SY33-8560-1
File No. S370-30

Systems

# DOS/VS LIOCS Volume 2 SAM Logic

**Release 29**

IBM

## THIS MANUAL....

.... is the second in a series of four volumes providing detailed information about the IBM Disk Operating System (DOS/VS) Logical IOCS programs. The four volumes are:

Volume 1: General Information and Imperative Macros, SY33-8559.

Volume 2: SAM, SY33-8560.

Volume 3: DAM and ISAM, SY33-8561.

Volume 4: VSAM, SY33-8562.

This manual is majorly intended for persons involved in program maintenance and for system programmers who are altering the program design. Program logic information is not necessary for the operation of the programs described.

General routines that apply to more than one access method or more than one file type are described in Volume 1. These routines include open/close, checkpoint/restart, and a number of transient routines. References to Volume 1 are made whenever required for a good understanding of the topics discussed.

This volume of the DOS/VS Logical IOCS Manuals consists of six parts:

1. LIOCS support for Unit Record files.
2. LIOCS support for Magnetic Tape files.
3. LIOCS support for sequential DASD files.
4. LIOCS support for Device Independent files.
5. LIOCS support for diskette files.
6. Charts.

Parts 1, 2, 3, 4, and 5 supply descriptions of the declarative and imperative macros, DTF tables, and initialization and termination procedures for each of the file types described. Part 6 supplies the detailed flowcharts associated with the descriptions in the first four parts.

The appendixes in the back of this manual provide maintenance personnel with the service aids:

1. Label list.
2. Messages cross-reference list.

Effective use of this publication requires an understanding of IBM System/370 operation and the Disk Operating System (DOS/VS) Assembler Language and its associated macro definition language. Reference publications for this information are listed below.

### PREREQUISITE PUBLICATIONS

- DOS/VS Data Management Guide, GC33-5372.
- DOS/VS Supervisor and I/O Macros, GC33-5373.
- OS/VS and DOS/VS Assembler Language, GC33-4010.
- DOS/VS LIOCS Volume 1, General Information and Imperative Macros, SY33-8559.

### RELATED PUBLICATIONS

- DOS/VS LIOCS Volume 3, DAM and ISAM, SY33-8561.
- DOS/VS LIOCS Volume 4, VSAM, SY33-8562.
- DOS/VS Messages, GC33-5379.

This volume of the DOS/VS Logical IOCS Logic Manuals contains detailed information on the logical IOCS support of unit record, magnetic tape, sequential access DASD, diskette, and device independent files.

This volume contains no general information apart from a brief introductory description of each of the file types covered. If you want to get an overall view of the concept of logical IOCS, or an idea of the functions performed by the imperative macros, refer to DOS/VS LIOCS Volume 1, General Information and Imperative Macros, SY33-8559. Volume 1 also contains descriptions of the generalized open/close routines, checkpoint/restart routines, and DASD file protect, VTOC dump, VTOC display, and message writer subroutines.

Information on all the logical IOCS items (modules, DTF tables, imperative macros, open and close routines, etc.) required for the particular file types discussed can be found in this manual. The only exceptions are certain common and special purpose routines which cannot be related to any specific file type or which apply to more than one file type; those routines are mentioned above.

The files discussed in this volume are divided into five parts:

- Unit Record files.
- Magnetic Tape files.
- Sequential access DASD files.
- Device independent files.
- Diskette files.

Files within a given group are presented in alphabetic sequence according to the last two letters of the DTFxx macro that defines the file (that is, DTFCD, DTFCN, ... DTFPT). Access to information on a particular file type can be made through the index. The information relating to a file type includes, in the order presented:

- The file definition (DTFxx) macro.
- The module generation (xxMOD) macro.
- The imperative LIOCS macros (GET, READ, etc.) used with the file.
- The special open and close routines, if applicable.
- The special purpose routines, such as message writers, if applicable.

Part 5 contains the generalized and detailed flowcharts of the imperative LIOCS macros supported by each of the data handling logic modules and the logical transients required for open, close, and other special functions.

The logic supporting each of the imperative macros has been flowcharted from macro language (source statement) listings. In some instances these flowcharts contain decision blocks to illustrate the logic included in the module for certain xxMOD macro parameter options. You should realize that these decisions do not appear in an assembly listing, but rather that a particular assembly listing is the result of these decisions being made at the time the logic module is generated.

The IBM Disk Operating System (DOS/VS) provides logical IOCS support for files on the following IBM devices generally categorized as unit-record equipment:

- 125(D) Display Operator Console
- 1017 Paper Tape Reader
- 1018 Paper Tape Punch
- 1255/1259 Magnetic Character Reader
- 1270/1275 Optical Reader/Sorter (These devices are not available in the United States)
- 1287 Optical Readers
- 1403 Printer
- 1419 Magnetic Ink Character Readers
- 1442N1 Card Read Punch
- 1442N2 Card Punch
- 1443 Printers
- 2245 Printer
- 2501 Card Reader
- 2520B1 Card Read Punch
- 2520B2/B3 Card Punch
- 2540R Card Reader
- 2540P Card Punch
- 2560 Multifunction Card Machine
- 2596 Card Read Punch
- 2671 Paper Tape Reader
- 3203 Printer
- 3210 or 3215 Console Printer-Keyboard
- 3211 Printer
- 3504 Card Reader
- 3505 Card Reader
- 3525P Card Punch
- 3525RP Card Punch with read feature
- 3881 Optical Mark Reader
- 3886 Optical Reader
- 5203 Printer
- 5425 Multifunction Card Unit

The files used with these devices are defined by a DTFxx declarative macro and the data handling LIOCS module is generated, except for console files, by an associated xxMOD macro. (The DTFCN declarative macro not only defines the file but also provides the data handling logic module for console files.)

The files described in this part include:

- Card - card readers and punches
- Console
- Optical Reader
- Magnetic Ink Character Recognition
- Printer
- Paper Tape
- Optical Reader/Sorter

## INITIALIZATION AND TERMINATION

Processing of a file by logical IOCS requires that the file be initialized, or opened, prior to the transfer of any data by the problem program. Likewise, when the transfer of all data is complete, the file is closed.

With the exception of magnetic ink and optical reader files, which are handled separately, all unit record files are opened by the unit record open logical transient phase, $$BOUR01, fetched by the Open Monitor (refer to Volume 1). On the other hand, unit record files (except magnetic ink character reader) require no special termination procedures and are closed by the Close Monitor which simply resets the open indicator in the DTF table for the file.

- 3504 Card Reader
- 3505 Card Reader
- 3525P Card Punch
- 3525RP Card Punch with read feature
- 3881 Optical Mark Reader
- 5425 Multifunction Card Unit

The files associated with these devices are defined by the DTFCD macro.

DTFCD Macro

Three types of DTF tables can be generated by the DTFCD macro. The DTF table type generated for a particular file depends on the TYPEFLE= parameter specified by the user in the DTFCD macro. The three table types are:

- DTFCD: Input (Reader) if TYPEFLE=INPUT (Figure 1).
- DTFCD: Output (Punch) if TYPEFLE=OUTPUT (Figure 2).
- DTFCD: Combined if TYPEFLE=COMBND (Figure 3). This parameter can be specified for 1442N1 or 2520B1 reader punch, or a 2540 punch with the punch feed read (PFR) feature.

The generated DTFCD table contains information describing the file and serves as a linkage to the CD logic module that is generated by a corresponding CDMOD macro.

| Bytes | Bits | Contents | Function |
|---|---|---|---|
| Bytes 40-47 as used for all files except 2560 and 5425 files. | | | |
| 40-43<br>(28-2B) | | LA &IOREG,0(14)<br>NOP 0 | Load user pointer register. |
| 44-47<br>(2C-2F) | | MVC 0(&BLKSIZE,13),0(14)<br>NOP 0<br>DC X'0000' | Move IOAREA to WORKA. |
| The following bytes (48-55) are added for 3504, 3505, and 3525 associated files. | | | |
| 48-51<br>(30-33) | | DC A(name)<br>B 16(15)<br>B 20(15)<br>DC F'0' | If ERROPT=name[2].<br>If ERROPT=SKIP.<br>If ERROPT=IGNORE.<br>If ERROPT=omitted. |
| 52-55<br>(34-37) | | DC A(ASOCFLE) | Address of associated DTF table[7].<br>(3525 only). |
| Bytes 40 onward as used for 2560 and 5425 files. | | | |
| 40-47<br>(28-2F) | | | Stacker select CCW. |
| 48-51<br>(30-33) | | LA &IOREG,0(14)<br>NOP 0 | |
| 52-57<br>(34-39) | | MVC 0(&BLKSIZE,13),0(14)<br>NOP 0<br>DC X'0000' | Move ICAREA to WORKA. |
| 58-63<br>(3A-3F) | | CLC 0(L,14),64(1) | Test for end-of-file.<br>L=4 if MODE=C; L=2 in other cases. |
| 64-67<br>(40-43) | | DC C'/*  '<br>DC X'0C001022' | End-of-file indicator if MODE=E.<br>In other cases. |
| 68-71<br>(44-47) | | DC A(name)<br>B 16(15)<br>B 20(15)<br>DC F'0' | If ERROPT=name[2].<br>If ERROPT=SKIP.<br>If ERROPT=IGNORE.<br>If ERROPT=omitted. |
| The following bytes are added for 2560 or 5425 associated files. | | | |
| 72-75<br>(48-4B) | | DC A(ASOCFLE) | Address of associated DTF<br>table[7]. |
| 76-81<br>(4C-51) | | MVC 0(&BLKSIZE,14),82(1) | Move card image to<br>ICAREA1. |
| 82<br>(52) | | DC &BLKSIZE.C' ' | Buffer for card image. |

[1]CMR only for 3504 and 3505.
[2]ERROPT for 2560, 3504, 3505, 3525, or 5425 READ file.
[3]3504, 3505, and 3525 with or without CONTROL=YES specified.
[4]2560, 3525, or 5425 with or without CONTROL=YES specified.
[5]2560, 3525, or 5425 without CONTROL=YES specified.
[6]Defaults to pocket2 for 3504, 3505, and 3525.
[7]Present only when 2560, 3525, or 5425 associated files are specified for the input DTF.

Numbers in parentheses are displacements in hexadecimal notation.

Figure 1. DTFCD: Input (reader) (2 of 2).

| Bytes | Bits | Contents | Function |
|---|---|---|---|
| 32-33 (20-21) | | LR 12,(RECSIZE)<br>NOPR 0 | Undefined records only. |
| 34-37 (22-25) | | LA &IOREG,4(14)<br>NOP 0 | Load user pointer register. |
| 38 (26) | 0-2 | | Not used. |
| | 3 | | 1 = 5425. |
| | 4 | | 1 = 2560. |
| | 5 | | 1 = 3525. |
| | 6 | | 1 = 1442 or 2596. |
| | 7 | | 1 = 2520B1. |
| 39 (27) | | DC C' ' | Blank for eject last card. |
| For all files except 2560 and 5425 files. | | | |
| 40-47 | | | Punch CCW. |
| 48-55 | | | Eject CCW for last card if 2520. |
| For 2540 files if CRDERR is specified. | | | |
| 48-55 | | | Retry CCW. |
| 56- | | DC CL80' ' | Save area card image. |
| For 3525 Punch/Interpret files. | | | |
| 48-55 | | | Load CCW. |
| 56-63 | | | Print CCW. |
| 64- | | DC 64C' ' | Print buffer. |
| For 3525 Associated files. | | | |
| 48-51 | | DC A(ASOCFIE) | Pointer to associated file. |
| For 2560 and 5425 files | | | |
| 40-47 | | DC D'0' | Eject CCW.<br>If FUNC=RP or RPW. |
| 48-55 | | | Stacker select CCW. |
| 56-63 | | | Punch and Feed CCW. |

[1]The bucket bytes handle undefined records.
[2]Valid for 3525 READ/PUNCH, PUNCH/PRINT, and READ/PUNCH/PRINT files.
[3]Valid for 3525 only.
[4]Defaults to pocket2 for 3525.

Numbers in parentheses are displacements in hexadecimal notation.

Figure 2. DTFCD: Output (punch) (2 of 3)

| Bytes | Bits | Contents | Function |
|---|---|---|---|
| 0-15 (00-0F) | | | CCB. |
| 16 (10) | 0-1 | | Not used. |
| | 2 | | CCBCL open; ignore option. |
| | 3 | | Not used. |
| | 4 | | OPENR relocates DTF address constants. |
| | 5-7 | | Not used. |
| 17-19 (11-13) | | | Address of logic module. |
| 20 (14) | | X'00' | DTF type. |
| 21 (15) | | | Command code (X'02' for 1442, X'C2' for 2520, 2540). |
| 22 (16) | | | Command code (X'01' for 1442, X'C9' for 2520, 2540). |
| 23 (17) | | | Command code (X'01' for 1442, X'09' for 2520, 2540). |
| 24-31 (18-1F) | | | CCW. |
| 32-35 (20-23) | | | Input area address. |
| 36-39 (24-27) | | | Output area address. |
| 40-41 (28-29) | | | Input blocksize. |
| 42-43 (2A-2B) | | | Output blocksize. |
| 44-49 (2C-31) | | MVC 0(&BLKS,13),C(14) | |
| 50-55 (32-37) | | MVC 0(&OUBL,14),C(13) | |
| 56-59 (38-3B) | | | End-of-file address. |
| 60-67 (3C-43) | | | Savearea. |
| 68-73 (44-49) | | MVC 1(&OUBL-1,13),0(13) | |
| 74-77 (4A-4D) | | MVI C(13),X'40' | |
| 78-79 (4E-4F) | | | Constant (blanks). |
| 80-83 (50-53) | | | Constant address (bytes 78-79). |

Figure 3. DTFCD: Combined reader/punch file.

CDMOD: PUT Macro Charts AJ-AM

Objective: To punch a card (normal or combined or associated files).

Entry: From a PUT macro expansion.

Exit: To the problem program.

Method: For regular output card files, the PUT routine punches the next sequential record in the file into a card. If WORKA=YES is specified in the DTFCD and CDMOD macros, the record is built in the workarea and then moved to the appropriate output area before punching is performed. If IOAREA2=YES is specified in the DTFCD and CDMOD macros, overlap of I/O and processing is possible by alternately switching output areas. (Dual I/O areas are not valid for associated files.)

For combined files (1442 or 2540 with PFR), IOAREA2 if it is specified, is used as the output area. If IOAREA2 is not specified, ICAREA1 serves as both the input and the output area.

For associated files, macro sequence checking is performed. The GET-PUT sequence must be maintained for these files.

Punching of a card in a 2560 or 5425 associated file can be initiated by:

- PUT for the punch file if FUNC=RP.
- PUT for the associated print file or GET for the next card if no printing is to be performed if FUNC=RPW.
- PUT for the punch file or PUT for the print file if FUNC=PW.

CONSOLE FILES

Console files (CN) are files associated with the system 3210 or 3215 console, or with the Display Operator Console (Models 115 or 125 only). A DTFCN macro defines these files as input, output, or combined files containing either fixed unblocked or undefined records. Combined files can contain only fixed-length records.

Console files are neither opened nor closed by logical IOCS.

DTFCN Macro

Logical IOCS support of console files differs from other files. One macro, DTFCN, generates both the DTF table and the logic module (Figure 4). The Version 5 DTFCN macro generates both table and module from statements contained in the source statement library. If TYPEFLE=INPUT is specified in the DTFCN macro, the module generated supplies the logic for both the GET and PUT functions. If TYPEFLE=OUTPUT is specified, the generated module supplies the logic for only the PUT function. If TYPEFLE=COMBND is specified, the generated module supplies the logic for the GET, PUT, and PUTR functions.

DTFCN: GET Macro Chart AN

Objective: To read from the system console, that is, to allow a record to be typed in from the console keyboard.

Entry: From a GET macro expansion to the label IJ2Xxxxx.

Exit: To the problem program.

Method: Upon entry to the GET routine, the CCW command code is set to hex '0A' for a read operation. If UNDEF is specified in the RECFORM= parameter in the DTFCN macro, the user specified BLKSIZE is moved into the byte count area of the CCW. (If FIXUNB is specified in the RECFORM= parameter in the DTFCN macro, the byte count area of the generated CCW automatically contains a count of 80.) An SVC 0 is then issued to read the record into ICAREA1. If a workarea was not specified in the DTFCN macro, control returns to the problem program immediately after completion of the I/O operation.

If WORKA=YES is specified in the DTFCN macro, the contents of ICAREA1 are moved to the workarea before control returns to the problem program.

DTFCN: PUT Macro Chart AP

Objective: To write a record on the system console printer.

Entry: From a PUT macro expansion to the label IJPTxxxx.

Exit: To the problem program.

Method: Upon entry to the PUT routine, the CCW command code is set to hex '09' for a write operation. If a workarea and TYPEFLE=OUTPUT are specified in the DTFCN macro, a test determines if a previous I/O operation is complete. If not, an SVC 7 is issued. On a combined I/O operation (TYPEFLE=INPUT) with a workarea specified, this test is bypassed.

If the file definition specifies undefined records (RECFORM=UNDEF parameter in the DTFCN macro) a check is made to determine if the BLKSIZE is greater than the record length, RECSIZE. (The byte count in the generated CCW is automatically equal to the BLKSIZE specified by the user.) If BLKSIZE is greater, the CCW byte count is modified to the value of RECSIZE.

Note: If RECSIZE is greater than BLKSIZE, the output record is truncated.

In all cases where a workarea is specified, the record is moved from the workarea to IOAREA1 before an SVC 0 is issued to write the record on the console printer. If TYPEFLE=OUTPUT is specified, control returns directly to the problem program. Otherwise, a test determines if a previous I/O operation is complete. If not, an SVC 7 is issued.

If no workarea is specified, the record to be written is available in IOAREA1, and an SVC 0 is issued directly. The routine then waits for completion of the I/O operation before control is returned to the problem program.

DTFCN: PUTR Macro Chart AQ

Objective: To write a message to or read a message from the console keyboard.

Entry: Form the PUTR macro expansion.

Exit: To the problem program.

Method: The PUTR macro processes fixed-length unblocked records only. Depending on whether or not a workarea is specified, one of the following methods is used:

a. No workarea specified. When the PUTR routine is entered, register 14 is saved and a branch is made to the PUT routine. INPSIZE and the input address are moved into the CCW and control is passed to the GET routine. The output area address and BLKSIZE are then restored in the CCW. The action bit in the CCW is turned off, register 14 is restored, and control is passed to the user.

b. Workarea specified. When the PUTR routine is entered, register 14 is saved and a branch is made to the PUT routine. The contents of register 0 is saved and that register is loaded with the input workarea address. INPSIZE and BLKSIZE in the DTFCN are interchanged. BLKSIZE and the address of the output area are then restored to the CCW. The CCW action bit is turned off, register 14 is restored, and control is passed to the user.

MAGNETIC INK CHARACTER RECOGNITION FILES

Magnetic Ink Character Recognition (MICR) files are input files processed by 1255/1259/1419 magnetic ink character readers, or 1270/1275 Optical Reader/Sorters. MICR type files are defined by a DTFMR macro and the processing logic module is generated by an associated MRMOD macro.

DTFMR Macro

The DTFMR Macro can generate two types of DTF tables (Figure 5). The type generated for a particular file depends on the ADDRESS= parameter specified by the user in the DTFMR macro. Although the two table types have been combined in this publication, the variation in the entries for ADDRESS=DUAL are noted in the table.

| Bytes | Bits | Function |
|---|---|---|
| 40-41 (28-29) | | Length of DTF table. |
| 42-43 (2A-2B) | | Device type indicator. |
| 44-45 (2C-2D) | | Record type. |
| 46-49 (2E-31) | | Reserved for future use. |
| 50-51 (32-33) | | I/O register. |
| 52-55 (34-37) | | End-of-file address. |
| 56-59 (38-3B) | | IOAREA2/1 address. |
| 60-63 (3C-3F) | | Document buffer size. |
| 64-65 (40-41) | | Blocking factor/Number of buffers. |
| 66-67 (42-43) | | I/O area size. |
| 68-71 (44-47) | | Record length. |
| 72-76 (48-4C) | | Sense information. |
| 77 (4D) | | Supervisor switch. |
| 78-79 (4E-4F) | | Logical class and unit numbers (secondary--for DUAL addressing only). |
| 80-81 (50-51) | | Register alignment bytes. |
| 82-83 (52-53) | | Logical class and unit numbers (primary--for DUAL addressing). |
| 84-87 (54-57) | | Document buffer size. |
| 88 (58) | | Command code (4C). |
| 89-91 (59-5B) | | Address of last byte of first document buffer. |
| 92 (5C) | | Command code (4C). |
| 93-95 (5D-5F) | | Address of last byte of last document buffer. |

Figure 5. DTFMR (2 of 4).

| Bytes | Bits | Function |
|---|---|---|
| 192-199 (C0-C7) | | CCW - Stacker Select. |
| 200-207 (C8-CF) | | CCW - TIC. |
| 208-215 (D0-D7) | | CCW - Control. |
| 216-223 (D8-DF) | | CCW - BN. |
| 224-231 (E0-E7) | | CCW - Read. |
| 232-239 (E8-EF) | | CCW - Sense. |
| 240-247 (F0-F7) | | CCW - Disengage. |
| | | For DUAL Address Adapter |
| 160-167 (A0-A7) | | CCW - Engage. |
| 168-175 (A8-AF) | | CCW - Read Buffer 1. |
| 176-183 (B0-B7) | | CCW - Sense. |
| 184-191 (B8-BF) | | CCW - NOP. |
| 192-199 (C0-C7) | | CCW - Read Buffer 2. |
| 200-207 (C8-CF) | | CCW - MOD Sense. |
| 208-215 (D0-D7) | | CCW - Read Buffer 1. |
| 216-223 (D8-DF) | | CCW - MOD Sense. |
| 224-231 (E0-E7) | | CCW - TIC to NCP. |
| 232-239 (E8-EF) | | CCW - NOP. |
| 240-247 (F0-F7) | | CCW - MOD CTL. |
| 248-255 (F8-FF) | | CCW - Stacker Select. |
| 256-263 (100-107) | | CCW - MOD Sense. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 5. DTFMR (4 of 4).

Entry: From a GET macro expansion to the label IJUGETCK.

Exit: To the problem program after a document buffer is filled with a stacker-selected document, or after error conditions are posted in the buffer.

Method: When more than three records remain in the document buffer area, the user's I/O register (ICREG) is loaded with the address of the next document buffer to be processed and the previous document buffer is reset to binary zeros. Control then returns to the user.

When three, or fewer than three, records remain in the document buffer area, a test determines if three or more document buffers are empty and it is necessary to engage the MICR type device to feed more documents. If it is necessary to issue engage commands, a branch is made to IJUENGCK where the necessary modifications are made to the CCW chain and the I/O operation executed.

If an engage is not required or after the engage commands are issued, the status of the document buffer is checked. If the buffer is ready for processing, the procedure followed is the same as for more than three records remaining in the I/O area. If the document buffer is not ready for processing, checks are made for errors and flags are set in the buffer as indicators of the error conditions found. This information is then passed to the user for further analysis.

MRMOD: READ Macro Charts PA-BB

Objectives:

1. To provide a pointer, in the user register ICREG, to the next document buffer to be processed in the document buffer area (Figure 6).
2. To issue engage commands to the MICR type device when necessary.

Entry: From a READ macro expansion to the label IJUREAD.

Exit: To the problem program after a document buffer is filled with a stacker-selected document, or after error conditions are posted in the buffer.

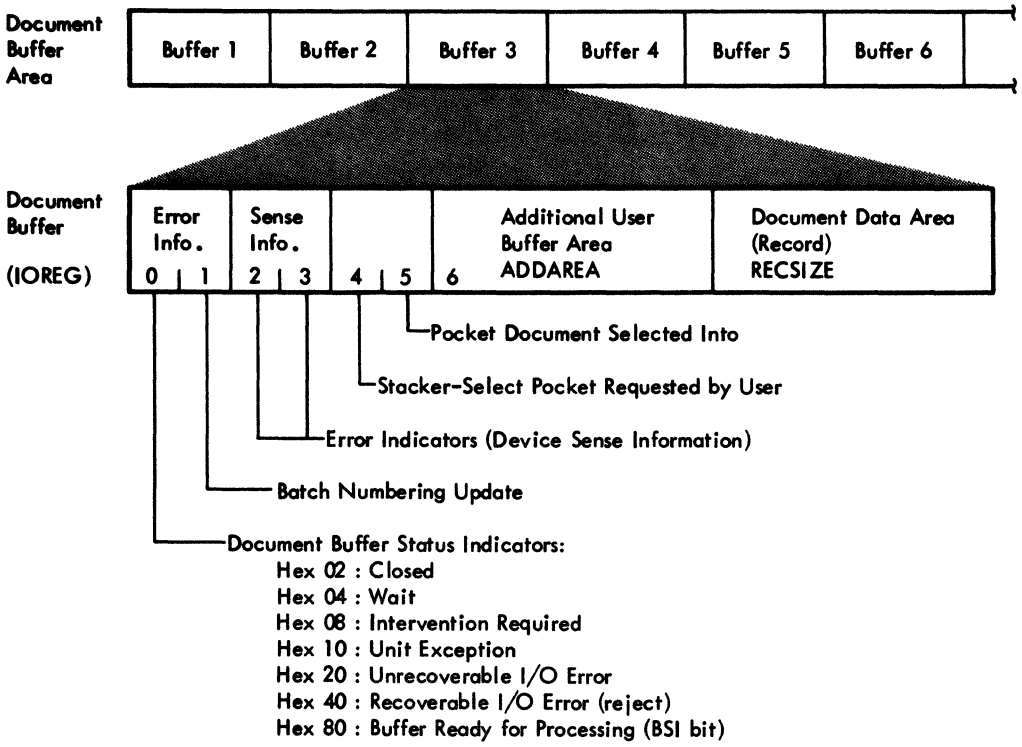



Figure 6. MICR buffer format (1255/1259/1270/1275/1419).

MRMOD: DISEN Macro Chart BD

Objective: To disengage the 1255/1259, 1270/1275, or 1419 MICR type device and stop the feeding of documents whenever necessary.

Entry: From a DISEN macro expansion to the label IJUDISEN.

Exit: To the problem program.

Method: The DISEN macro turns on the user disengage indicator (bit 0 of byte 21) in the DTF and checks for the type of address adapter used with the device. If dual addressing is specified, the NOP instruction in the CCW chain is changed to a disengage instruction, X'DF'. If single addressing is specified, the chaining bit is set on in the CCW chain.

Note: A DISEN macro must be issued before a LITE macro can be issued.

MRMOD: WAITF Macro Chart BD

Objective: To test whether any 1255/1259/1270/1275/1419 MICR devices specified in the macro operand (wait list) are operative and processing should continue, or if all the devices specified are inoperative and IOCS should enter the wait state.

Entry: From a WAITF macro expansion to the label IJUWAITF.

Exit: To the problem program.

Method: The WAITF macro loads a work register with the address of the DTF table for the file specified by the first operand. Tests are then made to determine if the file is in an operative condition. If the file is operative and ready, processing is resumed at the next sequential instruction following the macro expansion.

Note: When more than one file is specified in the operand of the WAITF macro (wait list), normal processing is resumed after the first operational file is detected, thereby omitting checking of any remaining files in the wait list. To accomplish this, the pointer in the wait list is repeatedly incremented by 4 (the length of each operand) until the end of the wait list is reached (the first nonzero byte). When the end-of-the-wait list has been detected, the pointer (register 14) then contains the address of the next sequential instruction following the macro expansion.

If the file currently being tested is not operational, the address of the DTF table for the next file specified in the macro operand is obtained and the tests are repeated for the next file.

If none of the files specified in the operand are operational, an SVC 29 is issued and the wait state is entered. If the system is operating in an MFS environment, processing is allowed to continue in another partition. When one of the files becomes operational, processing can resume in the partition in which the device is operating.

Initialization and Termination of MICR Type Files

Optical Reader/Sorter and Magnetic Ink Character Recognition files are some of the very few types of unit record files that are opened and closed by logical transients included in the system to handle a specific file type. These logical transients, $$BOMR01 and $$BCMR01, are fetched by the Open and Close Monitors respectively (refer to Volume 1).

$$BOMR01: Open MICR Type Files Chart BE

Objective: To open 1255/1259/1270/1275/1419 MICR type files and to initialize the document buffer area.

Entry: From the Open Monitor, $$BOPEN1, to the label FRSTINST.

Exit: To the TES processor, $$BOPEN.

Method: The $$BOMR01 routine clears the entire document buffer area and checks for device assignment. It then calculates the Physical Unit Block (PUB) entry address for the device, and determines from the PUB information which entry in the Supervisor table of DTF addresses (PDTABB) is to be used. The address of the DTF table is then inserted into the proper entry in the PDTABB table. Refer to DOS/VS Supervisor, SY33-8551, for the format and use of the PDTABB table.

The unit exception bit in the CCB is turned on and the remainder of the DTF is initialized. The open indicator (bit 0 of byte 30) is set on in the DTF to signal that the file is open and the TES Processor, $$BOPEN, is fetched to determine if more files are to be opened.

| Bytes | Bits | Function |
|---|---|---|
| 0-15 (00-0F) | | Dummy CCB. |
| 16 (10) | 0-1 | Not used. |
| | 2 | COBOL open; ignore option. |
| | 3 | Not used. |
| | 4 | DTF table address constants relocated by OPENR. |
| | 5-7 | Not used. |
| 17-19 (11-13) | | Address of logic module. |
| 20 (14) | | DTF type,(X'09'). DTF type, (X'0A' if HEADER=YES). |
| 21 (15) | | PIOCS switches. |
| | 0 | 1 = Open; 0 = Closed. |
| | 1 | 1 = Input. |
| | 2 | 1 = Control. |
| | 3 | 1 = Device is 1287. |
| | 4 | 1 = Header. |
| | 5 | Reserved for future use. |
| | 6 | 1 = RDLNE. |
| | 7 | Not used. |
| 22 (16) | | Not used. |
| 23 (17) | 0-6 | Not used. |
| | 7 | 1 = IIOCS posts a hopper empty condition to DTF. |
| 24-39 (18-27) | | CCB. |
| 40-47 (28-2F) | | Sense CCW. |
| 48-51 (30-33) | | Lost lines (equipment check). |
| 52-55 (34-37) | | After 9 retries for journal tape, or after 2 retries for documents. |
| 56-59 (38-3B) | | Wrong length records. |
| 60-63 (3C-3F) | | After 4 retries for journal tape, or after 2 retries for documents. |
| 64-67 (40-43) | | Keyboard corrections. |
| 68-71 (44-47) | | Count of data check errors. |
| 72-75 (48-4B) | | Lines marked. |

Figure 8. DTFCR (1 of 3).

| Bytes | Bits | Function |
|---|---|---|
| 130-131 (82-83) | | LR &RECS,13 |
| 132-133 (84-85) | | LR &IOR,13 |
| 134-135 (86-87) | | Sense. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 8. DTFOR (3 of 3).

CRMOD Macro

The module generated by the CRMOD macro provides the logic to perform the GET, CNTRL, and RDLNE functions for the 1287 (tape mode) optical reader, and the READ, WAITF, CNTRL, RESCN, and DSPLY functions for the 1287 (document mode) optical reader.

For 1287 operating in document mode, the logic module handles only unblocked records, and supports fixed and undefined record formats. Blocked records are also handled if the device is operating in tape mode.

The ORMOD is capable of generating many logic modules, each tailored to specific parameters. The number of different modules that can be generated is so great that it would be impractical to flowchart and describe every possible variation individually. To stay within practical limits, the internals of the OR module are flowcharted and described to indicate all variations.

CRMOD: CNTRL Macro Charts EG-EH

Objective: To execute a control operation for 1287 optical reader.

- When operating in tape mode, the control operation is either mark a line or read a complete line from the keyboard.
- When operating in document mode the control operation is either eject, eject and stacker-select, stacker-select, or increment the document.

Entry: From a CNTRL macro expansion.

Exits:

- Normal exit from the CNTRL routine is to the problem program.
- To the user's end-of-file routine when an EOF condition is reached.
- To the user's correction routine when a recoverable error occurs.

Method: The CNTRL routine ensures that any previous I/O operation is complete. It then sets the appropriate control command code (supplied by the user in general register 0) in the CCW and causes the control function to be performed.

If the control function is to read a line from the keyboard, the routine checks for a unit exception after the read operation is complete. If the control function is other than read a line from the keyboard, and a unit check occurs, the routine checks for command reject, late stacker-select, or recoverable errors.

If a unit exception or a read keyboard operation occurs, or if a recoverable type error occurs, the CNTRL routine branches to the address supplied in the DTF for the user's correction routine to attempt a recovery from the error before returning control to the problem program.

CRMOD: GET Macro Charts EJ-EN

Objective: To access a record from the 1287 (tape mode) optical character reader.

Entry: From the GET macro expansion.

Exit: To the problem program.

Exit: To the user-specified CCREXIT routine if a read error occurred or to the next sequential instruction following the WAITF macro expansion in the problem program if read was error-free.

Method: After checking for completion of the previous I/O, the routine tests for unit check. If unit check occurred, a check is made to determine if the cause of the interrupt is due to hardware or an unreadable character, lost line, etc. If a hardware error has occurred, a transient error routine is called, and a message is printed to the operator. If the error is due to an unreadable character or line, it is retried up to nine times (depending on error), posted to the user and then a branch is taken to the user's CCREXIT routine. On a return from CCREXIT, the operation is restarted from where the CCW chain was broken. Control is then returned to the problem program.

## DFR and DLINT Macros

The DFR and DLINT macros are used to build the format record that is required to read from the 3886 optical reader.

The DFR macro builds the first part of the format record, called the Document Information Record (Figure 9); it also generates two fields preceding the format record to provide information about that record. The first field is eight bytes long; when the field is opened or when the SETDEV macro is issued it contains the name of the format record. The second field is a two-byte binary field that contains the total length of the format record plus the two preceding fields. These two fields are not part of the format record and therefore not included when the format record is loaded into the 3886 control unit.

The DLINT macro builds the following for each line described:

- A line information record to describe an individual line (Figure 10).
- A field information record that describes an individual field on the line that is to be scanned (Figure 11).
- A field information record that describes an individual field on the line that is not to be scanned (Figure 12).
- A sync byte of X'FF' to indicate the end of the DLINT expansion.

In generating these records, the DLINT macro calls an inner macro (DLINTIN) 15 times to generate all fields on the line except the last field that is not to be scanned. The field information record for this last field is generated by the DLINT macro.

Figure 13 shows how the DFR and DLINT expansions combine to form a format record. An extra sync byte is generated to indicate the end of the format record.

| Bytes | Bits | Contents | Functions |
|---|---|---|---|
| 0 | | Address or character | Address of the right end of the field in BCD or Hex code of character used as field delimiter. |
| 1 | 0 | B'0' | Byte 0 represents character delimiter. |
| | | B'1' | Byte 0 represents address. |
| | 1 | B'1' | Indicates that the field is a critical field. |
| | 2-4 | B'000' | Not used. |
| | 5-7 | B'000' | Suppress high/low blanks - low fill. |
| | | B'001' | Suppress high/low blanks - high fill. |
| | | B'010' | Transmit all blanks - low fill. |
| | | B'011' | Suppress all blanks - no fill. |
| | | B'100' | Suppress all blanks - low fill. |
| | | B'101' | Suppress all blanks - high fill. |
| 2 | 0 | B'1' | Enable character edit for this field. |
| | 1-7 | | Field length. |
| 3 | 0-3 | B'0000' | Not used. |
| | 4-7 | B'0000' | Field is not to be scanned. |
| | | B'0001' | Field is mark read A font. |
| | | B'0010' | Field is mark read B font. |
| | | B'0101' | Field is numeric B font. |
| | | B'0110' | Field is alphanumeric B font (mode 1). |
| | | B'1001' | Field is numeric A font. |
| | | B'1010' | Field is alphanumeric A font (mode 1). |
| | | B'1011' | Field is alphanumeric A font (mode 2). |
| | | B'1100' | Field is NHP. |
| | | B'1101' | Field is Gothic (must have NHP feature). |
| | | B'1110' | Field is NHP (low sub). |

Figure 11. DLINT macro instruction expansion - field information record for a scannable field.

| Bytes | Bits | Contents | Functions |
|---|---|---|---|
| 0 | | | Address of the right end of the field in BCD. |
| 1 | 0 | B'1' | Byte 0 represents an address. |
| | 1-7 | B'0000000' | Not used. |
| 2-3 | | X'0000' | Not used. |

Figure 12. DLINT macro instruction expansion - field information record for a field that is not to be scanned.

DRMCD: CNTRL Macro Chart CA

Objective: To execute one of the following operations for the 3886 optical reader.

- Page mark current document when the document is ejected.
- Line mark indicated line when the document is ejected.
- Eject/stacker select the current document.

Entry: From the CNTRL macro expansion.

Exits:

- Normal exit from the CNTRL routine is to the problem program.
- To the user's end-of-file routine when an EOF condition occurs.
- To the user's COREXIT routine when an error occurs.

Method: The CNTRL routine sets up the field address and the length count in the CCW. If the CCW command is eject or stacker select, the routine builds a one-byte field to be passed to the 3886. The control CCW is then executed. When the I/O operation is completed, the routine checks for unit exception. If unit exception occurs, control is passed to the user's EOF routine. The routine then checks to see if any errors occurred. If an error occurred, it is posted to the user and control is passed to the user's COREXIT routine. On return from the COREXIT routine, control is passed to the problem program.

DRMCD: READ Macro Chart CA

Objective: To access one line of data from the 3886 optical reader.

Entry: From the READ macro expansion.

Exit: To the problem program.

Method: The READ routine reads a line of data from a document by first executing a scan CCW which scans one line of data and places that line in the 3886 buffer. A read CCW is then issued which reads the data into storage.

DRMCD: SETDEV Macro Chart CB

Objective: To allow the user to change the format record in the 3886 control unit.

Entry: From the SETDEV macro expansion.

Exits:

- Normal exit from the SETDEV routine is to the problem program.
- To the user's end-of-file routine if an EOF condition occurs.
- To the user's COREXIT routine when an error occurs.

Method: The SETDEV routine checks to see if the format record to be loaded from the 3886 control unit is in the format record area in the DTF. If so, the routine sets up the load format record CCW with the format record length and the format record address in storage. The routine then executes the CCW. If the format record is not in the format record area of the DTF, the routine loads the format record from the core image library. The routine then checks to see if the format record has the proper length. If so, the routine issues the load format record CCW to load the format record to the 3886 control unit. If any errors occur while this CCW is executed, they are posted to the user and control is passed to the user's COREXIT routine. Upon return from the COREXIT routine, control is passed to the problem program.

DRMCD: WAITF Macro Chart CB

Objective: To wait for I/O completion, to check for end-of-file conditions, and to indicate to the user if any errors have occurred.

Entry: From the WAITF macro expansion.

Exits:

- Normal exit from the WAITF routine is to the problem program.
- To the user's end-of-file routine when an EOF condition occurs.
- To the user's COREXIT routine when an error occurs.

Method: After checking for completion of the previous I/O operation, the routine moves the header record (20 bytes) to the address specified in the header parameter

| Bytes | Bits | Contents | Function |
|---|---|---|---|
| 40-43 (28-2B) | | X'00000000' | Not used. |
| 44-47 (2C-2F) | | | Start address of FR area in DTF. |
| 48-51 (30-33) | | | Address of four-byte pointer at the end of the FR area in the DTF. |
| 52-55 (34-37) | | | EOF routine address. |
| 56-63 (38-3F) | | | Scan CCW. |
| 64-71 (40-47) | | | Read CCW. |
| 72-79 (48-4F) | | | Read CCW. |
| 80-87 (50-57) | | | Control CCW. |
| 88-95 (58-5F) | | | Load format record CCW. |
| 96-99 (60-63) | | | COREXIT routine address. |
| 100-103 (64-67) | | | IOAREA1 area address. |
| 104-107 (68-6B) | | | Header area address. |
| 108-111 (6C-6F) | | | Exit indicator address. |
| 112 (70) | | | Start of FR area. |

Figure 14. DTFDR (2 of 2).

Initialization and Termination of Optical Reader Files

Optical reader files are opened by the logical transient $$BOCR01 that is fetched by the Open Monitor (refer to Volume 1). These files are closed by the Close Monitor, $$ECLOSE, which simply resets the open indicator in byte 21 of the DTF table.

$$BOCR01: Open Optical Reader Chart CC

Objective: To open an optical reader file.

Entry: From the Open Monitor ($$BOPEN1).

Exit: To $$BOPEN or cancel.

Method - 1287: If the optical reader file contains a header, this phase reads it into IOAREA1. If it does not contain a header, an I/O NOP is performed.

If a valid DTF type is found (indicating the presence or absence of the header), the routine returns to the Open Monitor ($$BOPEN) to determine if any more files need to be opened. The routine aborts the job if an invalid DTF type is present.

Method - 3886: This phase opens the file and loads the format record from disk into the DTF. If the format record is within

| Bytes | Bits | Contents | Function |
|---|---|---|---|
| 0-15 (00-0F) | | | CCB. |
| 16 (10) | 0 | | 1=2-line printer[3],[4]; 0=other. |
| | 1 | | 1=ERROPT[3],[4]; 0=omitted. |
| | 2 | | COBOL open; ignore option. |
| | 3 | | 1=3525; 0=other. |
| | 4 | | OPENR relocates DTF address constants. |
| | 5-7 | | 000=PRINT only<br>011=PUNCH/PRINT[3]<br>010=READ/PRINT[3]<br>101=READ/PUNCH/PRINT[3] |
| 17-19 (11-13) | | | Address of logic module. |
| 20 (14) | | X'08' | DTF type. |
| | | X'07' | DTF type for 2560 and 5425. |
| 21 (15) | 0 | | 1 = Open; 0 = Closed. |
| | 1 | | First time switch. |
| | 2 | | 1 = Control character. |
| | 3 | | 1 = Fixed unblocked records. |
| | 4 | | 1 = Variable unblocked records. |
| | 5 | | 1 = 2 I/O areas. |
| | 6 | | 1 = Workarea. |
| | 7 | | 1 = Print overflow channel 9. |
| For printer and card punch devices | | | |
| 22 (16) | | X'09' | Normal command code[5]. |
| 23 (17) | | X'09' | Control command code[5]. |
| 24-27 (18-1B) | | DC A(IOAREA1+x) | Address of data in IOAREA1. |
| 28-31 (1C-1F) | | | Bucket.[1] |
| 32-33 (20-21) | | LR 12,(RECSIZE)<br>NOPR 0 | For undefined records only. |
| 34-37 (22-25) | | LA &ICREG,4(14)<br>NOP 0 | Only if ICREG=(r). |
| 38-39 (26-27) | | | Bucket.[2] |
| 40-47 (28-2F) | | 11,*,X'60',1<br>9,IOAREA,X'20',121 | CCW--Set up Selective Tape List Control[6].<br>STLIST not specified. |
| 48-55 (30-37) | | 9,IOAREA,X'20',121<br>A(Name)<br>DC A(ASOFILE) | CCW--STLIST specified[6].<br>Address of user error routine (3211 only).<br>If ASCFILE=filename[3]. |
| For the 2560 and 5425 Multifunction Card Machine | | | |
| 22 (16) | | X'00' | Not used. |
| 23 (17) | | B'HHHHHH00' | Print head selection byte.<br>H=1 specifies the corresponding head. |

Figure 15. DTFPR (1 of 2).

PRMCD Macro

The module generated by the PRMCD macro provides the logic to perform the CNTRL, PRTOV, and PUT functions for a printer file. The logic module handles only unblocked records, and supports fixed, undefined, and variable record formats.

The PRMCD macro can generate many logic modules, each designed to handle the conditions specified by the macro parameters. Because it is not possible to describe all the variations, the PRMOD is flowcharted to show the internal decisions made for the specified parameters.

PRMCD: CNTRL Macro Chart CD

Objective: To control the carriage space and skip operations or the 3211 character folding.

Entry: From a CNTRL macro expansion.

Exit: To the next sequential instruction in the problem program following the CNTRL macro expansion.

Method: This routine provides completely independent control of the printer carriage. It also controls UCSB character folding on the 3211. When FOLD is specified, bits 0 and 1 of the byte to be printed are assumed to be ones. The FOLD and UNFOLD parameters permanently override the previous fold condition. It is used when the PRMCD macro parameter CTLCHR is not specified.

The PRMCD CNTRL routine waits for a previous I/O operation to finish and then inserts the control character into the CCW command code. The required carriage operation is started and control returns to the problem program without waiting for completion of the carriage operation.

PRMCD: PRTOV Macro Chart CE

Objective: To cause and control an overflow skip.

Entry: From a PRTOV macro expansion.

Exit: To the user's carriage overflow routine if the address of the routine is supplied, or to the problem program at the next sequential instruction following the PRTOV macro expansion.

Method: The PRTOV (printer overflow) macro instruction is used with a logical printer file to specify the operation to be performed on a carriage overflow condition.

PRTOV requires two parameters. The first parameter must be the name of the logical file specified in the DTF header entry. The second parameter must specify the number of the carriage tape channel (9 or 12) used to indicate the overflow condition. When an overflow condition occurs, IOCS skips the printer carriage to channel 1.

An optional third parameter causes a branch to a user routine instead of a skip to channel 1 on an overflow condition. This parameter specifies the symbolic name representing the address of the user's routine. In the user's routine, any desired function can be performed except another PRTOV.

PRMCD: PUT Macro Charts CF-GG

Objective: To print a line and space, or to print a line and skip the appropriate tape, if the Selective Tape Lister feature is available.

Entry: From a PUT macro expansion.

Exit: To the problem program or cancel.

Method: This routine causes a record to be printed on the output device. The logic determines if two I/O areas are used, if a workarea is specified, and if CTLCHR controls the carriage. A test is made to determine if CNTRL is specified. If so, the CTLCHR cannot be used; CTLCHR and CNTRL are mutually exclusive (if one is used, the other cannot be specified).

If the CTLCHR=ASA option is used, this routine translates the control character to EBCDIC.

If associated files are used, macro sequence checking is performed. Printing (PUT to a print file) may be omitted.

For 2560 and 5425 associated files, the print module initiates the read and/or punch command of the associated read and/or punch file processed by a CDMOD.

