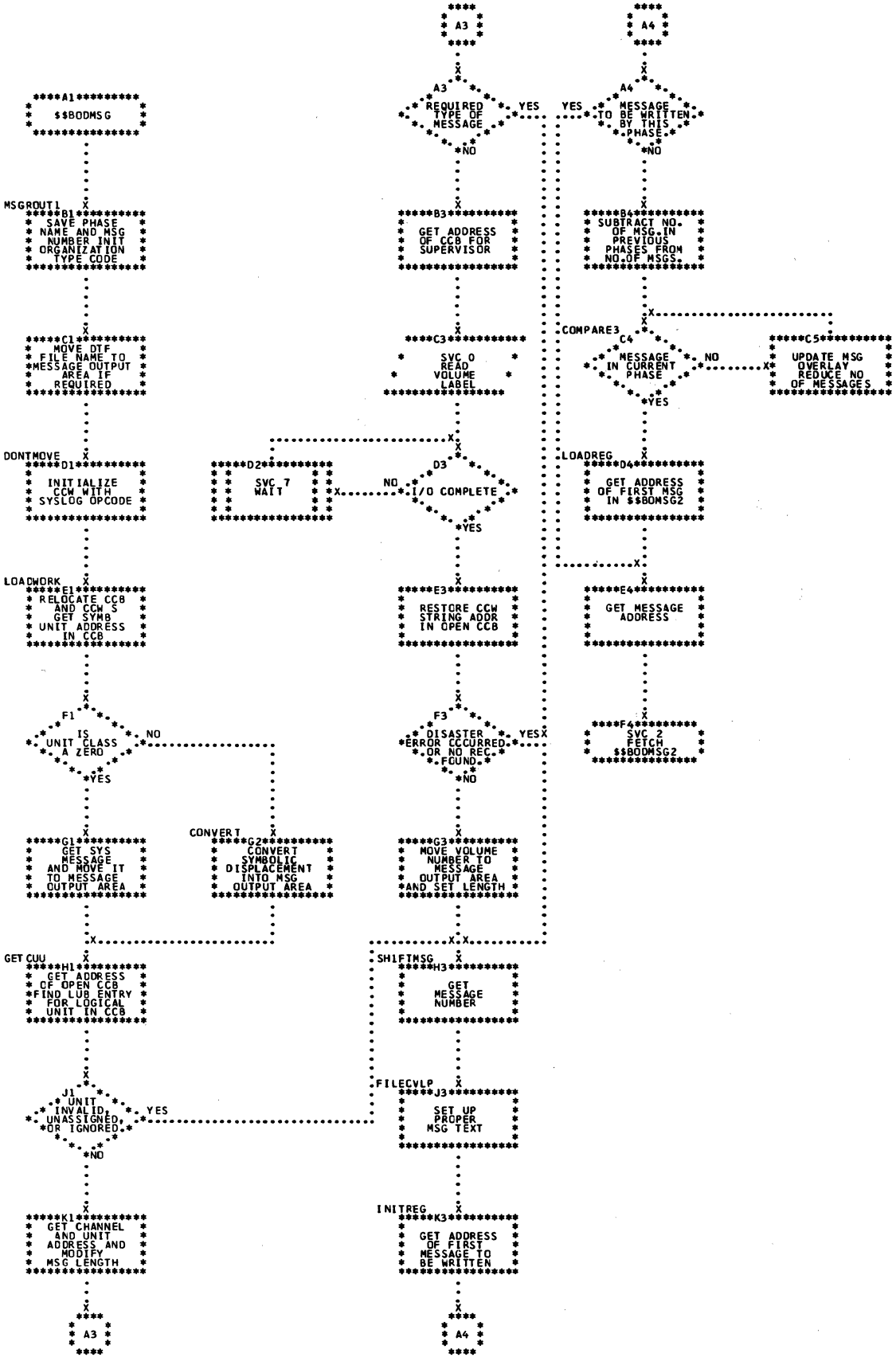


Chart GB. $$BODMS2: Diskette Open Error Message Writer, Phase 2 (Part 1 of 2)

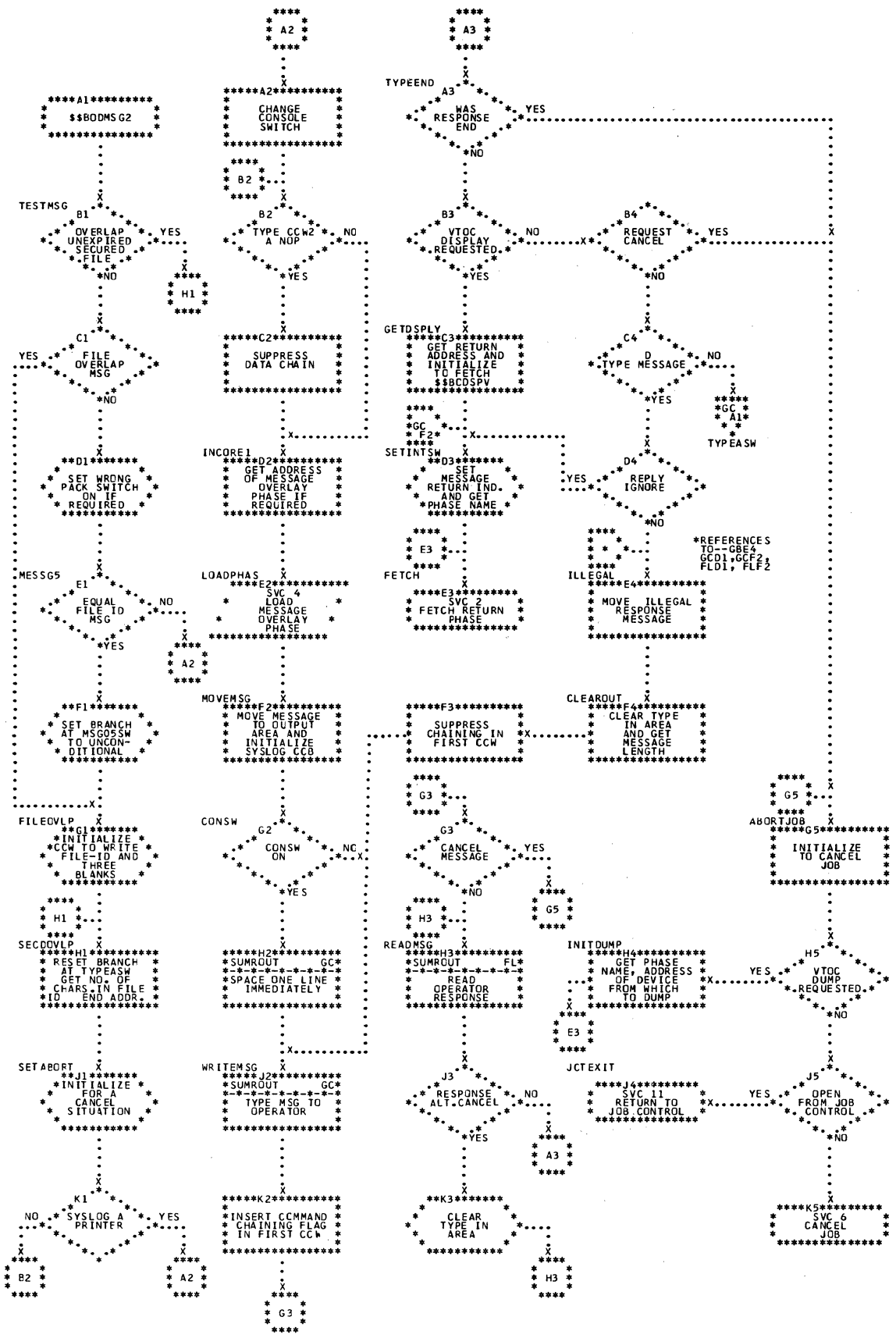


Chart GC. $$BODMS2: Diskette Open Error Message Writer, Phase 2 (Part 2 of 2)