If the Selective Tape List feature is used (1403 only), selected tapes are controlled through the use of a one-byte control field. This field is accessed by the optional operand, either STLSP=label, or STLSK=label, of the PUT macro. Figure 16 shows the format of the field specified by label.

| Bytes | Contents | Function |
|---|---|---|
| 0-15<br>(00-0F) | | CCB (X'08' in byte 2). |
| 16<br>(10) | X'20'<br>X'08' | CCBCL open; ignore option.<br>OPENR relocates DTF address constants. |
| 17-19<br>(11-13) | | Address of logic module. |
| 20<br>(14) | X'01' | DTF type. |
| 21<br>(15) | | Bit 0 OPEN indicator.<br>1 Two I/O areas.<br>2 EOF indicator.<br>3 Read error.<br>4-6 Not used.<br>7 Undefined record. |
| 22-25<br>(16-19) | MVI 26(1),X'07'<br>NOP 0(0) | NOP first SVC 0 if 2 areas.<br>If 1 area. |
| 26-27<br>(1A-1B) | SVC 0 | Read a record. |
| 28-29<br>(1C-1D) | SVC 0<br>NOPR 0 | Read another record if 2 areas.<br>If 1 area. |
| 30-31<br>(1E-1F) | LR &RECSIZE,14<br>NOPR 0 | Put record length in user's register.<br>No RECSIZE entry. |
| 32-35<br>(20-23) | A(&IOAREA2)<br>A(&IOAREA1) | If 2 areas.<br>If 1 area. |
| 36-39<br>(24-27) | A(&EOFADDR) | End-of-file address. |
| 40-43<br>(28-2B) | A(&ERROPT)<br>SR 0,0<br>SVC 6<br>B 12(15)<br>B 138(15) | Addr. of user's error routine if ERROPT=name.<br>ERROPT omitted.<br><br>ERROPT=SKIP.<br>ERROPT=IGNORE. |
| 44-47<br>(2C-2F) | A(&WLRERR)<br><br>B 12(15)<br>B 152(15)<br><br>B 152(15) | Address of user's WLR routine if<br>WLRERR=name.<br>WLRERR omitted and ERROPT=SKIP.<br>WLRERR and ERROPT both omitted, or<br>WLRERR omitted and ERROPT=IGNORE.<br>RECFORM=FIXUNB or omitted. |
| 48-55<br>(30-37) | X'02',&IOAREA1,X'C0',&BLKSIZE<br>X'06',&IOAREA1,X'C0',&BLKSIZE | CCW: if RECFORM=FIXUNB or omitted.<br>CCW: if RECFORM=UNDEF. |
| 56-63<br>(38-3F) | | Duplicate CCW. |
| 64-67<br>(40-43) | F'0' | Savearea for register 14. |
| 68-71<br>(44-47) | L &IOREG,48(1)<br>NOP 0(0) | Put input area address into user's register.<br>No IOREG entry. |

Figure 17. DTFPT: No translations, no shifts or deletes; device=2671.

| Bytes | Contents | Function |
|---|---|---|
| 0-15 (00-0F) | | CCB (X'00' in byte 2). |
| 16 (10) | X'20'<br>X'08' | CCBCI open; ignore option<br>DTF table address constants relocated by OPENR. |
| 17-19 (11-13) | | Address of logic module. |
| 20 (14) | X'01' | DTF type. |
| 21 (15) | | Bit 0 OPEN indicator.<br>1 Two I/O areas.<br>2 EOF indicator.<br>3-5 Not used.<br>6 Scanning.<br>7 Not used. |
| 22-25 (16-19) | MVI 26(1),X'07'<br>NOP 0(0) | NOP first SVC 0 if 2 areas.<br>If 1 area. |
| 26-27 (1A-1B) | SVC 0 | Read a record. |
| 28-29 (1C-1D) | SVC 0<br>NOPR 0 | Read another record if 2 areas.<br>If 1 area. |
| 30-31 (1E-1F) | H'0' | Record length field. |
| 32 (20) | C'02' | |
| 33-35 (21-23) | AL3(&IOAREA2)<br>AL3(&IOAREA1) | If 2 areas.<br>If 1 area. |
| 36-39 (24-27) | A(&EOFADDR) | End-of-file routine address. |
| 40-43 (28-2B) | A(0) | |
| 44-47 (2C-2F) | NOP 0(0) | |
| 48-55 (30-37) | X'02',&IOAREA1,X'C0',&CVBLKSZ<br>X'02',&IOAREA1,X'00',&BLKSIZE | CCW: if CVBLKSZ specified.<br>if CVBLKSZ not specified. |

Figure 19. DTFPT: Translation, shifts and deletes, with fixed unblocked records; device=2671 (1 of 2).

| Bytes | Contents | Function |
|---|---|---|
| 0-15 (00-0F) | | CCB (X'08' in byte 2). |
| 16 (10) | X'20'<br>X'08' | CCBCI open; ignore option.<br>DTF table address constants relocated by CPENR. |
| 17-19 (11-13) | | Address of logic module. |
| 20 (14) | X'01' | DTF type. |
| 21 (15) | | Bit 0 OPEN indicator.<br>1 Two I/O areas.<br>2 EOF indicator.<br>3-5 Not used.<br>6 Scanning.<br>7 Read error. |
| 22-25 (16-19) | MVI 26(1),X'07'<br>NCP 0(0) | NCP first SVC 0 if 2 areas.<br>If 1 area. |
| 26-27 (1A-1B) | SVC 0 | Read a record. |
| 28-29 (1C-1D) | SVC 0<br>NOPR 0 | Read another record if 2 areas.<br>If 1 area. |
| 30-31 (1E-1F) | H'0' | Record length field. |
| 32-35 (20-23) | A(&IOAREA2)<br>A(&IOAREA1) | If 2 areas.<br>If 1 area. |
| 36-39 (24-27) | A(&ECFADDR) | End-of-file routine address. |
| 40-43 (28-2B) | A(&ERROPT)<br><br>SR 0,0<br>SVC 6<br>B 16(15)<br>B 246(15) | Address of user's routine if ERROPT=name.<br><br>ERROPT omitted.<br><br>ERROPT=SKIP.<br>ERROPT=IGNORE. |
| 44-47 (2C-2F) | A(&WLRERR)<br><br>B 16(15)<br>B 260(15) | Address of user's WLR routine if WLRERR=name.<br><br>WLRERR omitted and ERROPT=SKIP.<br>WLRERR omitted and ERROPT=IGNORE omitted. |

Figure 20. DTFPT: Translation, shifts, and deletes, with undefined records; device=2671 (1 of 2).

| Bytes | Contents | Function |
|---|---|---|
| 0-15<br>(00-0F) | | CCB (X'88' in byte 2). |
| 16<br>(10) | X'08' | Indicates DTF table relocated by OPENR. |
| 17-19<br>(11-13) | | Address of logic module. |
| 20<br>(14) | X'01' | DTF type. |
| 21<br>(15) | | Bit 0 OPEN indicator.<br>1 Two I/O areas.<br>2 EOF indicator.<br>3 Read error.<br>4-6 Not used.<br>7 Undefined record. |
| 22-25<br>(16-19) | MVI 26(1),X'07'<br>NOP 0(0) | NOP first SVC 0 if 2 areas.<br>If 1 area. |
| 26-27<br>(1A-1B) | SVC 0 | Read a record. |
| 28-29<br>(1C-1D) | SVC 0<br>NOPR 0 | Read another record if 2 areas.<br>If 1 area. |
| 30-31<br>(1E-1F) | LR &RECSIZE,14<br>NOPR 0 | Put record length into user's register.<br>No RECSIZE entry. |
| 32-35<br>(20-23) | A(&IOAREA2)<br>A(&IOAREA1) | If 2 areas.<br>If 1 area. |
| 36-39<br>(24-27) | A(&EOFADDR)<br>SVC 50<br>H'0' | End-of-file address.<br>No EOFADDR operand. |
| 40-43<br>(28-2B) | A(&ERROPT)<br>SR 0,0<br>SVC 6<br>B 0(14)<br>B 4(14) | Addr. of user's error routine if ERROPT=name.<br>ERROPT omitted.<br><br>ERROPT=SKIP<br>ERROPT=IGNORE |
| 44-47<br>(2C-2F) | A(&WLRERR)<br>A(&ERROPT)<br>B 0(14)<br>B 8(14)<br><br>B 8(14) | Address of user's WLR routine if WLRERR=name.<br>WLRERR absent and ERROPT=name.<br>WLRERR absent and ERROPT=SKIP.<br>WLRERR and ERROPT both absent or WLRERR absent and ERROPT=IGNORE.<br>RECFORM=FIXUNB or absent. |
| 48-55<br>(30-37) | X'02',&IOAREA1,X'00',&BLKSIZE<br>X'06',&IOAREA1,X'00',&BLKSIZE | CCW: if RECFORM=FIXUNB or absent.<br>CCW: if RECFORM=UNDEF. |
| 56-63<br>(38-3F) | | Duplicate CCW. |
| 64-67<br>(40-43) | F'0' | Save area for register 14. |
| 68-71<br>(44-47) | L &IOREG,48(1)<br>NOP 0(0) | Put input area address into user's register.<br>No IOREG entry. |

Figure 21. DTFPT: No translation, no shifts or deletes; device=1017.

| Bytes | Contents | Function |
|---|---|---|
| 0-15<br>(00-0F) | | CCB (X'80' in byte 2). |
| 16<br>(10) | X'08' | Indicates DTF table relocated by OPENR. |
| 17-19<br>(11-13) | | Address of logic module. |
| 20<br>(14) | X'01' | DTF type. |
| 21<br>(15) | | Bit 0 OPEN indicator.<br>1 Two I/O areas.<br>2 EOF indicator.<br>3-5 Not used.<br>6 Scanning.<br>7 Not used. |
| 22-25<br>(16-19) | MVI 26(1),X'07'<br>NOP 0(0) | NOP first SVC 0 if 2 areas.<br>If 1 area. |
| 26-27<br>(1A-1B) | SVC 0 | Read a record. |
| 28-29<br>(1C-1D) | SVC 0<br>NOPR 0 | Read another record if 2 areas.<br>If 1 area. |
| 30-31<br>(1E-1F) | H'0' | Record length field. |
| 32<br>(20) | C'02' | |
| 33-35<br>(21-23) | AL3(&IOAREA2)<br>AL3(&IOAREA1) | If 2 areas.<br>If 1 area. |
| 36-39<br>(24-27) | A(&ECFADDR)<br>SVC 50<br>H'0' | End-of-file address.<br>No ECFADDR operand. |
| 40-43<br>(28-2B) | A(0) | |
| 44-47<br>(2C-2F) | NOP 0(0) | |
| 48-55<br>(30-37) | X'02',&IOAREA1,X'00',&CVBLKSZ<br>X'02',&IOAREA1,X'C0',&BLKSIZE | CCW: if CVBLKSZ specified.<br>if CVBLKSZ not specified. |
| 56-59<br>(38-3B) | L &ICREG,96(1)<br>NOP 0(0) | Put input area address into user's register.<br>No ICREG operand. |

Figure 23. DTFPT: Translation, shifts and deletes, fixed unblocked records; device=1017 (1 of 2).

| Bytes | Contents | Function |
|---|---|---|
| 0-15<br>(00-0F) | | CCB (X'88' in byte 2). |
| 16<br>(10) | X'08' | Indicates DTF table relocated by CPENR. |
| 17-19<br>(11-13) | | Address of logic module. |
| 20<br>(14) | X'01' | DTF type. |
| 21<br>(15) | | Bit 0 CPEN indicator.<br>1 Two I/O areas.<br>2 EOF indicator.<br>3-5 Not used.<br>6 Scanning.<br>7 Read error. |
| 22-25<br>(16-19) | MVI 26(1),X'07'<br>NOP 0(0) | NOP first SVC 0 if 2 areas.<br>If 1 area. |
| 26-27<br>(1A-1B) | SVC 0 | Read a record. |
| 28-29<br>(1C-1D) | SVC 0<br>NOPR 0 | Read another record if 2 areas.<br>If 1 area. |
| 30-31<br>(1E-1F) | H'0' | Record length field. |
| 32-35<br>(20-23) | A(&IOAREA2)<br>A(&IOAREA1) | If 2 areas.<br>If 1 area. |
| 36-39<br>(24-27) | A(&ECFADDR)<br>SVC 50<br>H'0' | End-of-file address.<br>No ECFADDR operand. |
| 40-43<br>(28-2B) | A(&ERROPT) | Address of user's error routine if ERROPT=name. |
| | SR 0,0<br>SVC 6 | ERROPT omitted. |
| | B 0(14)<br>B 4(14) | ERROPT=SKIP<br>ERROPT=IGNORE |
| 44-47<br>(2C-2F) | A(&WLRERR)<br>A(&ERROPT)<br>B 0(14)<br>B 8(14) | Address of user's WLR routine if WLRERR=name<br>WLRERR absent and ERROPT=name.<br>WLRERR absent and ERROPT=SKIP.<br>WLRERR and ERROPT both absent or WLRERR absent and ERROPT=IGNORE. |

Figure 24. DTFPT: Translation, shifts and deletes, undefined records; device=1017 (1 of 2).

| Bytes | Contents | Function |
|---|---|---|
| 0-15<br>(00-0F) | | CCB (X'8A' in byte 2). |
| 16<br>(10) | X'08' | Indicates DTF table relocated by OPENR. |
| 17-19<br>(11-13) | | Address of logic module. |
| 20<br>(14) | X'01' | DTF type. |
| 21<br>(15) | | Bit 0 OPEN indicator.<br>1 Two I/O areas.<br>2 Write error.<br>3 CLOSE indicator.<br>4-5 Retry counter.<br>6 Not used.<br>7 Undefined record. |
| 22-25<br>(16-19) | L &IOREG,48(1)<br>NOP 0(0) | Put output area address into user's register.<br>No IOREG operand. |
| 26<br>(1A) | X'&DELCHAR'<br>X'00' | Delete character.<br>No delete character specified. |
| 27<br>(1B) | | Not used. |
| 28-31<br>(1C-1F) | A(&IOAREA2)<br>A(&IOAREA1) | If 2 areas.<br>If 1 area. |
| 32-47<br>(20-2F) | X'01',&IOAREA1,X'00',&BLKSIZE<br>D'0'<br>X'01',&IOAREA1,X'00',&BLKSIZE<br>X'01',68,X'C0',C1 | CCW if RECFORM=FIXUNB.<br><br>CCW if RECFORM=UNDEF. |
| 48-55<br>(30-37) | | Duplicates CCW. |
| 56-59<br>(38-3B) | A(&TRANS)<br>A(0) | Address of user's translate table.<br>If no translation specified. |
| 60-63<br>(3C-3F) | A(&ERROPT)<br><br>SR 0,0<br>SVC 6<br>B 0(14) | Address of user's error routine if<br>ERROPT=none.<br>ERROPT absent.<br><br>ERROPT=IGNORE. |
| 64-67<br>(40-43) | STH &RECSIZE,54(1)<br>NOP 0(0) | Save user's record length.<br>No RECSIZE operand. |
| 68<br>(44) | X'&EORCHAR'<br>X'00' | End of record character.<br>No EORCHAR operand or RECFORM=FIXUNB. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 25. DTFPT: no shifts, device=1018.

PTMCD Macro

The PTMOD macro can generate ten logic modules providing the GET or PUT function for paper tape files. The particular module depends on DEVICE=, TRANS=, and SCAN= macro parameters. The user can obtain four different logic modules for the 2671 paper tape reader if DEVICE=2671 is specified or if DEVICE= is omitted. He can obtain four different logic modules for the 1017 paper tape reader if DEVICE=1017 is specified and two different logic modules for the 1018 paper tape punch if DEVICE=1018 is specified.

The TRANS= parameter specifies whether or not the generated module contains the logic needed to translate non-shifted punched paper tape characters into EBCDIC code on input (2671/1017 paper tape reader) or EBCDIC code into non-shifted punched paper tape characters on output (1018 paper tape punch).

The SCAN= parameter specifies whether or not the generated module contains the logic to handle records containing shift and/or delete characters.

All modules support the use of either 1 or 2 I/O areas. Decision blocks are included in the flowcharts to indicate the procedure followed for both 1 and 2 I/O areas.

PTMCD: GET Macro, No Translation, Device=2671 Chart CJ

Objective: To read a record from paper tape punched in EBCDIC code.

Entry: From a GET macro expansion.

Exit: To the problem program, or to the user's EOF routine.

Method: To support the use of either 1 or 2 I/O areas, the GET routine makes use of three key instructions contained in the DTF table. The three instructions, repeated here for convenience, are:

| | 1 Input Area | 2 Input Areas |
|---|---|---|
| IJE1MOVE | NOP 0(0) | MVI IJE1SVC1,X'07' (NOP first SVC 0 instruction) |
| IJE1SVC1 | SVC 0 | SVC 0 (first) |
| IJE1SVC2 | NOPR 0 | SVC 0 (second) |

If only one I/O area is specified for the file, the GET routine uses the 'first' SVC 0 instruction to read a record from the paper tape device. After the complete record has been read into the I/O area, the I/O area address in the CCW and the I/O area address in the DTF table are exchanged. (In this case, the exchange is meaningless because both addresses are the same when only one I/O area is used.) The record is then checked for errors and control is returned to the problem program.

If two I/O areas are specified for the file, the GET routine uses the 'first' SVC 0 instruction to read the first record only. After the first record is read, the first SVC 0 instruction is made a NOP by the execution of the instruction contained in the DTF table at the label IJE1MOVE. The first SVC 0 instruction is only restored if it is needed to handle error conditions. After checking for I/O complete and reading a complete record into the I/O area, the addresses of the two I/O areas are exchanged, and the record is checked for errors. The GET routine then NOPs the first SVC 0 instruction and uses the 'second' SVC 0 instruction to read the next record. From this point on, only the second SVC 0 instruction is used exclusively, except for handling error condition. The GET routine returns control to the problem program without waiting for completion of the I/O operation started by the second SVC 0 instruction.

PTMCD: GET Macro, Translation, No Shifted Code, Device=2671 Chart CJ

This GET routine functions the same as the GET routine for no translation. It differs only in the logic included to accomplish the required translation once a complete record has been read into the I/O area.

Only undefined records are checked for wrong length errors. For this, the user must specify one byte in excess of the longest record. If the residual count is zero, control is given to the wrong length error routine.

When physical IOCS indicates a permanent data check (bit 4 of the communication byte in CCB), the reader is stopped following the erroneous character so that the input area is not completely filled. Logical IOCS starts a read operation to obtain the remainder of the record, then exits to the error option.

End of File: Upon detection of an end-of-file condition (unit exception in CSW) by physical IOCS, a flag is set in the CCB (bit 1 of the communication byte). When logical IOCS detects this flag, an input area can still be processed. Thus, control to the EOF routine is only given at the following GET macro instruction.

PTMOD: GET Macro, No Translation, Device=1017 Chart CM

Objective: To read a record from the paper tape without performing the translating or editing procedures.

Entry: From the GET macro expansion.

Exit: To the user's program one instruction after the GET macro expansion (normal return), or to one of the other possible user's routines (end-of-file, wrong length, data error).

Method: The method is essentially that just described under Basic Principles.

PTMOD: GET Macro, Translation, Device=1017 Chart CM

Objective: To read a record from the paper tape and to perform the translate function, that is, to translate the paper tape code to an acceptable code.

Entry: From the GET macro expansion.

Exit: To the user's program one instruction after the GET macro expansion, or to one of the other possible user's routines.

Method: This method is essentially that just described under Basic Principles, except for the translate function. The translate function is performed using the TRANS table specified by the user, which must satisfy the requirements of the TR instruction.

PTMOD: GET Macro, Translation, Shifted Code, Fixed Unblocked Records, Device=1017 Chart CN

Objective: To read a record from paper tape using the translating and editing procedures.

Entry: From the GET macro expansion.

Exit: To the user's program or to the EOF routine.

Method: The method is described under Basic Principles, except for obtaining the correct number of bytes to complete the fixed-length record.

The number of characters specified by the user in OVBLKSZ is read in, translated, and compressed as described in the following section (Charts GL and GM). If the resulting record is shorter than that specified in BLKSIZE, additional characters are read in, translated, and compressed to complete the record. Additional reads are performed until the record length is equal to or greater than BLKSIZE. On the next GET macro instruction, the remaining characters, if any, are stored in the leftmost positions of the next input area to be read in. The I/O command is modified by the length of the remainder.

PTMOD: GET Macro, Translation, Shifted Code, Undefined Records, Device=1017 Chart CP

Objective: To read a record from the paper tape and to perform the translating and editing procedures.

Entry: From the GET macro expansion.

Exit: To the user's program one instruction after the GET macro expansion, or to one of the user's routines (end-of-file, wrong length, data error).

Method: Only the translating and editing functions are considered. The logic is described under Basic Principles. The editing functions consist of the following:

1. A scan for shift and/or delete characters is made using the Scan table. All entries of this table are zero except the entries for the shift and/or delete characters. Scanning is performed by a TRT instruction.

2. When a shift and/or delete character is encountered, scanning stops and the corresponding address is stored.

CCW1 Write the whole record.

CCW2 Write the EOR character in the UNDEF record format.

Insertion of shift characters in a record results in lengthening of the record. If the user does not use the option OVBLKSZ, several WRITE operations are required to punch the record. If CVBLKSZ has been specified with a value greater than that specified for BLKSIZE, the record can be punched in a single operation.

The following steps describe the logic of the editing function:

1. Determine the shift status of the first character in the record to include the correct shift character in the CCW0. This character will be the record header.

2. Perform a scan, using the correct scanning table, LSCAN or FSCAN, depending on the current shift status.

3. When the end of the record or a change in the shift status is encountered, the scanning operation (TRT instruction) is stopped and the segment is translated.

4. Test to make sure that the end of the record does not overstep the bounds of the output area (BLKSIZE or CVBLKSZ). If space is left, the remainder of the record is moved by one character for right-justification to include the new shift character. Then, the procedure is resumed at step 2. If no space is left, the segment considered is punched. The remainder of the record is moved to the beginning of the output area, for left-justification. Then, the procedure is resumed at step 2.

Note that CVBLKSZ is ignored for undefined records. For further information on CVBLKSZ, see Paper Tape File (DTFPT) in DOS/VS Supervisor and I/O Macros, GC33-5373.

| Bytes | Bits | Contents[1] | Function | Record[2] Format |
|---|---|---|---|---|
| 0-15 (00-0F) | | | CCB. | |
| 8 (8) | | Input: X'00'-X'63'<br>Output: X'00'-X'04' (Variable)<br>X'00' (Undefined) | Buffer offset length, ASCII | |
| 16 (10) | 0 | | First time entered MTMOD for a file | |
| | 1 | | Not used. | |
| | 2 | | CCBCL open; ignore option. | |
| | 3 | | American National Standard CCBCL | |
| | 4 | | DTF table address constants relocated by OPENR. | |
| | 5 | | 1 = spanned records | |
| | 6 | | 1 = ASCII | V |
| | | | 0 = EBCDIC | V,S |
| | 7 | | ASCII input: 1=Length Check | V |
| | | | ASCII output: 1=Buffer offset length=4 | V |
| 17-19 (11-13) | | | Address of logic module. | |
| 20 (14) | | X'11' | Nonstandard or unlabeled. | |
| | | X'12' | Standard labeled, output. | |
| | | X'13' | Standard labeled, input, backwards. | |
| | | X'14' | Standard labeled, input, forwards. | |
| 21 (15) | 0 | | First time switch:<br>1 = not first-time entry.<br>0 = first-time entry. | |
| | 1 | | 1 = blocked.<br>0 = unblocked. | |
| | 2 | | 1 = 2 I/O areas.<br>0 = 1 I/O area. | |
| | 3 | | 1 = workarea, | F,U,V |
| | | | 0 = no workarea. | F,U,V |
| | | | 0 = workarea, spanned | S |
| | 4 | | 1 = input,<br>0 = output | |
| | 5 | | 1 = backwards.<br>0 = forwards. | |
| | 6 | | 1 = checkpoint.<br>0 = no checkpoint. | |
| | 7 | | 1 = TRUNC required during Close. | |
| 22-29 (16-1D) | | | Symbolic filename. | |
| 30 (1E) | | | Same as command code in CCW.<br>(X'01', X'02', or X'0C'). | |

Figure 27. DTFMT: Data files (1 of 10).

| Bytes | Bits | Contents[1] | Function | Record[2] Format |
|---|---|---|---|---|
| 36 (24) | 0 | | DTFPH: 1 = yes, 0 = no. | |
| | 1 | | CCBCL indicator: 1 = yes, 0 = no. | |
| | 2 | | File type: 1 = input, 0 = output. | |
| | 3 | | FBCV switch: 1 = yes, 0 = no. | |
| | 4 | | EOF-EOV switch (output): 1 = EOF, 0 = EOV. | |
| | 5 | | Open indicator: 1 = open, 0 = closed. | |
| | 6 | | 1 = variable or spanned records. | V,S |
| | 7 | | 1 = undefined records. | U |
| 37-39 (25-27) | | | EOF address. | |
| 40-43 (28-2B) | | | Block count. | |
| 44-47[1] (2C-2F) | | EXH 11,12,24(15) | Forward. | F |
| | | EXLE 11,12,24(15) | Backward. | F |
| | | L &VARBLD,DBBLOCKER | If VARBLD parameter is used. | V |
| | | NOP 0(0) | | S |
| | | DC F'0' | DBBLOCKER1. | U |
| 48-51 (30-33) | | LA 14,1(14) | | F,V,S |
| | | BCTR 14,0+NCPRC | Backward. | F,V,S |
| | | L &RECSIZE,DBBLOCKER1 | If RECSIZE given. | U |
| | | NOP 0(0) | For input if not NCP. | U |
| 52-55 (34-37) | | L &ICREG,DBBLOCKER1 | If ICREG specified. | F |
| | | L &ICREG,DBBLOCKER5 | If ICREG specified. | V |
| | | L &ICREG,DBBLOCKER2 | If ICREG specified. | U |
| | | NOP 0(0) | If no ICREG. | |
| | | L &RECSIZE,IJFVSREC | If spanned input. | S |
| | | ST &RECSIZE,IJFVSREC | If spanned output. | S |
| 56-63 (38-3F) | | | CCW. | |

Figure 27. DTFMT: Data files (3 of 10).

| Bytes | Bits | Contents[1] | Function | Record[2] Format |
|---|---|---|---|---|
| 80-83 (50-53) | | DC Y(BLKSIZE)+Y(ELKSIZE-1) | Forward. | F |
| | | DC &(BLKSIZE)+Y(ELKSIZE+1) | Backward. | F |
| | | DC F'0' | DEELCCKER4. | V,S |
| | | LR 12,RECSIZE | (Bytes 80-81 only.) | U |
| | | DC H'0' | (Bytes 82-83.) | U |
| 84-87 (54-57) | | DC Y(RECSIZE-1) | (Bytes 84-85.) | F |
| | | DC 2X'00' | (Bytes 86-87) Output, standard labels. | F |
| | | DC A(ICAREA1+4) | 1 I/C area: DEBLOCKER5, EECDIC. | V,S |
| | | DC A(ICAREA2+4) | 2 I/C areas: DEBLOCKER5, EECDIC. | V,S |
| | | DC A(ICAREA1+BUFCFF) | 1 I/C area: DEBLCCKER5, ASCII. | V |
| | | DC A(ICAREA2+BUFCFF) | 2 I/C areas: DEBLOCKER5, ASCII. | V |
| | | DC 2X'00' | (Bytes 84-85 cutput cnly) Standard Labels. Reserved fcr CPEN. | F, U |
| | | B 28(15) | Input cnly, ERRCPT=cmitted. | U |
| | | B 24(15) | Input cnly, ERRCPT=SKIP. | U |
| | | B 28(15) | Input cnly, ERROPT=IGNCRE. | U |
| | | DC A(ERROPT) | Input cnly, ERROPT=ADDRESS | U |
| 88-91 (58-5B) | | DC A(WLRERR) | Input cnly WLRERR=ADDRESS. | For Fixed Length Reccrds cnly |
| | | B 24(15) | Input cnly, WLRERR cmitted and ERRCPT=SKIP. | |
| | | B 28(15) | Input cnly, WLRERR cmitted and ERRCPT=IGNCRE cr cmitted. | |
| | | DC 2X'00' | Output cnly, standard labels (bytes 88-89), reserved fcr CPEN. | |
| | | DC A(ERROPT) | Input cnly, WLRERR cmitted and ERRCPT=ADDRESS. | |
| 90-95 (5A-5F) | | DC 6X'00' | File Serial Number, Standard Labels, Output cnly. | |

Figure 27. DTFMT: Data files (5 cf 10).

| Bytes | Bits | Contents[1] | Function | Record[2] Format |
|---|---|---|---|---|
| 100-103 (64-67) | | DC A(WLRERR) | Input only, WLRERR=ADDRESS. | For Variable Length and Spanned Records |
| | | B 24(15) | Input only, WLRERR=omitted and ERROPT=SKIP. | |
| | | B 32(15) | Input only, WLRERR=omitted and ERROPT=IGNORE or omitted. | |
| 100-101 (64-65) | | DC 2X'00' | Output only, standard labels, reserved for OPEN. | |
| 104-107 (68-6B) | | DC A(ERROPT) | Input only, ERROPT=ADDRESS. | |
| | | B 28(15) | Input only, ERROPT=omitted. | |
| | | B 24(15) | Input only, ERROPT=SKIP. | |
| | | B 28(15) | Input only, ERROPT=IGNORE. | |
| | | DC A(ERROPT) | Output, Nonstandard labels only (Version 3 onward). ERROPT=ADDRESS | |
| 108-111 (6C-6F) | | DC 4X'00' | Volume sequence number. Standard labels, output only. | |
| 108-109 (6C-6D) | | DC 2X'00' | Standard labels, input only. Reserved for OPEN. | |
| 110-115 (6E-73) | | DC 6X'00' | File serial number. Standard labels, input only. | |
| 112-115 (70-73) | | DC 4X'00' | File sequence number. Standard labels, output only. | |
| 116-119 (74-77) | | DC A(ERROPT) | Output only, ERROPT=ADDRESS, standard labels only. | |
| 116-119 (74-77) | | DC 4X'00' | Volume sequence number Standard labels, input only. | |
| 121-123 (78-7B) | | DC 4X'00' | File sequence number. Standard labels, input only. | |

Figure 27. DTFMT: Data files (7 of 10).

| Bytes | Bits | Contents[1] | Functions | Record[2] Format |
|---|---|---|---|---|
| 100-103 (64-67) | | DC A(WLRERR) | Input only, WLRERR=ADDRESS. | For Spanned Records Only |
| | | B 24(15) | Input only, WLRERR=omitted and ERROPT=SKIP. | |
| | | B 32(15) | Input only, WLRERR=omitted and ERROPT=IGNORE or omitted. | |
| 100-101 (64-65) | | DC 2X'00' | Output only, standard labels, reserved for OPEN | |
| 102-107 (66-6B) | | File Serial Number | Standard labels, output only. | |
| 100-103 (64-67) | | DC 4X'00' | Output only, ERROPT=ADDRESS, Nonstandard labels only. | |
| 100-123 (64-7B) | | DC 24X'00' | Output only, ERROPT=omitted, Nonstandard labels. | |
| 104-107 (68-6B) | | DC A(ERROPT) | Input only, ERROPT=ADDRESS. | |
| | | B 24(15) | Input only, ERROPT=omitted. | |
| | | B 24(15) | Input only, ERROPT=SKIP. | |
| | | B 28(15) | Input only, ERROPT=IGNORE. | |
| 104-107 (68-6B) | | DC A(ERROPT) | Output only, ERROPT=ADDRESS, Nonstandard labels. | |
| 108-123 (6C-7B) | | DC 16X'00' | Output only, ERROPT=ADDRESS, Nonstandard labels. | |
| 108-111 (6C-6F) | | Volume Sequence Number | Standard labels, output only. | |
| 112-115 (70-73) | | File Sequence Number | Standard labels, output only. | |
| 116-119 (74-77) | | DC A(ERROPT) | Output only, ERROPT=ADDRESS, Standard labels. | |
| 120-123 (78-7B) | | DC 4X'00' | Output only, ERROPT=ADDRESS, Standard labels. | |
| 108-123 (6C-7B) | | DC 16X'00' | Input only, nonstandard labels. | |
| 108-109 (6C-6D) | | DC 2X'00' | Standard labels, input only, reserved for OPEN. | |
| 110-115 (6E-73) | | File Serial Number | Standard labels, input only. | |
| 116-119 (74-77) | | Volume Sequence Number | Standard labels, input only. | |
| 120-123 (78-7B) | | File Sequence Number | Standard labels, input only. | |

Figure 27. DTFMT: Data files (9 of 10).

| Bytes | Bits | Function |
|---|---|---|
| 0-15 (00-0F) | | CCB. |
| 16 (10) | 0-1 | Not used. |
| | 2 | COBOL open; ignore option. |
| | 3 | 1 = VOL1 label is at user specified density. |
| | 4 | 1 = DTF table address constants relocated by OPENR. |
| | 5-7 | Not used. |
| 17-19 (11-13) | | Address of logic module. |
| 20 (14) | | DTF type = X'10'. |
| 21 (15) | 0 | 1 = no rewind. |
| | 1 | 1 = rewind unload. |
| | 2 | 1 = workfile. |
| | 3 | 1 = read backward. |
| | 4 | 1 = write. |
| | 5 | 1 = POINTW. |
| | 6 | Not used. |
| | 7 | 1 = forward-space file before next operation. |
| 22-23 (16-17) | | Not used. |
| 24-25 (18-19) | | Record length. |
| 26-27 (1A-1B) | | Maximum BLKSIZE. |
| 28 (1C) | | Read command code (X'02' for read forward, X'0C' for read backward). |
| 29-31 (1D-1F) | | EOF address. |
| 32-39 (20-27) | | CCW. |
| 40-43 (28-2B) | | Block count, initialized 000C0000 for read forward, CC40C0C0 for read backward. |
| 44 (2C) | 0 | 1 = error routine. |
| | 1 | 1 = ignore. |
| | 2 | Not used. |
| | 3 | 1 = record fixed unblocked. |
| | 4-7 | Not used. |
| 45-47 (2D-2F) | | DC A(ERROPT) Address of error routine. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 28. DTFMT: Workfiles.

| Bytes | Bits | Contents | Function |
|---|---|---|---|
| 37-39 (25-27) | | | User label exit. |
| 40-43 (28-2B) | | DC F'0' | Reserved for OPEN. |
| 44-87 (2C-57) | | | EOV Routine |
| 88-89 (58-59) | | DC 2X'00' | Reserved for OPEN |
| 90-95 (5A-5F) | | DC 6X'00' | File Serial Number |
| 96-99 (60-63) | | DC 4X'00' | Volume Sequence Number |
| 100-103 (64-67) | | DC 4X'00' | File Sequence Number |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 29. DTFPH: Magnetic tape (2 of 2).

MTMOD MACRO

The MTMOD macro provides the logic modules required to process both magnetic tape data files and workfiles. In all, seven distinct modules can be generated; six for data files with various record formats, and one for workfiles. These modules supply the logic necessary to support all the logical IOCS imperative macros used with magnetic tape files.

Because some of the LIOCS imperative macros are used with both data files and workfiles and some are used only with one file type, the magnetic tape modules have been combined and flowcharted to illustrate the functions of the individual imperative macros.

DATA FILES

A separate logic module is generated by the MTMOD macro for data files containing records in each of the following formats:

1. Fixed, blocked and unblocked.
2. Variable and spanned, blocked and unblocked.
3. Undefined.

These modules provide the logic to perform the functions of the LIOCS imperative macros CNTRL, GET, PUT, RELSE, and TRUNC.

IF RECFORM=SPNBLK or SPNUNB, a workarea is required.

Note: The last two imperative macros are not used with undefined records.

WORKFILES

The workfile module is generated by the MTMOD macro, when TYPEFLE=WORK is specified, to allow a tape drive to be used for both input and output in one program. Although usually considered as an output file first, the function of a workfile can be switched at any time, thereby allowing records to be written on or read from the same file.

The logic of the workfile module uses the imperative macros READ and WRITE, instead of the GET and PUT used by data files, to transfer records between the device and main storage. The workfile module also differs from the data file modules in that blocked records, switching of I/O areas, and ASCII files are not supported.

In addition to the READ and WRITE imperative macros, the workfile module supports the NOTE, POINTS, POINTR, POINTW, and CHECK imperative macros. CHECK must always be issued after a READ or WRITE if the user wishes to ensure completion of the operation before issuing another instruction. NOTE is used with the POINTR and POINTW macros to position the tape to a

Method: This routine is for nondata operations on the file. For magnetic tape these operations are: backspace, forward space, rewind, write a tapemark, etc.

The routine puts the control operation code in the CCW, causes the control function to be performed, waits for the I/O operation to finish, and return control to the problem program.

MTMOD: FEOV Macro Chart DC

Objective: To cause an EOV to occur before a true EOV condition is reached.

Entry: From an FEOV macro expansion to the label:

IJFFFEO for files containing fixed-length records in EBCDIC.

IJFXFEO for files containing fixed-length records in ASCII.

IJFUFEO for files containing undefined records in EBCDIC.

IJFNFEO for files containing undefined records in ASCII.

IJFVFEO for files containing variable-length records in EBCDIC.

IJFSFEO for files containing spanned records in EBCDIC.

IJFRFEO for files containing variable-length records in ASCII.

Exit: To the problem program.

Method: The FEOV macro causes an end of volume condition to occur before the physical end of volume is reached. On an input file, the trailer labels are not checked and the user specified rewind option is executed. For an output file, if standard labels are specified, a trailer label is created, and the user rewind option is executed.

This routine sets the FEOV switch on in the DTF table and calls $$BOEOV1 to close the file and execute the user specified rewind option. If no rewind option is specified, a message is issued indicating the tape should be rewound and unloaded.

MTMOD: GET Macro Charts DD-DE

Objective: To access a logical record from a magnetic tape data file.

Entry: From a GET macro expansion to the label:

IJFFGET for files containing fixed-length records in EBCDIC.

IJFXGET for files containing fixed-length records in ASCII.

IJFUGET for files containing undefined records in EBCDIC.

IJFNGET for files containing undefined records in ASCII.

IJFVGET for files containing variable-length records in EBCDIC.

IJFSGET for files containing spanned records in EBCDIC.

IJFRGET for files containing variable-length records in ASCII.

Exit: To the problem program or to the EOF address.

Method: The GET routine accesses the next sequential logical record in a magnetic tape data file. The record is moved from the I/O area to the workarea if WORKA=YES is specified in the DTF. Otherwise, the record is available in the I/O area. Overlap occurs if two I/O areas are specified.

The routine performs deblocking, if required (for variable-length records, the user specified blocksize is obtained first), and then goes to a read/write subroutine. The read/write subroutine checks for first time, two I/O areas, checkpoint records, etc, and performs the required I/O operation. If ASCII is specified, the input is first translated to EBCDIC. If a workarea is specified, the GET routine then branches to the workarea subroutine that moves the contents of the I/O area into the workarea. If a workarea has not been specified, the address of the current I/O area is loaded into the user's IOREG. For variable-length, spanned and undefined records, the record size is calculated and made available to the user. The GET routine then returns control to the problem program.

If ERREXT is specified, input errors are returned to the problem program, and greater flexibility is allowed the user in attempting to continue processing.

MTMOD: NOTE Macro, Workfile Chart DF

Objective: To put the tape block count in register 1.

Exit: To the problem program.

Method: The PUT routine writes a record in the next sequential location on the file. If a workarea is specified, the PUT routine moves the contents of the workarea to the I/O area. For blocked (fixed, spanned and variable length) records, if the record fills the I/O area, the PUT routine writes the contents of the I/O area on the file. If unblocked or undefined records are specified, each PUT causes a record to be written on the file. If no workarea is specified, the user must build the physical record in the I/O area.

The PUT routine goes first to a workarea subroutine to determine if a workarea has been specified (for undefined records, the record size is obtained prior to this action). If a workarea is specified, the record is moved from the workarea to the I/O area and any required blocking is performed. If ASCII is specified, the output is also translated to ASCII. The PUT routine next goes to a read/write subroutine which checks for first time, two I/O areas, checkpoint records, etc, and performs the required I/O operation. The address of the next available I/O area is then loaded into the user's IOREG and the PUT routine returns control to the problem program.

If ERREXT is specified, additional errors are returned to the problem program, and greater flexibility is allowed the user in attempting to continue processing.

MTMOD: READ Macro, Workfiles Chart DJ

Objective: To read a physical record from tape.

Entry: From a READ macro expansion IJFWREAD.

Exit: To the problem program.

Method: The READ macro allows the user to access data on an input work file. This macro requires the user to specify the area the record is read into, and also allows the user to read only a portion of the record if he desires. The READ macro allows the user to read his records forward or backward. However, only one direction of reading is allowed for a particular file. Deblocking of blocked records is not performed by the READ macro.

The READ macro routine branches to the READ/WRITE Common Subroutine to test for record format and length and to set up the proper controls. This routine also determines if the entire physical record is to be read for undefined records. If not, only a portion of the record is read. WLR checking is not performed.

Before returning to the READ macro routine, a check of bit 7 of byte 21 in the DTF is made to determine if a POINTS macro was issued before the READ. If the bit is on, indicating the POINTS was issued, the tape is at loadpoint. It is, therefore, necessary to execute a forward-space file operation to bypass any labels and position the tape to the first data record. When this operation is complete, control returns to the READ macro routine.

The READ macro routine issues an EXCP to start a read operation and returns to the problem program.

MTMOD: RELSE Macro Chart DJ

Objective: To skip the remaining records in a block.

Entry: From a RELSE macro expansion to the label:

IJFFREL for files containing fixed-length records in EBCDIC.

IJFXREL for files containing fixed-length records in ASCII.

IJFVREL for files containing variable-length records in EBCDIC.

IJFSREL for files containing spanned records in EBCDIC.

IJFRREL for files containing variable-length records in ASCII.

Exit: To the problem program.

Method: This routine causes the remaining records in the input block to be bypassed. Conditions are set up that cause the next GET instruction for this file to read in a new physical record from the file. The GET instruction makes the first logical record of the new block available to the user.

The RELSE routine sets the end-of-block pointer to 0, and returns control to the problem program.

The RELSE macro can only be issued to an input file.

MTMOD: TRUNC Macro Chart DJ

Objective: To write the current block even if it is not full.

standard volume label, VOL1, and the standard file labels, HDR1, EOF1, and EOV1. (Refer to Volume 1 for the format of these labels.) User labels and nonstandard labels must be processed by the user.

Volume 1 of this set of four logic manuals contains a detailed discussion on the use of tape labels.

## Tape Open, Close and ECF/ECV Routines Charts 01 and 02

Note: Tape files are identified by a single job control statement, // TLBL. This statement combines the information formerly contained in the // TLAB and // VOL statements and allows for simplified standard tape label processing and tape file creation. Note, however, that Job Control still accepts either type statement, and that a mandatory change of job control statements by the user is not required.

The reader is cautioned, where reference is made to the // TLBL job control statement in the text and flowcharts in this manual, that the reference applies to the collective // VOL and // TPLAB statements as well. Also, the decision block (that appears in some of the flowcharts) that contains the question

TLBL INFORMATION IN DTF

determines if the DTF table contains the tape label information included by the DTFMT macros.

## $$BOMT01: Open Input Standard Labels, Forward Charts EA-EB

Objective: To open, in a forward direction, input tape files with standard labels.

Entry:

- From the Open Monitor, $$BOPEN1.
- From $$BOMT02.
- From a message writer phase to the label RELOCATE.

Input: From the label information stored on the SYSRES label information cylinder by Job Control (// TLBL card image) and from the tape currently being processed.

Exit: To $$BOMT07, phase 2 of Open Input Standard Labels, Forward, to store the updated TLBL information.

Method: This routine opens an input tape file in a forward direction. It reads the standard volume label (VOL1) and compares the volume serial number contained in the label to the volume serial number obtained from SYSRES. The routine then reads the standard tape file header label (HDR1) and compares it to corresponding entries specified in the SYSRES label information (// TLBL card).

All differences are noted by messages to the operator. If user labels are detected and the address of the user's label routine (LABADDR) is specified in the DTF table for the file, an exit is made to that routine to read and process the user's labels. After the tapemark signifying the end of the label area is read, $$BOMT07 is fetched.

Upon entry into the $$BOMT01 open phase, a test is made to determine if the phase was entered as a return from a message writer phase. If this is the case, a branch is made to the address specified by the particular response received from the operator for the message printed. This address is represented in the flowcharts by the exit(s) from the decision block following the message block.

The DTF table should contain tape label information. If entry to phase $$BOMT01 was not by return from a message writer phase, a test is made for the presence of this label information. If the DTF contains such information, it must then be used to update the SYSRES label information.

Next, the $$BOMT01 open phase reads the tape for a standard volume label, VOL1 (refer to Figures 26 and 27). If this label is not found, error message 4111A is printed. If the VOL1 label is found, the volume serial number is saved in VOLSAVE. Then the DTF table is checked to determine if the file is already open. If the open indicator is on, byte 36 of the DTF table is examined to determine if the file is defined for processing by physical IOCS (DTFPH). If the file is so defined, the volume sequence number entry in the // TLBL card image is incremented by 1 unless the entry is blank.

If the file is not already open, the volume serial number in the tape VOL1 label is compared to the file serial number (EBCDIC) or set identifier (ASCII) in the // TLBL card image. If the numbers are not the same, error message 4112A is printed. If the response to this message is IGNORE, the // TLBL file serial number (EBCDIC) or set identifier (ASCII) is made equal to the volume serial number. If the response is NEWTAP, indicating that the operator has mounted a new tape, checking of the standard volume label is repeated. If the

2. To the Message Writer, $$BCMTCM.

3. To the Standard Volume Label Rewriter, $$BONVOL.

Method: The $$BCMT03 routine checks and builds labels for output tape files. The routine:

1. Checks the information in the // TLBL card image and inserts the proper data when blank entries are encountered.

2. Reads the standard volume label from the tape and compares it to the information in the // TLBL card image.

3. Determines the availability of the tape from the standard file label.

4. Writes new standard file and user labels as required.

When the required labels have been written on the tape, a tapemark is written and the file is posted open in the DTF table. At this point, the tape is positioned at the beginning of the data area.

Upon entry into the $$BCMT03 phase, a check is made to determine if the phase was entered on a return from a message writer phase. If this is the case, a branch is made to the address determined by the response received from the operator for the message printed. This address is represented in the flowcharts by the exit(s) from the decision block following a message block.