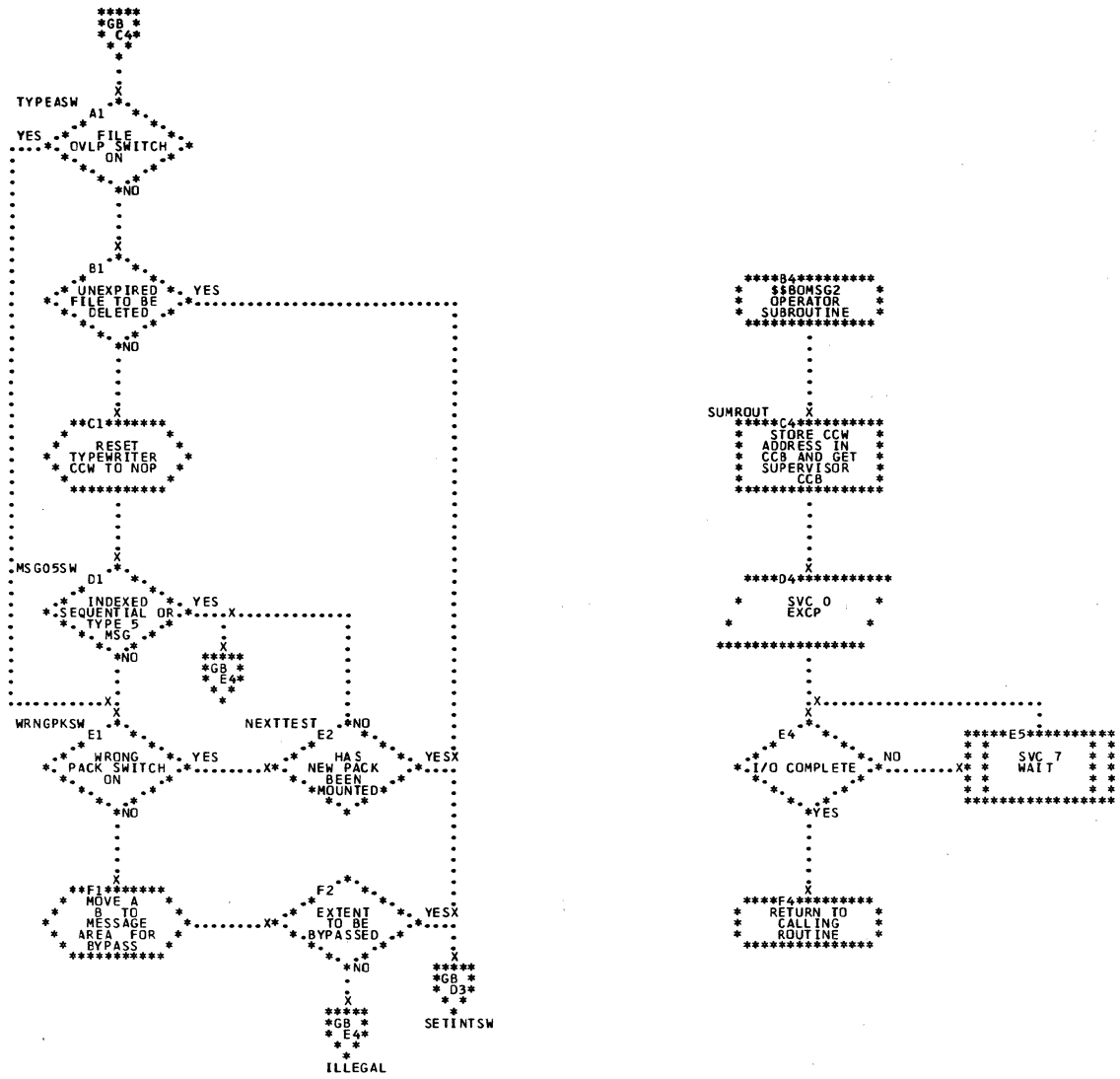


Chart GD. $$BODSMO: Diskette Data Security Message Writer

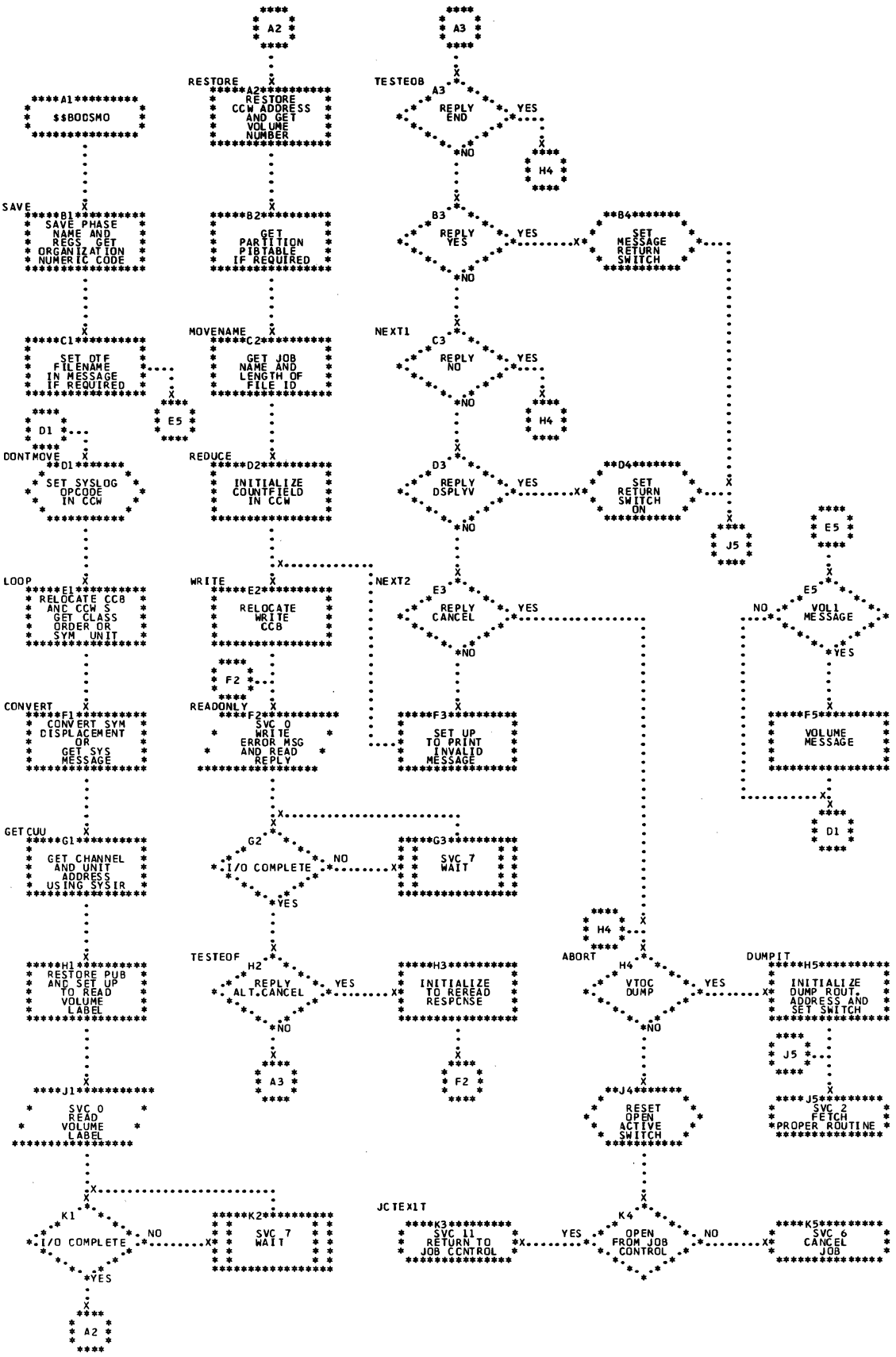


Chart HG. $$BOPEN4: 3340 DTF Device Type Update Open (Part 1 of 2)

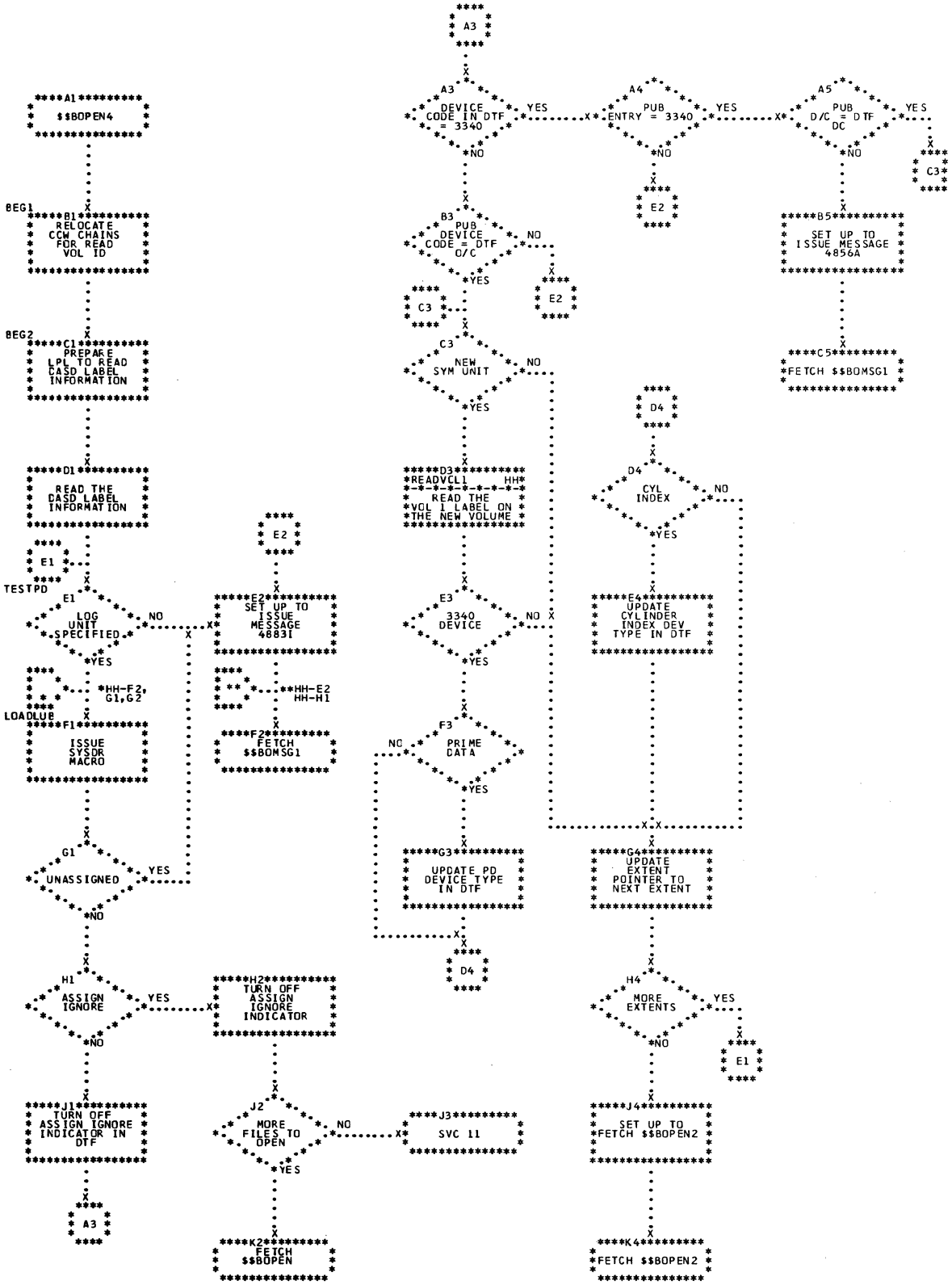


Chart HH. $$BOPEN4: 3340 DTF Device Type Update Open (Part 2 of 2)

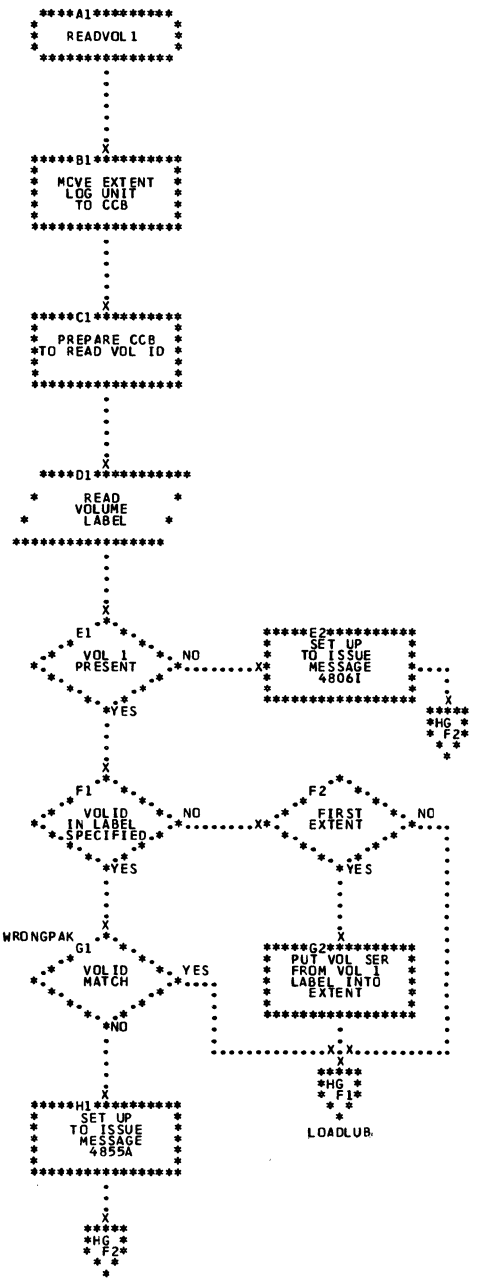


Chart JA. $$BODSPO: Diskette VTOC Display

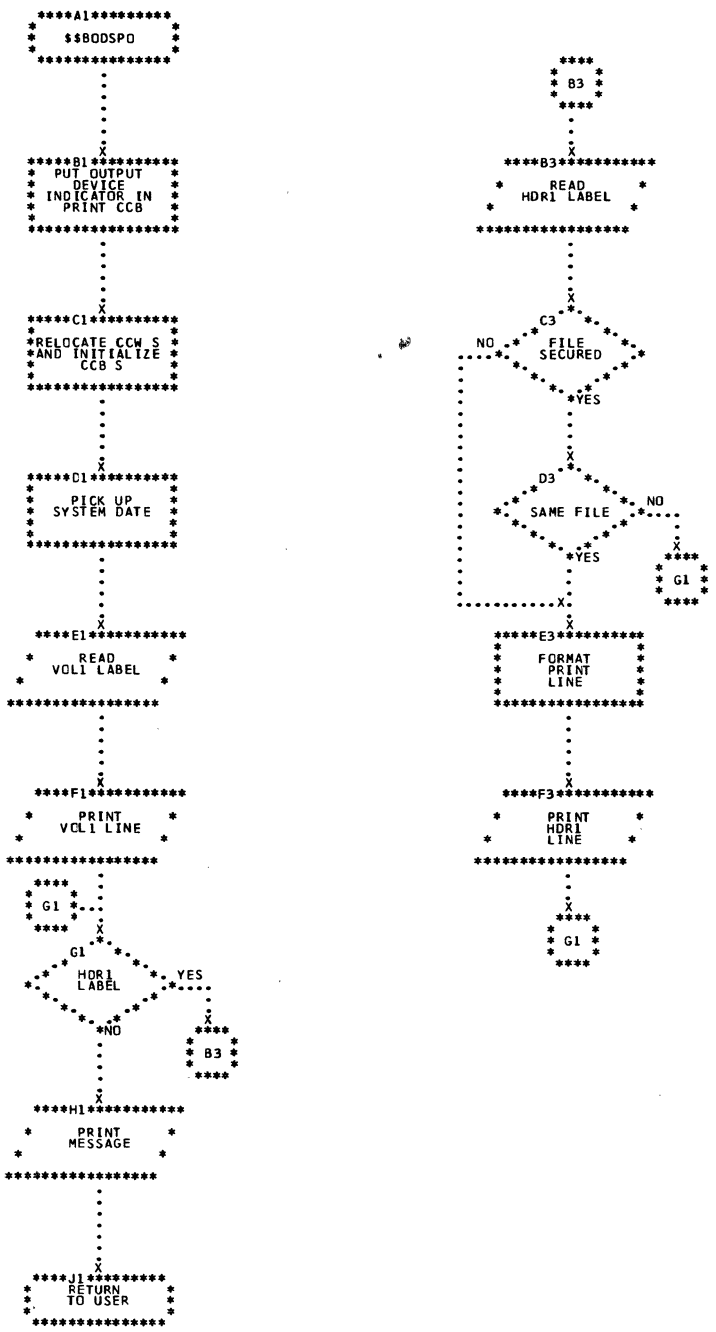


Chart JB. $$BOVDMO: Diskette VTOC Dump

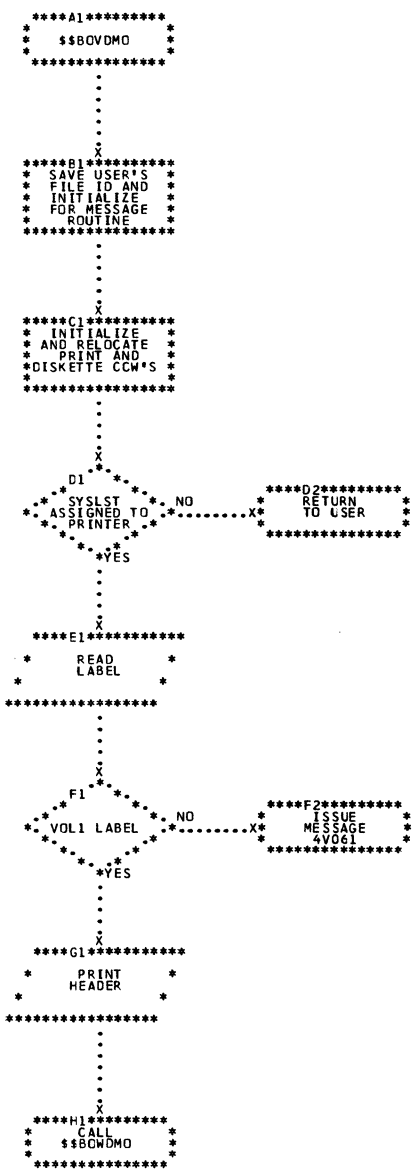
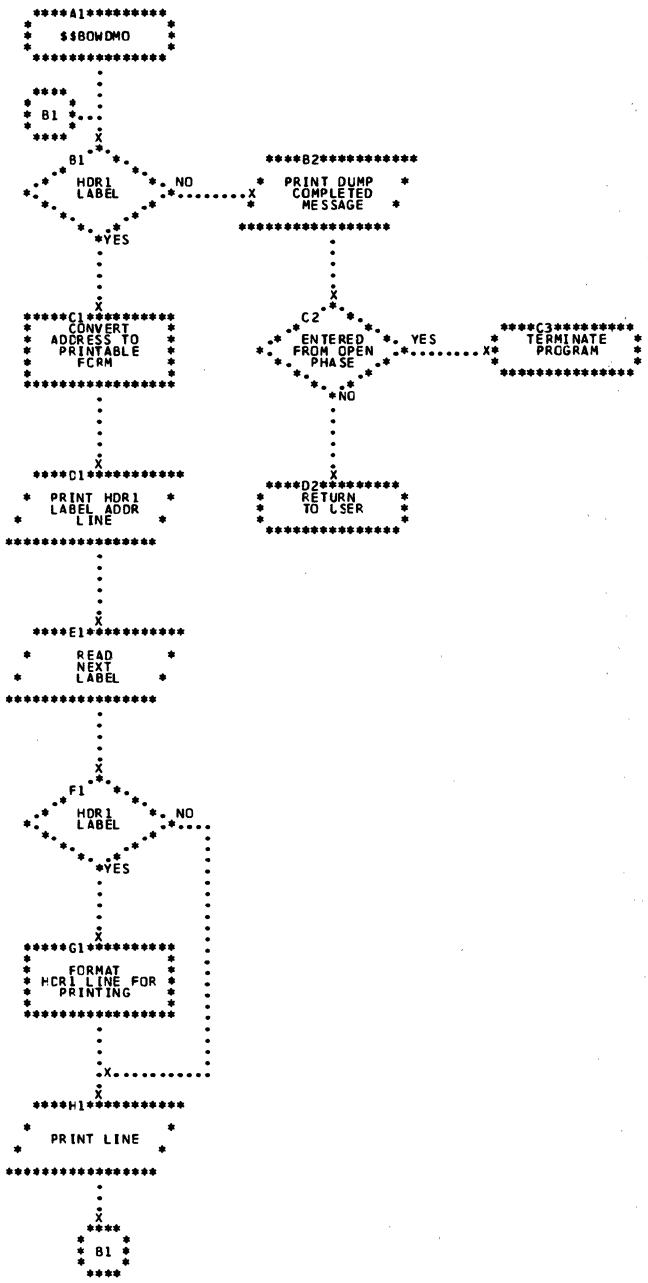


Chart JC. $$BOWDMO: Diskette List VTOC



APPENDIX A: LABEL CROSS-REFERENCE LIST

IOREQ00	FO
ISDISK	BO
ISNEXT	AT
ISOP	AP
ISRESTRD	EF
JCTEXIT	FK
JCTEXIT	GB
JCTEXIT	GD
KSET3800	BK
LABREAD	BI
LAST4NE	FF
LDDTF	AF
LDDTF	BG
LDDTF	BM
LOADONE	FE
LOADONE	FD
LOADONE	FG
LOADPHAS	FK
LOADPHAS	GB
LOADREG	GA
LOADRGS	FJ
LOADWORK	FJ
LOADWORK	GA
LOAD12	FI
LOOP	AQ
LOOP	GD
LTIOREG	AM
MAGTAPE	AU
MAGTAPEX	AU
MAGTAPE1	AU
MAGTAPE2	AU
MAGTAPE3	AU
MAGTAPE4	AU
MESSG5	GB
MICRCHCK	AT
MODCCWLG	FE
MODEXIT	AV
MODEXIT	BC
MODEXIT	BF
MODLOOP	BD
MODLOOP1	BD
MODLOOP2	BD
MODRESP	GB
MORE	AF
MORE	BM
MORE	EG
MORXTNTS	FF
MOVE	FF
MOVEMSG	FK
MOVEMSG	GB
MOVEMSG1	FD
MOVENAME	GD
MOVERCM	FE
MOVE1T	FI
MOVLOLIM	CA
MOVPOINT	FI
MSGROUT	FJ
MSGROUT1	GA
MSG05SW	GC
MUSYMUN	AQ
NCLOS	BG
NCLOS	BM
NEXT	AS
NEXT	BA
NEXT	BE
NEXT	FE
NEXT	FI
NEXTADR	FI
NEXTJIB	EE
NEXTONE	EE
NEXTPHAS	FE
NEXTTEST	GC
NEXT1	EG
NEXT1	GD
NEXT2	GD
NMFETCH	CB
NOALT	EE
NODUTYPE	AH