If a $$BCMT03 phase was not entered from a message writer phase, a test is made to determine if tape label information is contained in the DTF table. If so, it is necessary to update the // TLBL card image in main storage with information contained in the DTF table.

This routine then checks the following fields of the // TLBL card image and inserts default values, as necessary, when blank entries are encountered:

- Volume sequence number (EBCDIC) or file section number (ASCII) (default=0001).
- File sequence number (default=0001).
- Generation number (default=0001).
- Version number (default=01 for EBCDIC and 00 for ANSI Labels).
- Expiration date (default-retention period = 0).
- Creation date (today's date).

The block count in the DTF table is set to zero and a check is made for specification of no-rewind (NORWD). If NORWD is not specified, the tape is rewound to loadpoint. If NORWD is specified and the tape is not at loadpoint, a branch is made to location BACKSPC2, Chart LH. If the tape is at loadpoint, a check is made on the density of the tape mounted and assigned. After the density check, the tape is read in search of a standard volume label, VOL1.

If a VOL1 label is not found, error message 4110A is printed. If the operator response is a volume serial number, a new VOL1 label, followed by a dummy record to prevent possible detection of a noise record, is written. A response of NEWTAP indicates that the operator has mounted a new tape and the check of the VOL1 label is repeated.

If a standard volume label is found, the volume serial number is saved in VOLSAVE. If the file is not already posted open in the DTF table, the file serial number in the // TLBL card image is checked. If this field is blank, the volume serial number from the VOL1 label is inserted into the card image. The volume serial number and the // TLBL card file serial number (EBCDIC) or set identifier (ASCII) are compared. Any discrepancy is noted by error message 4112A. If the serial numbers are equal, the next record is read. If any additional volume labels are read, they are bypassed.

When the standard file label, HDR1, is read, the expiration date is checked to determine the availability of the tape. If the tape is available (that is, if the retention period has expired or the expiration date has been reached) or if no HDR1 label was found, the tape is backspaced and $$BCMT04 is fetched.

If the tape is not at loadpoint, the routine at BACKSPC2 is entered to backspace two records and read one record in a forward direction. This sequence of operations is executed in an attempt to locate and retrieve information from the EOF1 label of the preceding file. The file serial number (EBCDIC) or set identifier (ASCII), volume sequence number (EBCDIC) or file section number (ASCII), and the file sequence number +1 from the EOF1 label are inserted into the // TLBL card image and an exit is made from this phase to $$BOMT04 where a new header label is built and written for the file being opened. If no preceding file labels are found, the tape is rewound and unloaded and error message 4120I is printed.

Upon entry to the $$BOMT05 phase, a test is made to determine if the phase was entered on a return from a message writer phase. If this is the case, a branch is made to the address (in register 14) specified by the response received from the operator for the message printed. This address is represented in the flowcharts by the exit(s) from the decision block following the message block.

If the phase was not entered on a return from a message writer phase, the DTF table is examined to determine the direction of the file to be opened and the appropriate action is taken.

1. For nonstandard labeled input files opened in a forward direction, the labels are processed by the user if the address of the user's label routine is specified in the DTF table. Detection of a tapemark indicates the end of the label area, and the file is posted open.

2. For unlabeled input files opened in a forward direction, the tape is read in search of a tapemark. If a tapemark is not read, the tape is backspaced one record to return it to its original position. In either case, the file is posted open in the DTF table.

3. For nonstandard labeled output files opened in a forward direction, the labels are processed by the user if the address of the user's label routine is specified in the DTF table. If the tapemark option is specified in the DTF table, a tapemark is written after the last label. The file is then posted open.

4. For unlabeled output files opened in a forward direction, the tape is read to determine if the tape contains any labels. If a label is detected, error message 4125D is printed and the operator can elect either to ignore the label or to mount a new tape. If the operator chooses to ignore the label and the tape is a 9-track tape, it is backspaced, written, and again backspaced. This sets the mode and erases the label. If the tapemark option is specified in the DTF table, a tapemark is written. The file is then posted open in the DTF table.

5. For nonstandard labeled input files opened in a backward direction, the labels are processed by the user if the user's label routine address is specified in the DTF table and the file is posted open in the DTF table.

6. For unlabeled input files opened in a backward direction, the only check is for the existence of a tapemark. This test positions the tape properly. The file is then posted open in the DTF table.

7. For nonstandard labeled output files opened in a backward direction, the same action as in item 5 occurs.

8. For unlabeled output files opened in a backward direction, the same action as in item 6 occurs.

## $$BOMT06: Open Workfiles, Magnetic Tape Chart EH

Objective: To open a tape workfile.

Entry: From the Open Monitor, $$BOPEN1, to the label TRANSENT.

Input: From the tape being processed.

Exits:

1. To the TES processor, $$BOPEN.

2. To the Standard Volume Label Rewriter, $$BONVCL.

Method: This routine opens standard labeled or unlabeled, input or output, tape workfiles.

If this phase is reentered on a return from a message writer phase, a branch is made to the address specified by the response received from the operator for the message printed. This address is represented in the flowchart by the exit(s) from the decision block that follows a message block.

If this phase was entered from the Open Monitor the tape is rewound to loadpoint, unless the NORWD option is specified in the DTF table, and a read command issued.

If a tapemark is read, the file is immediately considered open and control is returned to the TES processor, $$BOPEN.

If a standard volume label (VOL1) is not read, blanks are inserted into the SDR record, the tape is backspaced, and a tapemark is written. The file is then considered Open and control is returned to the TES processor, $$BOPEN.

If a standard volume label is read, the volume serial number is saved in location VOLSAVE. Additional read commands are executed to bypass any additional volume labels. If a standard file label (HDR1) is read that is neither blank nor all zeros, the expiration date is checked to determine the availability of the tape. If the tape is available or if no HDR1 label is read,

information for the file into the open table at the end of the logical transient area. This information is passed to the required tape EOF/EOV phase fetched by the EOF/EOV Monitor. The file open bit in the PUB2 table is then turned off.

## $$BCMT01: Tape Close, EOF/ECV Input Forward Chart FA

Objective: To close an input tape file reading in a forward direction.

Entry: From the EOF/ECV Monitor, $$BCEOV1, to the label TRANSENT.

Input: From the SYSRES label information cylinder (// TLBL card image) and from the tape currently being processed.

Exits:

1. To the user's EOF routine if specified in the DTF table.
2. To phase $$BCMT02 for an EOV condition to switch to an alternate tape.
3. To CANCEL if an error condition occurs.

Method: This routine is called by the EOF/EOV Monitor. The routine reads and processes the standard trailer label. An exit and return are provided for processing user labels if LABADDR is specified. It checks to determine if an ECF or ECV condition is present. If an EOV condition exists, $$BCMT02 (alternate switching routine) is called. If an ECF condition exists, control passes to the user's ECF routine by branching to the user's EOFADDR.

Upon entry into the $$BCMT01 phase, a test determines if the phase was entered on a return from a message writer phase. If this is the case, a branch is made to the address specified (in register 14) by the response received from the operator for the message printed. This address is represented, in the flowcharts, by the exit(s) from the decision block following a message block.

If the phase was not entered on a return from a message writer phase, a check of the DTF table (byte 36) then determines if the file is being closed as a result of an FEOV (Force-End-of-Volume) macro being issued. If bit 3 of byte 36 is on, an FECV macro was issued. The tape is rewound, unless the NORWD option is specified in the DTF. No label checking is performed. Phase $$BCMT02 is then fetched to determine if an alternate tape is available.

If an FECV macro was not issued, the $$BCMT01 phase reads the tape in search of standard trailer labels if such are specified in the DTF table. Only the block count contained in these labels is checked against the block count contained in the DTF table. Any discrepancy is noted by error message 4131D. User labels are bypassed unless the user furnishes the name of his label processing routine. This phase checks DTF byte 16, bit 3, to determine if American National Standards CCBCL has been specified. If it has been specified, a first-time switch is set before yielding control to the user's label processing routine. The tape is rewound, unless the NORWD option is specified, and a check determines the type of standard trailer label read (that is, ECV1 or ECF1). An ECF1 label sets the ECF-ECV switch, bit 4 of byte 36 in the DTF table. If the switch is turned on, the file is posted closed in the DTF table and an exit is made to user's end-of-file address, ECFADDR. If the switch is off, indicating that an end-of-volume condition exists, a test determines if the file is a CCBCL file that must be rewound and unloaded. If so, the rewind-unload operation is executed. Phase $$BCMT02 is fetched to determine if an alternate tape is available.

If nonstandard labels are specified, exit is made to the user's label routine (LABADDR) to process these labels. If no labels are specified or if the user specifies the end-of-file in the user label routine, exit is made to the user's end-of-file routine, ECFADDR. If the user specifies end-of-volume, phase $$BCMT02 is fetched to determine the availability of an alternate tape.

If the user specifies neither end of volume nor end of file, message 4130A is printed and the operator can specify either ECF or ECV. If the operator's reply is ECF, an exit is made to the user's end-of-file routine, ECFADDR. If the operator's reply is EOV, a test determines if the file is a CCBCL file that must be rewound and unloaded. If so, the rewind-unload operation is executed. Phase $$BCMT01 then fetches phase $$BCMT02 to determine the availability of an alternate tape.

## $$BCMT02: Tape Close, Alternate Switching for ECV Chart FB

Objective: To switch tape drives when EOV is sensed.

Entry: From $$BCMT01 or $$BCMT04.

Input: From the trailer label of the active file and from the SYSRES label information (// TLBL card image) in main storage.

is complete, or if the address of the user's label routine is not specified, the $$BCMT03 phase proceeds as described for unlabeled files.

If standard labels are specified in the DTF table, the tape is read in search of the standard file header label, HDR1.

All user header labels are processed by the user's label routine (LABADDR), or they are bypassed if the address of the user's label routine is not specified. All file header labels preceding (in a backward direction) the standard HDR1 label are bypassed. When the HDR1 label is read, only the block count contained in the label is checked against the block count contained in the DTF table. Any discrepancy is noted by error message 4131D. After checking the block count, the $$BCMT03 phase proceeds as described for unlabeled files.

## $$BCMT04: Tape Close, EOV Output Forward Chart FD

Objective: To close an output tape file reading in a forward direction.

Entry: From the EOF/EOV monitor, $$BCEOV1, to the label TRANSENT.

Input: From the SYSRES label information cylinder (// TLBL card image) and from the tape being processed.

Exit: To phase $$BCMT02 to determine the availability of an alternate assignment.

Method: The $$BCMT04 phase closes the current active file by writing trailer labels and a tapemark for EBCDIC files and two tapemarks for ASCII files. It then fetches the alternate switching phase, $$BCMT02, to open the next tape reel.

Upon entry to the $$BCMT04 phase, a check determines the type of labels specified in the DTF table. If standard labels are specified, the version level of the DTF table is verified by examining bit 5 of byte 31 in the table. If the bit is on, the table was generated by a Version 2.1 onward DTFMT macro. It is then necessary to modify the corresponding entries in the // TLBL card image in main storage with the file serial number (EBCDIC) or set identifier (ASCII), volume sequence number (EBCDIC) or file section number (ASCII), and file sequence numbers from the DTF table. After the // TLBL card image is modified with the DTF table information, the following card entries are checked and the proper default value inserted if any are blank:

- File identification (FILEID) number (default = filename from the DTF table).
- Generation number (default = 0001).
- Version number (default = 01 for EBCDIC files and 00 for ASCII files).
- Expiration date (default = retention period set to 0 days).

Note: If expiration date is not absolute, the expiration date is calculated by adding the retention period to today's date.

After updating the // TLBL card image or if bit 5 of byte 31 in the DTF table is not on (indicating that the DTF table does not contain tape label information and updating is bypassed), the volume sequence number (EBCDIC) or file section number (ASCII) in the // TLBL card image is incremented by 1 and the entire updated card image is used to build a standard EOV1 file label.

The updated // TLBL card image is then written back onto the label information cylinder of SYSRES. If the DTF table does contain tape label information, it is also updated with the information from the // TLBL card image. The EOV1 label is then written on the tape.

An exit is made to the user's label routine (LABADDR) to process and write user labels as specified. After the user's label routine is complete, a tapemark is written to indicate the end of the label area. The tape is rewound, rewound and unloaded, or left positioned, as specified by the user in the DTF table. The block count in the DTF table is reset to zero, and phase $$BCMT02 is fetched to determine the availability of an alternate assignment.

If nonstandard labels are specified in the DTF table, an exit is made to the user's label routine to process these labels and the $$BCMT04 phase is concluded in the manner just described.

## $$BCMT05: Close Standard, Nonstandard, and Unlabeled Files, All Types Except Workfiles Charts FE-FF

Objective: To close a tape input or output file, reading in the forward or backward direction.

Entry: From the Close Monitor, $$BCLOSE, to the label TRANSENT, or from $$BCMT03.

Input: From the SYSRES label information cylinder (// TLBL card image) and from the tape being processed.

Method: This routine closes any magnetic tape workfile. It checks to determine if the last tape operation was a read or a write. If write, a tapemark is written, and the rewind option is tested. If read, the rewind option is tested. In either case, if NORWD was specified, the rewind function is bypassed. If NORWD was not specified, the tape is either rewound to the loadpoint, or, rewound and unloaded, as the user requires. The block count in the DTF table is reset to zero, and the Close Monitor is fetched.

## $$BCMT07: Tape Close, Alternate Switching for System Units Chart FH

Objective: To terminate and switch to an alternate drive when an EOV condition is encountered on an output tape file assigned to either SYSLST or SYSPCH.

Entry: From the EOF/EOV Monitor, $$BCEOV1, or from an output CP module.

Input: From the Supervisor I/O tables. Refer to DOS/VS Supervisor, SY33-8551.

Exits:

1. To the message writer phase, $$BMSGWR, if an alternate drive is not available.
2. To the Job Control open phase, $$BJCOPT, to open the alternate assignment.
3. To the logical IOCS module if a unit exception occurs while writing a tapemark.

Method: This phase closes the active tape output file by writing a tapemark and by rewinding and unloading the tape. The alternate device is determined, if specified, or an exit is made to the message writer phase, $$BMSGWR. If an alternate is not specified, or not available, the operator can mount a new tape on the same drive and continue processing.

Upon entry into the $$BCMT07, a test determines if the no rewind option is specified in job control switch byte JBCSW2 (byte 58 in the communications region). If the option is not specified, a tapemark is written and a test determines if the unit exception bit is on in the CCB (bit 7 of byte 4). If the bit is not on, alternate switching is not required. In this case, the tape is backspaced over the last tapemark written and control is returned to the logical IOCS module via an SVC 11.

If the unit exception bit in the CCB is on, alternate switching is required. The tape is rewound and unloaded and an SVC 22 is issued to seize the system (that is, to disable multiprogramming operation).

Note: Additional information concerning SVC 22 and the functions of the LUB, PUB, and JIB tables is found in DOS/VS Supervisor, SY33-8551.

The address of the Logical Unit Block (LUB) entry is calculated and the pointer to the Physical Unit Block (PUB) entry is obtained from the first byte of the LUB entry. The address of the PUB assigned to the LUB is calculated, and the PUB mode byte (byte 5) is saved for later use.

The second byte of the LUB entry contains a pointer to an entry in the Job Information Block (JIB) table if any JIB's are specified (chained) for the LUB. If there is an alternate PUB assigned, it is stored in a JIB. The first JIB in the chain is examined for a stored alternate PUB. If the JIB does not contain an alternate PUB, the next JIB in the chain is checked. When a stored alternate assignment is found, the pointer to that PUB is obtained from the JIB and inserted into the first byte of the LUB. Any remaining JIB's are scanned and updated. The standard PUB assignment is stored in the last JIB in the chain.

After the alternate assignment is checked for availability, the mode byte saved from the standard PUB is inserted into the alternate PUB. The $$BCMT07 phase concludes by issuing an SVC 22 to release the system (restore multiprogramming operation), and phase $$BJCOPT is fetched to open the alternate assignment.

If an alternate assignment is not found, or not available, initialization to print message 4121A takes place.

## $$BCMT0M and $$BOMT0W: Tape Open Message Writers Chart FK

Note: The tape open message writer phases $$BOMT0M and $$BOMT0W perform the same function. The two phases differ only in the messages printed and the calling phases.

Objective: To print messages to the operator and analyze the operator's response to a particular message.

Entries:

1. Phase $$BOMT0M--from tape open phase $$BOMT02, $$BOMT03, $$BOMT04, $$BOMT06.

## $$BMSGWR: Tape Open/Close Message Writer Chart FL

Objective: To print messages to the operator and analyze the operator's response to a particular message.

Entry: From tape Open/Close phases:

$$BOMT05

$$BCMT01

$$BCMT02

$$BCMT03

$$BCMT07

Input: Pointer, received from the calling phase, to indicate the desired message.

Exits:

1. To the calling phase if a legal response to the message is received, or
2. To CANCEL if no console is available, or if the response is END (end-of-communication) or illegal.
3. To $$BJCCPT if entered from tape close phase $$BCMT07.

Messages:

4117D NO TM FOUND ON READBK

4119A FILE UNEXPIRED

4121A NO ALTERN DRIVE ASSGN

4122I EOV ENCOUNTERED

4125D VOL1 LBL FOUND

4126I EOV ENCOUNTERED

4127A EOV WHILE WRITING ECF

4130A ECF OR EOV INQUIRY

4131D BLOCK COUNT ERROR

4140A NO ALTERN DRIVE ASSGN

Note: Messages not requiring printing of label information may also be passed to the $$BMSGWR phase.

Method: The $$BMSGWR routine is used for message writing and response interrogation.

The calling routine passes the following information to the $$BMSGWR routine:

- Register 0. The number of the desired message.
- Register 12. The RETRY reply return address.
- Register 13. The NEWTAP or EOV reply return address.
- Register 14. The IGNORE or EOF reply, or the information type message return address.
- The last two bytes in the name of the phase returned to.

Each message contained in this phase consists of four sections:

1. Option byte. Indicates action to be taken after printing the message or valid response(s).

   Bit 0 = EOV or ECF

   Bit 1 = IGNORE

   Bit 2 = NEWTAP

   Bit 3 = Double message

   Bit 4 = RETRY

   Bit 5 = Information message (no response required)

   Bit 6 = System unit possible and CANCEL not legal response

   Bit 7 = Leave DTF blank.

2. Displacement byte. Contains displacement, in bytes, from the start of the label to the field(s) to be printed.
3. Length byte. Contains the length, in bytes, of the label information to be printed.
4. Message.

The $$BMSGWR message writer phase assembles and writes the required message and receives and analyzes the response received from the operator. The phase then passes control to the proper open phase or to the cancel routine. If control is returned to an open phase, bit 4 of byte 89 in the communications region is set to indicate to the open phase that entry is made on a return from a message writer phase.

Sequentially-organized DASD (SD) files are contained on DASD devices, and are processed by the Sequential Access Method. These files, defined by the DTFSD macro, are either input or output data files, or work files.

A sequential DASD file contains DASD records that are processed with a beginning DASD address and that continue in order through the records on successive tracks, cylinders, and volumes to the ending address.

A sequential DASD file is contained within one or more sets of limits called extents. These extents are specified by the user with job control cards (// DLBL and/or // EXTENT). If the logical file consists of more than one extent, each extent is accessed in the sequence specified by the user. The records within each extent must be adjacent and contained within one volume (pack or cell). The extents need not be adjacent, and they may be on more than one volume.

The data handling logic modules for files defined for logical IOCS by the DTFSD macro are provided by the associated module generation macro, SDMODxx, where the xx is determined by the record format and function of the file.

Sequential DASD files are opened and closed by logical transient routines that are fetched by the open and close monitors (refer to Volume 1). The open routines provide procedures for checking each file before any records are processed. The close routines provide procedures for terminating each file after all records are processed.

Sequential DASD files can also be defined for physical IOCS if the user intends to use physical IOCS macros, such as EXCP, WAIT, etc. These files are defined by a DTFPH macro.

In addition, sequential DASD files can be defined by the device independent macros, DTFDI and DTFCP. These files are described in the section "Device Independent Files".

## STORAGE AREAS

### INPUT/OUTPUT AREAS

The logical IOCS GET-PUT macro instructions allow the programmer to use one or two I/O areas and process records either in a work area or in an I/O area.

When blocked records are to be processed in an I/O area with no work area specified, (or when unblocked records are to be processed in two I/O areas, with no work area specified) the DTFSD macro instruction defines the register IOREG. Logical IOCS uses this register to specify the address of the logical record that is currently available for processing by the problem program.

If variable-length blocked records are built directly in an output area(s) with no work area specified, the DTFSD macro instruction specifies another register, VARBLD. This register provides the programmer with the remaining space in the output area after each PUT instruction has been issued.

### MODULE SAVE AREAS

If the RDONLY=YES parameter is included in the module generation macro, the module is reentrant and must never be modified by the problem program. Each DTF referencing the module must be associated with a 72-byte, doubleword aligned save area which is used by the module during execution. The address of the save area is passed to the module in register 13.

If the module is to be shared by DTFs in different tasks, the module must be made reentrant. This is done by associating a unique save area with each DTF.

In sequential DASD, the save area contains user general registers, module general registers, switches and other information needed by the module. Figures 30 through 39 illustrate the format of the save area for each logic module.

SDMCDFO - Fixed-Length Output

| Byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Displ. DEC 0 / HEX 0 | User Register 9 | | | | User Register 10 | | | |
| 8 / 8 | User Register 11 | | | | User Register 12 | | | |
| 10 / 10 | User Register 13 | | | | User Register 14 | | | |
| 24 / 18 | Module Register 15 | | | | Module Register 0 | | | |
| 32 / 20 | Module Register 1 | | | ** | X'FF'* | | | |
| 40 / 28 | Module Register 10 | | | | Module Register 11 | | | |
| 48 / 30 | Module Register 12 | | | | Module Register 13 | | | |
| 56 / 38 | Count Field of Previous Record | | | | H | H | R | Previous I/O Area Address |
| 64 / 40 | Previous I/O Area Address (continued) | | Current I/O Area Address | | | | | |

*Indicates to OPEN that no more DTFs are to be opened.

**If ERREXT=YES, bytes 32-39 contain the parameter list that includes the address of the DTF and the storage address of the block in error.

Figure 31. SDMCDFO save area.

SDMCDVO - Variable-Length Output

| Byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Displ. DEC HEX 0 0 | User Register 6 | | | | User Register 7 | | | |
| 8 8 | User Register 8 | | | | User Register 9 | | | |
| 16 10 | User Register 10 | | | | User Register 11 | | | |
| 24 18 | User Register 12 | | | | User Register 13 | | | |
| 32 20 | Module Registers 6, 7, or 10 | | | | Module Registers 7 or 8 | | | |
| 40 28 | Module Registers 8 or 14 | | | | Module Register 0 or 15 | | | |
| 48 30 | Module Register 0 | | | | Module Register 1 | | | ** |
| | Module Register 1 | | | | X'FF'* | | | |
| 56 38 | Not used | | | | | | | |
| 64 40 | Not used | | | | | | | |

*Indicates to OPEN that no more DTFs are to be opened.

**If ERREXT=YES, bytes 52-59 contain the parameter list that includes the address of the DTF and the storage address of the block in error.

Figure 34. SDMCDVO save area.

SDMCDUO - Undefined Output

| Byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Displ. DEC HEX | User Register 7 | | | | User Register 8 | | | |
| 0 0 | | | | | | | | |
| 8 8 | User Register 9 | | | | User Register 10 | | | |
| 16 10 | User Register 11 | | | | User Register 12 | | | |
| 24 18 | User Register 13 | | | | Module Register 12 | | | |
| 32 20 | Module Register 13 | | | | Module Register 14 | | | |
| 40 28 | Module Register 15 | | | | Module Register 0 | | | |
| 48 30 | Module Register 1 ** | | | | X'FF'* | | | |
| 56 38 | Not used | | | | | | | |
| 64 40 | Not used | | | | | | | |

*Indicates to CPEN that no more DTFs are to be opened.

**If ERREXT=YES, bytes 45-55 contain the parameter list that includes the address of the DTF and the storage address of the block in error.

Figure 37. SDMCDUO save area.

SDMCDUU - Undefined Input with Update

| Byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Displ. DEC HEX | User Register 8 | | | | User Register 9 | | | |
| 0 0 | | | | | | | | |
| 8 8 | User Register 10 | | | | User Register 11 | | | |
| 16 10 | User Register 12 | | | | User Register 13 | | | |
| 24 18 | Module Register 8 | | | | Module Register 9 | | | |
| 32 20 | Module Register 12 | | | | Module Register 13 | | | |
| 40 28 | Module Register 14 | | | | Module Register 15 | | | |
| 48 30 | Module Register 0 | | | | Module Register 1 ** | | | |
| 56 38 | X'FF'* | Last Record Switch | | | Read Data CCW (Bytes 0-3) | | | |
| 64 40 | Read Data CCW (Bytes 4-7) | | | | | | | |

*Indicates to CPEN that no more DTFs are to be opened.

**If ERREXT=YES, bytes 52-59 contain the parameter list that includes the address of the DTF and the storage address of the block in error.

Figure 38. SDMCDUU save area.

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| &Filename | 0-15 (0-F) | | Command Control Block (CCB). |
| | 16 (10) | 0 | 1 = Dequeue old volume extents. |
| | | 1 | 1 = Dummy OPEN to obtain extents from label track. |
| | | 2 | 1 = File assigned 'IGN' (CCBOL). |
| | | 3 | 1 = Track hold option specified. |
| | | 4 | 1 = DTF relocated by OPENR. |
| | | 5 | 1 = Input trailer labels to be processed at close time (CCBOL only). |
| | | 6 | 1 = Spanned processing. |
| | | 7 | 1 = CCBOL end-of-extent option specified. |
| | 17-19 (11-13) | | Address of logic module. |
| | 20 (14) | | DTF type for OPEN/CLOSE (X'20' = sequential access DASD files). |
| | 21 (15) | 0 | 1 = 2321 (Version 1/2 only). |
| | | 1 | 1 = Blocked file. |
| | | 2 | 1 = Work file. |
| | | 3 | 1 = Work area specified. |
| | | 4 | 1 = Not a Version 1 type table. |
| | | 5 | 1 = Open, 0 = Closed. |
| | | 6 | 1 = Input, 0 = Output. |
| | | 7 | 1 = User labels specified. |
| | 22-28 (16-1C) | | Filename (DTF Name). |
| | 29 (1D) | | Device Type Code:<br>X'00' = 2311<br>X'01' = 2314, 2319<br>X'02' = 2321<br>X'04' = 3330.<br>X'08' = 3340 general<br>X'09' = 3340 35MB<br>X'0A' = 3340 70MB. |
| | | | Note: In previous versions, last byte of filename contains device type code. |
| | 30-35 (1E-23) | | Address of Format 1 label in VTOC (BBCCHHR). |
| | 36-37 (24-25) | | Volume sequence number. |
| | 38 (26) | | Open communications byte. |
| | | | Input File |
| | | 0 | 1 = No more extents. |
| | | 1 | 1 = Update file. |
| | | 2 | 1 = Process trailer labels. |
| | | 3 | 1 = Exit to user's EOF routine. |
| | | 4 | 1 = Next extent on new volume. |
| | | 5 | 1 = Return to close routine. |
| | | 6 | 1 = Process header labels. |
| | | 7 | 1 = Extent switch. |

Figure 40. DTFSD: Data files (1 of 12).

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| | | | Fixed Length Record Modules: |
| | | 0 | Not first entry after Open (INPUT and UPDATE). |
| | | | Not first write after Open (OUTPUT). |
| | | 1 | Short record (INPUT and UPDATE without truncation). |
| | | 2 | Partial block written (OUTPUT). |
| | | 3 | ERROPT=SKIP (INPUT). |
| | | | TRUNCS=YES (OUTPUT). |
| | | 4 | End-of-file record written (OUTPUT). |
| | | | End of extent (UPDATE). |
| | | 5 | Truncation not specified (used by OPEN routines). |
| | | 6 | Write block of records (UPDATE). |
| | | 7 | End of file (UPDATE). |
| | | | |
| | | | Variable Length Record Modules: |
| | | 0 | Not first entry after OPEN (INPUT and UPDATE). |
| | | | Write record (OUTPUT). |
| | | 1 | Wrong length record (INPUT). |
| | | | TRUNCS=YES (OUTPUT). |
| | | | Second GET operation performed (UPDATE). |
| | | 2 | Return to close routine (OUTPUT). |
| | | | Update specified (UPDATE). |
| | | 3 | Not first entry after OPEN (OUTPUT). |
| | | 4 | New extent required by CLOSE. |
| | | 5 | Capacity of I/O area exceeded (OUTPUT). |
| | | | Second GET required (UPDATE). |
| | | 6 | Not first read (INPUT). |
| | | | Second GET issued (UPDATE). |
| | | 7 | Unnecessary to read (INPUT). |
| | | | Track capacity exceeded (OUTPUT). |
| | | | Save record count (UPDATE). |
| | | | |
| | | | Undefined Length Record Modules: |
| | | 0 | Not first entry after OPEN (ALL modules). |
| | | 1 | Save record count (UPDATE). |
| | | 2 | Return to close routine (OUTPUT). |
| | | 3 | Second GET issued (UPDATE). |
| | | 4 | Not used. |
| | | 5 | PUT command issued (UPDATE). |
| | | 6 | End of file reached (UPDATE). |
| | | 7 | Multi-track operation (UPDATE). |
| | | | |
| | 74-75 (4A-4B) | | Block size minus one. |
| | | | |
| | 76-80 (4C-50) | | CCHHR = Extent lower limit and record number. Field is used as a search argument bucket by the logic modules. |
| | | | |
| | 81 (51) | 1 | 1 = FEOVD has been issued (output only) |
| | | | |
| | 81-83 (51-53) | | Address of user wrong-length record routine if input file. |
| | | | Track capacity counter if output file |

Figure 40. DTFSD: Data files (3 of 12).

The following section is added tc the DTFSD table for fixed-length reccrd input files.

| DTF Assembly Label | Bytes | Bits | Functicn |
|---|---|---|---|
| If RECFORM=FIXBLK and TRUNCS=YES: | | | |
| | 136-143 (88-8F) | | Read Ccunt CCW. |
| | 144-151 (90-97) | | Ccunt field input area. |
| | If CONTROL=YES, the following secticn is added: | | |
| | 152-167 (98-A7) | | Ccntrcl CCB. |
| | 168-175 (A8-AF) | | Ccntrcl CCW. |
| If UPDATE=YES: | | | |
| | 136-143 (88-8F) | | Search ID Equal CCW. |
| | 144-151 (90-97) | | TIC CCW. |
| | 152-159 (98-9F) | | Verify CCW. |
| | If CONTROL=YES, the following secticn is added: | | |
| | 160-175 (A0-AF) | | Ccntrcl CCB. |
| | 176-183 (B0-B7) | | Ccntrcl CCW. |

Figure 40. DTFSD: Data files (5 cf 12).

The following section is added tc the DTFSD table for fixed-length reccrd output files.

| DTF Assembly Label | Bytes | Bits | Functicn |
|---|---|---|---|
| | 136-143 (88-8F) | | Search ID Equal CCW. |
| | 144-151 (90-97) | | TIC CCW. |
| | 152-159 (98-9F) | | Verify CCW. |
| | If CONTROL is nct specified: | | |
| | 160-163 (A0-A3) | | End-cf-extent rcutine address (primarily used by CCBOL compiler). |
| | If CONTROL=YES: | | |
| | 160-175 (A0-AF) | | Ccntrcl CCB. |
| | 176-183 (B0-B7) | | Ccntrcl CCW. |
| | 184-187 (B8-BB) | | End-cf-extent routine address (primarily used by CCBCL ccmpiler). |

Figure 40. DTFSD: Data files (7 cf 12).

The following section is added to the DTFSD table for variable length record and undefined length record input files

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| If UPDATE=YES: | | | |
| | 144-151 (90-97) | | Search ID Equal CCW. |
| | 152-159 (98-9F) | | TIC CCW. |
| | 160-167 (A0-A7) | | Verify CCW. |
| | 168-175 (A8-AF) | | Count field input area. |
| | 176-183 (B0-B7) | | Count field save area if one I/O area. |
| | 184-191 (B8-BF) | | Count field save area if two I/O areas. |
| | If CONTROL=YES:* | | |
| | 192-207 (C0-CF) | | Control CCB. |
| | 208-215 (D0-D7) | | Control CCW. |

The following section is added to the DTFSD table for variable length spanned record update files.

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| | 216-219 (D8-DB) | | Logical record length. |
| | 220-223 (DC-DF) | | RX type instruction. |
| | 224 (E0) | 0 | Not used. |
| | | 1 | 1 = Skip segment. |
| | | 2 | 1 = Spanned first time. |
| | | 3 | 1 = Null segment. |
| | | 4 | 1 = Spanned PUT return. |
| | | 5 | Not used. |
| | | 6 | Not used. |
| | | 7 | 1 = No update |
| | 225-227 (E1-E3) | | Pointer in logical record. |
| | 228-235 (E4-EB) | | Count save area. |
| | 236-239 (EC-EF) | | Extent status save area. |

*These bytes are always generated when spanned processing is specified.

Figure 40. DTFSD: Data files (9 of 12).

The following section is added tc the DTFSD table for variable length spanned record output files.

| DTF Assembly Label | Bytes | Bits | Functicn |
|---|---|---|---|
| | 200-203 (C8-CB) | | Lcgical record length. |
| | 204-207 (CC-CF) | | RX type instructicn. |
| | 208 (D0) | 0 | Nct used. |
| | | 1 | Nct used. |
| | | 2 | 1 = Leading segrert. |
| | | 3 | 1 = Output blcck truncated. |
| | | 4 | 1 = Erd of track. |
| | | 5 | 1 = Track truncated. |
| | | 6 | 1 = Save count. |
| | | 7 | 1 = Vclumes spanned. |
| | 209-211 (D1-D3) | | Pcinter in logical record. |
| | 212-219 (D4-DB) | | Ccunt save area. |
| | 220-223 (DC-DF) | | Extent status save area. |

*These bytes are always generated when spanned processing is specified.

Figure 40. DTFSD: Data files (11 of 12).

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| &Filename | 0-15 (0-F) | | Command Control Block (CCB). |
| | 16 (10) | 0-1 | Not used |
| | | 2 | 1 = File assigned 'IGN' (CCBCL). |
| | | 3 | 1 = Track hold option specified. |
| | | 4 | 1 = DTF relocated by CPENR. |
| | | 5-7 | Not used. |
| | 17-19 (11-13) | | Address of logic module. |
| | 20 (14) | | DTF type for CPEN/CLOSE (X'20' = sequential access DASD files). |
| | 21 (15) | 0 | 0 = disk device. |
| | | 1 | 1 = CLOSE macro is not to delete Format 1 and Format 3 file labels. |
| | | 2 | 1 = Work file. |
| | | 3 | Type of open: 1 = Point, 0 = Normal. |
| | | 4 | 1 = Routine entered from close routine. |
| | | 5 | 1 = File opened.<br>0 = File closed. |
| | | 6 | Not used. |
| | | 7 | 1 = Reentry to close routine. |
| | 22-28 (16-1C) | | Filename (DTF Name). |
| | 29 (1D) | | Device Type Code:<br>X'00' = 2311<br>X'01' = 2314, 2319<br>X'04' = 3330.<br>X'08' = 3340 general<br>X'09' = 3340 35MB<br>X'0A' = 3340 70MB.<br><br>NOTE: In previous versions, last byte of filename contains device type code. |
| | 30-31 (1E-1F) | | Track capacity counter. |
| | 32-35 (20-23) | | Address of Format 1 label in VTOC (CCHR). |
| | 36 (24) | | Extent sequence number. |
| | 37 (25) | | Open Communications Byte. |
| | | 0-2 | Not used. |
| | | 3 | 1 = Symbolic unit in DTF. |
| | | 4 | 1 = Next extent or new volume. |
| | | 5 | 1 = Extent opened. |
| | | 6-7 | Not used. |
| | 38 (26) | | Lower head limit. |
| | 39 (27) | | Upper head limit. |

Figure 41. DTFSD: Workfiles (1 of 3).

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| | 61 (3D) | | Switch byte used by logic module. |
| | | 0 | 1 = First write entry indicator. |
| | | 1 | 1 = Write update indicator. |
| | | 2 | 1 = POINTS macro issued. |
| | | 3 | Not first record on track. (RECFORM=UNDEF). |
| | | 4 | 1 = Track upper limit reached. |
| | | 5 | Not used. |
| | | 6 | 1 = Check after read/write. |
| | | 7 | Not used. |
| | 62-63 (3E-3F) | | Maximum record length. |
| | 64 (40) | | Verify chain bit. |
| | 65-67 (41-43) | | Address of user's ECF routine. |
| | 68 (44) | | Logical indicators. |
| | | 0 | 1 = ERROPT = address. |
| | | 1 | 1 = ERROPT = IGNORE. |
| | | 2 | 1 = Fixed-length unblocked records. |
| | | 3 | 1 = Verify specified. |
| | | 4 | 1 = ERROPT = SKIP. |
| | | 5 | 1 = Reread after read error. |
| | | 6 | Not used. |
| | | 7 | Not used. |
| | 69-71 (45-47) | | Address of user read/write error routine. |
| | 72-143 (48-8F) | | CCW chain for work files (see Figure 24). |
| | 144-151 (90-97) | | Input area for Verify CCW and Read Count CCW. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 41. DTFSD: Workfiles (3 of 3).

DTFPH MACRO

When physical IOCS macro instructions are used to process a sequential DASD file with standard labels, and the user wishes to have the labels checked, the file must be defined by a DTFPH (Define The File for PHysical IOCS) macro. To define a sequentially-organized DASD file in this manner, the parameters specified in the operand of the DTFPH macro instruction must include DEVICE=2311/2314/3330/3340/2321 and MOUNTED=SINGLE. Figure 42 illustrates the DTF table generated to define the file for physical IOCS.

| Bytes | Bits | Function |
|---|---|---|
| 39 (27) | | Sequence number of current EXTENT being opened. |
| 40 (28) | | Sequence number of last EXTENT opened (not a consecle EXTENT entry). |
| 41-43 (29-2B) | | Address of user's label routine. |
| 44-47 (2C-2F) | | Address of IOAREA1. |
| 48-51 (30-33) | | CCHH address of user's label track. Initially X'80000000'. |
| 52-53 (34-35) | | Lower head limit (HH) X'0000' if type 1; X'00nn' if type 128 (n = head limit). |
| 54-57 (36-39) | | EXTENT upper limit (CCHH). |
| 58-59 (3A/3B) | | BB seek address:<br>=X'0000' if disk device.<br>=X'00nn' if 2321 where 'nn' = bin number. |
| 60-63 (3C-3F) | | EXTENT lower limit (CCHH). |
| 64 (40) | | Record number. 1 = Input, 0 = Output. |
| 65-67 (41-43) | | Not used. |
| 68-71 (44-47) | | CCHH control bucket.<br>CCHH = X'1309C413' if 2321 - type 1.<br>CCHH = X'00C80009' if 2311, X'00C80013' if 2314 or 2319, or X'01940012' if 3330 - type 1.<br>CCHH = X'130904nn' if 2321 - type 128.<br>CCHH = X'00C8C0nn' if 2311, 2314, 2319 - type 128<br>CCHH = X'019400nn' if 3330 - type 128<br>CCHH = X'015CC00B' if 334C 35MB<br>CCHH = X'02B8000B' if 334C 70MB<br>where nn = current upper head number. |
| 72 (48) | | Record number. |
| 73 (49) | | Not used. |
| 74-75 (4A-4B) | | Not used. |
| 76-80 (4C-50) | | CCHHR bucket = extent lower limit and record number. |
| 81-83 (51-53) | | Not used. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 42. DTFPH: Sequential disk (2 of 2).

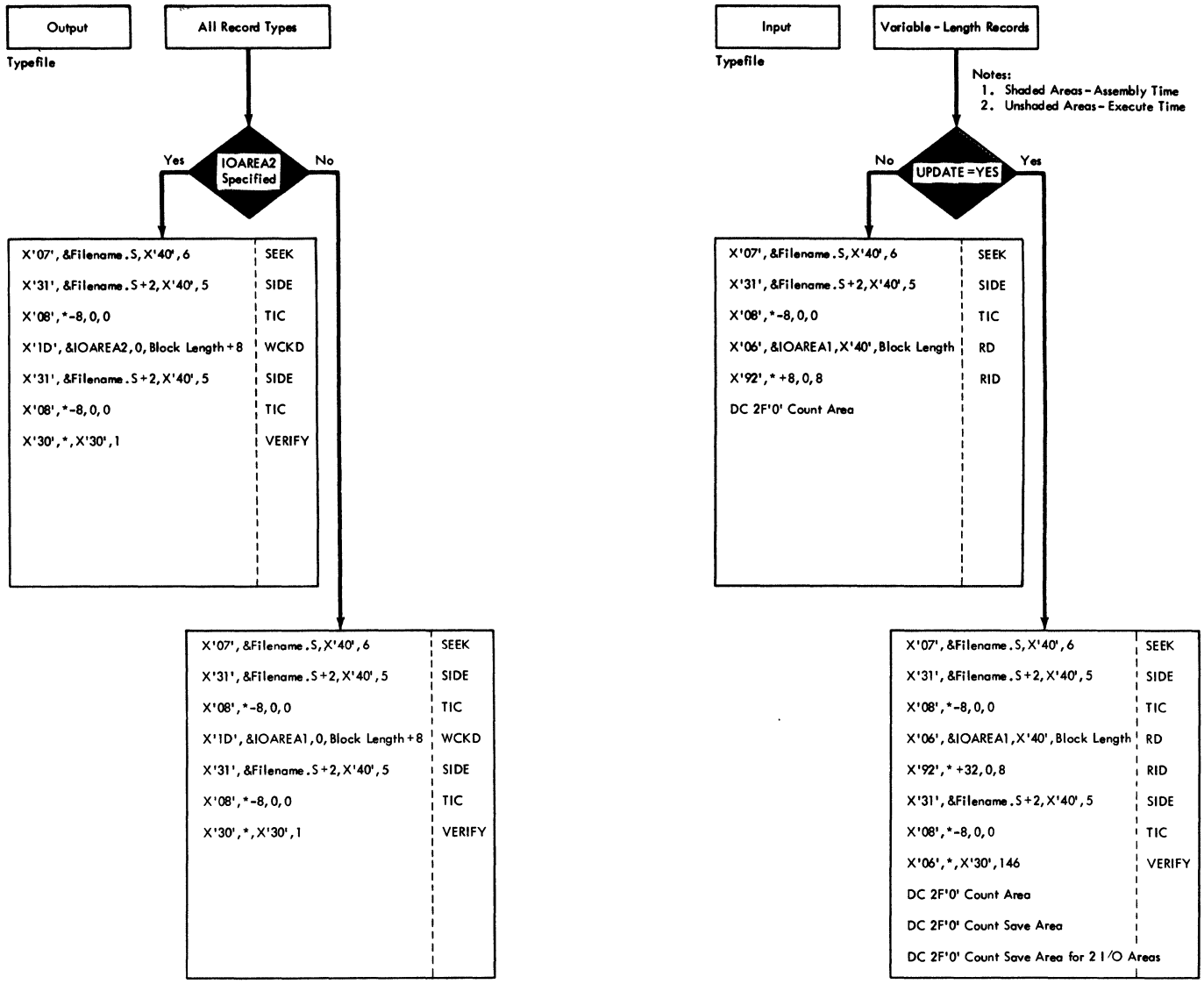



Figure 43. DTFSD channel programs (2 of 3)

If the ASSGN IGN function is to be used for an SD LIOCS file, it is the user's responsibility to test for the IGNORE indicator posted in the DTF after an OPEN has been issued to the file. If this indicator is on, the user should not issue I/O to that file.

FIXED-LENGTH RECORD MODULES

The basic modules for accessing fixed-length records are:

- SDMODFI - Fixed-length input records.
- SDMODFO - Fixed-length output records.
- SDMODFU - Fixed-length input records for update.

Each of these modules actually consists of two modules (depending on whether TRUNC=YES is specified as an SDMODFx parameter) to provide a total of six different modules for sequential DASD fixed-length record formats.

The modules are generalized routines that work with one or more unique DTF tables to perform their various functions. Each module can function using:

- A work area (optional).
- 1 or 2 I/O areas.
- Error options (if specified at module generation time).
- ERET macro (if ERREXT=YES is specified at module generation time).
- Blocked or unblocked fixed-length records.
- CNTRL macro (control function, if specified at module generation time).
- Update function (input files).
- RDONLY option (if specified at module generation time).
- Track hold function, if specified at module generation time (input files with update function).

The options listed must be defined by the DTFSD macro statement.

SDMODFI With Truncation: GET Macro Charts GA-GB

Objective: To read fixed-length blocked or unblocked records from a sequential DASD file with provisions for truncation.

Entry: From the GET macro expansion.

Exit: To the problem program.

Method: This logic module reads fixed-length blocked or unblocked records and makes the logical record available to the user in a work area, if one is specified.

If blocked records are being read, and TRUNCS=YES parameter has been specified, truncated records may be included in the file. The module must use the data length from each count field to determine the length of the record to be read. The first GET macro issued to the file results in two separate read operations. The first operation reads the count information to determine the address and length of the data area of the record. The search argument is initialized with the count identifier field. The data length from the count field is compared with the defined block size from the DTF table. If the data length is greater than the block size, this routine initializes the Read Data CCW with the block size and sets the incorrect length indicator in the CCB. Otherwise, the Read Data CCW is initialized with data length.

The second read operation reads the data area of the record and the count field of the next consecutive record in the file. The count field is used to update the search argument and byte count field of the Read Data CCW. Subsequent GET macros read the data area of the record and then the count field of the next consecutive record.If unblocked records are being read, or TRUNCS=YES parameter has not been specified, only one read operation is performed at a time, and only the data area of the record is read.

If the extent upper limit is exceeded, this module issues an SVC 2 to fetch $$BOPEN to open the next extent for the file. Errors are also processed if ERROPT is specified as a DTF and module parameter and if the problem program has specified error routines. If ERREXT is specified, additional errors are returned to the problem program for processing.

If the extent upper limit is exceeded while in the PUT routine logic, the next extent is opened and the partial block or end-of-file record is written as the first record of the next extent.

SDMODFO Without Truncation: Close Routine Chart GE

Objective: To write any remaining records and/or the end-of-file record in sequence on a DASD file.

Entry: From the close B-transient, $$BCSDC1.

Exit: To the close B-transient, $$BOSDC1 via an SVC 9.