NOHOLD	CC
NOP	AE
NOPEN	AF
NORESTR	EF
NOSECUR	AF
NOTAPE	AH
NOTCON	BE
NOTFOUND	AV
NOTFOUND	BF
NOTF1	FI
NOUTMC	AL
NOVALDT	AM
NVALTP	AH
NVALTP	AM
OCRTEST	BE
OPENCNAM	AW
OPEN3340	AH
OTHER	BM
OTHEROWN	EG
OTPT3540	BA
OUTAPE	AI
OUTPTFL	CA
OWNERCK	EF
PAST3800	BK
PHTEST	CB
PRELIOCS	CA
PREP	FD
PRINT	BE
PRINTER1	FE
PRINTER1	FD
PRINTER1	FH
PRINTM	BF
PRT/PCH	BL
PTAPE	BA
PTAPE1	BA
PTAPE2	BA
PTAPE3	BA
PTAPE4	BA
PTAPRDR	BA
PUNCH	BL
PUNCHM	BE
PUNCHM	BF
PUTTER2	FI
RDINIT	AE
READ	BL
READDISK	FE
READDISK	FH
READF3	FF
READM	BF
READMSG	FD
READMSG	GB
READONLY	GD
REDUCE	GD
REENTRY	AF
REENTRY	BG
REENTRY	BM
REGSVE	AE
RELADR	BC
RELATIVE	AQ
RELOAD8	FF
RELOAD8	FI
RELOC	BG
RELOC	BO
RELOCATE	FG
REREAD	FD
RESET9	CA

APPENDIX B: MASTER ERROR MESSAGE LIST

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

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

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

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

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

1. LIOCS Volume 1: General Information and Imperative Macros,
2. LIOCS Volume 2: SAM,
3. LICCS Volume 3: DAM and ISAM,
4. LIOCS Volume 4: SAM for DASD