Method: This routine determines if there are any logical records to be written on the file. If so, it initializes the count field with the data length and initializes the Write CCW with the data length plus eight. It also turns the partial block switch on. This phase then branches to the PUT routine to write the partial block on the file. Control then returns to this routine.

The end-of-file switch is turned on and control branches to the PUT routine again to set up and write the end-of-file record. This phase then exits to $$BCSDC1 via an SVC 9.

SDMODFO With Truncation: TRUNC Macro Chart GE

Objective: To cause a truncated record to be written.

Entry: From the TRUNC macro expansion.

Exit:

- To the SDMODFO PUT logic.
- To the problem program.

Method: A test determines whether there are any records in the output area to be written. If none, control returns to the problem program. Otherwise, the phase sets a truncation switch to indicate that the TRUNC macro instruction has been issued, and control branches to the SDMODFO PUT logic to write the truncated record.

SDMODFO Without Truncation: PUT Macro Chart GF

Objective: To write records in sequence on a DASD file.

Entry: From the PUT macro expansion.

Exit: To the problem program.

Method: This logic module writes only fixed-length blocked or unblocked records. However, if the close routine turns the partial block switch on, a truncated record may be written, because no padding is used to fill out the last block of records.

If the user specifies a work area, this routine moves the logical record from the work area to the output area. It then updates the current I/O area address by the logical record size, and determines if the end of the block has been reached. If not, control returns to the user.

When the block is full, the search argument record number is updated and the block address, key length, and data length are set in the count field. Then, the block is written on the file.

After the record is written, the search argument is updated, and if the extent upper limit is exceeded, $$BCPEN is called to open the next extent. Control then returns to the problem program.

If ERROPT is specified as a parameter, errors are processed as each record is written. If ERREXT is specified, additional errors are returned to the problem program for processing.

SDMODFU With Truncation:

GET Macro Charts GG-GJ

PUT Macro Chart GN

Objective: To read fixed-length blocked or unblocked records to be updated (optionally) by the problem program, and rewrite the updated records on a sequential DASD file.

Entry:

- From the GET macro expansion.
- From the PUT macro expansion.

Exit: To the problem program.

Method: This logic module reads fixed-length blocked or unblocked records, that are to be updated optionally by the

If ERROPT is specified as a parameter, the module tests for errors as each record is read. If ERREXT is specified, additional errors are returned to the problem program for processing. If a wrong-length record error occurs, the residual count is tested. If the residual count is zero, a genuine wrong length record error exists. Otherwise, the residual count is decreased by the logical record size until the result is zero or negative. If negative, a genuine wrong-length record error exists. If zero, the record is a valid short record, and is processed as a normal record.

When a PUT macro is issued to the file, the module sets a switch in the DTF table to indicate that a PUT command has been issued. If the problem program has specified a work area, the module moves the logical record to the output area. Control then returns to the problem program.

Because the routine that writes the record is incorporated within the GET macro logic, a GET macro or a CLOSE macro must be issued to actually write the record. After the initial entry to the module, a test determines whether another record is needed. If not, the module moves the logical record to a work area, if specified by the problem program, and control returns to the problem program. If the track hold operation has been specified, every read operation reads a record and holds the track.

If another record is needed, the module determines whether a record must be written out first. If the PUT-issued switch is on, the problem program has issued a PUT macro and the record in the output area is written before reading the next record. If the PUT-issued switch is off, the module ignores that record in the output area and reads the next record. In either case, if the track hold option has been specified, the module issues an SVC 36 to free the held track before reading the next record.

If the extent upper limit is reached, the module writes the remaining records for the extent, if necessary, and then issues an SVC 2 to fetch $$BOPEN to open the next extent for the file.

When end of file is reached, the module writes any remaining records and frees any tracks that have been held (if the track hold option is specified). The module then fetches $$BOPEN to exit to the problem program's end-of-file routine.

## SDMODFU: Close Routine:

### With Truncation Charts GG-GJ

Objective: To write any remaining records in sequence on a DASD file.

Entry: From the close B-transient, $$BOSDC1.

Exit: To the close B-transient, $$BOSDC1.

Method: The close routine uses the GET macro logic of its respective module, and has the same entry point to the module as the GET macro.

The routine determines whether there are any remaining records to be written. If so, the remaining records are written and control returns to the close B-transient, $$BOSDC1, via an SVC 9.

### Without Truncation Charts GK-GM

Objective: To write any remaining records in sequence on a DASD file.

Entry: From the close B-transient, $$BOSDC1.

Exit: To the close B-transient, $$BOSDC1.

Method: The close routine uses the GET macro logic of its respective module, and has the same entry point to the module as the GET macro.

The routine determines whether there are any remaining records to be written. If so, the remaining records are written and control returns to the close B-transient, $$BOSDC1, via an SVC 9.

## SDMODFU: CNTRL Macro, Fixed-Length Records Chart GN

Objective: To perform a nondata operation.

Entry: From the CNTRL macro expansion.

Exit: To the problem program.

Method: The CNTRL macro instruction causes a seek operation on a DASD device or a restore operation on a 2321. The routine waits for the completion of any previous I/O operation. It then initializes the control CCB with the symbolic unit address for the file, moves the control command code into the control CCW, and loads the address of the CCB into register 1. The routine issues an SVC 0 to perform the control operation and returns control to the problem program.

## SDMODFU: RELSE Macro, Fixed-Length Records Chart GN

Objective: To cause a physical read operation to be performed when the next GET macro is issued by the problem program.

Entry: From the RELSE macro expansion.

Exit: To the problem program.

Method: This routine, used only in conjunction with blocked input records, causes the remaining records in an input block to be bypassed. It sets the current pointer to the end of the input area, so that the next GET macro instruction causes a new physical record to be read into the input area. The first logical record of that block is made available to the problem program.

Exit: To the problem program.

Method: This routine, used only in conjunction with blocked input records, causes the remaining records in an input block to be bypassed. It sets the current pointer to the end of the input area, so the next GET macro instruction causes a new physical record to be read into the input area, and makes the first logical record of that block available to the problem program. If spanned records are being processed, the entire block of logical spanned records is bypassed.

SDMCDVO: PUT Macro Charts HD-HF

Objective: To write variable-length blocked or unblocked records in sequence on a DASD file.

Entry: From the PUT macro expansion.

Exit: To the problem program.

Method: Because variable-length records are written, this module must keep track of the number of bytes remaining on the track for each record processed (if all records are unspanned), and must calculate the number of bytes remaining in the output area for the problem program.

For each record to be written, this routine increases the output area address and accumulated data length by the current record size. A test then determines if the record fits on the track. If the record fits, a test determines whether the output area is full. If not full, the record is moved to the output area (if a work area is specified), addresses are updated and the amount of space remaining in the output area is calculated. Control returns to the problem program.

If the output area is full, but not exceeded, the module moves the record to the output area (if a work area is specified) and calculates the remaining track capacity. It then writes the record on the file.

If the record does not fit on the track, or the output block has been exceeded, a test determines if any records have been previously processed and moved into the output area. If so, the record(s) already in the output area is written as a truncated record and as the last record on the track. The record that did not fit is moved to the first part of the output area.

If records have not been previously placed in the output area, and the current record does not fit on the track, the module updates the DASD address to the next track, and writes the record as the first record on that track. If the extent upper limit is reached, it fetches $$BOPEN to open the next extent.

If spanned processing is specified, a logical record in the user's work area of the length specified in the RECSIZE register is divided by LIOCS into segments to make full use of the space available in each physical record and device track. Processing proceeds for each segment in a manner similar to unspanned records. In addition to the bytes of data, each segment contains a segment descriptor word indicating its sequence as the only, first, middle, or last segment in the construction of the logical record.

A spanned record does not span volumes on output. If there is not enough space on the current volume to complete a spanned record, the module rereads the last block of the previous spanned record and truncates it, if necessary, to the last segment. The remainder of the track is then erased. An 8-byte record consisting of a four-byte block descriptor word and a four-byte null segment is written on each remaining track to the end of the extent(s) on the current volume. Finally, an attempt is made to put the entire spanned record on the next volume.

In rereading the last block of the previously spanned record, a reopening of one or more previous extents may be necessary. If so, the module interfaces with the OPEN transients by setting indicators in the DTF to show that a preceding extent on the current or previous volume is to be reopened.

The module also processes errors if ERROPT is specified as an SDMODVO parameter and the problem program has specified error routines. If ERREXT is specified, additional errors are returned to the problem program for processing. If spanned processing is specified and ERROPT=[SKIP,IGNORE], the physical record on which the error occurred is ignored. The remaining spanned record segments, if any, are written.

SDMODVC: Close Routine Chart HH

Objective: To write any remaining records and/or the end-of-file record in sequence on a DASD file.

Entry: From the close transient, $$BOSDC1.

Exit: To the close transient, $$BOSDC1.

the problem program on a sequential DASD file. The records are returned to the same location from which they were read.

Entry: From the PUT macro expansion.

Exit:

- To the problem program.
- To the close B-transient, $$BCSDC1.

Method: For unspanned record, this routine first moves the logical record from a work area to the output area, if a work area has been specified by the problem program.

The I/O area address is then updated by the logical record length, and the routine determines whether the end of the block has been reached. If it has not been reached, an update switch is set, and control returns to the problem program.

If the end of the block has been reached, the routine determines whether a second read operation has been performed. If so, tests are made to ensure that the I/O operation has been completed.

This routine then initializes the search argument with the address of the record to be written, and the Read/Write CCW with the length of the record. It modifies the Read/Write CCW to write data, and issues an SVC 0 to write the record. When I/O is completed, the routine determines whether the track hold option has been specified. If so, the track of the record just written is freed via an SVC 36.

Spanned record processing proceeds for each segment in a fashion similar to unspanned records. The device is initially repositioned to the first block of the logical record by using the pointer stored in the DTF table. If the logical record spans several extents, the DTF is reset by decrementing the extent sequence number by 1 and by using an AND (X'44') to the open communications byte. The extent where the logical record begins can now be reopened by fetching $$BOPEN. The physical record blocks are then updated from the logical record in the user's work area. Null segments are recognized, but not assembled.

The routine next tests to see if entry to the module was from the close routine. If so, control returns to the close B-transient, $$BCSDC1. If end of file has not been reached, control returns to the GET macro logic (if specified), or to the problem program. If end of file has been reached, control returns to the problem program, after setting the end-of-file bit on in the CCE.

If ERROPT has been specified as a SDMODVU parameter, and if the problem program has specified error routines, the module also processes errors. If ERREXT is specified, additional errors are returned to the problem program for processing.

SDMODVU: Close Routine Chart HR

Objective: To write any remaining records in sequence on a DASD file.

Entry: From the close B-transient, $$BCSDC1.

Exit: To the close B-transient, $$BOSDC1.

Method: This routine sets the current pointer to the end of the I/O area, and then determines whether there are any records in the output area to be written. If not, control returns to the close B-transient, $$BOSDC1 via an SVC 9.

If any records are to be written, control branches to the PUT macro logic to write the remaining records and return to the close routine.

SDMODVU: RELSE Macro Chart HR

Objective: To cause a physical read operation to be performed when the next GET macro is issued by the problem program.

Entry: From the RELSE macro expansion.

Exit: To the problem program.

Method: This routine, used only in conjunction with blocked input records, causes the remaining records in an input block to be bypassed. It sets the current pointer to the end of the input area, so the next GET macro instruction causes a new physical record to be read into the input area, and makes the first logical record of that block available to the problem program. If spanned records are being processed, the entire block of logical spanned records is bypassed.

SDMODVU: CNTRL Macro, Chart HR

Objective: To perform a nondata operation.

Entry: From the CNTRL macro expansion.

Exit: To the problem program.

Method: The CNTRL macro instruction causes a seek operation on a DASD device , or a restore operation on a 2321. The routine

neither is specified, the wait loop is bypassed.

Next, a test determines if the record fits on the track. If the record fits, the module calculates the space remaining on the track after the record is written and posts that information in the DTF table. Then, it modifies the I/O area address in the Write CCW and issues an SVC 0 to write the record.

If ERROPT is specified as a parameter, errors are processed as each record is written. If ERREXT is specified, additional errors are returned to the problem program for processing.

If the record does not fit, a test determines if the extent upper limit has been reached. If so, the routine fetches $$BOPEN to open a new extent. If not, the routine updates the address to the next available track, initializes the record capacity bucket, and sets the record number to 0. In either case, after the write operation, the routine returns control to the problem program after an I/O wait (if two I/O areas are specified or if a work area is not specified).

SDMODUO: Close Routine Chart JE

Objective: To write an EOF record.

Entry: From the close transient, $$BOSDC1.

Exit: To the close transient, $$BCSDC1 or SDMODUO PUT logic.

Method: The routine waits for I/O completion if two I/O areas or a work area is specified. A test determines if there is enough room left on the track to write an EOF record. If not, the SDMODUO PUT undefined record routine is entered, and the search address is updated to the next available track in the current extent. If another track is not available, $$BOPEN is called in to open a new extent.

When control returns to this routine, the proper I/O area is selected, the record data length is set to 0, and the EOF record is written. After a wait for I/O completion, control returns to the close transient, $$BOSDC1 via an SVC 9.

SDMODUU:

GET Macro Chart JF

PUT Macro Chart JG

Objectives: To read a physical record from a DASD file, and to rewrite the record in the same location if the record requires updating.

Entry: From the GET or PUT macro expansion.

Exit: To the problem program.

Method - GET Logic: This module reads undefined length unblocked records and makes them available to the user in a work area, if one is specified. If track hold is specified, each read operation reads a record and holds a track.

The first time through the module, a switch is turned on, the count field and data area of the first record are read, and the count field of the next record is read. On each subsequent entry, the data area is read and the count field of the next sequential record is read. A test determines if the track hold option is specified. If so, the track is freed so that the data can be read. A test also determines if two I/O areas or a work area is specified, so that another GET operation can be initiated.

In either case, control returns to the problem program so that the record can be updated.

If ERROPT is specified as a parameter, errors are processed as each record is read. If ERREXT is specified, additional errors are returned to the problem program for processing.

Method - PUT Logic: This module writes the records (updated optionally) on the DASD file and returns control to the problem program. When end of file is reached, the module processes the last record before returning control to the problem program.

If ERROPT is specified as a parameter, errors are processed as each record is written. If ERREXT is specified, additional errors are returned to the problem program for processing.

SDMODUU: CNTRL Macro, Undefined-Length Records Chart HR

Objective: To perform a nondata operation.

Entry: From the WRITE macro expansion.

Exit: To the problem program.

Method: Two types of write operations may be specified by the problem program (SQ and Update). If SQ is specified in the operand of the WRITE macro, a sequential format write (write count, key, and data) is performed. If UPDATE is specified in the operand, a nonformat write (write data) is executed. A WRITE UPDATE should always be preceded by a READ macro instruction.

This macro causes a record to be written from the area defined by the WRITE macro to the file. The length of the record to be written is specified in the operand of the WRITE macro instruction only if records of undefined format are being written. If fixed-length unblocked records are being written, the record length is defined in the DTF table. Record blocking is not handled by the WRITE macro because it is a responsibility of the problem program.

This routine initializes the CCW chain to write count, key and data, and write data. It also initializes a verify CCW if the update option has been specified. If a WRITE UPDATE macro is issued, this routine reinitializes the CCW chain to write data, and sets the verify CCW to read data. This read data operation is followed by a read count operation in order to obtain the count field ID of the next sequential record. It then issues an SVC 0 to write the record.

If a WRITE SQ macro has been issued, the routine determines whether the current record fits on the track, or if the track limit has been reached on a previous read operation. If either condition exists, control branches to a routine to update the search address. The routine determines whether the end of the extent has been reached. If so, it fetches $$BCPEN to open a new extent. An SVC 0 is then issued to write the record. The track capacity is decreased by the effective length of the record just written. If the routine has been entered from the close routine, control passes to the check routine to determine if the input/output operation has been completed. Otherwise, control passes to the problem program.

SDMODW: Close Routine Chart KD

Objective: To write any remaining records on a DASD work file.

Entry: From the close B-transient, $$BCSDC1.

Exit: To the close B-transient, $$BOSDC1, via an SVC 9.

Method: This routine performs a branch-and-link operation to the WRITE macro routine to write any records remaining in the output area, and check the write operation. Upon return from the CHECK macro routine, this routine issues an SVC 9 to return to the close B-transient, $$BOSDC1.

SDMODW: CHECK Macro Chart KE

Objective: To ensure that a previously issued READ or WRITE macro has been satisfactorily completed.

Entry: From the CHECK macro expansion.

Exit:

- To the problem program.
- To the problem program's end-of-file routine.

Method: This routine waits for the completion of the input/output operation started by a READ or WRITE macro instruction. If the problem program has specified ERROPT as a parameter, this routine checks for read or write errors. If no error has occurred, this routine checks for a write update operation. If so, and the track hold option has been specified, the routine issues an SVC 36 to free the held track. Control then returns to the problem program.

If a read or a write error has occurred, and the problem program has specified an error routine, control branches to the user's error routine to process the error. Upon return, if a read error routine has been specified, the count field of the next record is read. If the ignore option has not been specified, the routine returns to the READ macro routine to read the next record. If the ignore option has been specified, control returns to the problem program.

SDMODW: NOTE Macro Chart KF

Objective: To pass identification of the last physical record that was read or written to the problem program.

Entry: From the NOTE macro expansion.

Exit: To the problem program.

Method: The CNTRL macro instruction causes a seek operation on a disk device. This routine isolates the Seek CCW in the CCW chain by setting off the command chaining bit, and issues an SVC 0 to perform a control seek. When the I/O operation is completed, the routine turns the command chaining bit on and control returns to the problem program.

SDMOD: FEOVD Macro Chart KG

Objective: To force end of volume in sequential disk processing.

Entry: From $$BOSDEV.

Exit: To $$EOSDEV to close the current volume and open a new one.

Method: The FEOVD macro instruction causes and end of volume condition to occur before physical end of volume has been reached. If forced end of volume is specified, $$BOSDEV is fetched to close the current volume and open a new volume.

INITIALIZATION AND TERMINATION PROCEDURES

When sequential access DASD files (DTFSD) are opened, and the file is on more than one volume, only one extent is processed at a time, so only one volume need be on-line at a time.

Job control accepts label information supplied by VOL, DLAB, and XTENT statements (not for 3330), as well as information on the simplified DLBL and EXTENT statements provided by Version 3 onward. Job control stores this DASD label information on the SYSRES DASD label information cylinder. The IES Processor, $$BCPEN, prepares to read the label information from the SYSRES label information cylinder into the logical transient area, and then fetches $$BOSD00.

The sequential DASD open logical transients read the DASD label information from SYSRES into storage. The format of the SYSRES DASD label information is illustrated in DOS/VS IIOCS Volume 1, SY33-8559. If the file is an input file, the open transients compare the file label information with the SYSRES DASD label information to determine if the logical file is correct, if the serial numbers are equal, and if the label extent limits are equal to or greater than the limits of the incoming extent. The extent limits are posted in the DTF.

If the logical file is an output file, the open logical transients create file labels and write them in their appropriate location and sequence. Extent limits are checked to ensure that no extent overlaps the Volume Table of Contents (VTOC) limits, or overlaps an already existing file that is still active.

Disk work files are supported as single-volume, single-pack files and are always opened as output file.

When a file is closed, the close logical transient determines whether a block of data remains to be processed. If so, the logic module is reentered to complete the processing. Upon return, file labels are deleted if so specified. Otherwise, the file labels are updated and rewritten if the file is an output file or a work file. Control returns to the close monitor or the problem program.

SEQUENTIAL DASD OPEN/CLOSE LOGIC

Open/Close Sequential DASD Files Chart 03

When a DASD file is processed sequentially (DTFSR or DTFSD specified), OPEN initially:

- checks the standard label(s) on the volume, (or on the first volume of a multivolume file),
- makes any additional labels on the first volume available for checking, and
- locates the first extent on the first volume and makes it available for processing.

Logical IOCS processes one extent at a time in the sequence specified by the user's job control // EXTENT cards. When logical IOCS detects the end of the current extent, it branches to the end-of-extent routine. OPEN then locates the next extent specified by the control cards and makes it available for processing. If the next extent is the first extent of a different volume used by the file, OPEN checks the standard labels on that volume and makes any additional user labels available to the user for checking.

OPEN (Input Sequential DASD) Chart 04

If the file to be opened is normal input, the extents are read and checked as needed. User labels are read and checked if LABADDR is specified. The file labels are checked against the DLBL information. The open indicator for the file is turned on and control returns to the user.

## $$BOSDI1: SD Open Input, DLBL Extents Charts LE-LF

Objective:

- To control the sequence of operations required for opening each file extent.
- To provide an entry to the user's trailer label routine (if specified), at each end-of-volume.
- To provide an entry to the user's end-of-file routine (if specified) upon reaching the end of the last extent.

Entry: From $$BOSD00, $$BOSD01, $$BOSD02, and reentry from $$BOSDI2, $$BOSDI3, and $$BOMSG1.

Exits:

- To $$BOSDI5 to post the DIB.
- To $$BOSDI2 to continue OPEN processing.
- To $$BODQUE to dequeue old extents.
- To $$BOSDI3 to process user labels.
- To $$BOPEN if the last DLBL extent has been processed and another file is to be opened.
- To $$BOMSG1 for operator communication.
- Return to problem program when all files have been opened.

Method: If a system unit other than SYSLNK is being opened, this routine gets extent information for the DTF from the data information block (DIB). $$BOSDI5 is the overlay phase that posts the DIB. Otherwise, it continues at BYPASSX.

$$BOSDI1 tests for availability of DLBL extents. If no more are available, an exit is made to the user's end-of-file address if no trailer labels are to be processed. If the file has been previously opened, the next consecutive DLBL extent to be opened is read, and a test determines if this extent is for another volume.

Upon encountering a new volume, trailer labels are processed for the previous volume (if LABADDR was specified with a DTFSD), by exiting to phase 3 of open input. The volume label is read and checked to ensure that the proper pack is mounted. If the volume label is all right, the Format 4 label is read and checked. The VTOC limits from this label are saved, and initialization is performed to fetch the next phase. The routine exits to $$BOSDI2.

## $$BOSDI2: SD Open Input, Extent to DTF Chart IG

Objective: To obtain extent information for the DTF table as required by an attempt to access a record beyond the limits of the current extent.

Entry: From $$BOSDI1, and reentry from $$BOMSG1 and $$BODSMW.

Exits:

- To $$BOSDI1 to bypass current extent and process the next one.
- To $$BOSDI4 to continue initialization of the DTF table.
- To $$BODSMW to print message if data secured file is unccuntered.
- To $$BOPEN if the last DLBL extent is processed and another file remains to be opened.
- To $$BOMSG1 for operator communication to display error message.

Method: The routine reads the Format 1 label for the file and ensures that no discrepancies exist between the DLBL and Format 1 label. The extents within the label are scanned for one that either matches or falls around the limits of the incoming extent. The scanning process continues until a proper match is found, or until all the extents have been exhausted by reading the labels in the chain (if any are present). The extent limits are then posted in the DTF table. The file is indicated as being open, and additional initialization is performed depending on the type of DTF being opened.

The format 1 label is checked for the data security indicator. If it is ON and the file has not been opened, $$BODSMW is fetched to put out a data security message. Otherwise, any user header labels are processed, and control branches to $$BOSDI4 to continue initialization of the DTF table.

## $$BOSDI3: SD Open Input, User Labels Chart IH

Objective: To read user labels and give control to the user for processing them. To rewrite any labels updated by the user.

Entry: From $$BOSDI1 or $$BOSDI2, and reentry from $$BOMSG1, $$BOFLPT, or the user's label routine via an SVC 9.

- To $$BOSDI5 to post the DIB.
- To $$BOPEN if the open processing for the file is complete and another file remains to be opened.
- To $$BODQUE to dequeue extents.
- To $$BOMSG1 for operator communication.
- To $$BOSIGN to check for device assignment.
- Return to problem program when all files have been opened.

Method: This phase reads each extent record from the SYSRES label cylinder, tests for various conditions in their appropriate order, and fetches the phase required for further processing. If the normal sequence is interrupted by the entry of an extent from the console, the phase finds the next DLBL record by using the sequence number of the last extent processed before the extent was entered from the console. $$BCSDI5 is the overlay phase that posts the DIB for a system file.

The processing required for each extent record depends on whether:

1. The file being opened is a system file.
2. The file is already open.
3. The extent is on another volume.
4. The extent is entered from the console.
5. The extent is the last one for the file.
6. The extent is to be bypassed, either for file protection or because it is a duplicate.
7. User labels are specified.
8. File protect is specified.

## $$BCSIGN: SD Open Ignore Chart IN

Objective: To check for the CCBCL Open/Ignore function.

Entry: From $$BCSDO1.

Exits: To $$BOSDO1.

Method: This routine determines whether the COBOL Open/Ignore function has been specified. If so, and the device is unassigned or assigned IGN, the open is bypassed. If the device is assigned, the open is continued.

If the Open/Ignore option has not been specified, and the device is unassigned or assigned IGN, the job is aborted. Otherwise, the open continues.

The routine also determines whether the assigned device is the correct device. If not, the job is aborted. Otherwise, control returns to $$BCSDO1 to continue processing.

## $$BOSDO2: SD Open Output, Volume Label Chart IO

Objective: To read and verify the standard volume label (VOL 1) and VTOC label (Format 4), preventing any extent from overlapping the VTOC.

Entry: From $$BOSDO1, $$BOSDO6, $$BOSDO7, and reentry from $$BOMSG1.

Exits:

- To $$BOSDO1 if an extent overlaps the VTOC.
- To $$BOSDO3 to continue processing a sequential file.
- To $$BOSDO4 to complete the opening of a compiler file.
- To $$BOSDO8 to prevent the user from creating identical labels in the VTOC.
- To $$BOMSG1 for operator communication.

Method: The volume and Format 4 labels are read and verified, VTOC limits are saved, and the extent limits are checked against the VTOC limits for overlap. For each new volume that is opened for the file, an exit is made to $$BOSDO8 to prevent the user from creating identical labels in the VTOC.

For an opened SYSLNK file, this routine exits to $$BOSDO4 after getting the VTOC limits to complete the opening of the file. Otherwise, $$BOSDO3 of open output is fetched to further process a sequential file.

## $$BOSDO3: SD Open Output, Extent Overlap Charts IP-IQ

Objective: To prevent opening any extent that overlaps an already existing file that is still active.

Entry: From $$BOSDO2 or $$BOSDW1, and reentry from $$BOSDO8 or $$BOMSG1.

$$BCSDO7: SD Open Output, Extents from Console Chart LW

Objective: To enter operator-provided extent information from the console.

Entry: From $$BOSDO1, $$BCDSPW, or $$BO2321.

Exits:

- To $$BO2321 to complete conversion of 2321 extent.
- To $$BOSDC2 to process the new extent.
- To $$BODSPV to display the VTOC.
- To $$BOVDMP for a more extensive VTOC dump.

Method: This routine initiates a no more available extents message and reads the operator's reply (if a 3210 or 3215 has been assigned to SYSLOG). If the operator did not cancel the job, it is assumed that an extent was entered, which is then checked for validity. If the extent is valid, this routine exits to $$BOSDO2 to process it.

$$BCSDO8: SD Open Output, Delete Label Chart LX

Objective: To prevent creation of identical file labels.

Entry: From $$BOSDO2, $$BCSDW1, $$BODAO1, or $$BCIS03, and reentry from $$BOMSG1.

Exits:

- To $$BODAC1 for Direct Access Method.
- To $$BOSDC3 for all other uses.
- To $$BOMSG1 for operator communication.

Method: This routine uses the 44-byte filename from the DLBL record as a key to search the VTOC for any identical filename. It deletes any identical label found if the expiration date is passed. Otherwise, the operator has the option of canceling the job or deleting the identical label.

$$BCSDC9: SD Open Output, Extent to DTF Chart LY

Objective: To update the DTF table.

Entry: From $$BCSDO4.

Exits:

- To $$BCSDO6 if user labels are specified.
- To $$BCFLPT if file protect is specified.
- To $$BCPEN if open processing for the file is complete and another file remains to be processed.
- To $$BCMSG1 for operator communication.
- Return to problem program when all files have been opened.

Method: This routine posts appropriate extent information in the DTF table. It then tests to check if user header labels are to be processed. If that is the case, it fetches $$BCSDC6. Otherwise, it tests if the extent is to be file-protected. If yes, it fetches $$BOFLPT; if no, it exits to $$BCPEN to open the next file or it returns to the problem program if there are no more files to be opened.

$$BOSDW1: SD Open Work File, Volume Label Charts MA-MB

Objective: To read and verify the standard volume label (VOL 1) and VTOC label (Format 4), preventing any extent from overlapping the VTOC (Volume Table of Contents).

Entry: From $$BOSD00, $$BOSD01, $$BOSD02, $$BOSDW2, and reentry from $$BCMSG1.

Exits:

- To $$BCSDO3 to continue processing a work file extent.
- To $$BCSDO8 to prevent duplicate file labels.
- To $$BCPEN if the last extent has been processed and another file remains to be opened.
- To $$BCMSG1 for operator communication.

Method: This routine determines whether the symbolic unit specified in the DLBL statement is assigned and whether it can be used as a work file. It reads the volume label and, if the device is a disk, determines if a correct disk pack is mounted. It reads the VTOC label and ensures that no extent overlaps the VTOC. If the VTOC has not been checked for a duplicate filename, $$BOSDO8 is fetched to eliminate possible duplication. Subsequent exits are to $$BOSDO3. (See $$BCSDO3 SD Open Output, Extent Overlap.)

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 SD file remains to be closed, control returns to the close monitor, $$BCLOSE. If ISAM files are being processed, control returns to $$BCIS0A. Otherwise, control returns to the problem program.

## $$BCDQUE: Dequeue Extent JIBs Chart MJ

Objective: To find the Job Information Block (JIB) 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 sequential DASD open phase $$BCSD01, $$BOSD06, or $$BCSDI1 to the label DEQUERTN.

Exit: To the problem program if no files remain to be opened, or to the TIS 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 to be returned to, if specified, phase $$BCDQUE issues an SVC 22 to seize the system; that is, to suspend multiprogramming operation. The phase then locates the proper 2-byte entry in the LUB table for the logical unit specified and examines the second byte of the IUB entry to determine if any JIBs are chained to the LUB. If JIBs are chained to the IUB; 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.

When all the chained JIBs have been checked, or if no JIBs are chained to the IUB, phase $$BODQUE issues a second SVC 22 to release the system for multiprogramming operation. Phase $$BODQUE then fetches the calling phase or the first phase of the TIS processor, $$BOPEN, if the name of the calling phase was not supplied and there is another file to be opened. If the name of the calling phase was not supplied and there are no other files to be opened, phase $$BCDQUE returns control to the problem program via an SVC 11.

## $$BCSDEV: SD Close Chart MK

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

Entry:

- From the FEOVD macro.
- From $$BCSD05 (phase 5 of open sequential output).
- From IICCS via SVC 9.

Exits:

- To the TIS processor $$BOPEN.
- To the close phase $$BCSD02.
- To the problem program.

Method: For an output file, an end of volume marker is written and the DTF is set up so that the next record is written on a new volume. The end of volume marker is a normal end of file record.

For an input file, a check is made to determine if update has been specified. If it is necessary to rewrite any updated records, an exit is made to the module close routine. End of volume is posted in the DTF, any remaining extents on the volume are bypassed, and the first extent on the next volume is opened.

## $$BO2321: SD Open Output , 2321 Extents from Console Chart MI

Objective: To complete conversion of 2321 extent information entered from the console.

Entry: From $$BOSD07.

Exit: To $$BOSD07.

Method: This routine completes the conversion of a 2321 extent limit.

Device independent files are those files defined by either a DTFCP macro or DTFDI macro.

The DTFCP macro defines files used by IBM compilers: COBOL, FORTRAN and PL/I.

The DTFDI macro defines files assigned to the device independent system units SYSRDR, SYSIPT, SYSPCH, and SYSLST. The DTFDI macro and its associated DIMOD macro, therefore, provide DOS/VS Assembler users with the same capabilities extended by DTFCP.

## COMPILER FILES

Compiler (CP) files are files provided specifically for IBM internal programs such as COBOL, FORTRAN, and PL/I. These files, defined by a DTFCP macro, provide limited device independence. Because this file definition does not conform to standards established for other logical IOCS component and is tested for use only by IBM internal programs, it is not documented in any other System Library publications.

Some of the differences between CP components (DTFCP and CPMOD) and other LIOCS components are:

1. No provision for separate assembly.
2. Error recovery not the same.
3. The DTFCP table is not self-initializing; that is, the user must initialize the table if the file is reopened.

### DTFCP Macro

A DTFCP macro instruction can be used for each file that has fixed, unblocked records, and limited device independence. When the file is opened, a channel program for reading/writing on a particular device is built in the DTFCP table area (Figures 44, 45, and 46). Only the following devices can be accessed by DTFCP: IBM 1442, 1443, 1403, 3211, 2501, 2520, 2540, 2560, 3504, 3505, 3525, 2400/3400 series, 2311, 2314, 2319, 3330, 3340, 3540, and 5425.

The DTF table generated at assembly time is initialized by the DTFCP open phase according to the device type. The device type is found by the open phase by checking the device type set in the PUB table entry. After it is found, the proper indicators are set in the DTF table, and the work areas and CCW's are modified. Standard labels are not required on tape files.

The DTFCP header card is followed by a series of parameter cards describing the file, and specifying symbolic addresses of routines and areas used for processing the file. Because keyword parameters are specified, the parameter cards may appear in any order. This group of cards generates the necessary logical IOCS DTFCP tables during assembly.

The parameter cards following the DTFCP header card have keyword entries in the operand field. All cards used in the DTFCP macro instruction, except the last, have continuation punches in column 72.

Filename: This is the first entry in a DTFCP macro instruction. It assigns a symbolic name to the file, which appears in all I/O macro statements referencing this file. The symbolic name of the file is in the name field, and DTFCP is in the operation field.

DEVADDR=SYSXXX: This parameter specifies the symbolic unit to be associated with the file. If SYSPCH is assigned to the IBM 2540 or 2520 punch units, the CP Open transient allows the error recovery procedures generated by DTFCP when DEVADDR=SYSPCH.

IOAREA1=NAME: The I/O area to be used by the file is defined by an address expression.

IOAREA2=NAME: If two I/O areas are needed for overlapped GET/PUT processing, this parameter is required.

IOREG=(n): For input files, this parameter specifies the general purpose register (n) into which IOCS inserts the address of the next logical record available for processing. For output files, IOCS inserts the address of the area where the user can build the next logical record. Any register 2-12 may be specified. The same register may be used for different files.

This parameter must be specified whenever two I/O areas are used.

RECSIZE=n: For fixed-length records, this parameter specifies the number of characters in the record. I/O routines use

| Bytes | Bits | Function |
|---|---|---|
| 0-15 (00-0F) | | CCB. |
| 16 (10) | 0-1 | Not used. |
| | 2 | COBOL open; ignore option. |
| | 3 | X'10' indicates an unlabeled FORTRAN tape. |
| | 4 | DTF table address constants relocated by OPENR. |
| | 5 | Used by FORTRAN (Sequential Disk Backspace and Rewind). |
| | 6 | 1 = ASCII<br>0 = EBCDIC |
| | 7 | FORTRAN is calling DTFCP. |
| 17-19 (11-13) | | Logic module address. |
| 20 (14) | | DTF type X'32' except in the case of disk assigned to units SYS000 to SYSnnn. In this case, a DTFCP open phase changes it to X'20'. |
| 21 (15) | | Open indicators: X'02' input, X'00' output, except for tapes assigned to SYS000 to SYSnnn when X'00' = input and X'08' = output.<br><br>X'08' DISK=YES indicator. |
| | 0 | 1 = No rewind, 0 = Rewind. |
| 22-28 (16-1C) | | Filename (see byte 29). |
| 29 (1D) | | Device type code:<br>X'00' = 2311<br>X'01' = 2314, 2319<br>X'04' = 3330.<br>X'08' = 3340 general<br>X'09' = 3340 35MB<br>X'0A' = 3340 70MB. |
| 30-35 (1E-23) | | File address for disk; block count if bit 7 of byte 16 is on. |
| 36-37 (24-25) | | Volume sequence number or workarea. |
| 38 (26) | | Open switch. |
| 39 (27) | | Sequence number of current extent. |
| 40 (28) | | Sequence number of last extent, or X'80' if 1442 punch. |
| 41 (29) | | X'80' indicates request for standard label tape OPEN. |
| 42 (2A) | | X'80' device is a 2560.<br>X'40' device is a 5425.<br>X'20' device is a DASD.<br>X'10' device is a tape.<br>X'08' device is a printer.<br>X'04' device is a punch.<br>X'02' device is a reader. |

Figure 44. DTFCP: DISK=YES (1 of 3).

| Bytes | Bits | Function |
|---|---|---|
| End-of-table if DTF is defined for an input file. | | |
| 120-127 (77-7F) | | Second CCW for output. |
| 128-151 (80-97) | | Verify CCW's for output. |
| End-of-table if DTF is defined for output file and DEVADDR does not equal SYSPCH. | | |
| 152-159 (98-9F) | | 2540 punch error recovery CCW 1. |
| 160-167 (A0-A7) | | 2540 punch error recovery CCW 2. |
| 168-231 (A8-E7) | | Reserved. |
| When the CP open initializes the table and determines that the device is a 2540 punch, the following bytes in the table are changed: | | |
| 30 (1F) | | X'FF' indicator to DTFCP open phases and logic module. |
| 32-35 (20-23) | | Instruction to load user I/O area to I/O register. |
| 48-55 (30-37) | | CCW. |
| 56-63 (38-3F) | | 2540 punch error recovery CCW 1. |
| 64-71 (40-47) | | 2540 punch error recovery CCW 2. |
| 72-151 (48-97) | | 80-byte card image, savearea 1. |
| 152-231 (98-E7) | | 80-byte card image, savearea 2. |
| When the CP open initializes the table and determines that the device is a 2560 or 5425, the following bytes in the table are changed: | | |
| 32-35 (20-23) | | Instruction to load user I/O area to I/O register. |
| 48-55 (30-37) | | First output CCW. |
| 56-63 (38-3F) | | Second output CCW. |
| 64 (40) | | Stacker select character V for ASA. |
| 65 (41) | | Stacker select character W for EBCDIC. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 44. DTFCP: DISK=YES (3 of 3).

| Bytes | Bits | Function |
|---|---|---|
| 44 (2C) | 0 | 1 = input, 0 = output. |
| | 1 | 1 = eject needed for a reader-punch, 0 = no eject. |
| | 2 | 1 = not first pass, 0 = first pass. |
| | 3 | 1 = 2 I/O areas, 0 = 1 I/O area. |
| | 4 | 1 = 2540 punch. |
| | 5 | 1 = SYSLST or SYSPCH. |
| | 6 | 1 = SYSLST or SYSPCH on output tape. |
| | 7 | Reserved for future use. |
| 45-47 (2D-2F) | | IOAREA2 address. |
| 48-55 (30-37) | | CCW. |
| End-of-table is DTF is defined as output file and DEVADDR is not equal to SYSPCH. | | |
| 56-63 (38-3F) | | 2540 punch error recovery CCW 1. |
| 64-71 (40-47) | | 2540 punch error recovery CCW 2. |
| 65-67 (41-43) | | EOF address, input only. |
| End-of-table if DTF is defined as input file. | | |
| 72-151 (48-97) | | 80-byte card image, savearea 1. |
| 152-231 (98-E7) | | 80-byte card image, savearea 2. |
| If the device is a 2560 or 5425, bytes 56 onward contain the following information: | | |
| 56-63 (38-3F) | | Second output CCW. |
| 64 (40) | | Stacker select character V for ASA. |
| 65 (41) | | Stacker select character W for EBCDIC. |
| 66-75 (42-4B) | | Reserved for future use. |
| 76-235 (4C-EB) | | First I/O area. |
| 236-237 (EC-ED) | | Reserved. |
| 238-317 (EE-13D) | | Second I/O area. |
| 318-319 (13E-13F) | | Reserved. |

[1]DTF type X'30' found in DOS LIOCS Version 1 only.

Numbers in parentheses are displacements in hexadecimal notation.

Figure 45. DTFCP: DISK=NO (2 of 2).

| Bytes | Bits | Function |
|---|---|---|
| 40-47 (28-2F) | | CCW. |
| End of table if DTF is defined as output file and DEVADDR is not equal to SYSPCH. | | |
| 48-55 (30-37) | | 2540 punch error recovery CCW 1. |
| 56-63 (38-3F) | | 2540 punch error recovery CCW 2. |
| 57-59 (39-3B) | | EOF address, input only. |
| End of table if DTF is defined as input file. | | |
| 64-143 (40-8F) | | 80-byte card image, savearea 1. |
| 144-223 (90-13F) | | 80-byte card image, savearea 2. |
| For 2560 and 5425 bytes 48 onwards contain the following information: | | |
| 48-207 (30-CF) | | IOAREA1. |
| 208-209 (D0-D1) | | Reserved. |
| 210-369 (D2-171) | | IOAREA2. |
| 370-371 (172-173) | | Reserved. |
| 372-451 (174-1C3) | | Compare area. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 46. DTFCP: DISK= parameter omitted (2 of 2).

CPMOD Macro

The CPMOD macro is used with IBM internal programs only. Therefore, the CPMOD macro is not documented in any other System Library publications--see Compiler Files.

The CPMOD macro generates nine different logic modules. Each CPMOD logic module is flowcharted and described in detail. Output modules where RETRY=NO is specified are not flowcharted because they are the same as the other output modules except for the 2520 and 2540 punch error recovery procedures.

The GET-PUT logic modules for two I/O areas, with IOPTR and RETRY omitted handles all other cases except those modules where IOPTR=YES is specified.

The parameters of the CPMOD macro are keyword parameters. Because keyword parameters are specified, the parameter cards after the header card may appear in any order. This group of cards generates the necessary logical IOCS CPMOD during assembly.

The parameter cards following the CPMOD header card have key entries in the operand field. All cards used in the CPMOD macro

Exit: To the problem program, to the user's EOF routine, or to $$BOPEN to get a new diskette extent.

Method: This routine makes a logical record available to the user in the I/O area. When an end-of-file is sensed, a branch to the user's EOF routine is made.

When an end-of-extent condition is found on a diskette and no more extents are available, a branch is made to the user's EOF routine. If there are more diskette extents, $$BOPEN is fetched to get another diskette extent, and processing continues.

The routine is similar to the GET two I/O areas routine except the logic to handle I/O area switching is not included.

## CPMOD: GET Macro, IOPTR=YES Chart NB

Objective: To read a record into the I/O area pointed to by register 0 (IOPTR).

Entry: From a GET macro expansion.

Exit: To the problem program or EOF address.

Method: When the parameter IOPTR=YES is included, the CP module generated allows the user to use any area in main storage (other than a storage protected area) as an I/O area. The I/O area is pointed to by the address loaded into register 0. With this exception, this routine is the same as the GET with one I/O area.

## CPMOD: PUT Macros, Two I/O Areas Chart NC

Objective: To write a logical record with overlap.

Entry: From a PUT macro expansion, or from an SD workfile close.

Exits:

1. To the problem program.
2. To the user's EOF address, if EOF occurs on an output device other than a printer.
3. To $$BOMT07 if EOF occurs on an output tape assigned as SYSLST or SYSPCH.
4. To $$BERRTN error recovery routine, if an error occurs on a 2540 or 2520 punch.
5. To $$BERRTN if the upper extent is exceeded on a DASD file.
6. To $$BOPEN to get a new extent if the upper extent is exceeded on a diskette file.

Method: If entered for the first time, the PUT routine performs an I/O operation immediately. (After the initial entry, this I/O operation is bypassed.) When the I/O operation is complete, the routine checks for an EOF condition. If an end-of-file is detected on a unit record device (other than a printer), control is passed to the user's end-of-file routine. If an end-of-file is detected on a magnetic tape device assigned to SYSPCH or SYSLST, the PUT routine fetches phase $$BOMT07 to determine if an alternate device is available. If it is not necessary to handle an EOF condition, the addresses of the two I/O areas are exchanged, another I/O operation is performed, and control is returned to the problem program.

The I/O subroutine first tests to determine the device type, and appropriate action is taken. If the device is a DASD or diskette, a routine is initialized to determine if various specified limits have been exceeded, and to update the seek address and count. When an end-of-extent condition is found on a diskette and no more extents are available, $$BOPEN is fetched to get another extent, and processing continues. The input/output operation is performed, and control returns to the problem program.

If an error occurs and the device is an IBM 2540 or a 2520 punch, the error recovery transient, $$BERRTN, is called. When an error recovery is complete, control returns to the module.

Note: If RETRY=NO is specified as a parameter in the CPMOD macro, the error recovery facility is not present in the module. $$BERRTN is also called to cancel the job if the upper extent is exceeded on a DASD file.

## CPMOD: PUT Macro, 1 I/O Area Chart ND

Objectives:

1. To write a logical record.
2. To read or write FORTRAN ASCII tape records.

Entry: From PUT macro expansion, or from and SD workfile close.

- 2560 MFCM
- 3504 Card Reader
- 3505 Card Reader
- 3525 Card Punch
- 5203 Printer
- 5425 Multifunction Card Unit

DTFDI Macro

The DTFDI macro defines the file for device independent system units and generates a table containing the information necessary to describe the file for processing by the DIMOD logic module (Figure 47).

Multivolume input and output diskette files are supported. For multivolume diskette input files, processing continues until a diskette associated with each extent provided is processed. No volume sequence checking is done. Neither the multivolume indicator nor the volume sequence number in the HDR1 label is examined. Sequencing of volumes is totally controlled by the volume serial numbers on the extent cards.

| Bytes | Bits | Function |
|---|---|---|
| 0-15 (00-0F) | | CCB. |
| 16 (10) | 0-1 | Not used. |
| | 2 | COBOL open; ignore option. |
| | 3 | Not used. |
| | 4 | DTF table address constants relocated by OPENR. |
| | 5-7 | Not used. |
| 17-19 (11-13) | | Address of logic module. |
| 20 (14) | | DTF Type = X'33' |
| 21 (15) | | Open/Close indicators - X'02' = input, X'00' = output. |
| 22-28 (16-1C) | | Symbolic filename. |
| 29 (1D) | | DASD or diskette device indicators<br>X'00' = 2311<br>X'01' = 2314, 2319<br>X'04' = 3330<br>X'08' = 3340 general<br>X'09' = 3340 35MB<br>X'0A' = 3340 70MB. |
| 30-35 (1E-23) | | DASD address of Format 1 label. |
| 36-37 (24-25) | | DASD or diskette volume sequence number. |
| 38 (26) | 0 | Open communications switch.<br>1 = No more extents -- diskettes |
| | 1-3 | Not used |
| | 4 | Always 1 |
| | 5-7 | Not used. |

Figure 47. DTFDI (1 of 3).

| Bytes | Bits | Function |
|---|---|---|
| 74-75 (4A-4B) | | Logic module constants<br>X'0020' DASD output<br>X'0018' DASD input<br>X'0008' Diskette devices<br>X'0000' Non-DASD devices |
| 76-80 (4C-50) | | Count field CCHHR (0CHR0 for diskettes). |
| 81 (51) | | Key length. |
| 82-83 (52-53) | | Data length. |
| 84-87 (54-57) | | Instruction to load IOREG with correct I/O area address. |
| 88-103 (58-67) | | Seek, Search CCWs.<br>Seek, Read/Write CCW for diskette files. |
| 104-111 (68-6F) | | TIC CCW.<br>NOP CCW for diskette output files; unused for diskette input files. |
| 112-119 (70-77) | | Input/output CCW. |
| 120-127 (78-7F) | | Second output CCW. |
| 128-151 (80-97) | | Verify CCWs for output. |
| 152-159 (98-9F) | | Error CCW1. |
| 160-167 (A0-A7) | | Error CCW2. |
| 168-231 (A8-E7) | | Savearea (64 bytes). |
| 232-235 (E8-EB) | | DC A(WLRERR) if WLRERR=Address.<br>B 28(15) if ERROPT= omitted.<br>B 25(15) if ERROPT=SKIP.<br>B 28(15) if ERROPT=IGNORE. |
| 236-239 (EC-EF) | | DC A(ERROPT) if ERROPT=Address.<br>B 0(15) if ERROPT= omitted.<br>B 24(15) if ERROPT=SKIP.<br>B 28(15) if ERROPT=IGNORE. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 47. DTFDI (3 of 3).

than diskette, or SYSLST or SYSPCH assigned to an output tape.

5. To the user's WLR or ERRCPT routine, if specified in the DTFDI macro parameters, to handle other error conditions.

6. To $$BCPEN to get a new extent if the upper extent limit is exceeded on a diskette file.

Method: The PUT routine determines the device type and selects the proper I/O operation to write the record to the file:

1. If the device is a reader or a magnetic tape, an SVC 0 is issued directly.

2. If the device is a DASD, the number of the record to be written is checked and the CCHHR seek address is updated, if necessary, before the SVC 0 is issued. If an end-of-extent condition is found on a diskette, $$BCPEN is fetched to get another diskette extent, and processing continues.

3. If the device is a printer or punch, the control code is determined, converted to EBCDIC code if ASA is specified (refer to Appendix F), and the control function is performed. When the control function is complete, an SVC 0 is issued to write the record.

After the record is written, tests determine if a unit exception or error condition occurred. If a unit exception occurred, the following action is taken:

1. For printers, the unit exception is ignored.

2. For output tapes assigned to either SYSLST or SYSPCH, phase $$BCMT07 is fetched to determine the availability of an alternate device.

3. For all other devices, the address of the user's ECF routine is obtained from the DTF table and used as the return address from the PUT routine.

Other error conditions are handled as specified by the user in the DTFDI macro parameters.

## DIMOD: PUT Macro, 2 I/O Areas Charts PF-PF

Objectives: To write the next sequential logical record to the file from a user specified output area and to provide overlap through the use of two I/O areas.

Entry: From a PUT macro expansion in the problem program.

Exits:

1. To the next sequential instruction in the problem program following the PUT macro expansion after the record is written to the file.

2. To phase $$BCMT07 if an EOF condition is reached on an output tape assigned to either SYSLST or SYSPCH.

3. To the user's EOF routine if an end-of-file condition is reached on devices other than diskette, or SYSLST or SYSPCH assigned to an output tape.

4. To phase $$BERRTN if a punch error occurs.

5. To the user's WLR or ERRCPT routine, if specified in the DTFDI macro parameters, to handle other error conditions.

6. To $$BCPEN to get a new extent if the upper extent limit is exceeded on a diskette file.

Method: The PUT routine, when two I/O areas are employed, functions in the same manner as a PUT with one I/O area. The difference between the two routines is in alternately exchanging the addresses of the two I/O areas each time a PUT macro is issued. Even though two areas are used, the exchange of addresses makes it possible for the user to insert the record to be written at the same address for each PUT macro issued.

## $$BERPTP: 1018 Paper-Tape Punch Error Recovery Charts PH-PJ

Objective: To attempt a recovery from a punch data-check condition or a 1018 paper-tape punch with the Error Correction Feature.

Entry: From an output PTMOD, or from $$BCLOSP if closing a file using two I/O areas.

Exits:

1. To the calling routine after the error is corrected.

2. Automatic job termination if the unrecoverable error must not be ignored (for shifted codes).

7. To $$BOMSG1 to issue the error message, 4883I INVALID LOGICAL UNIT.

8. To $$BODUCP if the file is a diskette file.

Method: This routine is called by the Open Monitor when the file is found to be a DTFCP or DTFDI type (Version 1 DTFCP excluded). Because the logic modules for both file types provide for device independence, this phase is needed to determine the device type associated with the logical unit. The PUB is located for the logical unit and the DTF is initialized according to the device type (refer to Figures 44, 45, and 46).

The LDIOREG routine initializes the user's I/O register.

If the device is tape, a check determines if Job Control has already opened the device. If the file is open and if there are more files to open, the TES processor, $$BOPEN, is fetched. If no more files remain to be opened, control returns to the problem program. If the file has not been opened, $$BOCPT1 is called to open the input files, and $$BOCPT2 is called to open the output files.

If device type cannot be determined, an error message is issued and the job is canceled. If the device is a tape drive and found to be file-protected, an error message is issued and the operator may either insert a ring in the tape reel or cancel.

## $$BOCCP02: Open Device Independent Files - Phase 2 Chart CB

Objective: To open the DTFCP tables of the tape resident system type (Version 1 DTFCP).

Note: A tape resident type DTFCP table is generated for DOS if the DISK= parameter is omitted from the DTFCP macro.

Entry: From $$BOCCP01, or from $$BOMSG1.

Exits:

1. To $$BOPEN, if additional files remain to be opened.

2. To the problem program, if no more files remain to be opened.

3. To $$BOMT06, if the device type is tape and the CP type is input.

4. To $$BOMSG1, to print the following messages:

   4883I INVALID LOGICAL UNIT

   4884D NEED FILE PROTECT RING

Method: This phase functions in the same manner as $$BOCP01, except when it is entered from a message routine. If this is the case, a sense command is issued and a test is made for file protect. If the file is not file-protected, $$BOMT06 is fetched. If it is, a message is printed, via $$BOMSG1.

## $$BOCCP03: Open Device Independent Files - Phase 3 Chart CC

Objective: To open the DTFCP and DTFDI tables for unit record files (Version 1 DTFCP excluded).

Entry: From $$BOCCP01.

Exits:

1. To $$BOPEN if additional files remain to be opened.

2. To the problem program, if no additional files remain to be opened.

Method: The routine at the label UNTRCP initializes an I/O area. The addresses of the I/O area in the CCW and the alternate I/O area are modified to bypass the control character if the device is a printer or a punch. This routine also re-initializes the DTF table if the device is a reader-punch.

If the device is a 1403, 3203, or 5203 printer with the Universal Character SET (UCS), a set mode command is given to suppress data checks.

If the device is a 2560 or 5425, the command code required for the specified hopper is inserted in the first CCW for input files. For output files, two CCWs are loaded; the first for stacker selection, the second for the punch operation.

This phase functions in the same manner as $$BOCCP01 to open unit record files.

$$BOCPT2: Open DTFCP and DTFDI Output Tape Charts RD-RF

Objective: To determine if output tape file is to be created with labels and to fetch the necessary routine if it is; to open the file if it will be unlabeled.

Entries:

1. From $$BOCP01 if the file is defined by DTFCP or DTFDI for output and the device is a tape.
2. From $$BOCPM2 after a message has been issued and a non-cancel reply has been received.
3. From $$BOCPT3 returning for retry.

Exits:

1. To $$BOCPT3, the open transient for labeled output tapes.
2. To the TES processor, $$BOPEN, if more files remain to be opened.
3. To the problem program, if no more files remain to be opened.
4. To the message writer, $$BOCPM2, when an error condition occurs.

Method: Upon initial entry from $$BOCP01, this phase searches for label information on SYSRES as supplied by TLBL (or TPLAB) job control statements and then reads the first record on the output tape to check for a VOL1 label or a tapemark.

The presence of either the VOL1 label or TLBL information requires the creation of a new HDR1 label. This phase tests for file protection, load point, and 1600 BPI and then fetches and transfers control to $$BOCPT3 for the actual label checking and writing.

If an unlabeled output tape file is to be opened, this routine determines if the tape has a tapemark. If the tape has a tapemark, it is retained. If the tape lacks a tapemark, no tapemark is written.

Note: If bit 6=1 in byte 16 of the DTFCP, and if DISK=YES, then input data is in ASCII mode.

$$BOCPT3: Open DTFCP and DTFDI Labeled Output Tape Charts RG-RJ

Objective: To open labeled output tape files.

Entries:

1. From $$BOCPT2 if output tape is already labeled or if label information was supplied by TLBL [TPLAB] job control statements.
2. By return from $$BOCPT2 after a message has been issued, and a non-cancel reply has been received.

Exits:

1. To the TES processor, $$BOPEN, if more files remain to be opened.
2. To the problem program if no more files remain to be opened.
3. To the message writer, $$BOCPM2, when an error condition occurs.
4. To phase $$BOCPT2 for retry if entry to this phase was neither from $$BOCPT2 nor a return from the message writer.
5. To the Standard Volume Label Rewriter, $$BCNVCL, if the volume label must be rewritten according to the user specified density.

Method: After relocating the CCB's and CCW's for the tape I/O routines, the mode and density are set for the user's file tape. Then the instructions necessary to search for TLBL information on SYSRES are initialized with necessary disk and storage addresses. The Communication Region is then tested to see if entry is a return from the message writer. If it is, reentry will be to the label PROCRUN2 to execute the next subroutine of the procedure group unless a more specific return address exists in the linkage register.

If entry was from $$BOCPT2, the proper series of subroutines are executed as chosen by the procedure pointer passed to this phase in the register equated as PROCPTR.

Effectively, the action of these subroutines is such that for standard labeled output tapes:

1. Expiration date is checked.
2. If label information is provided by TLBL (or TPLAB) cards, a new HDR1 label is written.
3. If label information is not provided, a dummy header is written.
4. No additional standard header or user header labels are written.

## $$BCLOSP: Punch File Close Charts SC-SD

Objective: To close DTFCP, DTFDI, and DTFCD punch files and recover possible errors occurring when the last card in the file is punched. To close DTFPT and to check the last record if the output file has two I/O areas.

Entries:

- From the Close Monitor, $$BCLCSE, to the entry point BEGINRTN.
- From $$BERPTP, if an error has been detected on a 1018 paper-tape punch using a DTFPT output file with two I/O areas during the last punch operation and if error recovery procedure is in process.

Exits:

- To the Close Monitor $$BCLOSE if there are additional files to be closed.
- To the problem program if no additional files remain to be closed.
- To $$BERPTP if an error occurred during the last punch operation on a 1018 paper-tape punch using a DTFPT output file with two I/O areas.

Method: This routine first determines the device type. If the device is a 1442 punch or a 1442 reader-punch, the routine exits to the Close Monitor or to the problem program if no additional files remain to be closed. Depending upon the file type and whether there are one or two I/O areas, the following actions are taken:

1. For a DTFCD file with two I/O areas where the device is a 2540 punch, an error, if it has occurred, is corrected on the card preceding the last card. Then, any error detected on the last card is corrected.
2. For a DTFCD file with one I/O area where the device is a 2540 punch, any error detected on the last card is corrected.
3. For a DTFCD file with two I/O areas where the device is a 2520 punch, any error detected on the last card is corrected.
4. For a DTFCD file with one I/O area where the device is a 2520 punch, no error recovery is needed and the routine takes the proper exit.
5. For a DTFCP and DTFDI files (one or two I/O areas) where the device is a 2540 punch, error recovery is performed first on the card preceding the last card punched. Then, any error detected on the last card is corrected.
6. For a DTFCP and DTFDI files (one or two I/O areas) where the device is a 2520 punch, any error on the last card is corrected.
7. For a DTFPT output file with two I/O areas, the checking of the last record is performed, and in case an unrecoverable error occurred, a channel program is reissued to punch the entire erroneous record if the ERROPT operand is coded in the DTFPT.

In all cases, whenever an error card is repunched correctly, the message 4000I RETRY is printed on SYSLOG to inform the operator that the error was retried.

## $$BCCPM1 and $$BOCPM2: DTFCP/DTFDI Message Writers Chart SE

Objective: To write messages on SYSLOG and to process operator responses to the messages issued.

Entry: From the DTFCP and DTFDI tape open routines, $$BOCPT1 and $$BCCPT2.

Exits:

1. For certain messages, automatic job termination.
2. For CANCEL response or no console, job canceled.
3. For IGNORE, NEWTAP, RETRY responses or information-type messages, the next phase to be executed is fetched.

$$BCCPM1 Messages:

4111A - NO VOL1 LBL FOUND

4112A - VOL SERIAL NO. ERROR

4113D - NO HDR1 LABEL FOUND

4114A - FILE SEQ NO. ERROR

4115A - FILE SER. NO. ERROR

4116A - VOLUME SEQ. NO. ERROR

4132D - ERROR IN FILE ID

4133D - ERROR IN HDR LBL

Diskette input/output files are processed by the Sequential Access Method. These files, defined by the DTFDU macro, are either input or output data files.

A diskette file contains records that are processed from a beginning diskette address and that continue in sequential order through the records on successive tracks, cylinders, and volumes to the ending address.

A diskette file is contained within one or more sets of limits called extents. These extents are specified in the file label on the diskette for input files, and are computed and stored in the file label for output files by the open routines. The user can identify the files to be processed through the // DLBL and // EXTENT job control cards. The records within each extent must be adjacent on a volume. Only one extent is allowed per volume, but files may cross diskette volume boundaries. If the logical file consists of more than one extent, each extent is accessed in the sequence specified by the user.

The data handling logic modules for files defined for logical IOCS by the DTFDU macro are provided by the associated module generation macro, DUMODFx, where x is determined by the function of the file.

Diskette files are opened and closed by logical transient routines that are fetched by the open and close monitors (see DOS/VS LIOCS Volume 1). The open routines provide procedures for checking each file before any records are processed. The close routines provide procedures for terminating each file after all records are processed.

Diskette files can also be defined for physical IOCS if the user intends to use physical IOCS macros (such as EXCP and WAIT). These files are defined by a DTFPH macro. In addition, diskette files can be defined by the device independent macros, DTFCP and DTFDI. These files are described under "Device Independent Files".

## Record Format

Logical records in a diskette file can only be in fixed-length format. The diskette is initialized to 128-byte sectors; therefore the maximum record size may not exceed 128 bytes. The format of the record is specified by the user in the DTFDU macro instruction which defines the file.

## STORAGE AREAS

### INPUT/OUTPUT AREAS

The logical IOCS GET-PUT macro instructions allow the programmer to use one or two I/O areas and to process records either in a workarea or in an I/O area.

Using DTFDU, it is possible to logically block the individual records in the I/O areas by command chaining the input and output operations. This allows logical IOCS to read or write multiple individual records when the device is being addressed. In subsequent discussions, the term "chained records" is used to describe this method of reading and writing.

When chained records are to be processed in an I/O area with no workarea specified (or when non-chained records are to be processed in two I/O areas with no workarea specified), the DTFDU macro instruction must include the IOREG parameter. Logical IOCS uses this register to specify the address of the logical record that is currently available for processing by the problem program.

### MODULE SAVE AREAS

If RDONLY=YES is included in the module generation macro, the module is reentrant and must never be modified by the problem program. Each DTF referencing the module must have a 72-byte doubleword aligned save area associated with it. This save area is used by the module during execution. The address of the save area is passed to the module in register 13.

If the module is to be shared by DTFs in different tasks, the module must be made reentrant. This is done by associating a unique save area with each DTF.

For diskettes, the save area contains the user's general registers, switches, and other information needed by the module. Figures 48 and 49 illustrate the format of the save area for each logic module.

DTFDU MACRO

To process a diskette file of data records, the file must first be defined by the declarative macro DTFDU (Define The File for Diskette Unit). This macro describes the characteristics of the logical file, indicates the type of function being performed, defines the format of the record being processed, and specifies the storage areas and routines used for the file. A DTF table is then generated according to the parameters specified in the operands of the DTFDU macro instruction. Figure 50 illustrates the DTF table generated for diskette files.

| Bytes | Bits | Contents | Function |
|---|---|---|---|
| 0-15 (0-F) | | | Command Control Block (CCB) |
| 16 (10) | 0-3 | B'0000' | Not used. |
| | 4 | | 1 = DTF relocated by OPENR. |
| | 5-7 | B'000' | Not used. |
| 17-19 (11-13) | | | Address of logic module. |
| 20 (14) | | X'1A' | DTF type for OPEN/CLOSE (X'1A' = diskette file). |
| 21 (15) | 0 | | 1 = Command chained file. |
| | 1-2 | B'00' | Not used. |
| | 3 | | 1 = Workarea specified. |
| | 4 | B'0' | Not used. |
| | 5 | | 1 = Open; 0 = Close. |
| | 6 | | 1 = Input; 0 = Output. |
| | 7 | | Not used. |
| 22-28 (16-1C) | | | Filename. |
| 29 (1D) | | X'06' | Device type code. (X'06' = 3540). |
| 30-35 (1E-23) | | C'00CHR00' | Address of HDR1 label in VTOC. |
| 36-37 (24-25) | | | Volume sequence number. |
| 38 (26) | | | Open communications byte |
| | | | Input File |
| | 0 | | 1 = No more extents |
| | 1-2 | B'00' | Not used |
| | 3 | | 1 = Exit for user's EOF routine |
| | 4 | | 1 = Next extent on new volume |
| | 5-6 | | Not used |
| | 7 | | 1 = Extent switch. |
| | | | Output File |
| | 0 | | 1 = No more extents |
| | 1 | | 1 = Extents needed at Close time |
| | 2-3 | B'00' | Not used |
| | 4 | | 1 = Next extent on new volume |
| | 5 | | 1 = Extent entered via console |
| | 6-7 | | Not used. |

Figure 50. DTFDU Table (Part 1 of 3)

| Bytes | Bits | Contents | Function |
|---|---|---|---|
| 92-95 (5C-5F) | | | Logical record size. |
| 96-99 (60-63) | | | Address of last byte of the I/O area. |
| 100 (64) | | | Logical indicators. |
| | 0 | | 1: ERROPT=address |
| | 1 | | 1: ERROPT=IGNORE |
| | 2 | | 1: ERROPT=SKIP |
| | 3 | | Not used |
| | 4 | | 1 = Two I/O areas |
| | 5-7 | | Not used. |
| 101-103 (65-67) | | | Address of user's error handling routine. |
| 104 (68) | | | CCW count (write command only). |
| 105 (69) | | | Allowed operations |
| | 0 | | 1 = Allow read commands |
| | 1 | | 1 = Allow write commands |
| | 2 | | 1 = Suppress unit check or C4/C6 |
| | 3-7 | B'00000' | Not used. |
| 106 (6A) | | X'00' | Sector factor (X'00'=128). |
| 107 (6B) | | X'00' | Reserved. |
| 108 (6C) | 0 | | 1 = Write protect |
| | 1 | | 1 = No feed at EOF |
| | 2 | | 1 = Check multivolume sequence |
| | 3 | | 1 = Multivolume file |
| | 4 | B'0' | Not used |
| | 5 | | 1 = C6s written (update ERMAP) |
| | 6 | | 1 = READ/WRITE security |
| | 7 | B'0' | Not used. |
| 109-111 (6D-6F) | | X'000000' | Not used. |
| 112-119 (70-77) | | | Feed CCW. |
| 120-127 (78-7F) | | | Define cps CCW (output); 8X'00' (input). |
| 128-135 (80-87) | | | Seek CCW. |
| 136-143 (88-8F) | | | TIC CCW. |
| 144-X (90-Y | | X=143+8*(# of CCWs) Y=8F+8*(# of CCWs) | Read/write data CCWs; 1, 2, 13, or 26 read/write CCWs. |
| X+1 (Y+1) | | | NOP CCW (output only). |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 50. DTFDU Table (Part 3 of 3)

| Bytes | Bits | Function |
|---|---|---|
| 39 (27) | | Sequence number of current extent being opened. |
| 40 (28) | | Sequence number of last extent opened (not a console extent entry). |
| 41-43 (29-2B) | | Not used. |
| 44-47 (2C-2F) | | Address of IOAREA1. |
| 48-51 (30-33) | | Not used. |
| 52-53 (34-35) | | X'0000' |
| 54-57 (36-39) | | Extent upper limit (CCHR). |
| 58-59 (3A-3B) | | Not used. |
| 60-63 (3C-3F) | | Extent lower limit (CCHR). |
| 64 (40) | | Record number. 1=Input, 0=Output. |
| 65-67 (41-43) | | Not used. |
| 68-71 (44-47) | | CCHR control bucket. CCHR = X'CC49001A' for 3540 (output only). |
| 72 (48) | | Record number. |
| 73 (49) | | X'10' - multivolume file (input)<br>X'40' - last volume on multivolume file (input). |
| 74 (4A) | | Record size (maximum of 128). |
| 75 (4B) | | Not used. |
| 76-80 (4C-50) | | CCHR bucket = extent lower limit and record number (output). |
| 81-83 (51-53) | | Not used. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 51. DTFPH Table for Diskette (Part 2 of 2)

3. $$BOPEN is fetched to open the next extent.

If end-of-extent has been exceeded and there are no more extents, the following action is taken:

1. The ERMAP record is updated if bad spot records were written on the diskette because of an ERET RETRY situation.

2. $$BOPEN is fetched.

If the ERROPT parameter has been specified, errors are processed as each record or chain of records is written. If ERREXT is specified, additional errors are returned to the problem program for further processing.

DUMCDFC: Close Processing Chart UE

Objective: To write any remaining records in sequence on the diskette file.

Entry: From the close transient, $$BODIC4.

Exit: To $$BODIC4, via an SVC 9.

Method: This routine sets on the partial block switch and determines if there are any logical records to be written to the file. If so, the chain of CCWs to write the proper number of records is set up; then the PUT routine is given control to write the short chain to the file. After that the CLOSE routine regains control, the module switches are set off, and a branch is made to the PUT routine to exit to the close transient $$BODIC4.

INITIALIZATION AND TERMINATION PROCEDURES

When a diskette file (DTFDU) is opened, and the file is on more than one volume, only one extent is processed at a time, so only one volume need be online at a time.

Job Control accepts label information supplied by DLBL and EXTENT statements. Job Control stores this DASD label information on the SYSRES DASD label information cylinder. The IES Processor, $$BOPEN, prepares to read the label information from the SYSRES label information cylinder into the logical transient area, and then fetches $$B35400.

The diskette open logical transients read the DASD label information from SYSRES into storage. The format of the SYSRES DASD label information is illustrated in DOS/VS ICCS Volume 1, SY33-8559. If the file is an input file, the open transients compare the file label information with the SYSRES DASD label information to determine if the logical file is correct and if the serial numbers are equal.

If the logical file is an output file, the open logical transients create file labels and write them in their appropriate location. Extent limits are determined and overlapped, expired file labels are deleted.

When a file is closed, the close logical transient determines whether a block of data remains to be processed. If so, the logic module is re-entered to complete processing. The file labels are updated and rewritten if the file is an output file. Control returns to the close monitor or the problem program.

DISKETTE OPEN/CLOSE LOGIC

Open Diskette Files, General Chart 08

When a diskette file is processed (DTFDU specified), OPEN initially takes care of the following functions:

1. The standard label(s) on the volume, or on the first volume of a multivolume file, is checked.

2. The first extent on the first volume is located and made available for processing.

Logical IOCS processes one extent at a time in the sequence specified by the user's EXTENT statements. When logical IOCS detects the end of the current extent, it branches to the end-of-extent routine. OPEN then locates the next extent specified by the control statements and makes it available for processing. For each subsequent extent used by the file, OPEN checks the standard labels on that new volume (see Charts UC and UD for general OPEN flow).

OPEN (Input Diskette) General Flow Chart 09

If the file to be opened is a normal input file, the extents are read and checked as needed. The file labels are checked against the DLBL information. The open indicator for the file is turned on and control returns to the user.

| | 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 | S |
| DTFPH | A | A | N | A |

A = Always feed at close time.
S = User can suppress feed at close time.
N = Never feed at close time.
NA= Not applicable.

Figure 52. This table indicates if diskettes are fed at close time

$$B3540I: Diskette Open Input, DLBL Extents Charts VA-VC

Objective: This phase is used to control the sequence of operations required for opening each file extent. It also provides an entry to the user's end-of-file routine, if specified, when the end of the last extent is reached or if the multivolume indicator in the HDR1 label indicates last extent for DTFDU. Furthermore it checks for the CCBCI Open/Ignore function.

Entry: From $$B35400. Reentry from $$BCDMSG.

Exits:

1. To $$BODIO1 to process volume labels.
2. To $$BOPEN if the last DLBL extent has been processed and another file is to be opened.
3. To the user's end-of-file routine.
4. To $$BODMSG for operator communication.

Method: If a system unit is being opened and the DTF does not indicate open, this routine gets extent information for the DTF from the DIB. Otherwise, both system and programmer units are handled identically.

$$B3540I tests for the availability of DLBL extents. If no more extents are available or if the multivolume indicator indicates last volume, an exit is made to the user's end-of-file routine. If the file has been opened previously, the next consecutive DLBL extent to be opened is read.

If the COBOL Open/Ignore function has been specified and the device is unassigned or assigned to IGN, the open is bypassed. If the device is assigned, open is continued.

If the Open/Ignore function has not been specified and the device is unassigned or assigned to IGN, the job is canceled. Otherwise, open continues.

This routine also determines whether the assigned device is the correct device and ensures that only one file is open on the device. If this is not the case, the job is canceled; otherwise, this transient passes control to $$BODIC1.

$$BCDIC1; Diskette Volume Label Processor, Charts VD-VE

Objective: This phase reads the volume label and checks it for validity. It also ensures that the proper volume is mounted and requests further information from the operator if a secured volume is being opened.

Entries:

1. $$B3540I, $$B3540C, and $$BCDIO8.
2. Reentry from $$BCDMSG and BCDSMO.

Exits:

1. To $$BCDIO2 if open output.
2. To $$BCDIO5 if open input.
3. To $$BCDSMO if a secured volume.

If the Open/Ignore option has not been specified and the device is unassigned or assigned to IGN, the job is canceled; otherwise, the open procedure is continued.

This routine also determines whether the assigned device is the correct device and ensures that only one file is open on that device. If not, the job is canceled; otherwise, this transient fetches $$BCDIC1.

$$BCDIC2: Diskette Open Output, Determine Extents and Delete HDR1 Labels Charts WD-WF

Objective: To determine the extent limits for the file on the diskette. To prevent a duplicate file being created, to delete both overlapped and duplicate expired files, and to determine the new HDR1 label address.

Entry: From $$BCDIO1.

Exits:

1. To $$BODIO3 to create a new HDR1 label.
2. To $$BODMSG for operator communication.

Method: Two passes are made through all of the HDR1 records.

Pass 1. All HDR1 labels are examined to determine if the files are write-protected or unexpired. If either of these conditions exists, the file name is compared with the file name of the new file; if they are equal, the job is canceled. Otherwise, the open continues. The highest upper extent limit track plus 1 of any write-protected or unexpired file is made the lower extent limit of the new file.

Pass 2. All HDR1 labels are re-read and the first delete control record that is encountered (either read or written) is made the address of the new HDR1 label. All overlapped files are deleted. All expired, non-write-protected files with duplicate file names area also deleted. Then this transient fetches $$BCDIO3.

$$BCDIC3: Diskette Open Output, Create/Write New HDR1 Label Charts WG-WJ

Objective: To build and write a new HDR1 label for the file being opened.

Entry: From $$BODIO2.

Exits:

1. To $$BCDIC7 to initialize the DTF.
2. To $$BCDMSG for operator communication.

Method: This routine verifies that at least one complete track is available and that a new HDR1 label address was found. If either of these conditions is not met, the job is canceled; otherwise, the new HDR1 label for the file is created and written out on the diskette.

$$BCDIC7: Diskette Open Output, Initialize DTF Table Charts WK-WI

Objective: To update the DTF table.

Entry: From $$BODIO3.

Exits:

1. To $$BCPEN to open the next file.
2. To $$BCDIC4 if Close needed an extent.
3. To $$BCDMSG for operator communication.
4. To the problem program.

Method: This routine posts appropriate extent information in the DTF table and in the DIB in case of a system file.

$$BCDUCP: Diskette DTFCP/DTFDI Open, Set Up Skeleton DTF Charts WM-WN

Objective: To prepare a compiler of device independent DTF for a diskette file, so that normal diskette open phases can be used to complete the open procedure.

Entry: From $$BOCP01.

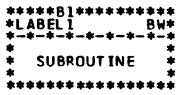
Exit: To $$BOPEN1 to call the proper diskette open phase.

## EXPLANATION OF FLOWCHART SYMBOLS

**DESCRIPTION**


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

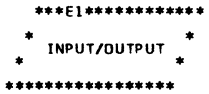
*B2
IF ANY ADDITIONAL EXPLANATION IS REQUIRED, ITS LOCATION ON THE CHART IS IDENTIFIED BY AN ASTERISK AND THE BLOCK ID.




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.

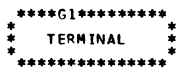

An instruction, or group of instructions, that changes portion of a routine or initializes a routine for a given condition.


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.

**EXAMPLE**

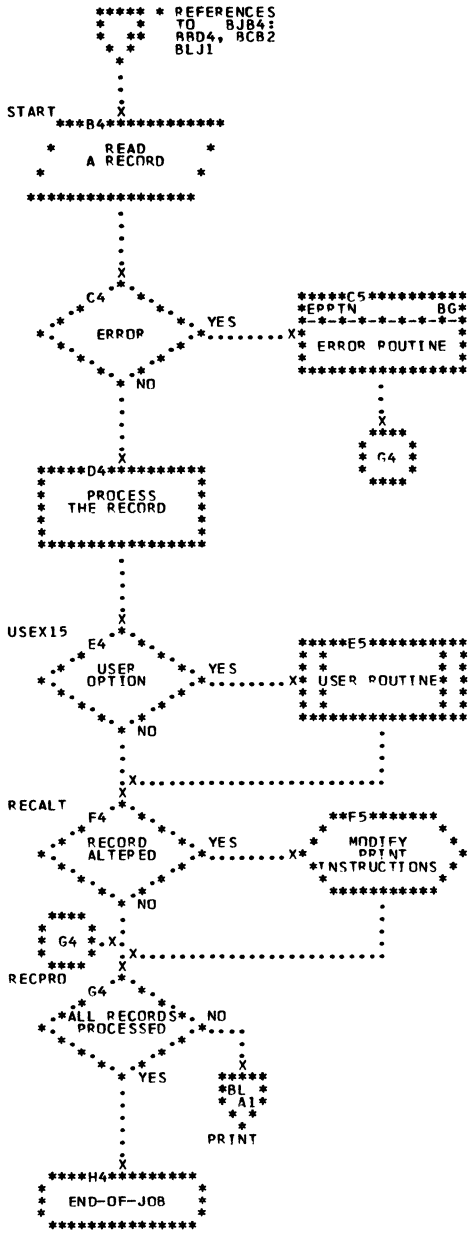
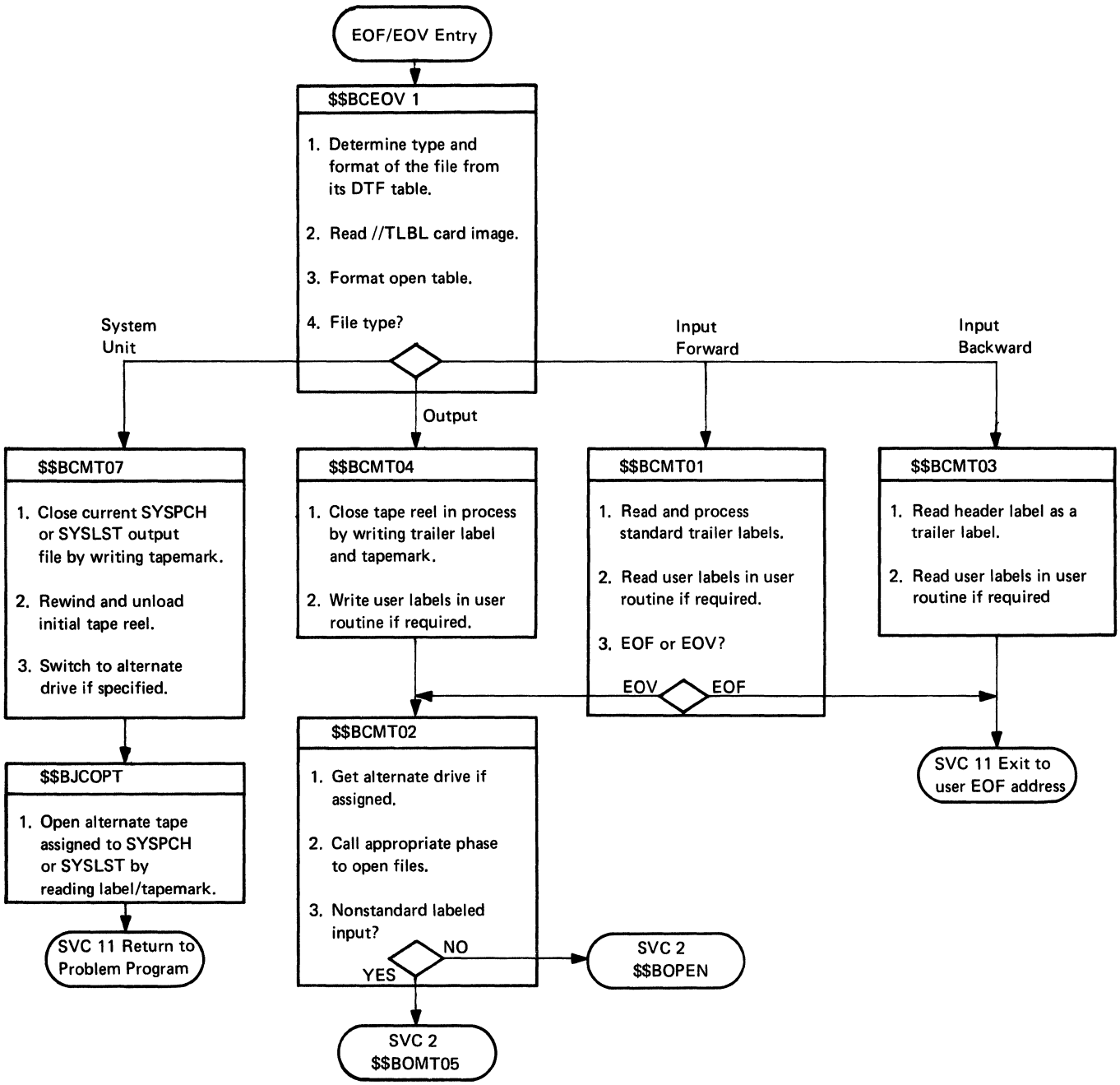


Chart 02. Magnetic Tape Clcse and EOF/EOV Routine



## Chart 04. Sequential Access DASD Open, Input Files

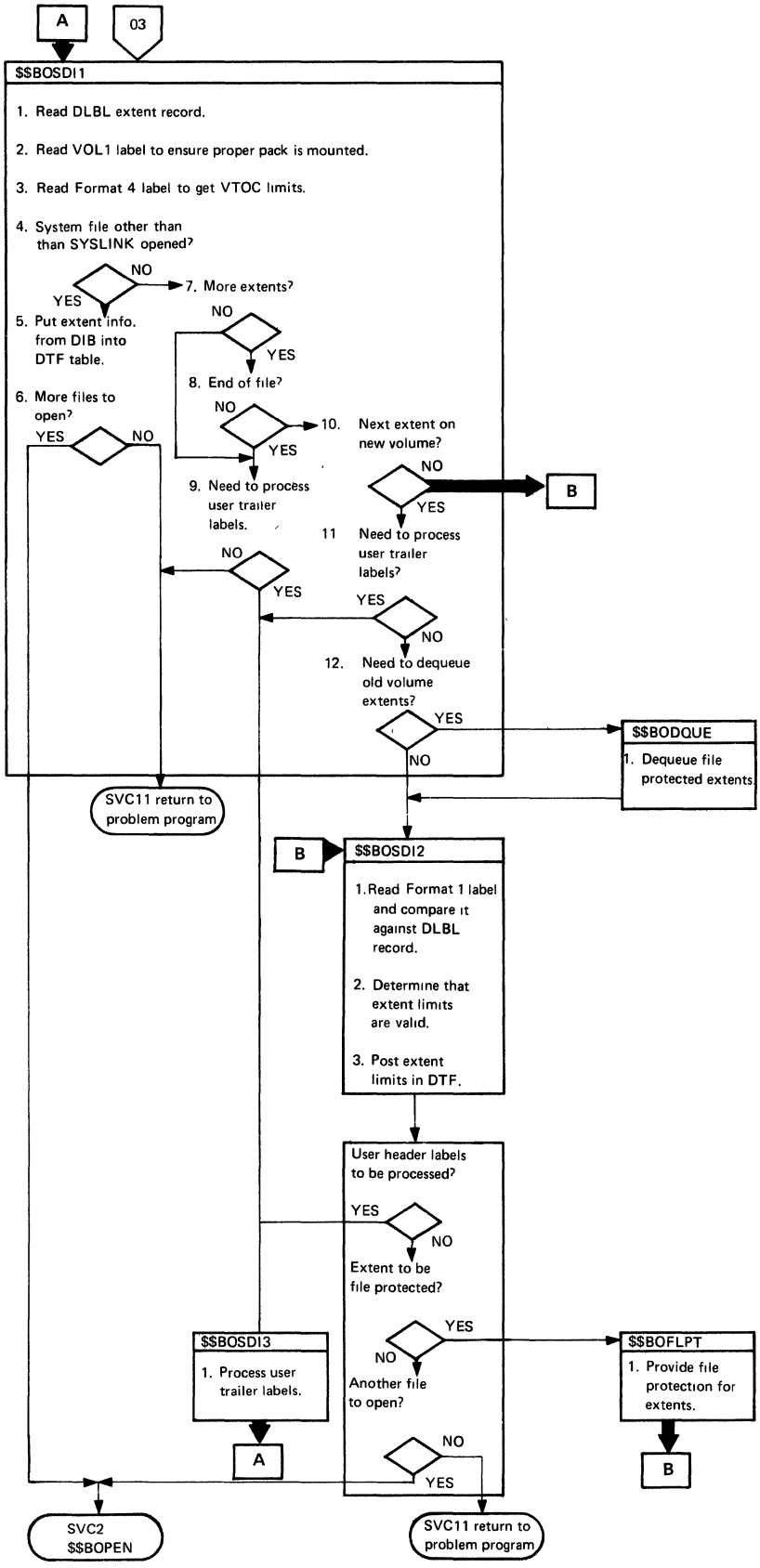


Chart 06. Sequential Access DASD Open, Output Files (2 of 2)

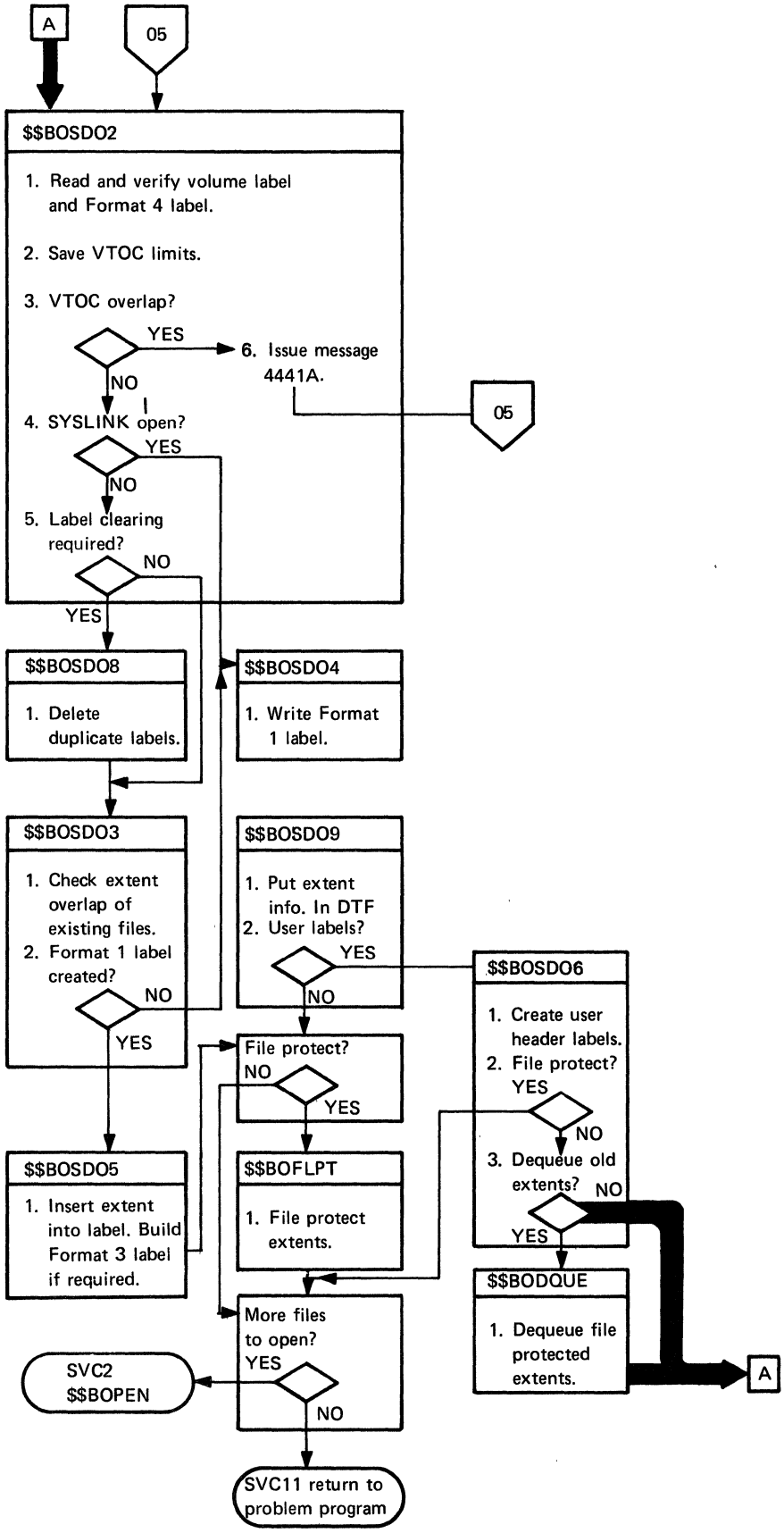
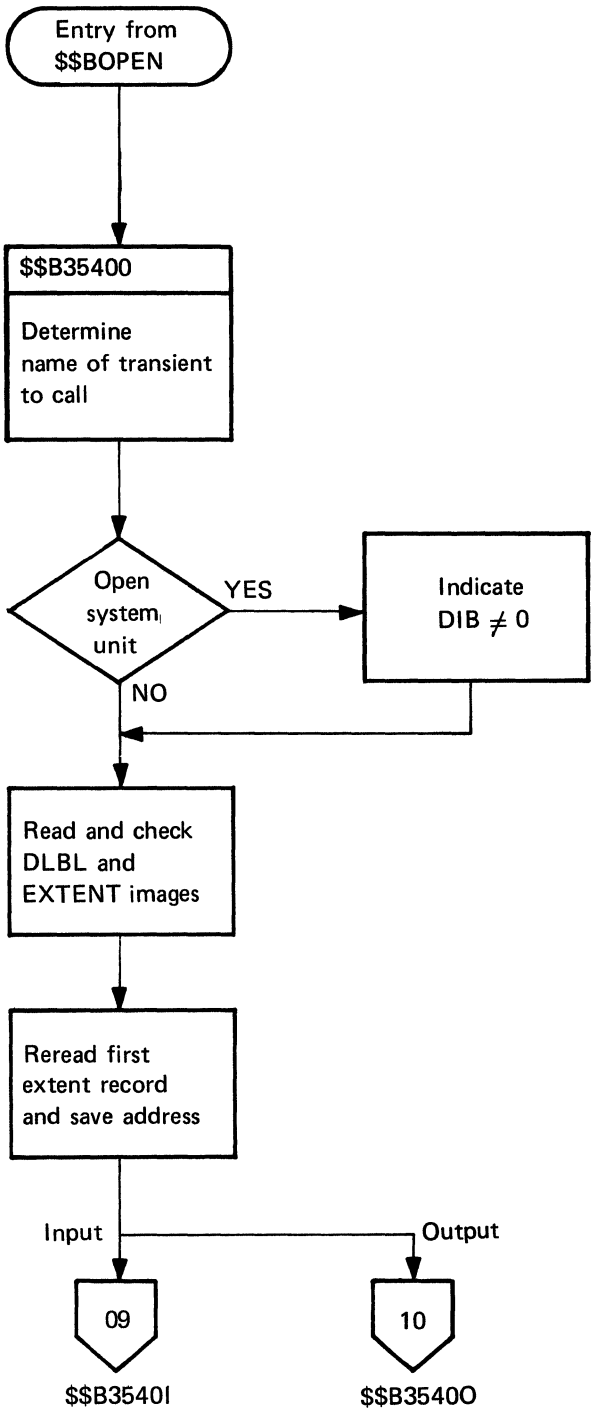