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

Message Number	Phase	Chart	Volume	Message
4110A	$$BOMT03	ED	2	NO VOL1 LBL FOUND TLBL=xxxxxx filename SYSxxx=cuu
	$$BCCPT3	RJ	2	
4111I	$$BOMT01	EA	2	NO VOL1 LBL FOUND filename SYSxxx=cuu
	$$BCCPT4	RL	2	
4112A	$$BOMT01	EA	2	VOL SERIAL NO. ERROR TLBL=xxxxxx filename SYSxxx=cuu
	$$BOMT03	ED	2	
	$$BOCPT3	RH	2	
	$$BOCPT4	RL	2	
4113D	$$BOMT01	EA	2	NO HDR1 LBL FOUND filename SYSxxx=cuu
4113I	$$BOCPT4	RM	2	
4114A	$$BOMT01	EB	2	FILE SEQ NO. ERROR filename SYSxxx=cuu
	$$BOCPT4	RL	2	
4115A	$$BOMT01	EA	2	FILE SER. NO. ERROR TLBL=xxxxxx filename SYSxxx=cuu
	$$BCCPT4	RL	2	
4116A	$$BOMT01	EA	2	VOLUME SEQ. NO. ERROR filename SYSxxx=cuu
	$$BOCPT4	RL	2	
4117D	$$BOMT02	EC	2	NO TM FOUND ON READBK filename SYSxxx=cuu
	$$BOMT05	EG	2	
4118D	$$BOMT02	EC	2	FILE ID ERROR, READBK filename SYSxxx=cuu
4118I				
4119A	$$BOMT03	EE	2	FILE UNEXPIRED filename SYSxxx=cuu
	$$BOMT06	EH	2	
	$$BOCPT3	RH	2	
4120I	$$BOMT03	EE	2	TAPE POSITIONED WRONG filename SYSxxx=cuu
4121A	$$BCMT07	FH	2	NO ALTERN DRIVE ASSGN SYSxxx=cuu
4122I	$$BCMT07	FH	2	EOV ENCOUNTERED SYSxxx=cuu
4123D	$$BOMT02	EC	2	WRONG POSITN, READBK filename SYSxxx=cuu
4123I				
4124I	$$BOMT04	EF	2	TOO MANY UHL'S filename SYSxxx=cuu
4125D	$$BOMT05	EG	2	VOL1 LBL FOUND filename SYSxxx=cuu
4125I				
4126I	$$BCMT02	FB	2	EOV ENCOUNTERED filename SYSxxx=cuu
4127A	$$BCMT05	FE	2	EOV WHILE WRITING EOF
4130A	$$BCMT01	FA	2	EOF OR EOV INQUIRY filename SYSxxx=cuu
4131D	$$BCMT01	FA	2	BLOCK COUNT ERROR filename SYSxxx=cuu DTF=xxxxxx
	$$BCMT03	FC	2	LBL=xxxxxx
4132D	$$BOMT01	EA	2	ERROR IN FILE ID filename SYSxxx=cuu
	$$BOCPT4	RL	2	
4133D	$$BOMT01	EB	2	ERROR IN HDR LBL filename SYSxxx=cuu
	$$BOCPT4	RL	2	

Figure 17. Master Error Message List (Part 1 of 8)

Message Number	Phase	Chart	Volume	Message
4140A	$$BCMT02	FB	2	NO ALTERN DRIVE ASSGN filename SYSxxx=cuu
4151I	$$BOMT01	EA	2	HDR1 LBL INFORMATION filename SYSxxx=cuu
	$$BOMT04	EF	2	
4170A	$$BJCOPT	EK	2	FILE PROTECTED TAPE filename SYSxxx=cuu
4171A	$$BJCOP1	EL	2	UNEXPIRED FILE SYSxxx=cuu
4172A	$$BJCOP1	EL	2	INVALID LABEL SET SYSxxx=cuu
4183I	$$BOMT01	EA	2	INVALID LOGICAL UNIT filename SYSxxx=cuu
	$$BOMT02	EC	2	
	$$BOMT03	ED	2	
	$$BOMT06	EH	2	
	$$BJCOPT	EK	2	
	$$BCMT01	FA	2	
	$$BCMT05	FE	2	
	$$BCMT07	FH	2	
4184D	$$BOCPT2	RE	2	NEED FILE PROTECT RNG filename SYSxxx=cuu
	$$BOCPT2	RF	2	
	$$BOCPT3	RJ	2	
4185I	$$BOMRCE	AC	2	INVALID FORMAT RECORD
	$$BOMRCE	AD	2	
4000I	CDMOD	AL	2	RETRY
	$$BCLOSP	SC	2	
	$$BCLOSP	SD	2	
	$$BERRTN	PG	2	
4n00I	IJJGSDVH	3-93	4	NO LABEL SPACE IN VTOC
4400I	$$BODIO3	WG	2	
4n01I	IJJGSDVH	3-93	4	NO FORMAT 1 LABEL or NO RECORD FOUND
4201I	$$BOIS02	LC	3	
	$$BOISOA	NA	3	
4301I	$$BOSIO5	VF	2	
4202I	$$BCISCA	NA	3	NO RECORD FOUND
4n03I	IJJGDAI1	3-66	4	NO FORMAT 3 LABEL FOUND
	IJJGDAI2	3-67		
	IJJGSDI3	3-42		
	IJJGSDI4	3-43		
	IJJGSDW3	3-48		
4n04I	IJJGSDVH	3-93	4	NO FORMAT 4 LBL IN VTOC
4204I	$$BOIS02	LC	3	NO FORMAT 4 LBL IN VTOC or NO RECORD FOUND
4n05I	$$BOPEN2	AQ	1	UNRECOVERABLE I/O ERROR
	$$BOPLBL	AK	1	
	$$BCLLBL	BJ	1	
	$$BOSDW1	MB	2	
	$$BCCPT1	SB	2	
	IJJGDARL	3-97	4	

Figure 17. Master Error Message List (Part 2 of 8)