Chart 08. Diskette Open, General Flow



## Chart 10. Diskette Open, Output Files

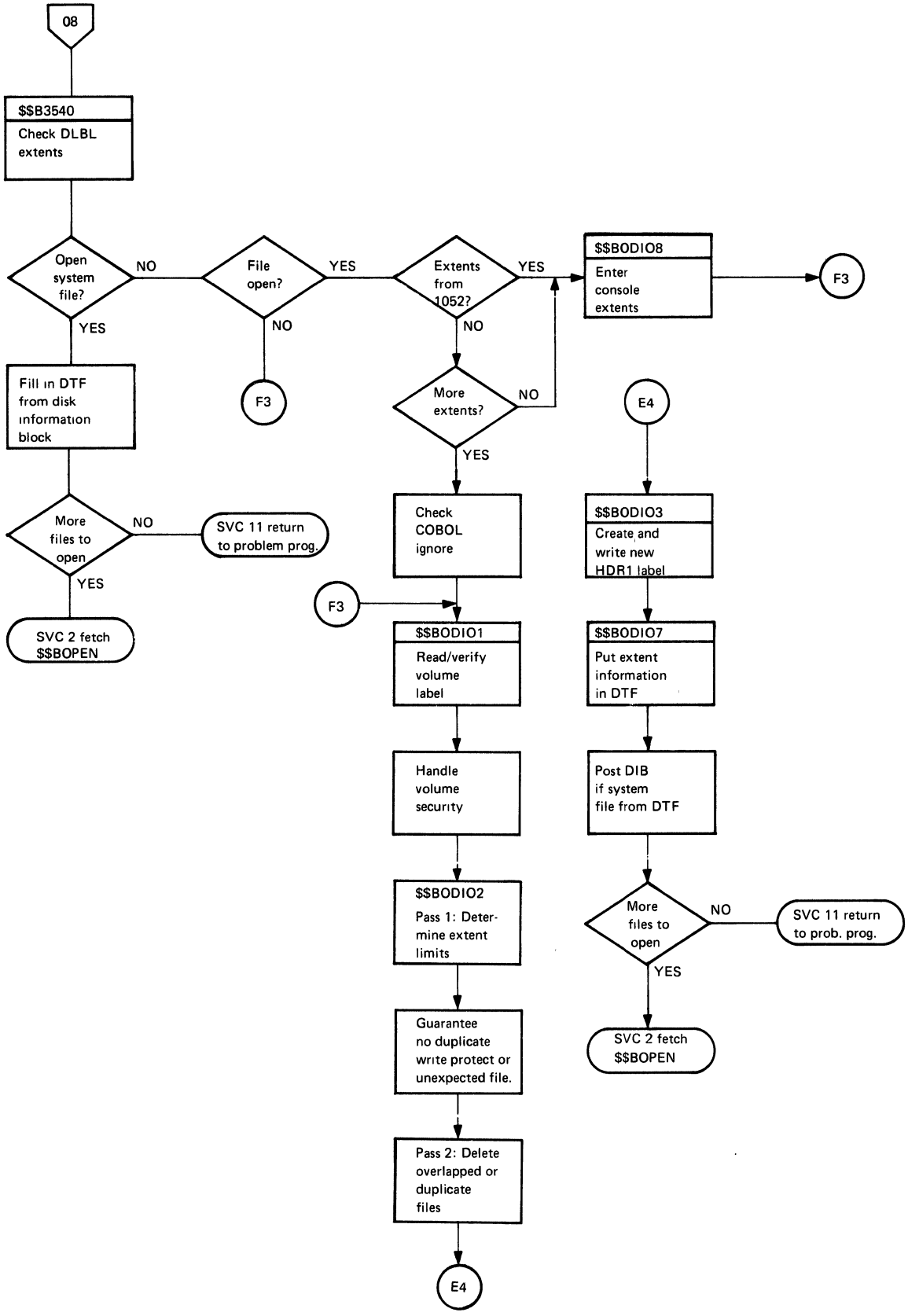


Chart AA. $$BOUR01: Cpen Unit Record (1 cf 2)

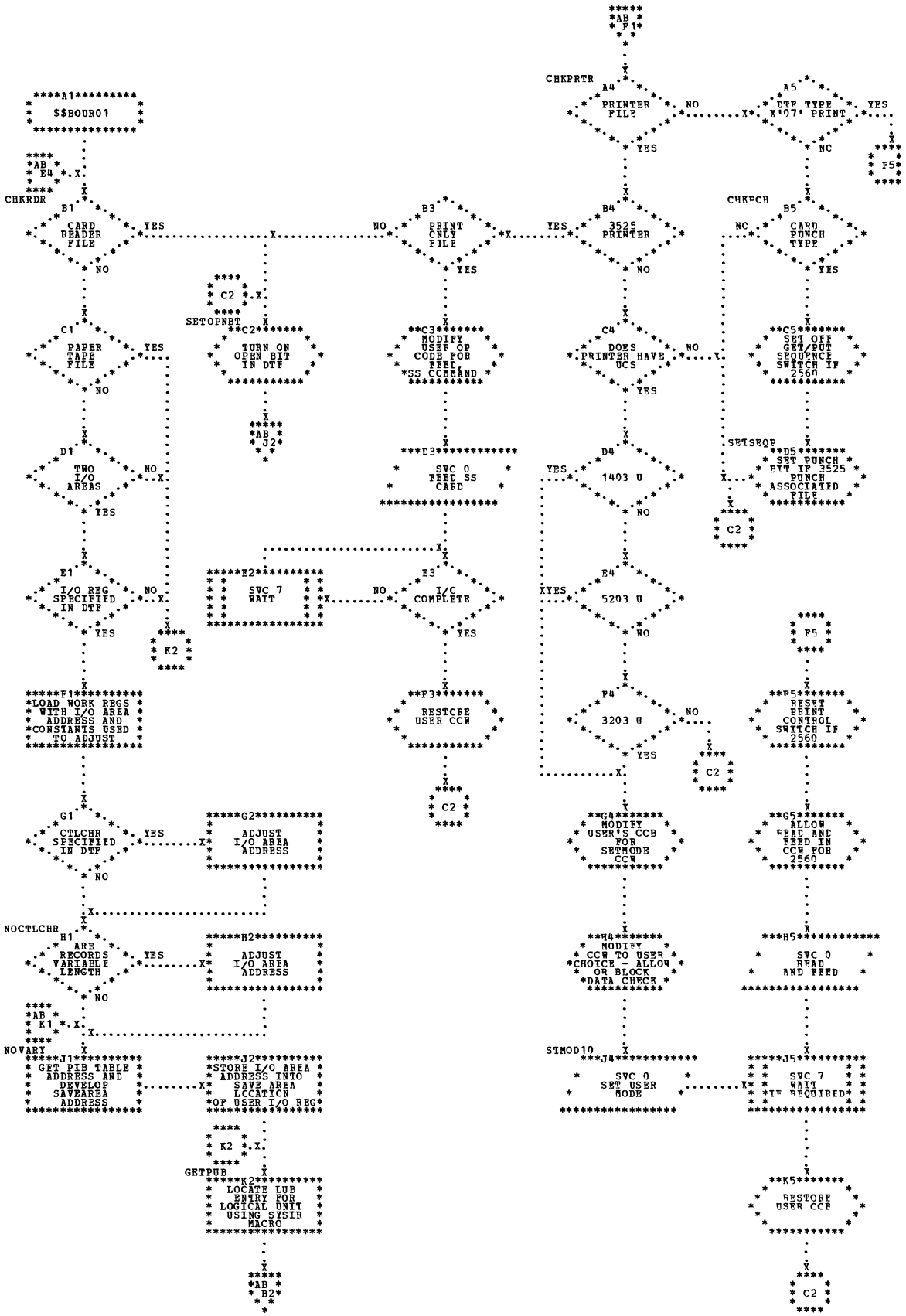
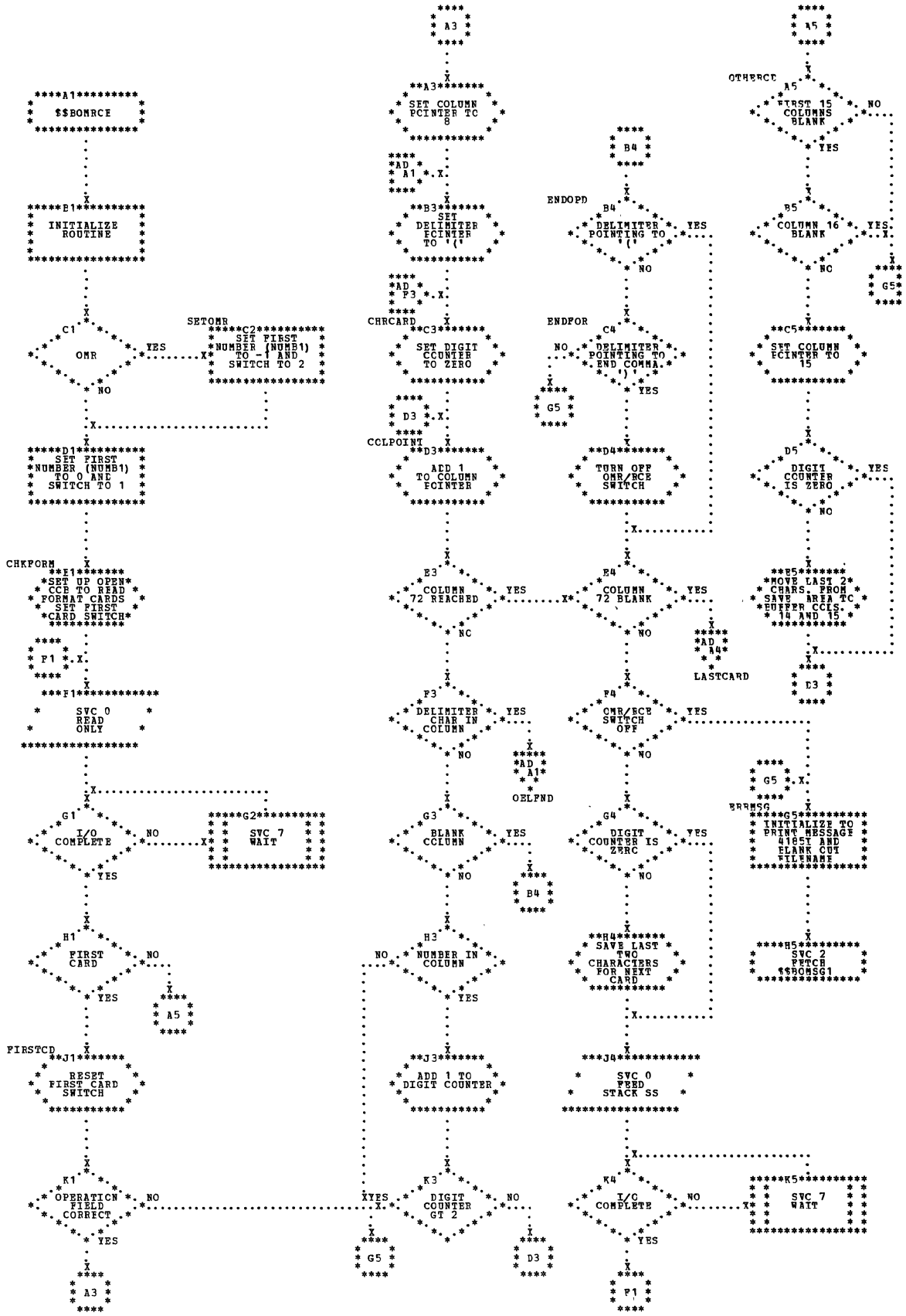


Chart AC. $$BCMRCE: CMR and RCE Open Routine (1 of 2)



## Chart AE. CDMOD: CNTRL Macro

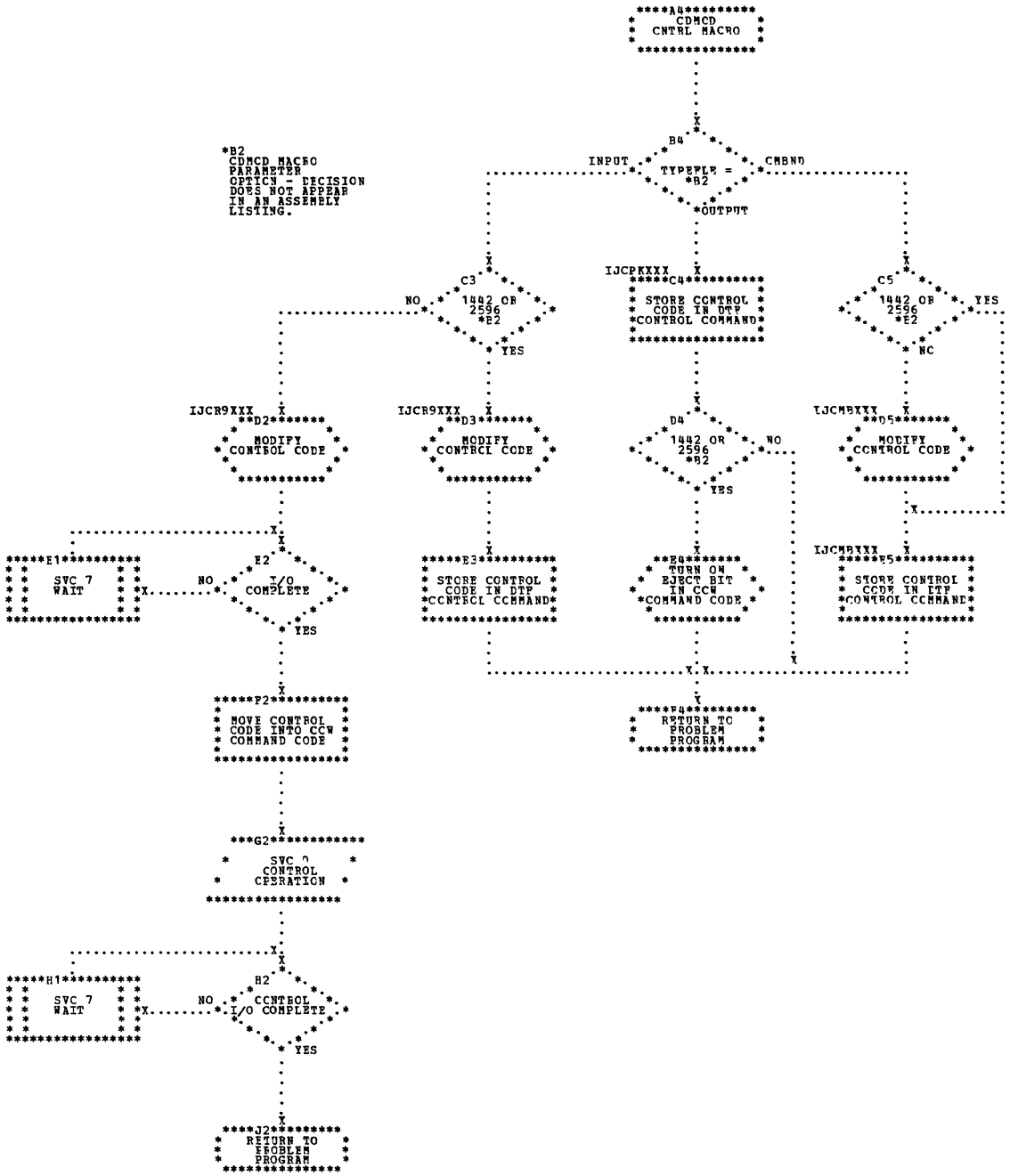


Chart AG. CDMOD: GET Macro (2 of 3)

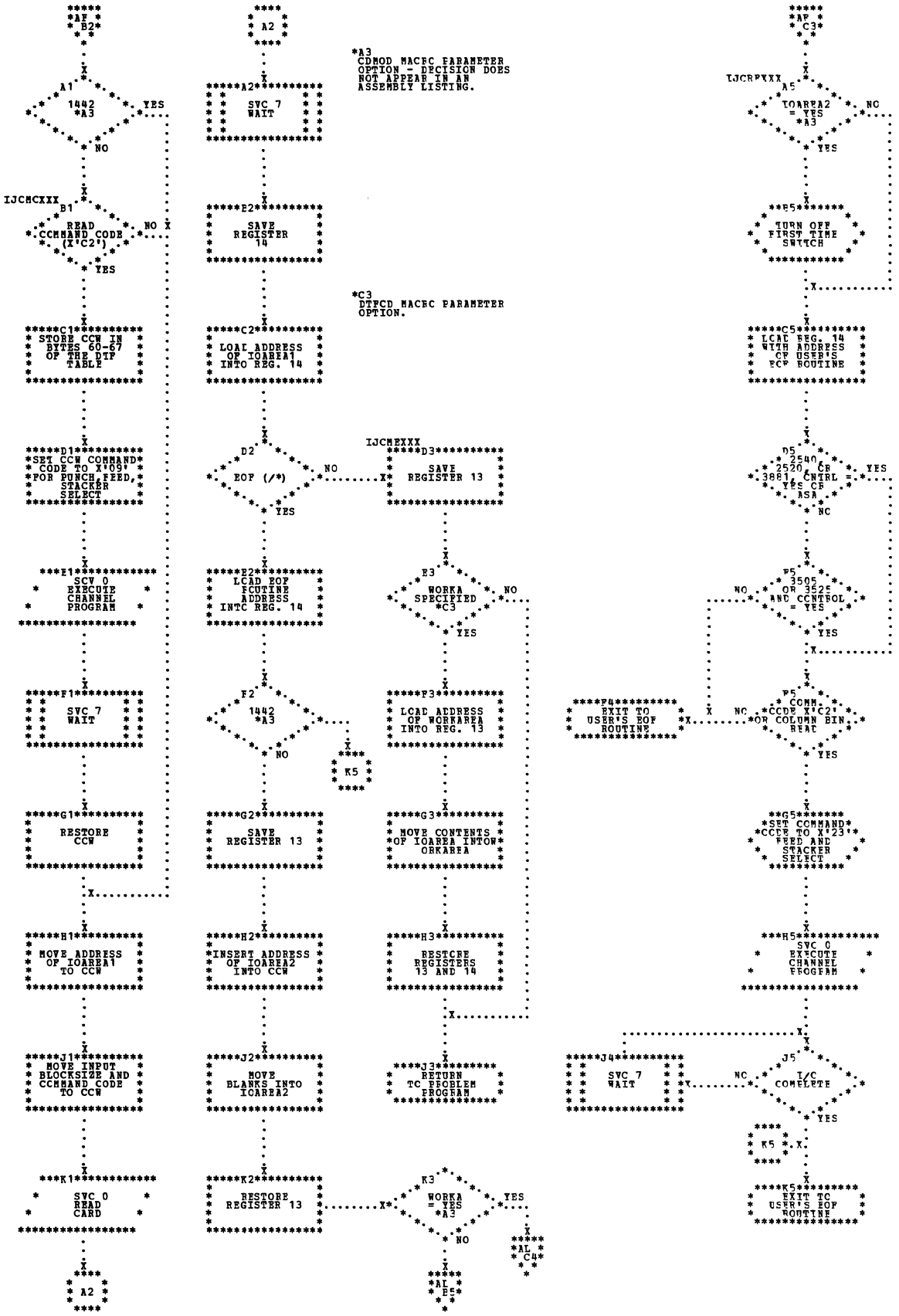


Chart AJ. CDMOD: PUT Macro (1 of 4)

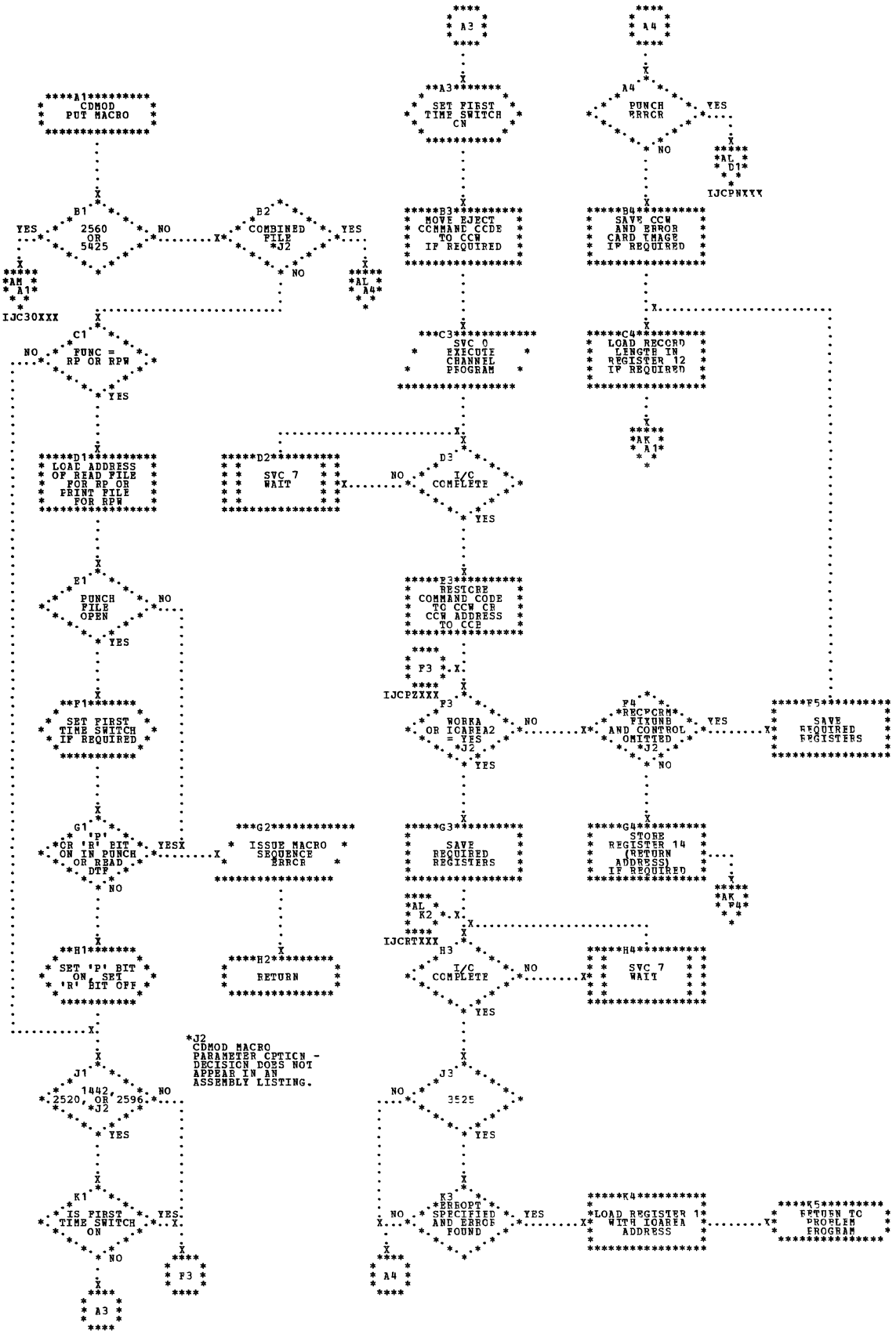


Chart AL. CDMOD: PUT Macro (3 of 4)

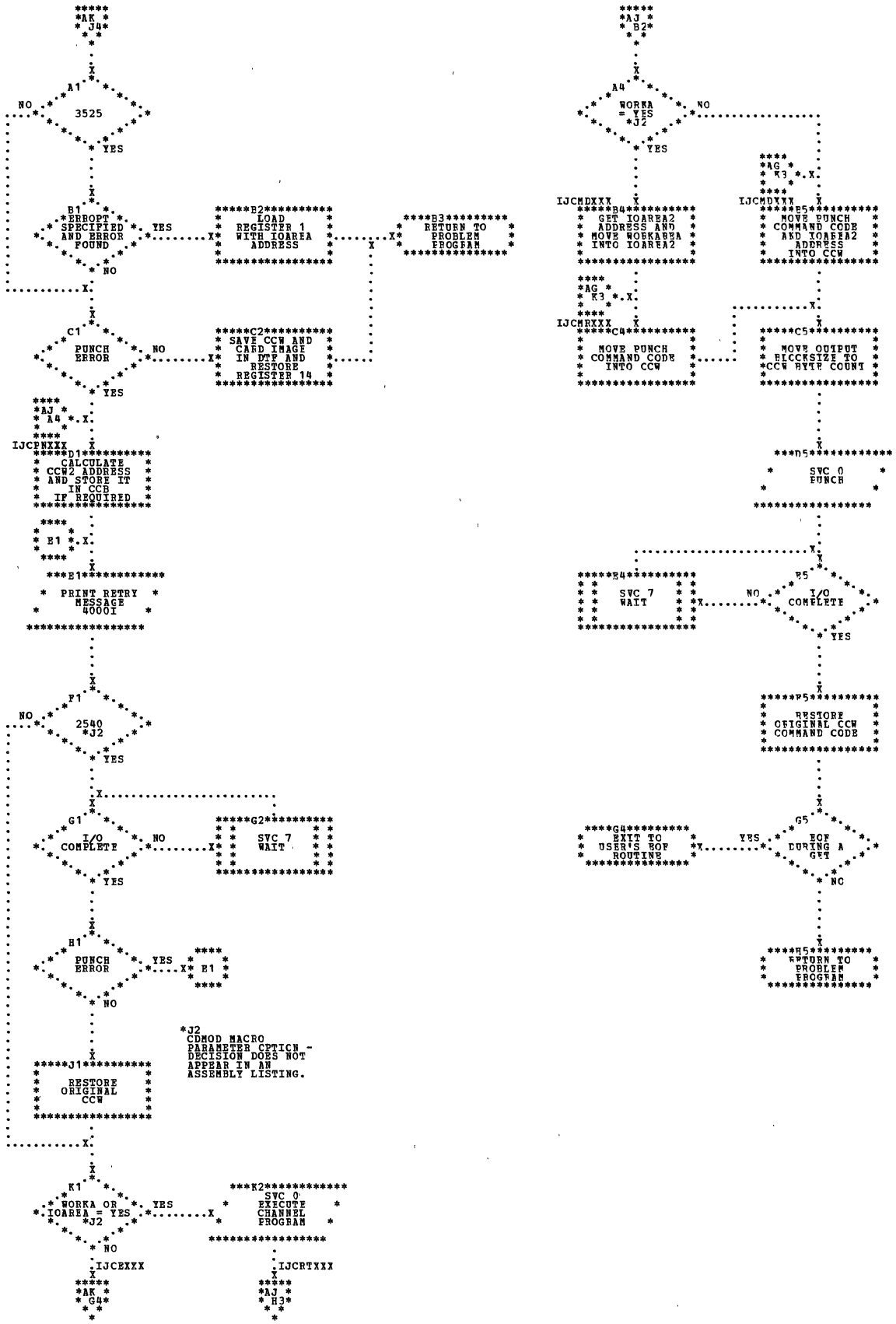


Chart AN. DTFCN: GET Macro

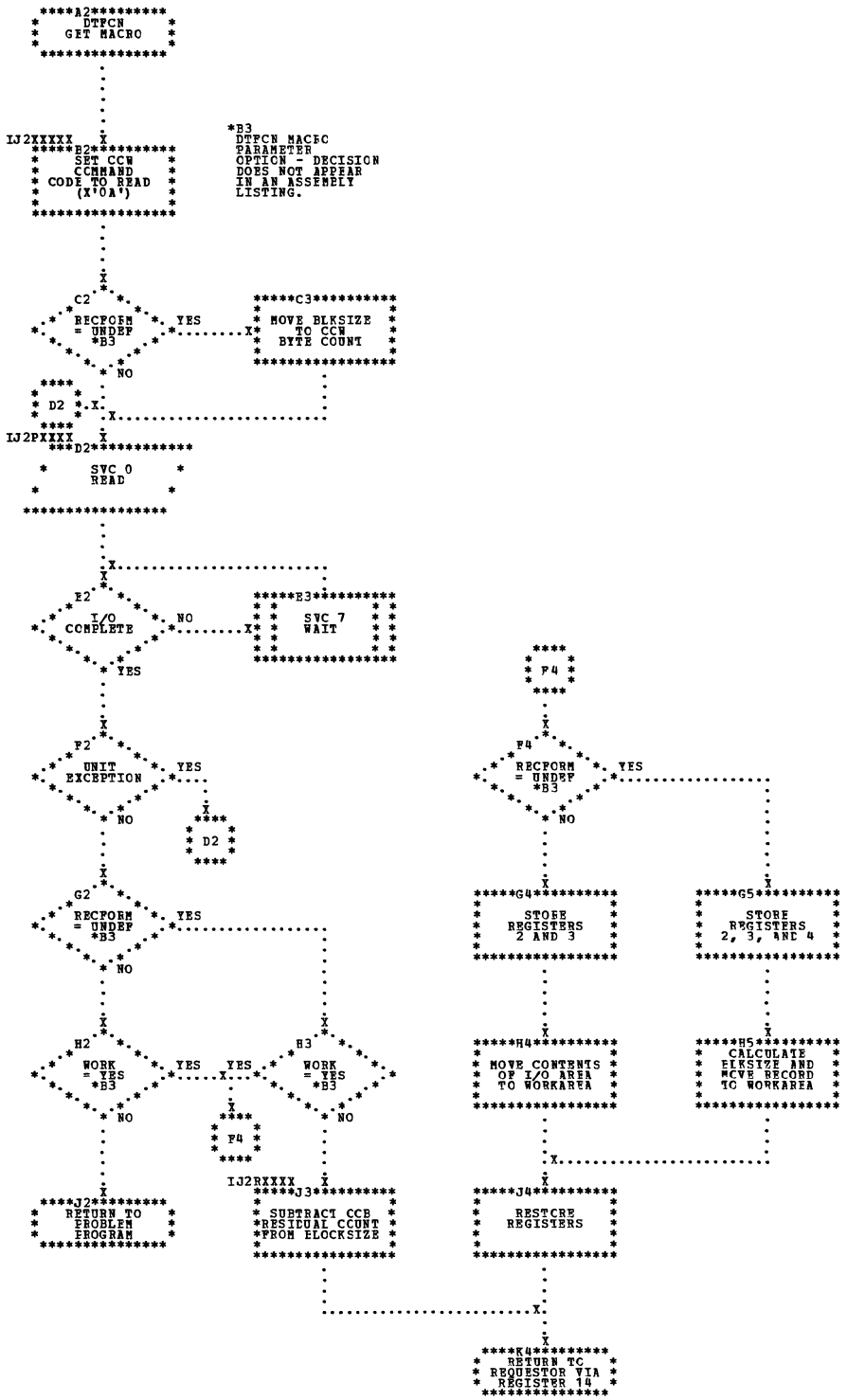


Chart AQ. DTFCN: PUTR Macrc

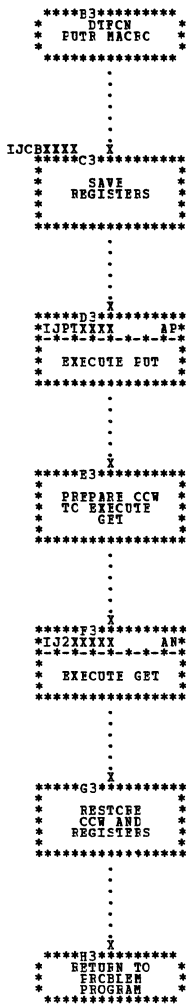


Chart BB. MRMOD: GET, READ, and CHECK Macrcs (2 cf 2)

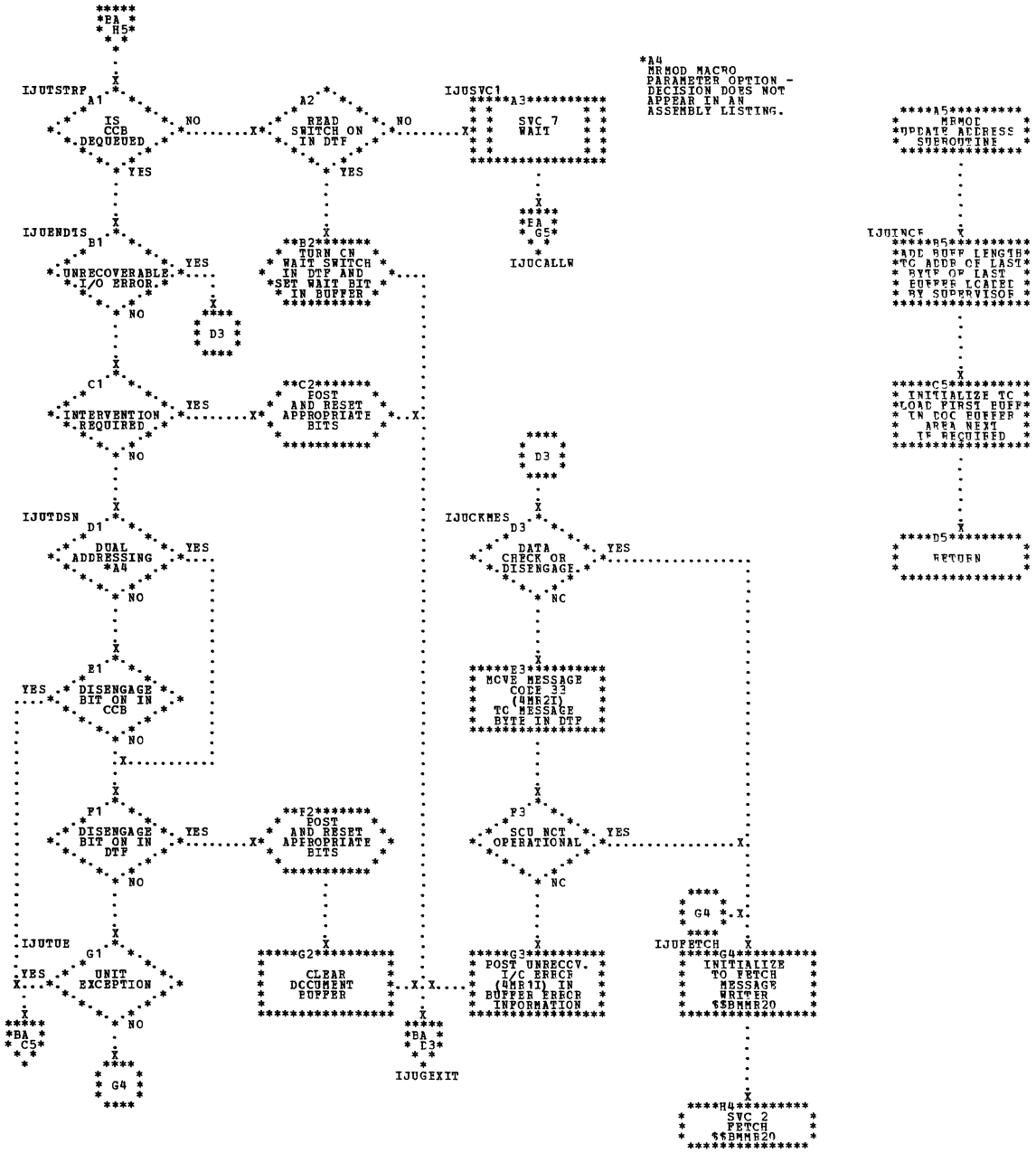


Chart BD. MRMCD: DISEN and WAITF Macros

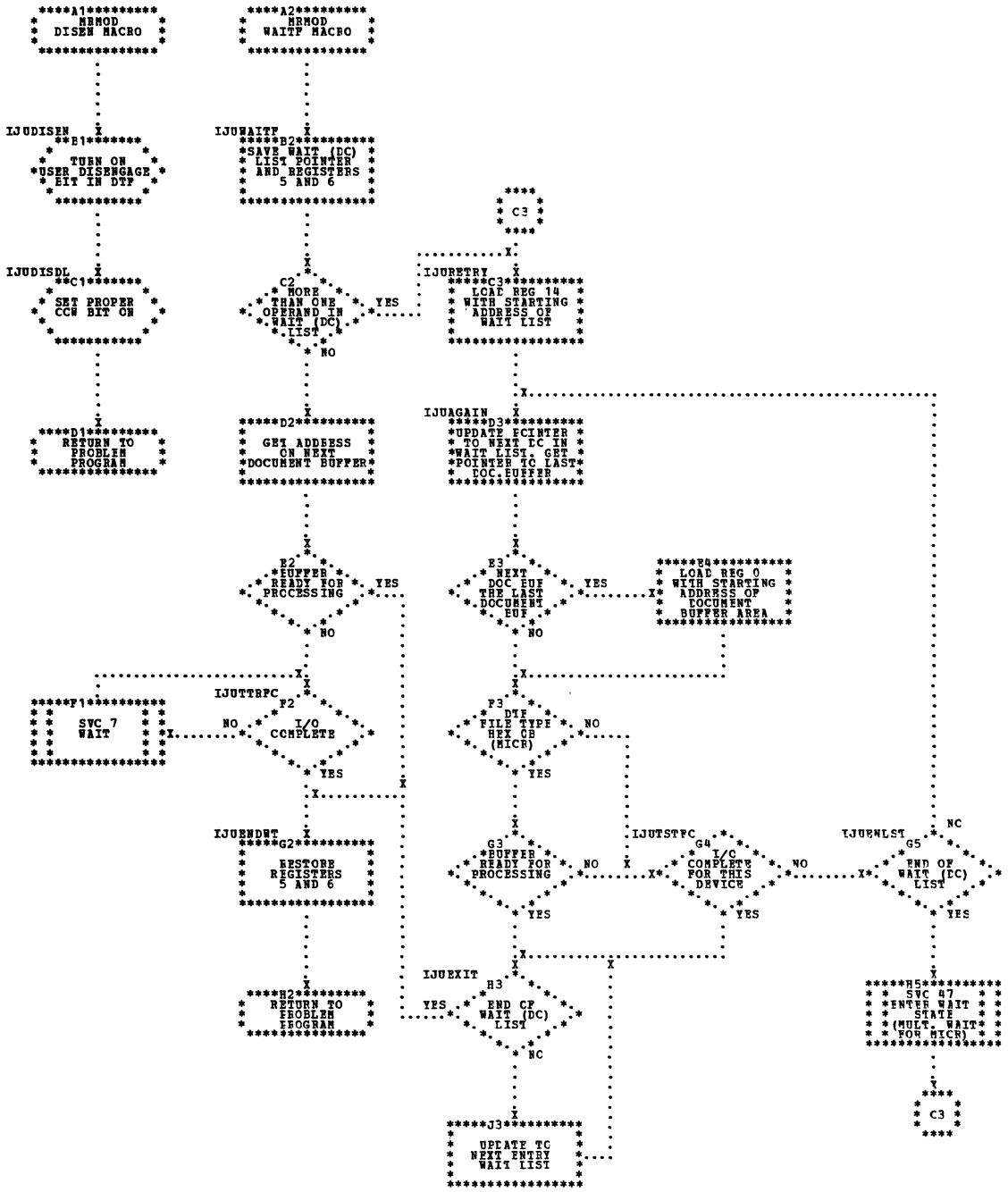


Chart BF. $$BMMR20: MICR Message Writer

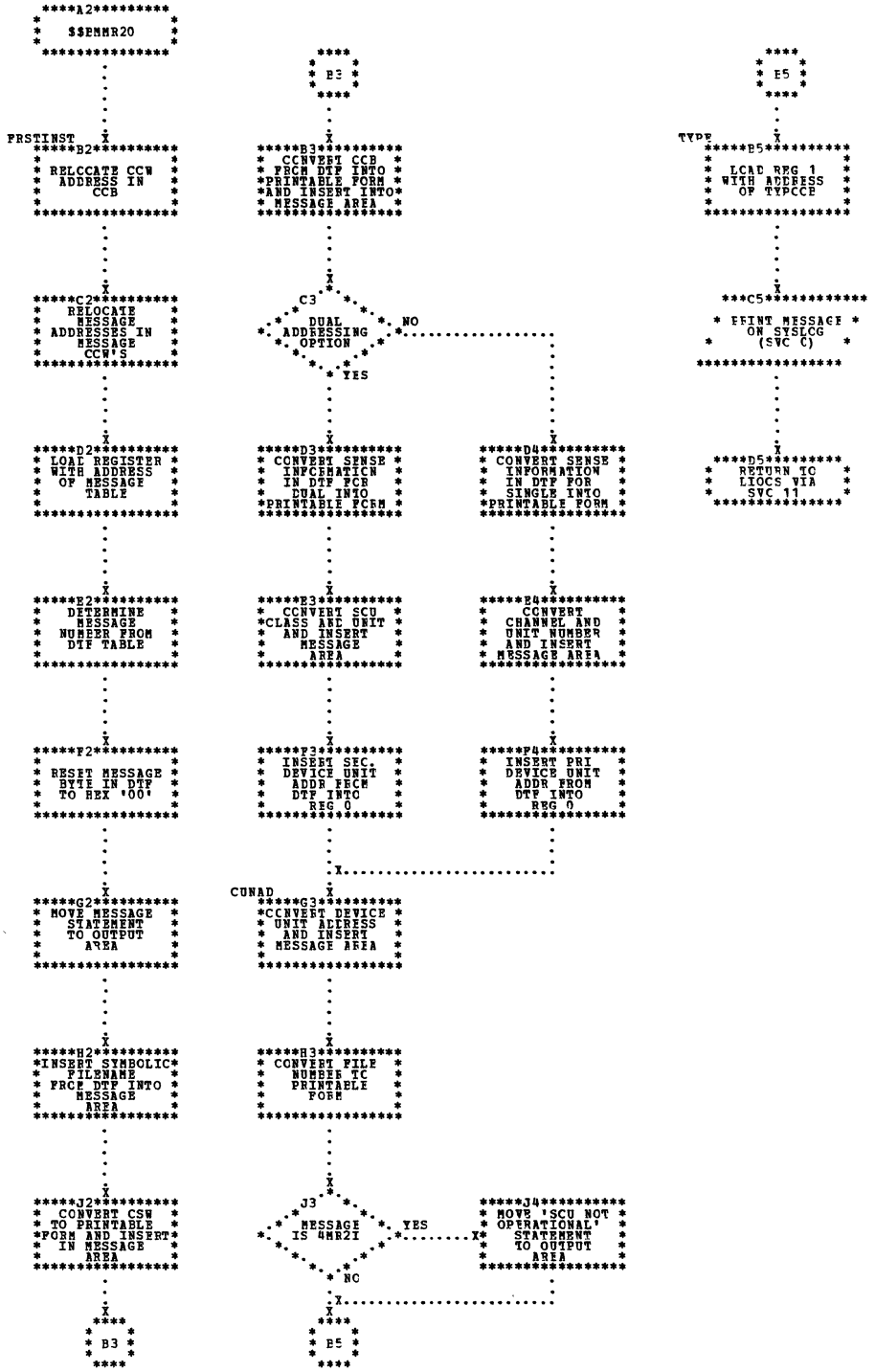


Chart BH. CRMCD: CNTRL Macro (2 of 2)

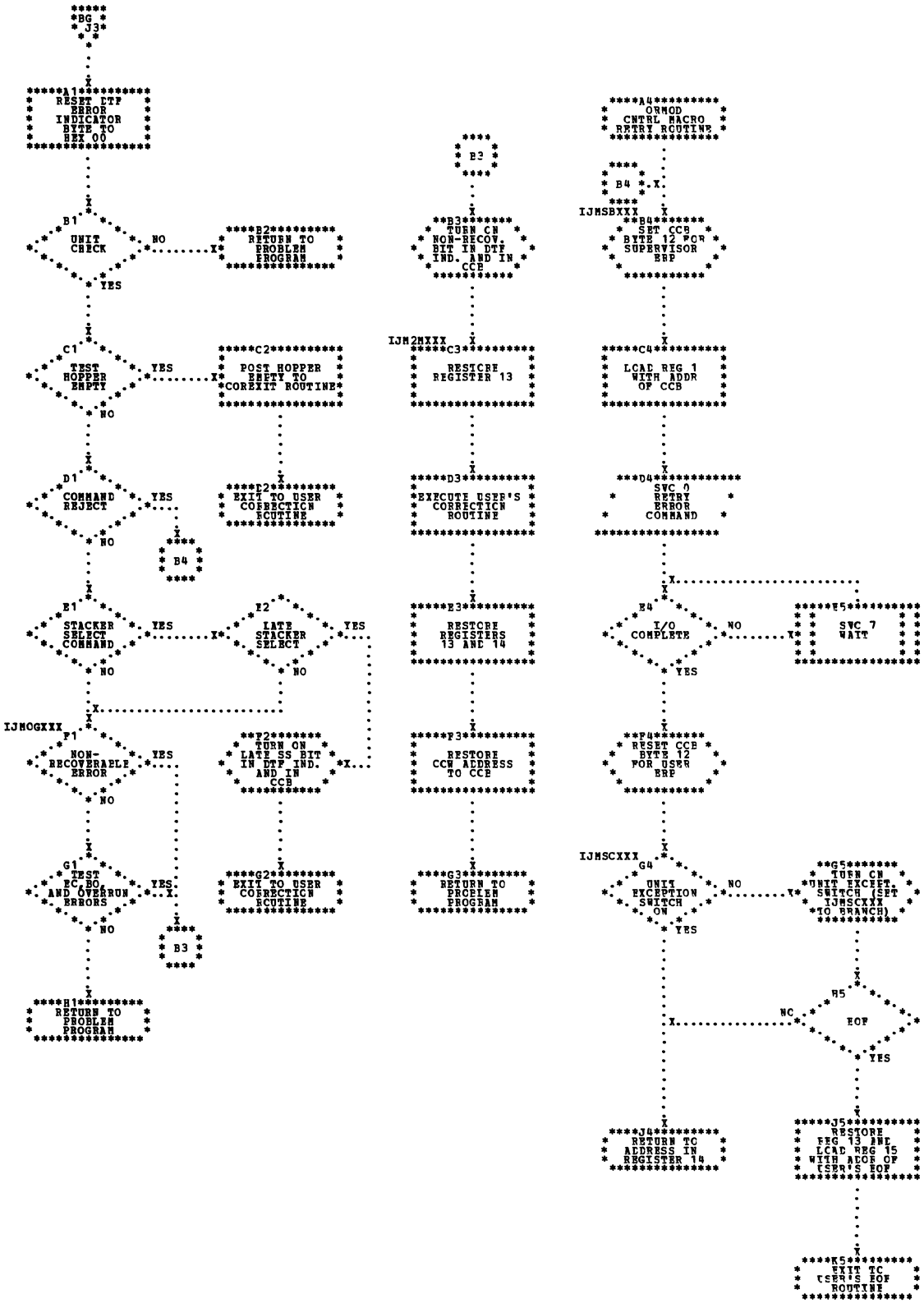


Chart BK. CRMOD: GET Macro, Unblocked Records (2 of 3)

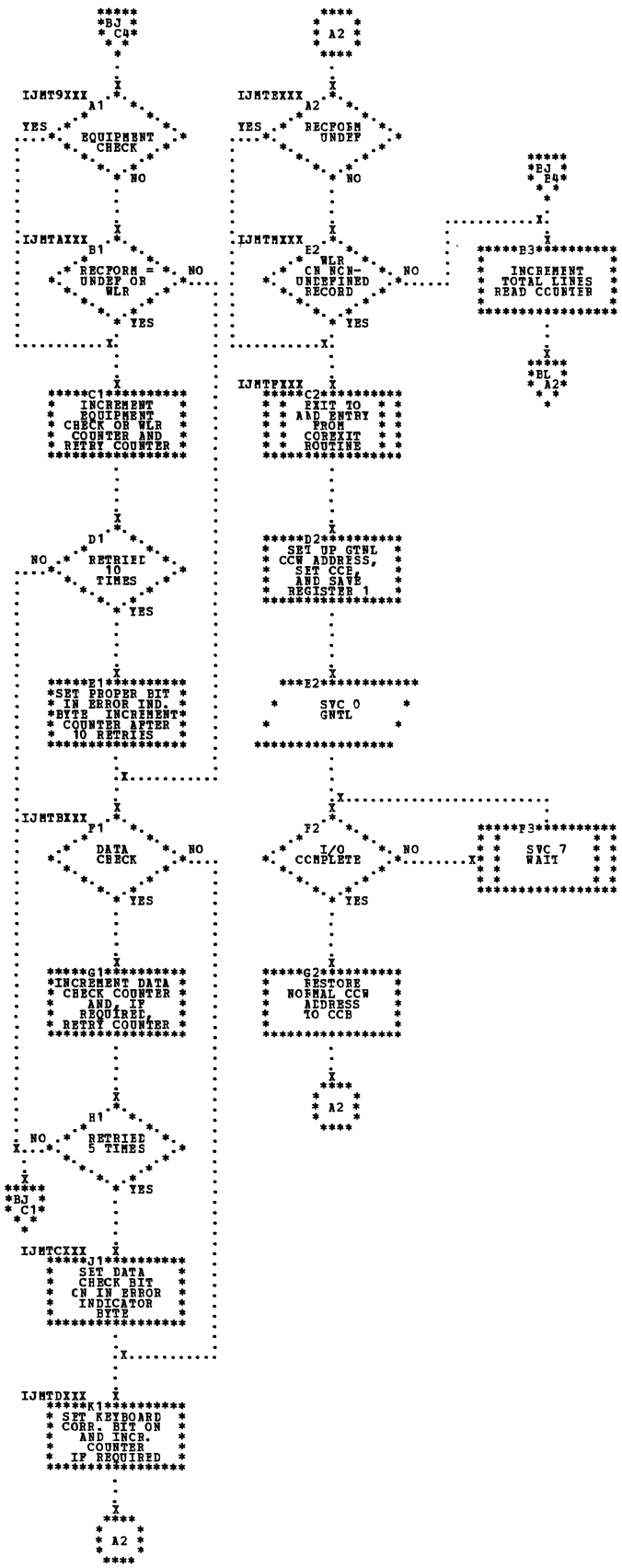


Chart BM. CRMCD: GET Macrc, Blccked Records (1 of 2)

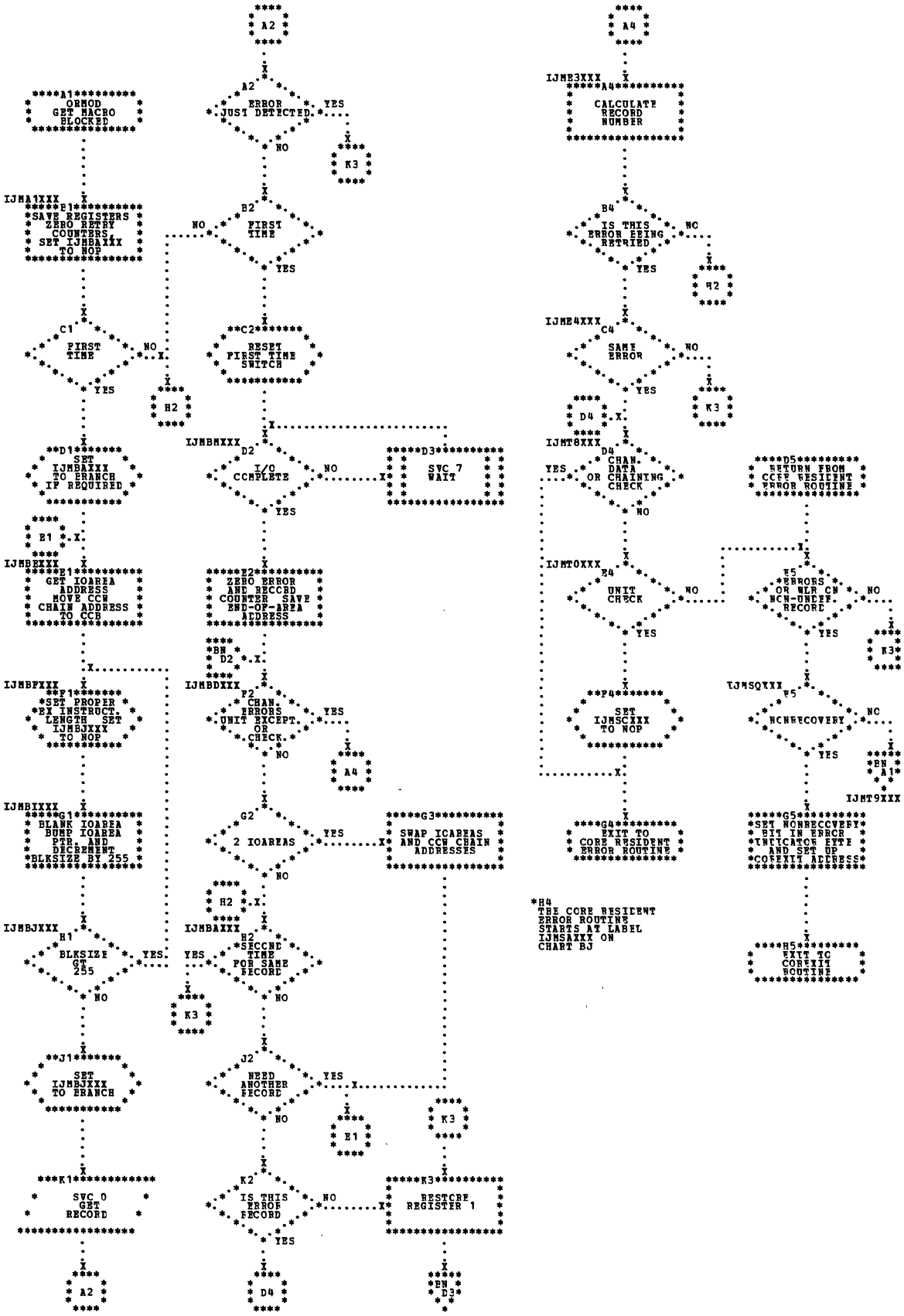


Chart BP. CRMCD: RDLINE Macrc

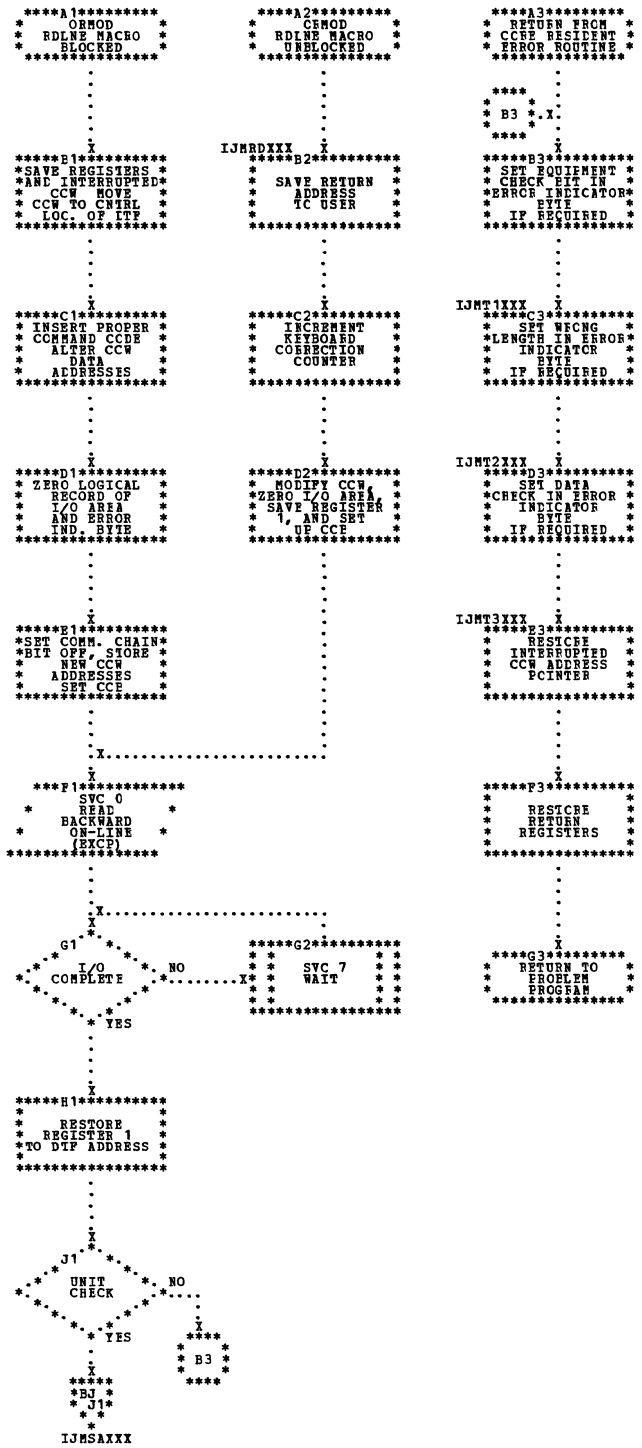


Chart BR. CRMOD: RESCN Macro

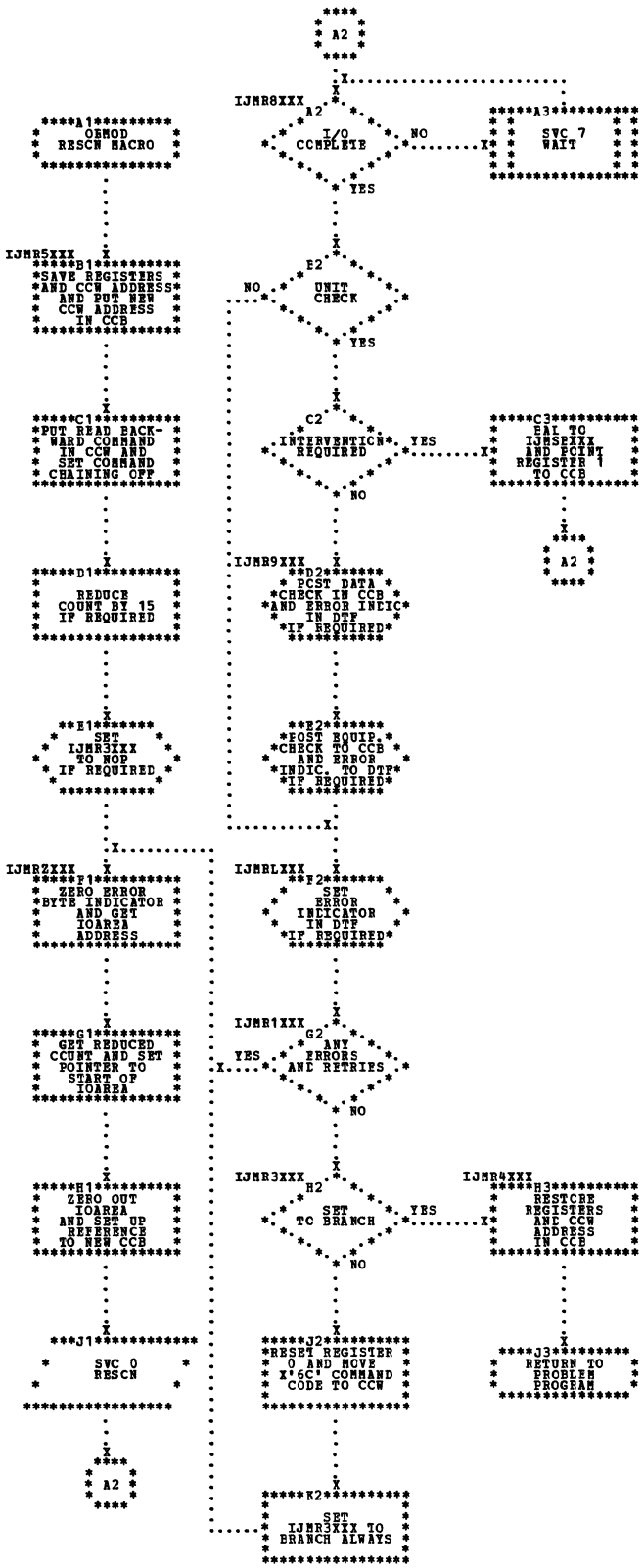


Chart BT. CRMOD: WAITF Macro (2 of 3)

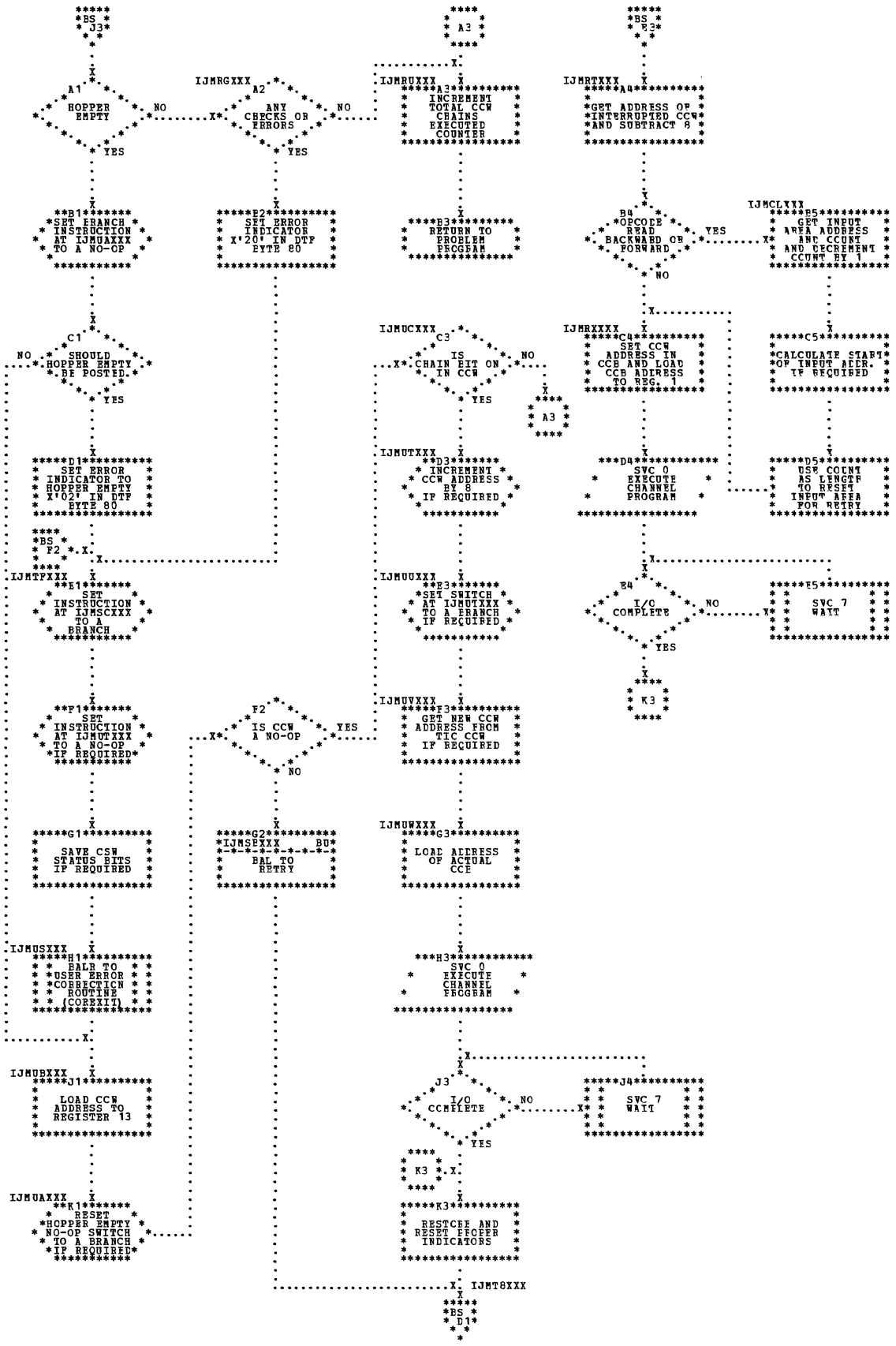


Chart CA. DRMOD: CNTRL and READ Macros

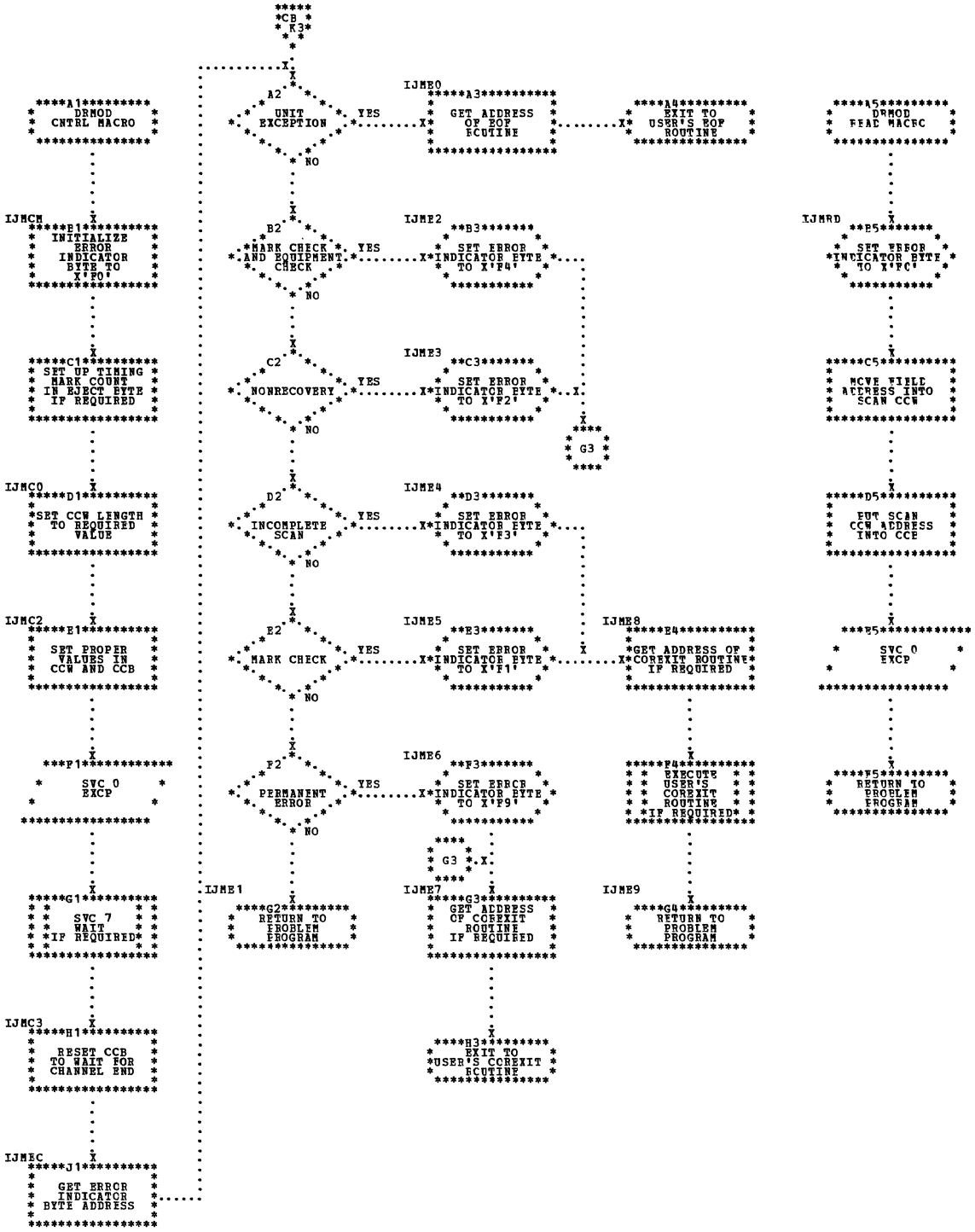
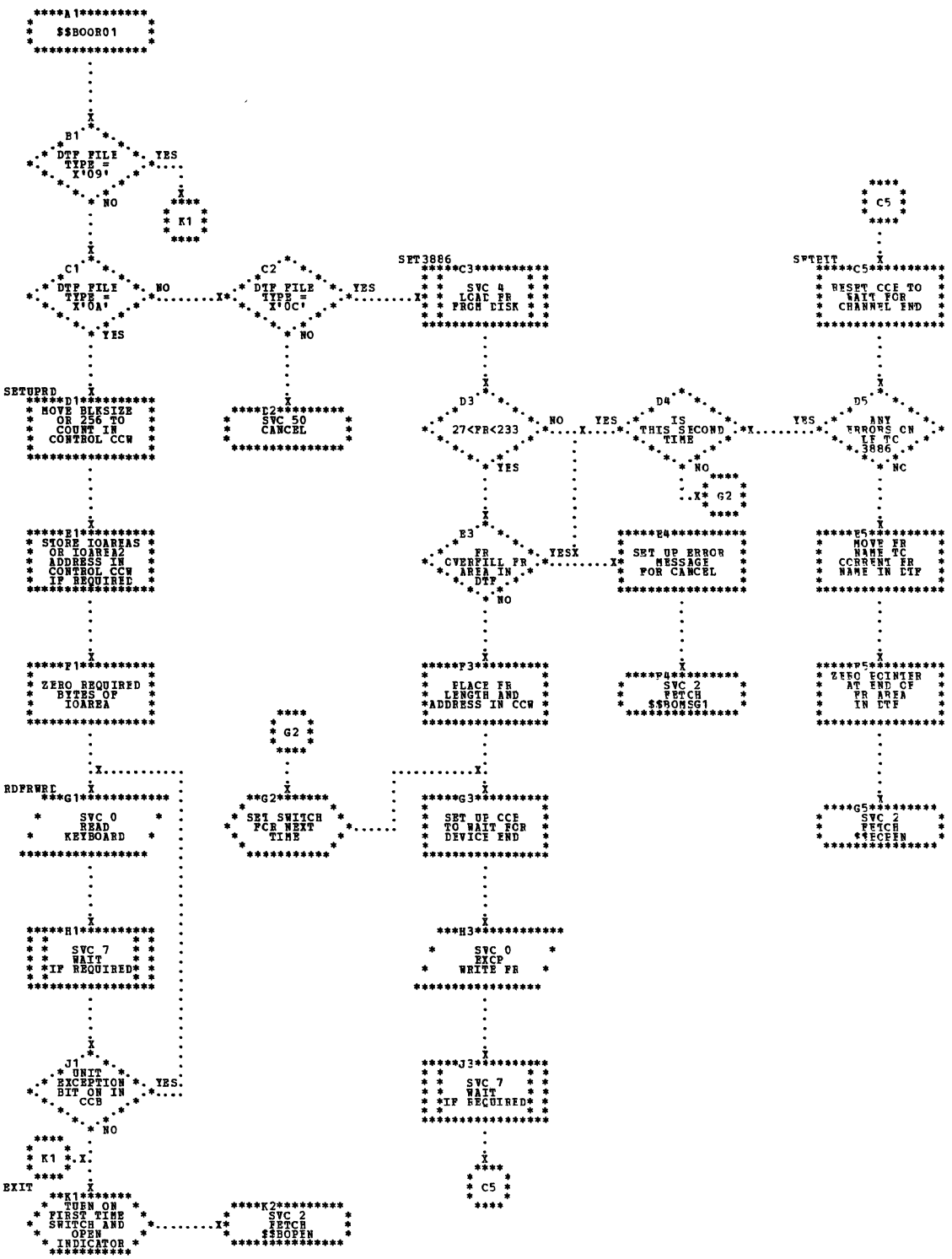


Chart CC. $$BOOR01: Open Optical Reader



## Chart CE. PRMOD: PRTCV Macro

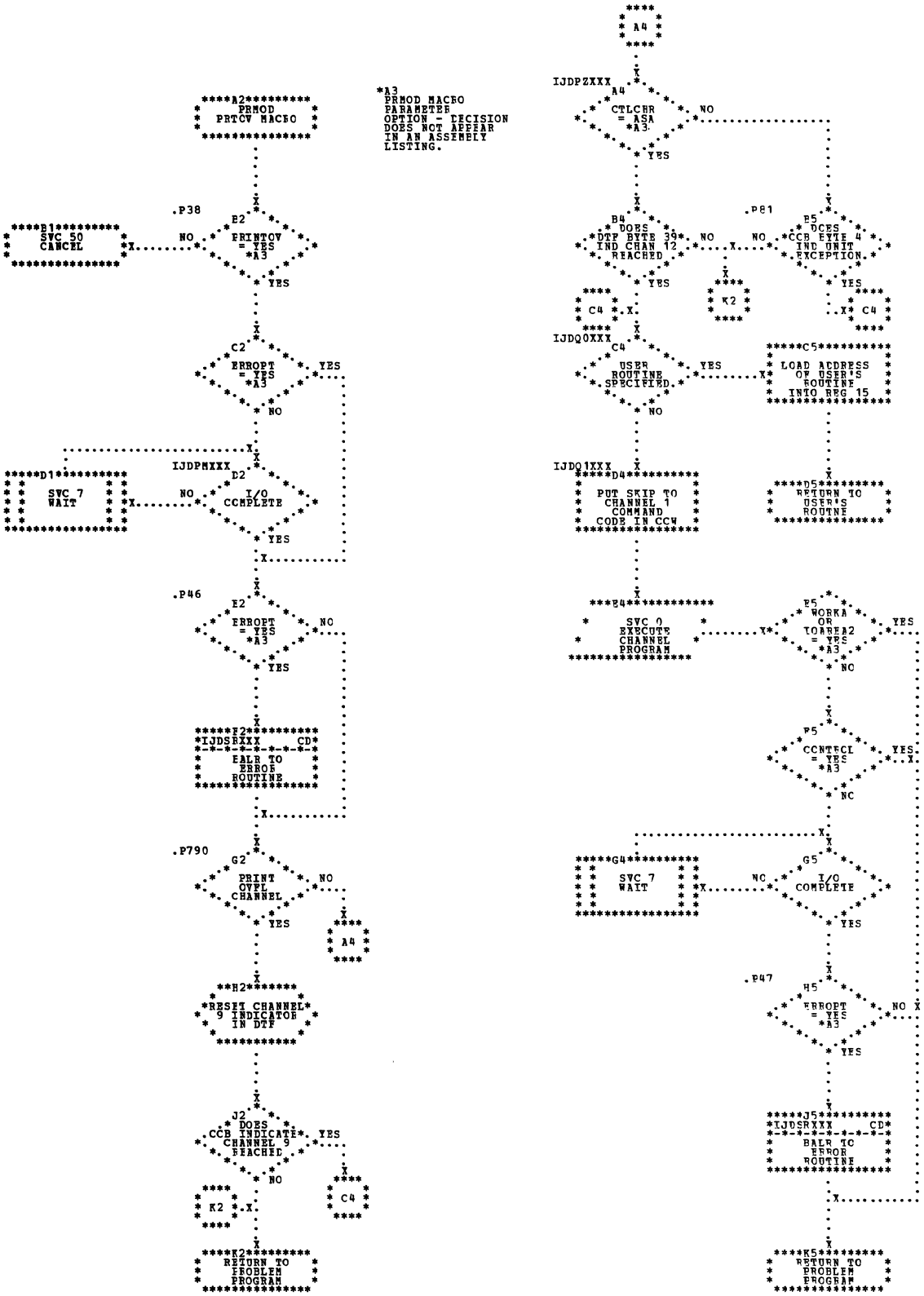


Chart CG. PRMCD: PUT Macrc (2 cf 3)

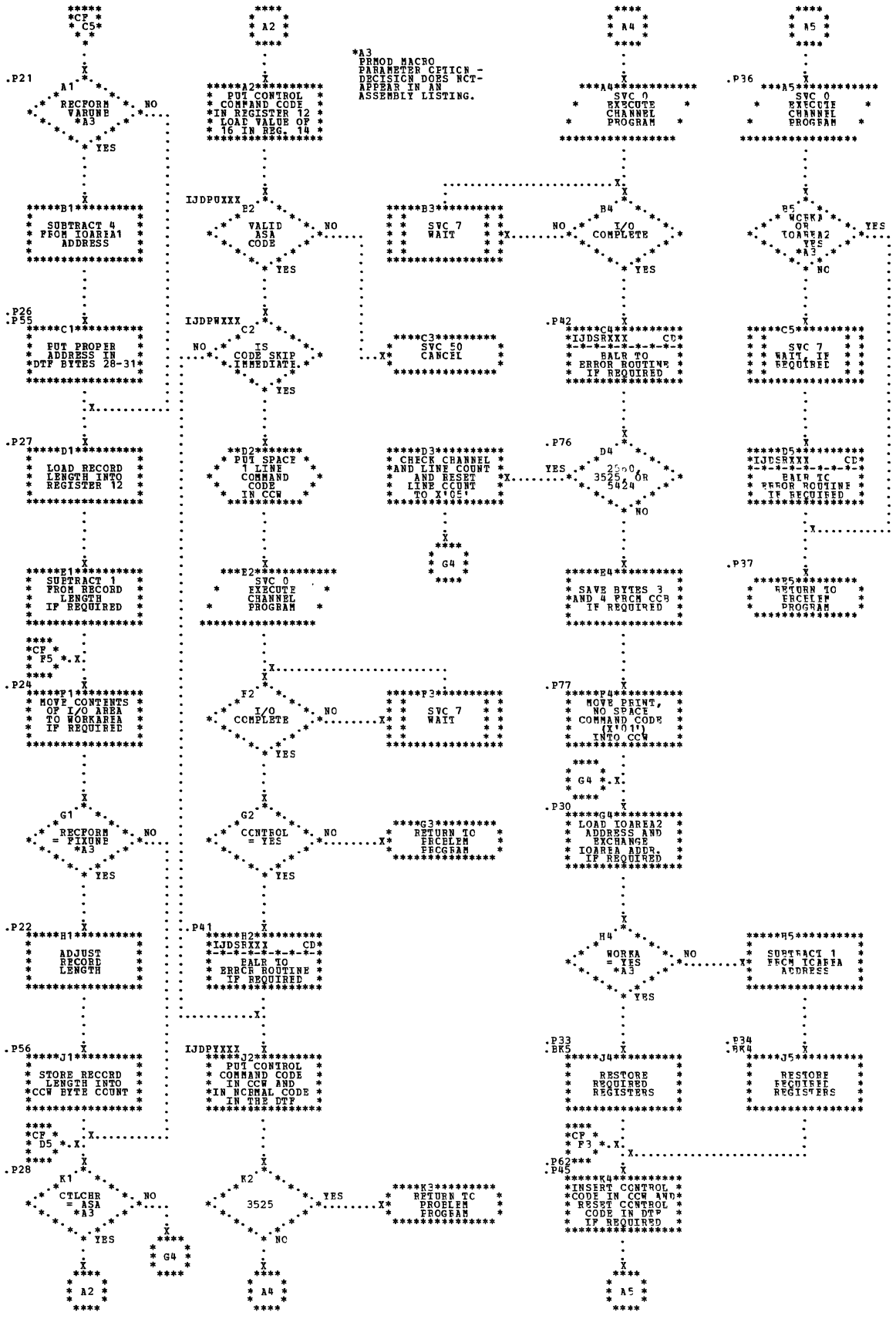


Chart CJ. PIMOD: GET Macro, No Translation, and GET Macro, Translation, No Shifted Code, Device=2671

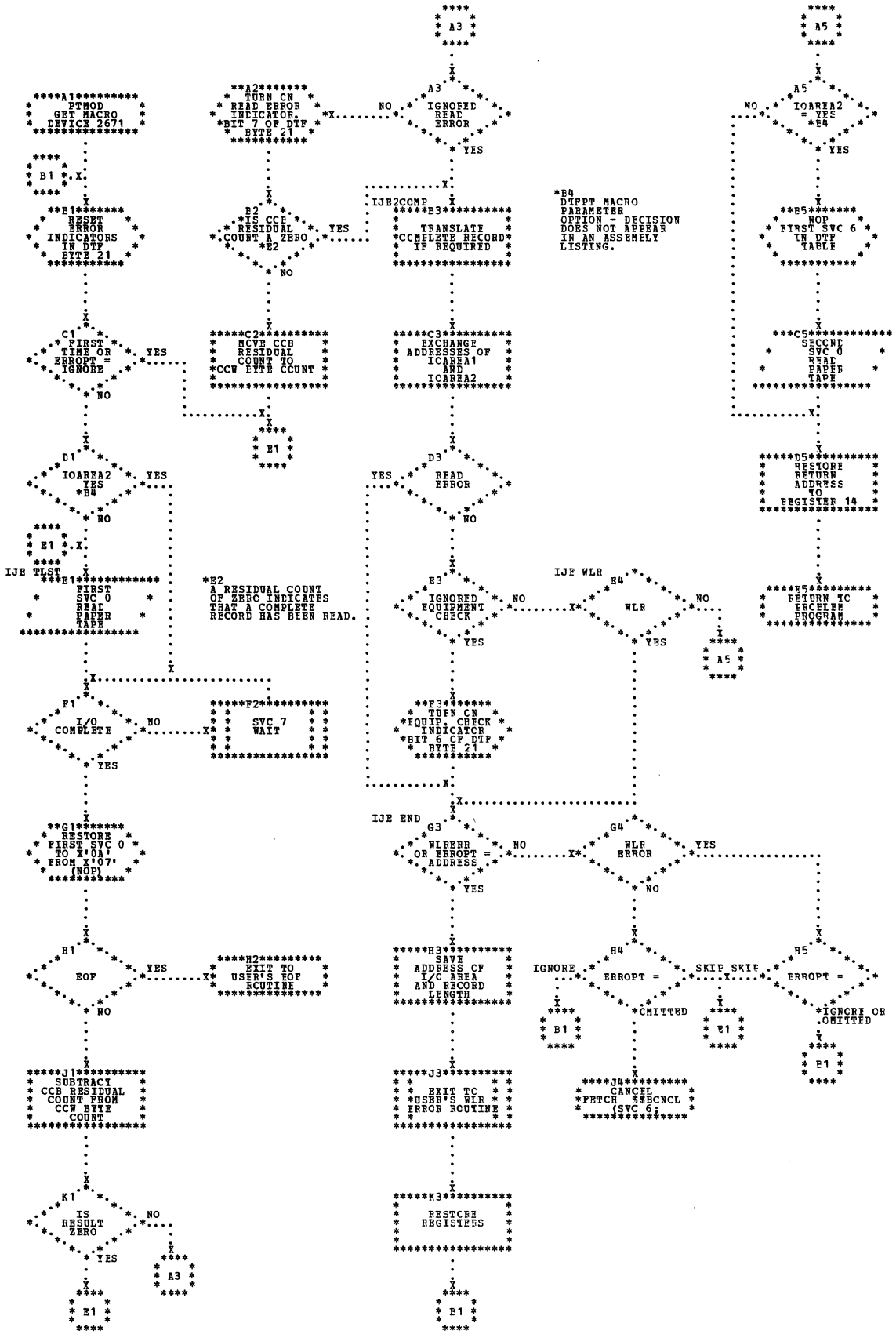


Chart CL. PTMCD: GET Macrc, Translatiоn, Shifted Ccde, Undefined Reccrds, Device=2671

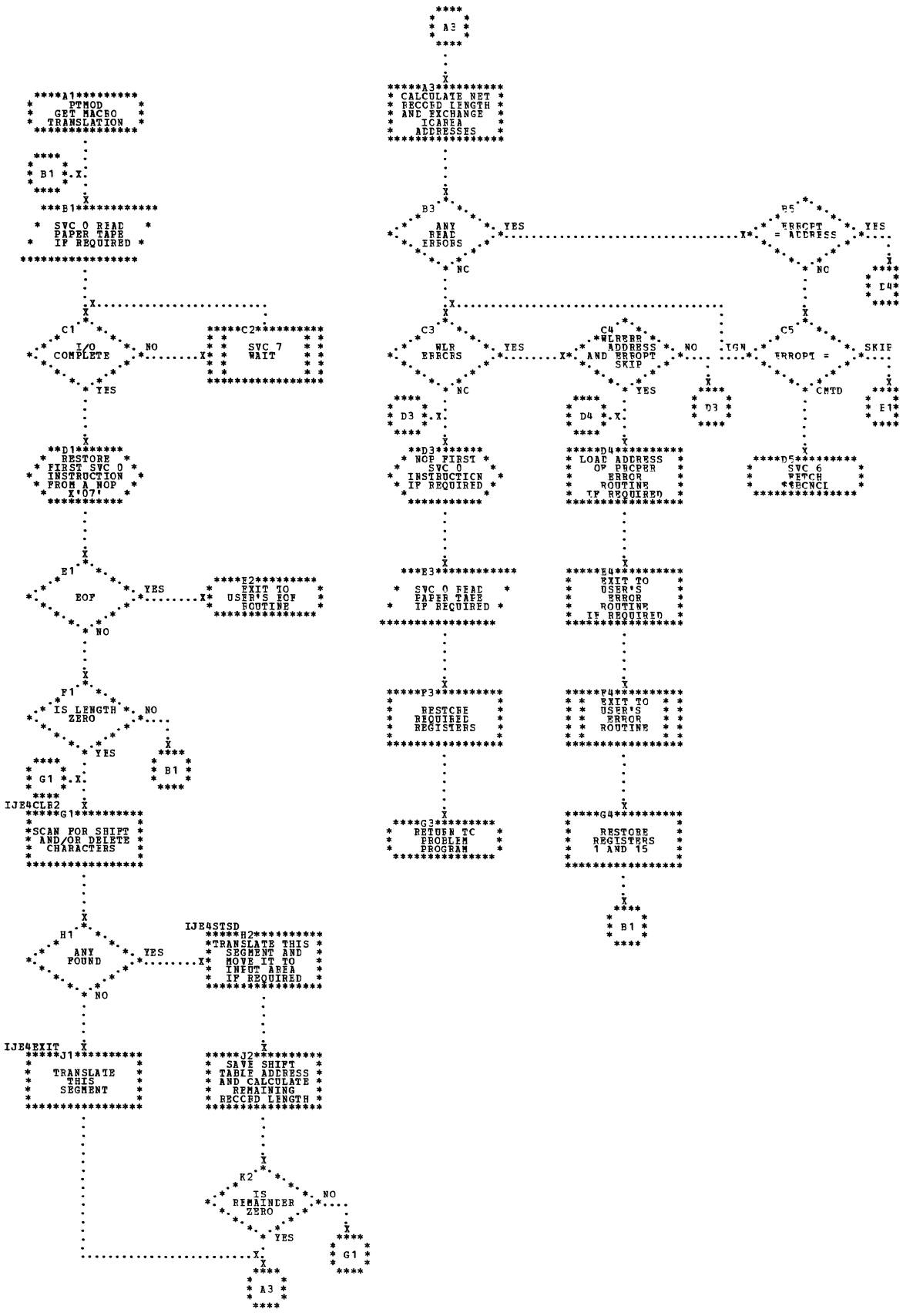


Chart CN. PTMOD: GET Macrc, Translation, Shifted Code, Fixed Unblocked Reccrds, Device=1017