Message Number	Phase	Chart	Volume	Message
4n06I	IJJGSDGC	3-97	4	NO STANDARD VOL 1 LABEL or NO RECORD FOUND
4206I	$$BOIS02	LC	3	
	$$BCISOA	NA	3	
4306I	$$BODIO1	VD	2	
4506I	$$BODIC4	WT	2	
4806I	$$BOPEN4	HH	1	
4n07I	IJJGSDRL	3-96	4	NO RECORD FOUND
4307I	$$B35400	VC	2	
4407I	$$B3540I	WC	2	
4n08D	$$BOKUL1	3-11	4	NO UTLO FILE MARK FOUND or NO RECORD FOUND
	$$BIKUL1	3-5		
	$$BOULI1	3-5.1		
	$$BOULO1	3-11.1		
4608D	$$BODACL	CQ	3	
4n20I	IJJGSDSF	3-23	4	PROC. DATA NOT ACCESSIBLE
4n31D	IJJGSDI2	3-41	4	VOLUME SEQUENCE ERROR
4332I	$$BODIO5	VH	2	
4n33A	IJJGDAO3	3-62	4	EQUAL FILE IN VTOC
	IJJGSDC4	3-30		
4433A	$$BODIC2	WD	2	
4n34I	IJJGSDO6	3-32	4	CURRENT FILE LBL DELETED
	IJJGSDW3	3-48		
4n36I	IJJGSDW3	3-48	4	NO MORE AVAIL/MATCH EXTENT
4337I	$$B3540I	VA	2	CHAINING TO SYSTEM UNIT
	$$BODIC6	VM		
4437I	$$B35400	WA	2	
	$$BODIC7	WL		
4n38D	$$BIKUL1	3-5	4	USER HDR LBL IS NOT STD
	$$BOULI1	3-5.1		
4639D	$$BODACL	CQ	3	USER TRL LBL IS NOT STD
4n40A	IJJGSDC4	3-30	4	EXTENT OVERLAY ON ANOTHER
	IJJGDAO3	3-62		
4240I	$$BOIS02	LC	3	EXTENT OVERLAPS ANOTHER
4n41A	IJJGDAC3	3-62	4	EXTENT OVERLAP ON VTOC
	IJJGSDO4	3-30		
4241I	$$BOIS02	LC	3	
4n42A	IJJGSDI4	3-43	4	NO MATCHING EXTENT
4243I	$$BORTV1	NF	3	INV EXTENT HI/LO LIMITS

Figure 17. Master Error Message List (Part 3 of 8)

Message Number	Phase	Chart	Volume	Message
4n44A	IJJGDAC3	3-62	4	OVERLAP ON UNEXPIRED FILE
	IJJGSDO4	3-30		
4n45I	IJJGDACX	3-58	4	TOO MANY EXTENTS
	IJJGSDSF	3-23		
	IJJGSDXT	3-34		
4245I	$$BOIS06	LI	3	
4445I	$$BODIO8	WQ	2	
4246I	$$BOIS07	MA	3	DISCONT INDEX EXTENTS
4n47A	IJJGSDW1	3-46	4	EXTENTS NOT ON SAME UNIT
4n48I	IJJGSDSF	3-23	4	SYSIN/SYSOUT UNSUPPORTED
4348I	$$B35400	UE	2	
4249I	$$BOIS05	LG	3	DATA TRACK LIMIT INVALID
4n50A	IJJGSDXT	3-34	4	NO MORE AVAILABLE EXTENTS
4450A	$$BODIO8	WP	2	
4n51I	IJJGDACX	3-58	4	SYSUNITS NOT IN SEQUENCE
4252I	$$BOIS05	LE	3	DISCONT TYPE 1 EXTENTS
4n54I	IJJGDART	3-68	4	DSKXTN ENTRY TABLE FULL
4254I	$$BOIS05	LG	3	
	$$BORTV2	NG		
4n55A	IJJGDAVC	3-59	4	WRONG PACK, MOUNT nnnnnn
	IJJGSDIP	3-98		
4355A	$$BOSDI1	LF	2	
4855A	$$BOPEN4	HH	1	
4856A	$$BOPEN4	HG	1	WRONG MODULE SIZE
4n58I	IJJGDAO1	3-60	4	NO EXTENT FOR OUTPUT FILE
	IJJGSDRL	3-95		
4358I	$$B35400	UF	2	
4n59A	IJJGSDO2	3-29	4	INVALID EXTENT
	IJJGSDI4	3-43		
	IJJGDACX	3-58		
	IJJGSDO5	3-31		
	IJJGSDLP	3-98		
4n59I	IJJGSDRL	3-95	4	
4359I	$$BODIO5	VK	2	
	$$BODIO6	VL		
4459I	$$BODIO3	WG	2	
4859I	$$BOPEN2	AQ	1	

Figure 17. Master Error Message List (Part 4 of 8)

Message Number	Phase	Chart	Volume	Message
4n60I	IJJGSDC1	3-28	4	NO EXTENTS, ALL BYPASSED
	IJJGSDI1	3-39		
	IJJGSDSF	3-23		
	IJJGSDW1	3-46		
	IJJGDAO1	3-60		
	IJJGDAC2	3-61		
	IJJGDAO4	3-63		
4360I	$$B35401	VB	2	
4n61I	IJJGSDRL	3-95	4	INVALID DLBL FUNCTION
	IJJGDARL	3-57		
4261I	$$BOIS01	LA	3	
	$$BORTV1	ND		
4361I	$$B35400	UF	2	
4861I	$$BOPEN2	AP	1	
4262I	$$BOIS05	LG	3	NO PRIME DATA EXTENT
	$$BORTV1	ND		
4263I	$$BOIS07	MB	3	LOAD FILE NOT CLOSED
4364I	$$BODIC5	VF	2	INVALID HDR1 LABEL
	$$BODIO6	VM		
4n66A	IJJGSDI4	3-43	4	1 TRACK USER LBL EXTENT
	IJJGSDC5	3-31		
	IJJGDAO2	3-61		
4266I	$$BOIS05	LF	3	
4n67I	IJJGSDO4	3-30	4	CVH PROCESSING FAILURE
	IJJGSDVH	3-93		
	IJJGVD00	3-77		
	IJJGVD10	3-85		
	IJJGDAC3	3-62		
4n68A	IJJGSDI4	3-43	4	USER LBLS EXHAUST FIRST EXTENT
	IJJGSDC5	3-31		
4n68D	IJJGDAC2	3-61	4	
4269I	$$BOIS07	MC	3	FILE IS OPEN FOR ADD
4270I	$$BORTV2	NG	3	1ST XTNT CD NOT INDX VOL
4271I	$$BOIS01	LA	3	EXTENT INFO NEEDED
4272I	$$BOIS08	MD	3	MOD AND DTF INCOMPATIBLE
4n73D	IJJGDAMX	3-52	4	LMOD NOT CURRENT LVL

Figure 17. Master Error Message List (Part 5 of 8)

Message Number	Phase	Chart	Volume	Message
4n74I	IJJGSDBS	3-37	4	BLKSIZE OPEN FAILURE
	IJJGSDW1	3-46		
4n75I	IJJGSDBS	3-37	4	BLKSIZE NOT MULT OF RECSIZE
	IJJGSDI2	3-41		
4n76D	IJJGSDLP	3-98	4	VOL SER NOT XXXXXX
	IJJGDAVC	3-59		
4n77A	IJJGSDXT	3-34	4	EXTENT ENTRY ERROR - RETRY
4n79I	$$BCFLPT	FA	1	GETVIS FAILED
	$$BOPLBL	AK		
	$$BCLLBL	BJ		
	$$BOPEN2	AQ		
	$$BCEOV1	EM	2	
	$$BCCPT1	SB		
	$$BOSVLT	3-4	4	
	$$BOSFBL	3-3		
	IJJGMFBA	3-18		
	IJJGSDO4	3-30		
	IJJGSDVH	3-93		
	IJJGSDUL	3-100		
	IJJGDARL	3-57		
	IJJGDAI2	3-67		
	IJJGVD00	3-77		
	IJJGVD10	3-85		
	IJJGSDFP	3-102		
	IJJGMMBF	3-21		
	IJJGDAMX	3-52		
4n80I	$$BOSFBL	3-3	4	INVALID FILE TYPE
4880I	$$BOPEN1	AH	1	
	$$BCLOS2	BK		
	$$BCEOV1	EM	2	
4n81I	IJJGSDRL	3-95	4	NO LABEL INFORMATION
	IJJGSDMO	3-22		
	IJJGSDSF	3-23		
	IJJGDARL	3-57		
4881I	CLOSE	BP	1	
	$$BCLLBL	BJ		
	$$BOPEN2	AN		
	$$BOPLBL	AK		
	$$BCCPT1	SA	2	
	$$BCEOV1	EM		
	$$B35400	UF		
4282I	$$BOIS07	MA	3	ISAM NULL FILE

Figure 17. Master Error Message List (Part 6 of 8)

Message Number	Phase	Chart	Volume	Message
4n83I	IJJGSDGC	3-97	4	INVALID LOGICAL UNIT
	IJJGSDNV	3-99		
	IJJGSDRL	3-95		
	IJJGDACX	3-58		
4383I	$$B3540I	VC	2	
4483I	$$B3540O	WC	2	
4883I	$$BOPEN4	HG	1	
	$$BOCP01	QA	2	
	$$BOCP02	QB		
	$$BOCP11	QE		
	$$BOCP12	QF		
	$$BOUR01	AB		
4884D	$$BOPEN1	AJ	1	NEED FILE PROTECT RNG filename SYSxxx=cuu
	$$BOCP02	QB	2	
	$$BOCP11	QE		
	$$BOCP12	QF		
4885I	$$BOPENC	AW	1	SYSxxx AND SYSyyy ARE ASSIGNED TO THE SAME PHYSICAL UNIT
4886D	$$BOPEN1	AJ	1	TAPE UNIT NOT READY
4887I	$$BERRTN	PG	2	SYS FILE EXTENT EXCEEDED
4888I	$$BERRTN	PG	2	EOF ON SYSTEM FILE
4n89I	IJJGSDSF	3-23	4	WORKFILE NOT SUPPORTED FOR SYSFIL
4n90I	IJJGSDVH	3-93	4	NO JIBS AVAILABLE
	IJJGVD00	3-77		
	IJJGSDFP	3-102		
4890I	$$BCFLPT	FB	1	
4n93I	IJJGSDRL	3-96	4	UNRECOVERABLE I/O ERROR
4n94I	IJJGSDCI	3-94	4	CISIZE INCORRECT
	IJJGMIOI	3-24		
4n95I	IJJGSDRL	3-96	4	(PHASENAME) NOT IN SVA
	$$BOSFBL	3-3		
	IJJGMLLM	3-19		
	IJJGDAMX	3-52		
4n96I	IJJGSDSF	3-23	4	IMPROPER DTFSD SYSFIL OPEN
4n97I	IJJGDAC3	3-62	4	OVLAP EXPIRED SECRD FILE
4n98I	IJJGSDC4	3-30	4	OVLAP UNEXPRD SECRD FILE
	IJJGDAO3	3-62		
4n99D	IJJGSDI2	3-41	4	DATA SECURED FILE ACCESSED
	IJJGDAI1	3-66		

Figure 17. Master Error Message List (Part 7 of 8)

Message Number	Phase	Chart	Volume	Message
4MR1I	MRMOD	BB	2	EXTERNAL INTERRUPT I/O ERROR filename SYSxxx
4MR2I	MRMOD	BB	2	SCU NOT OPERATIONAL filename SYSxxx
4P01I	$$BERPTP	PH	2	DATA CHECK SYSxxx=cuu
4P02D	$$BERPTP	PH	2	DATA CHECK SYSxxx=cuu
4V03I	$$BODSPW	FF	1	NO RECORD FOUND filename SYSxxx
4V04I	$$BODSPW $$BOVDMP	FF FG	1	NO RECORD FOUND filename SYSxxx, or NO FORMAT 4 LBL IN VTOC filename SYSxxx
4V06I	$$BOVDMO $$BOVDMP	JB FG	1	NO STANDARD VOLUME LABEL filename SYSxxx
4V09I	$$BODSPW $$BOWDMP	FE FI	1	NO RECORD FOUND filename SYSxxx
4V67I	IJJGVD00 IJJGVD10	3-83 3-90	4	CVH PROCESSING FAILURE
4V90I	IJJGVD10	3-85	4	NO JIBS EXIST
4V95A	$$BODSPV IJJGVD10	FD 3-85	1 4	SYSLOG OR SYSLST
4V96A	$$BODSPV OJJGVD10	FD 3-85	1 4	SYSLST NOT A PRINTER

Figure 17. Master Error Message List (Part 8 of 8)

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

APPENDIX C: ASCII CONVERSION TABLES

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

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

Figure 18. ASCII to EBCDIC Conversion (Part 1 of 2)

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

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

Figure 18. ASCII to EBCDIC Conversion (Part 2 of 2)

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

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

Figure 19. EBCDIC to ASCII Conversion (Part 1 of 2)

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

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

Figure 19. EBCDIC to ASCII Conversion (Part 2 of 2)

APPENDIX D: MASTER INDEY FOR VSE/ADVANCED FUNCTIONS LIOCS

This Master Index contains references to the VSE/Advanced Functions Logical IOCS manuals. The number(s) after each entry is the key to the manual(s) in which the information is found. The keys correspond to the following manuals.

1. VSE/Advanced Functions LIOCS Volume 1: General Information and Imperative Macros , LY24-5209.
2. VSE/Advanced Functions LIOCS Volume 2: SAM , LY24-5210.
3. VSE/Advanced Functions LIOCS Volume 3: DAM and ISAM , LY24-5211.
4. VSE/Advanced Functions LIOCS Volume 4: SAM FOR DASD , LY24-5212.

INDEX

VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 1, LY24-5209, contains a master index to LIOCS Volumes 1, 2, 3, and 4.

VSE/Advanced Functions: LIOCS Volume 1 (File No. S370/4300-30) Printed in U.S.A. LY24-5209-0

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

READER'S COMMENT FORM

VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 1
General Information and Imperative Macros
LY24-5209-0

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

_____

_____

_____

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

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 812 BP
1133 Westchester Avenue
White Plains, New York 10604

Fold and tape | Please Do Not Staple | Fold and tape

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

VSE/Advanced Functions: LIOCS Volume 1 (File No. S370/4300-30) Printed in U.S.A. LY24-5209-0

Cut or Fold Along Line