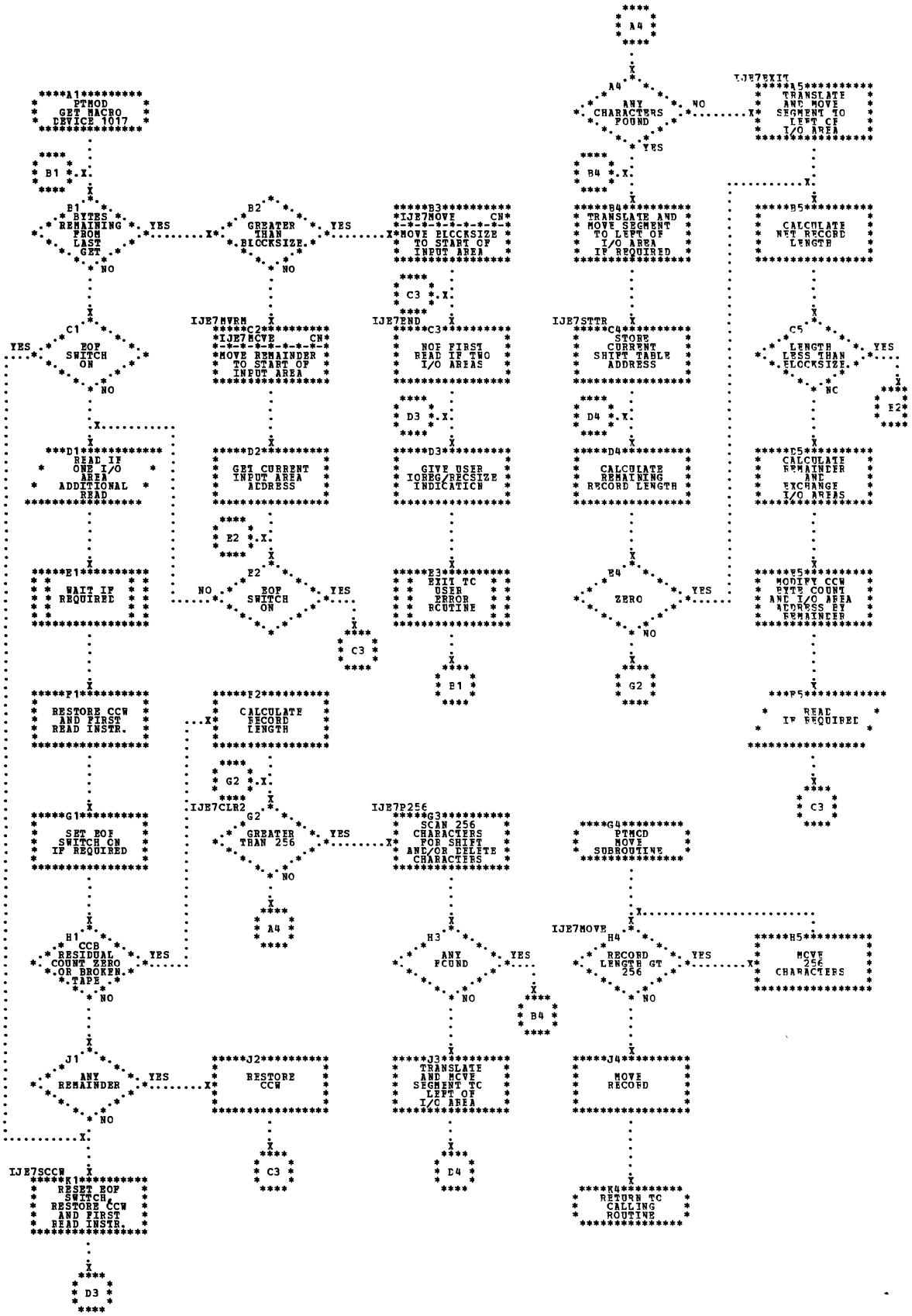


Chart CQ. PIMOD: PUT Macro, No Shifted Code, Device=1018

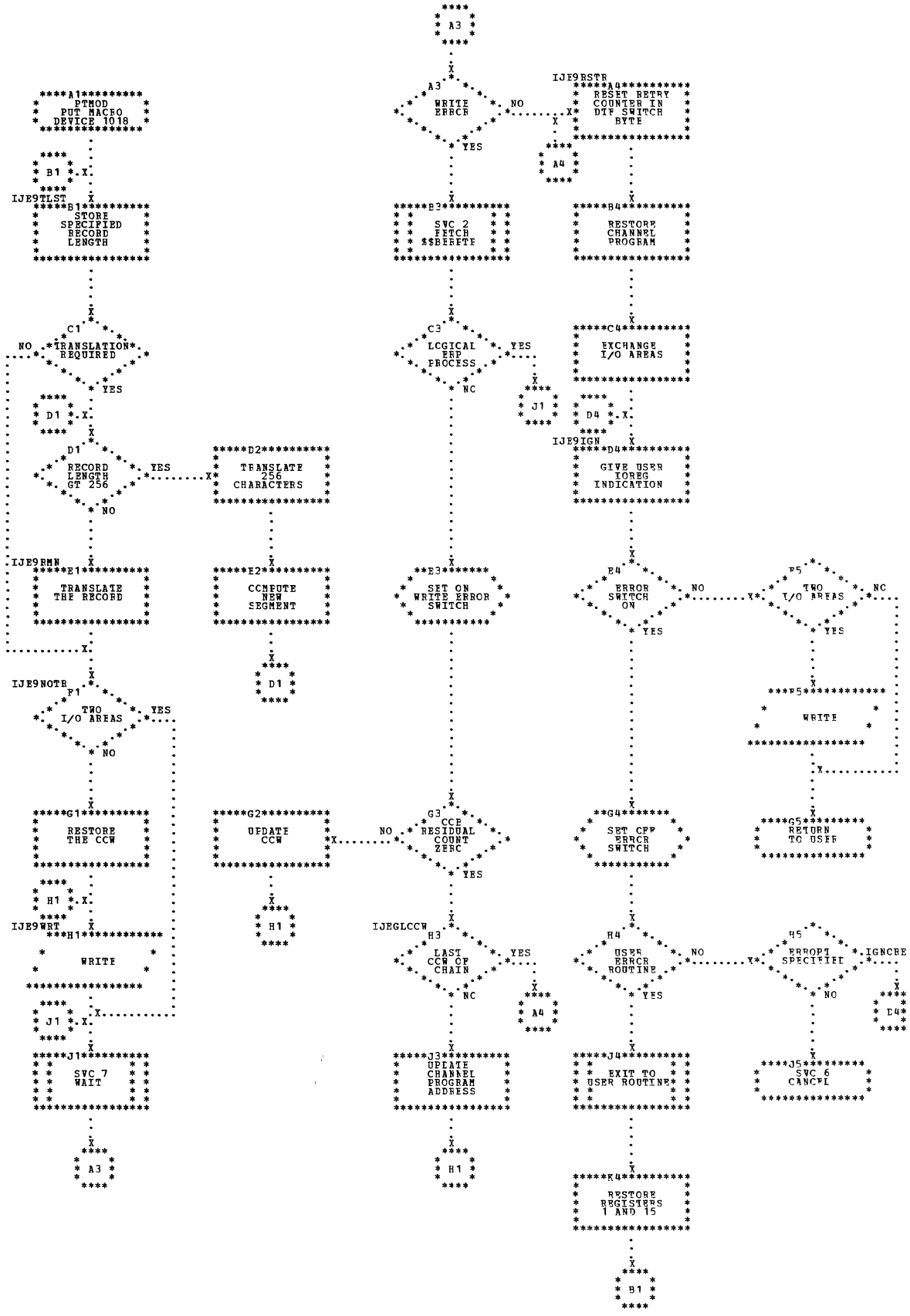
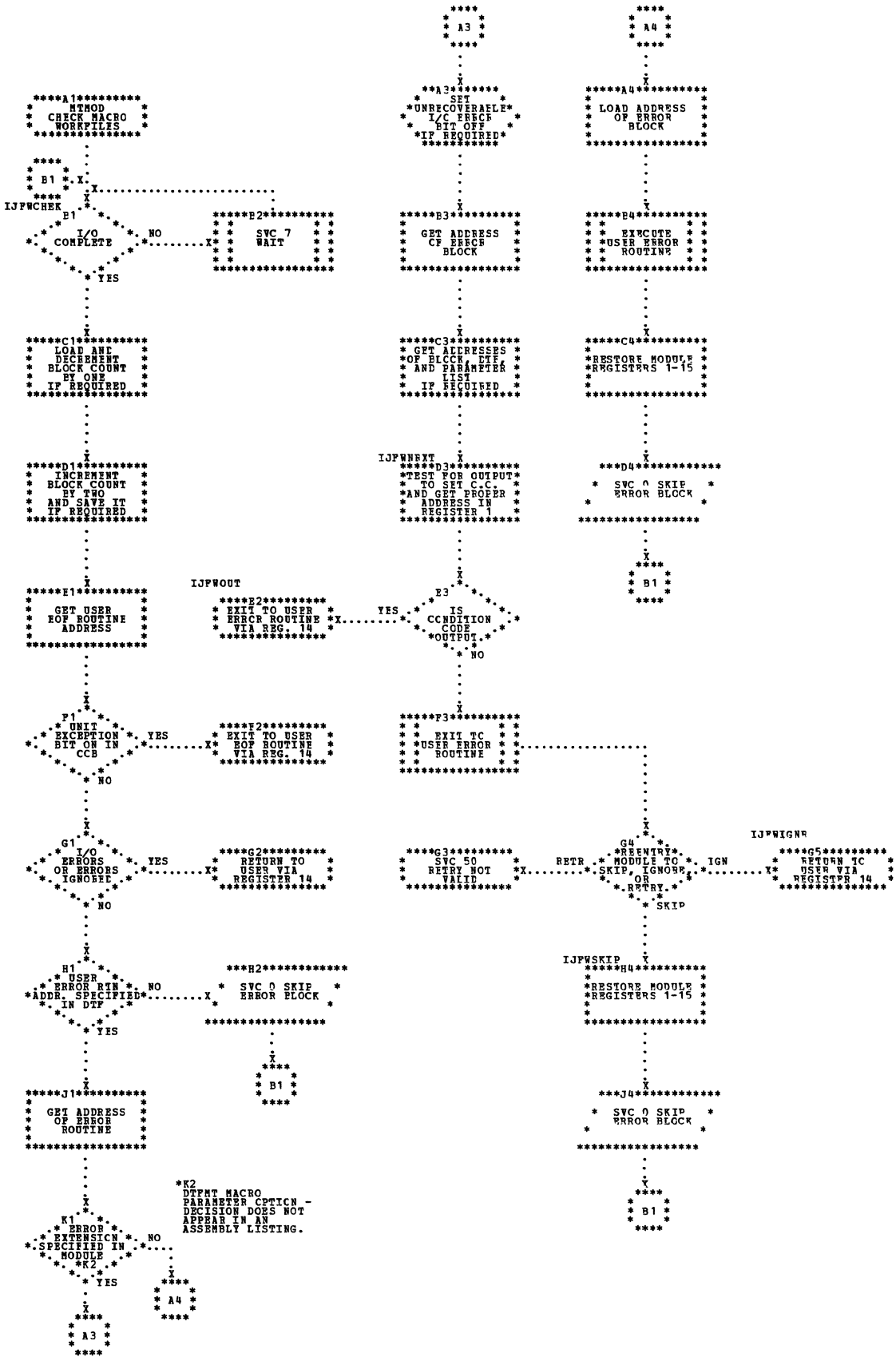
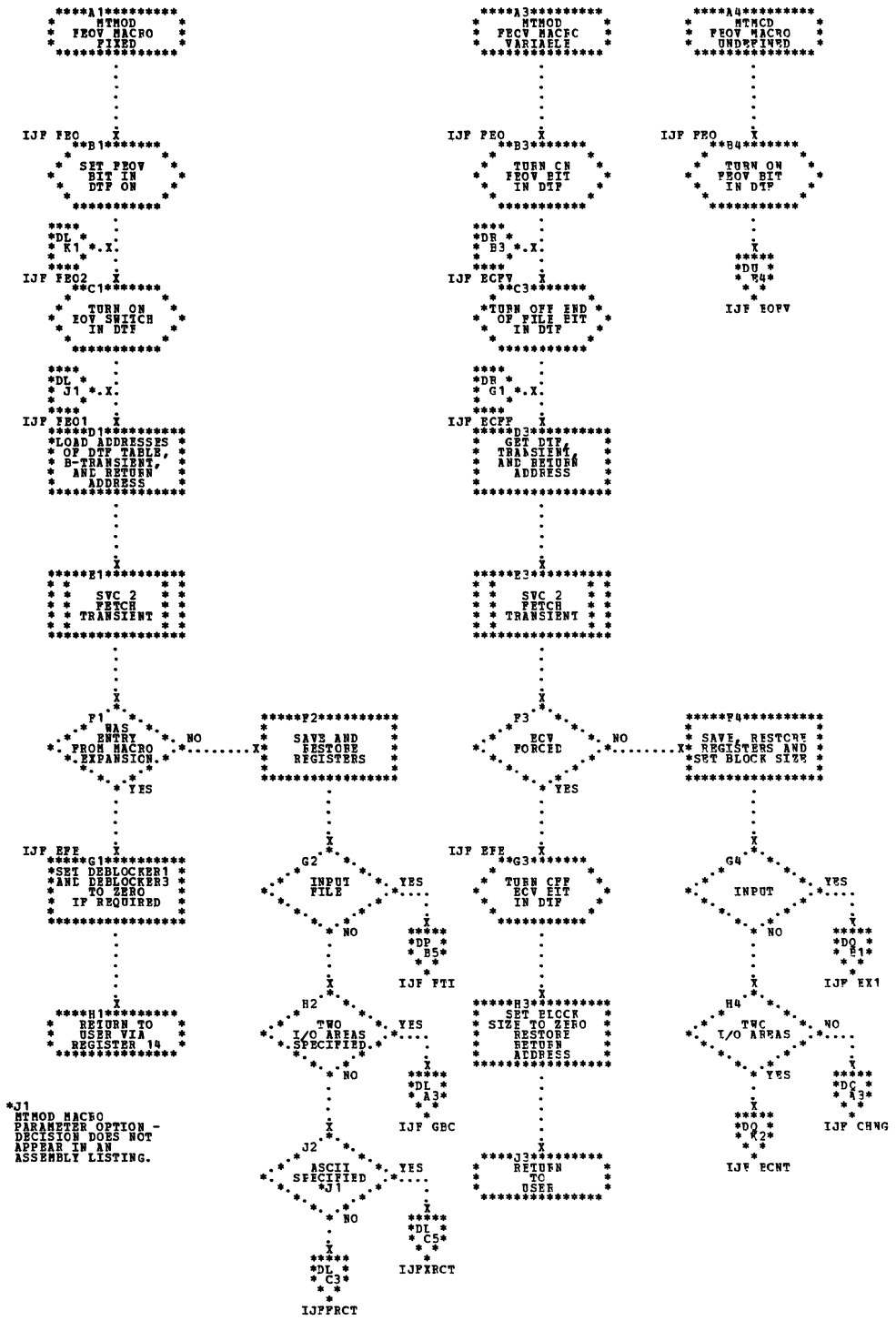


Chart DA. MIMCD: CHECK Macro, Workfile



## Chart DC. MTMOD: FEOV Macrc



## Chart DE. MTMCD: GET Macrc, Spanned Records Routines

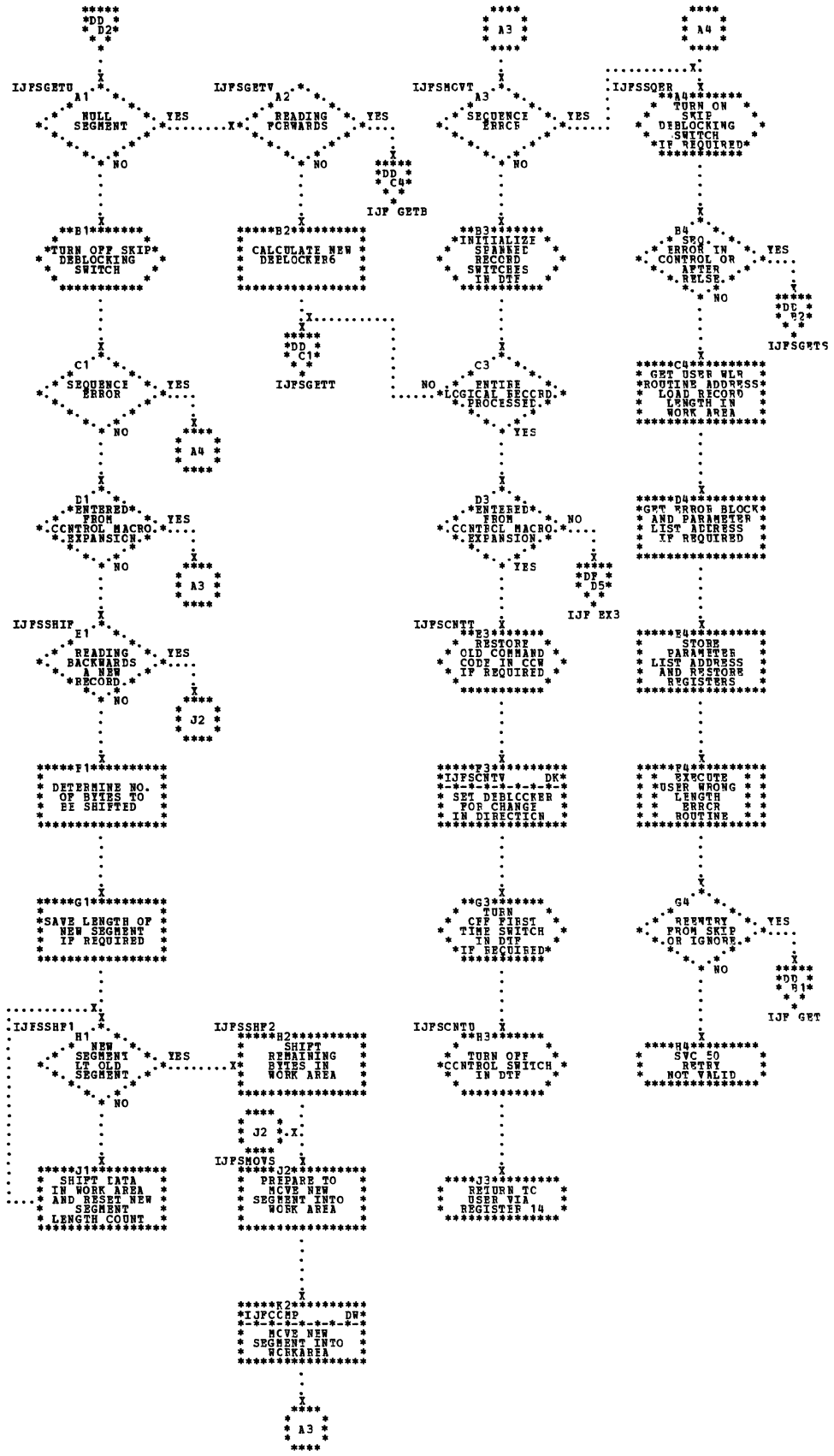


Chart DG. MTMOD: PUT Macrc

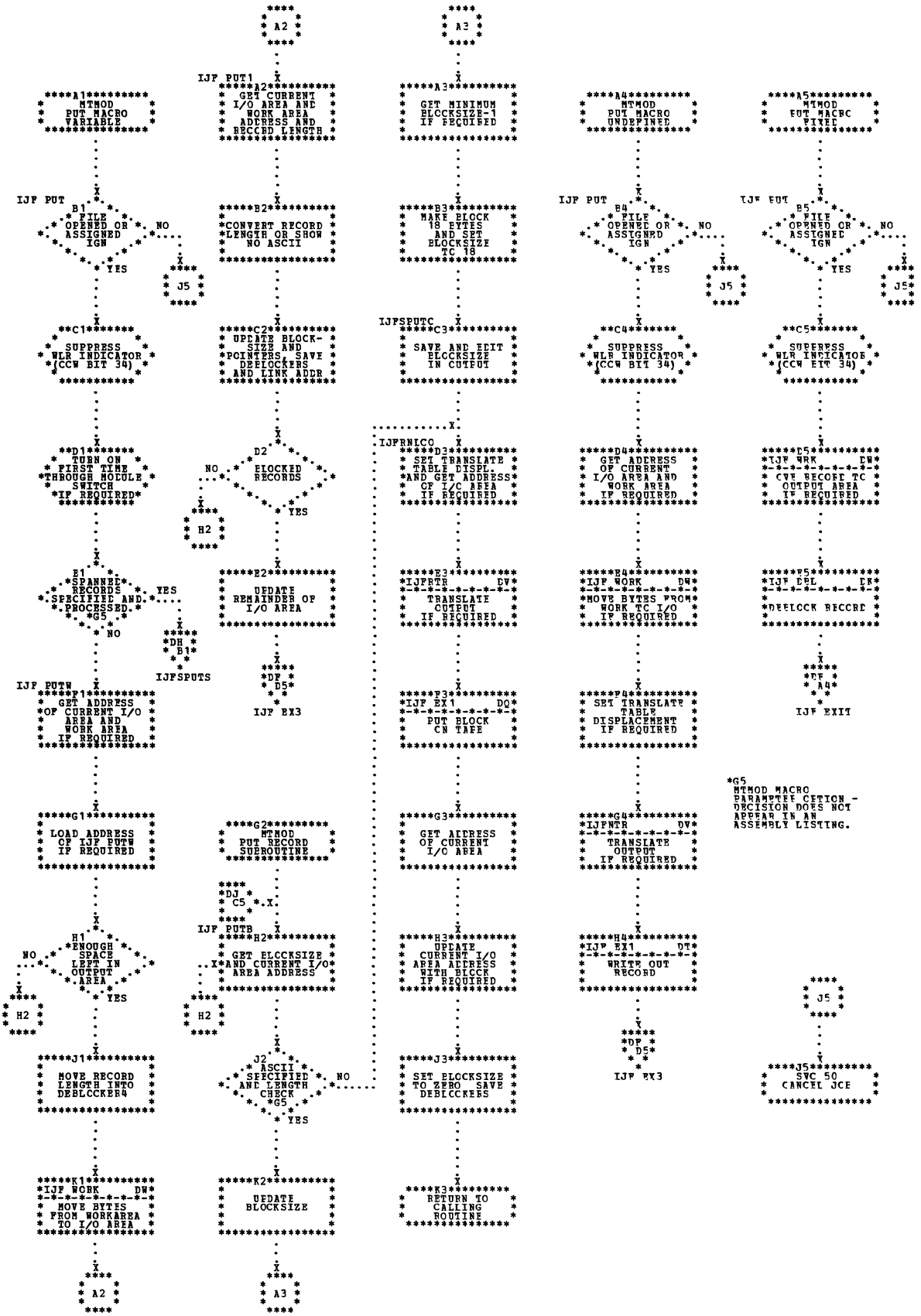


Chart DJ. MTMOD: READ and WRITE Macros, Workfiles, and RELSE and TRUNC Macros

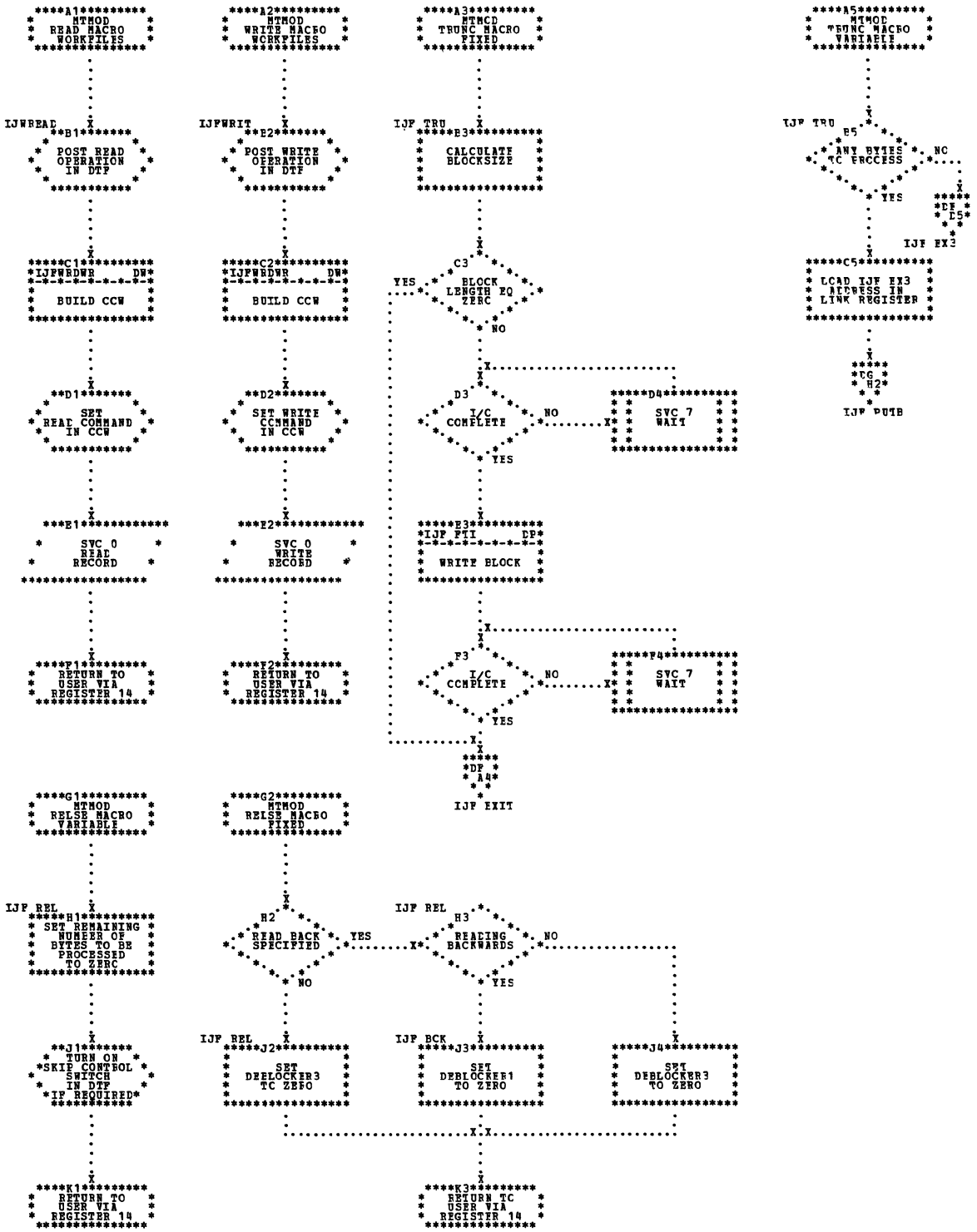


Chart DL. MTMOD: Read/Write Subroutine, Fixed Length Records (1 of 2)

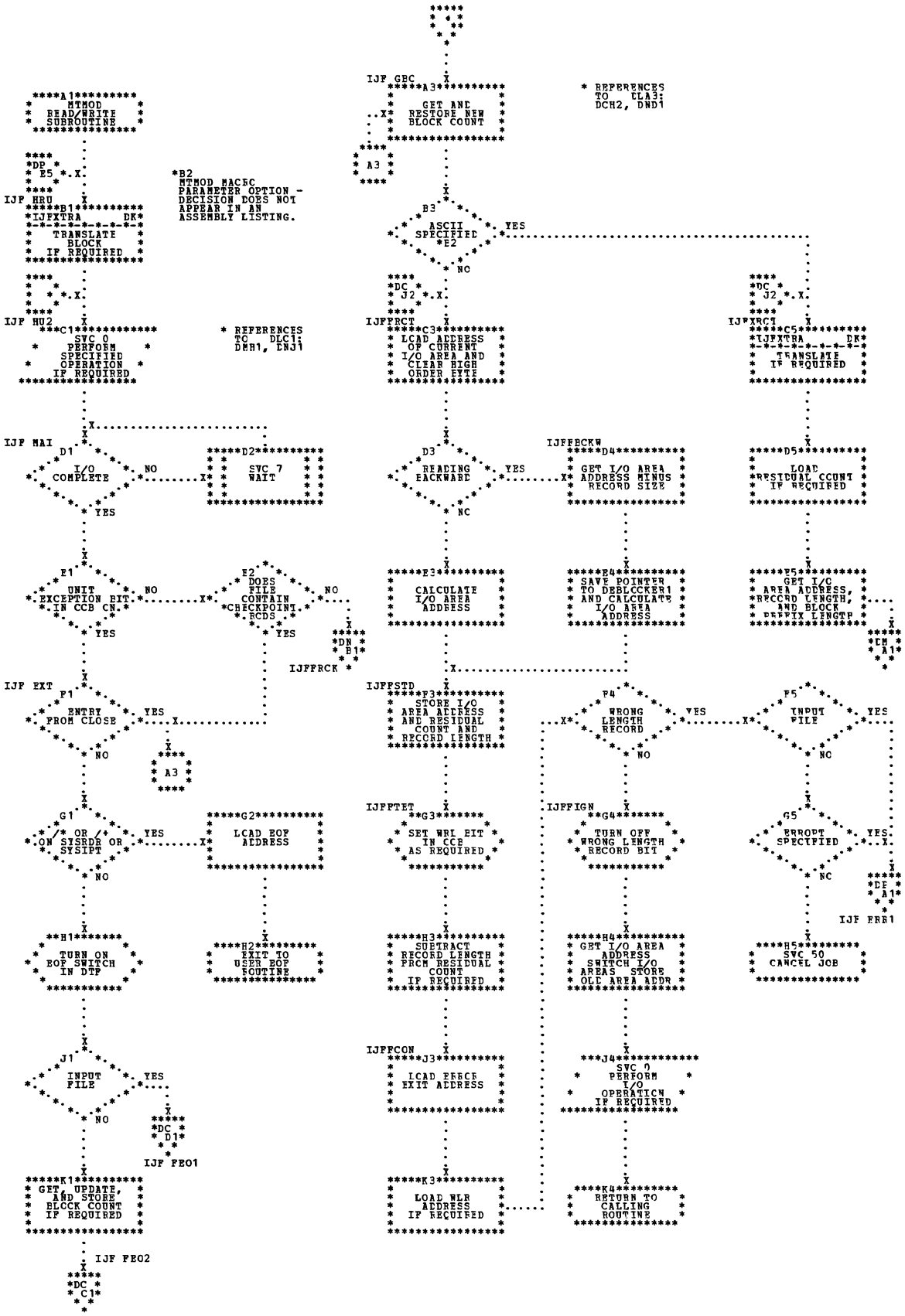


Chart DN. MTMCD: Bypass Checkpoint Records Routine, Fixed Length Records

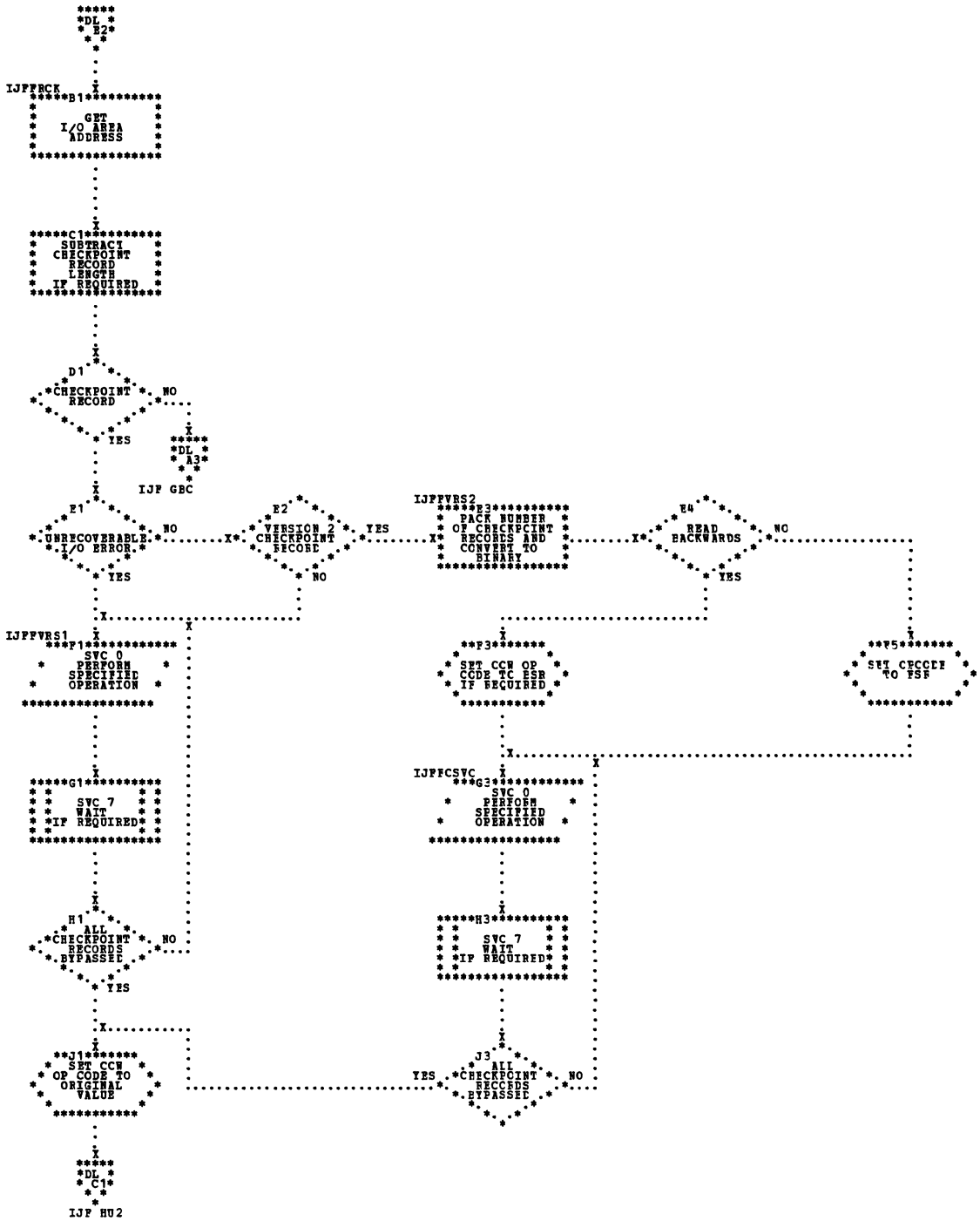


Chart DQ. MTMOD: Read/Write Subrcutine, Variable Length Records (1 cf 2)

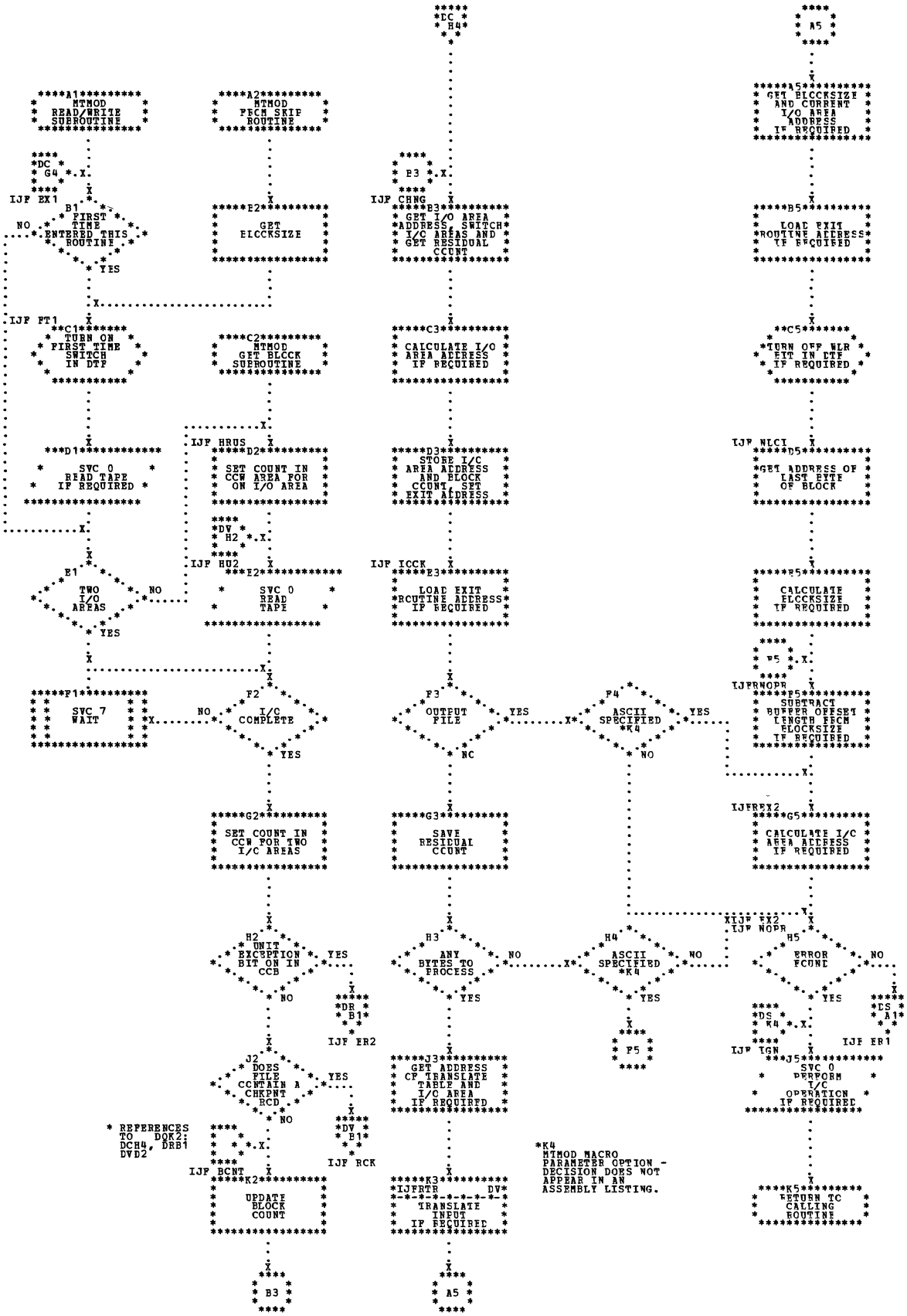


Chart DS. MTMOD: Error Exit Routine, Variable Length Records

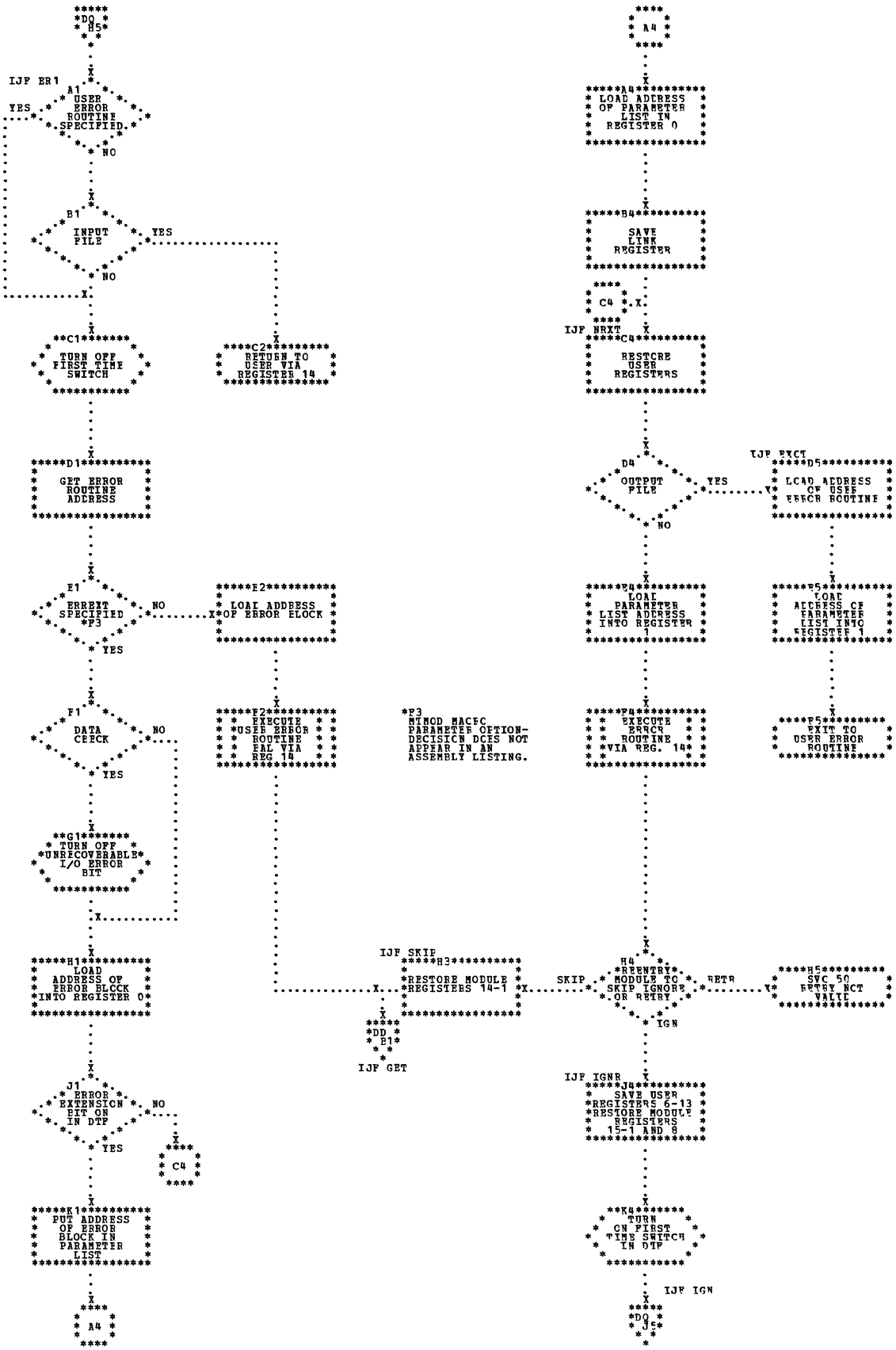
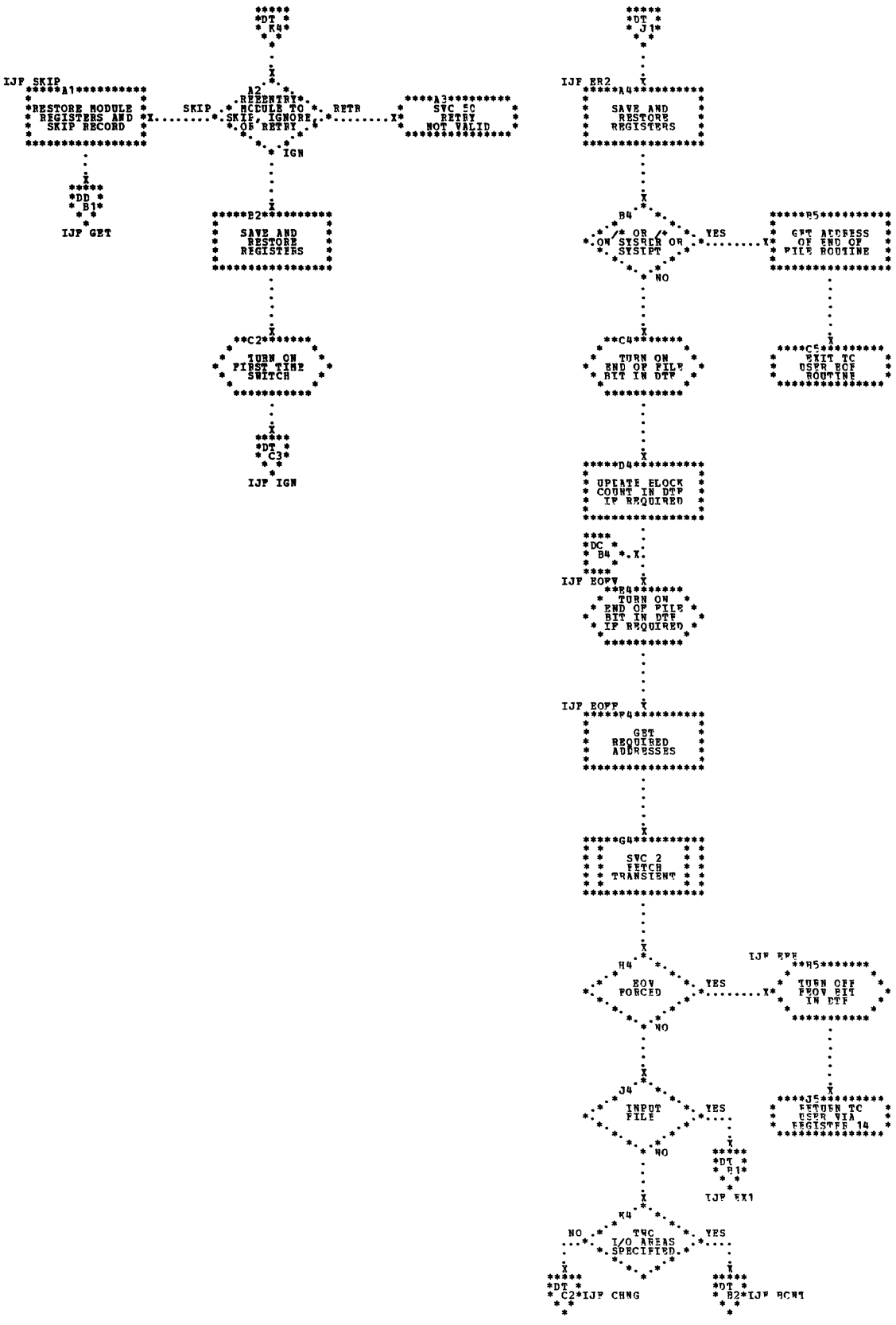


Chart DU. MTMOD: Read/Write Subrcutine, Undefined Length Records (2 cf 2)



## Chart DW. MTMCD: Work Area Subroutine

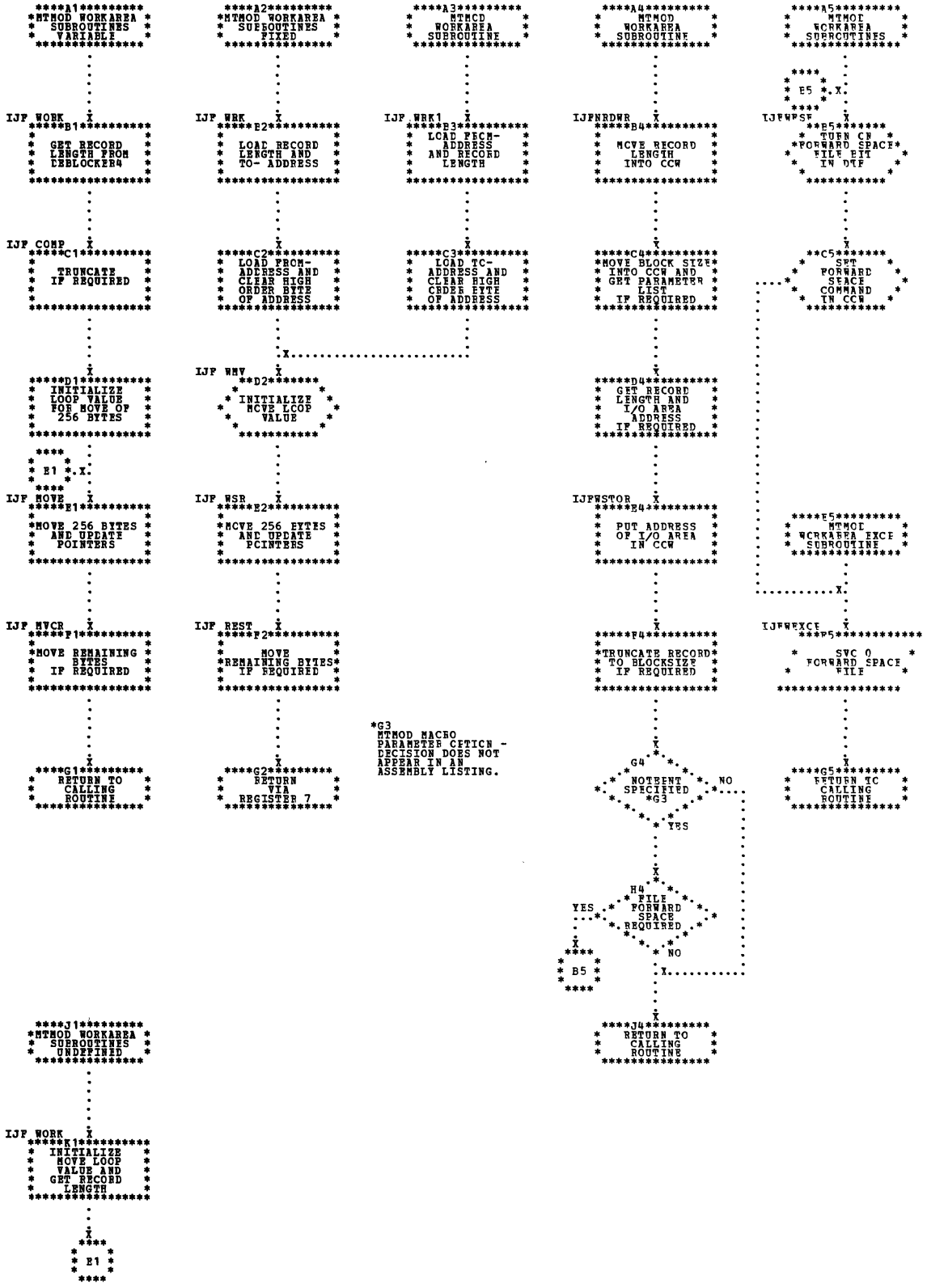


Chart EB. $$BCMT01: Open Input Standard Labels, Forward (2 of 2)

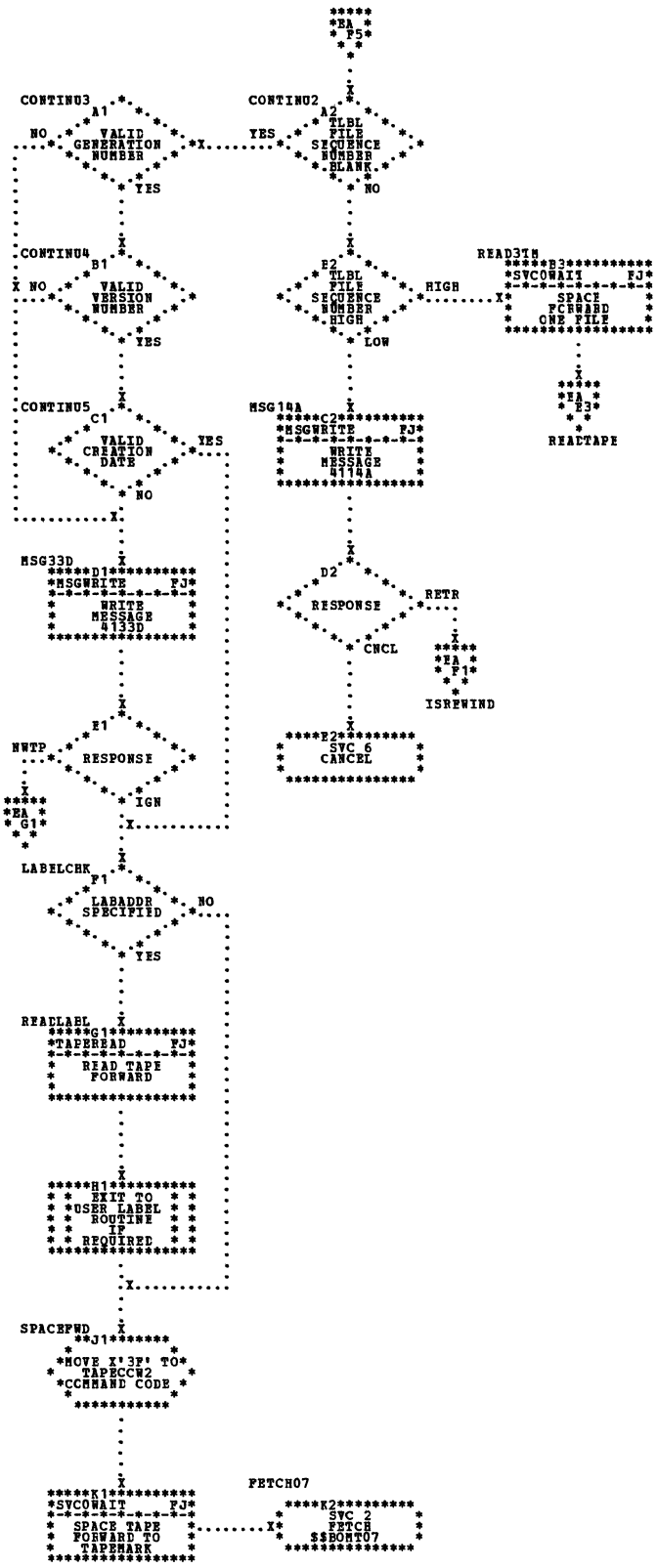


Chart ED. $$BOMT03: Open Output Standard Labels, Forward, Phase 1 (1 of 2)

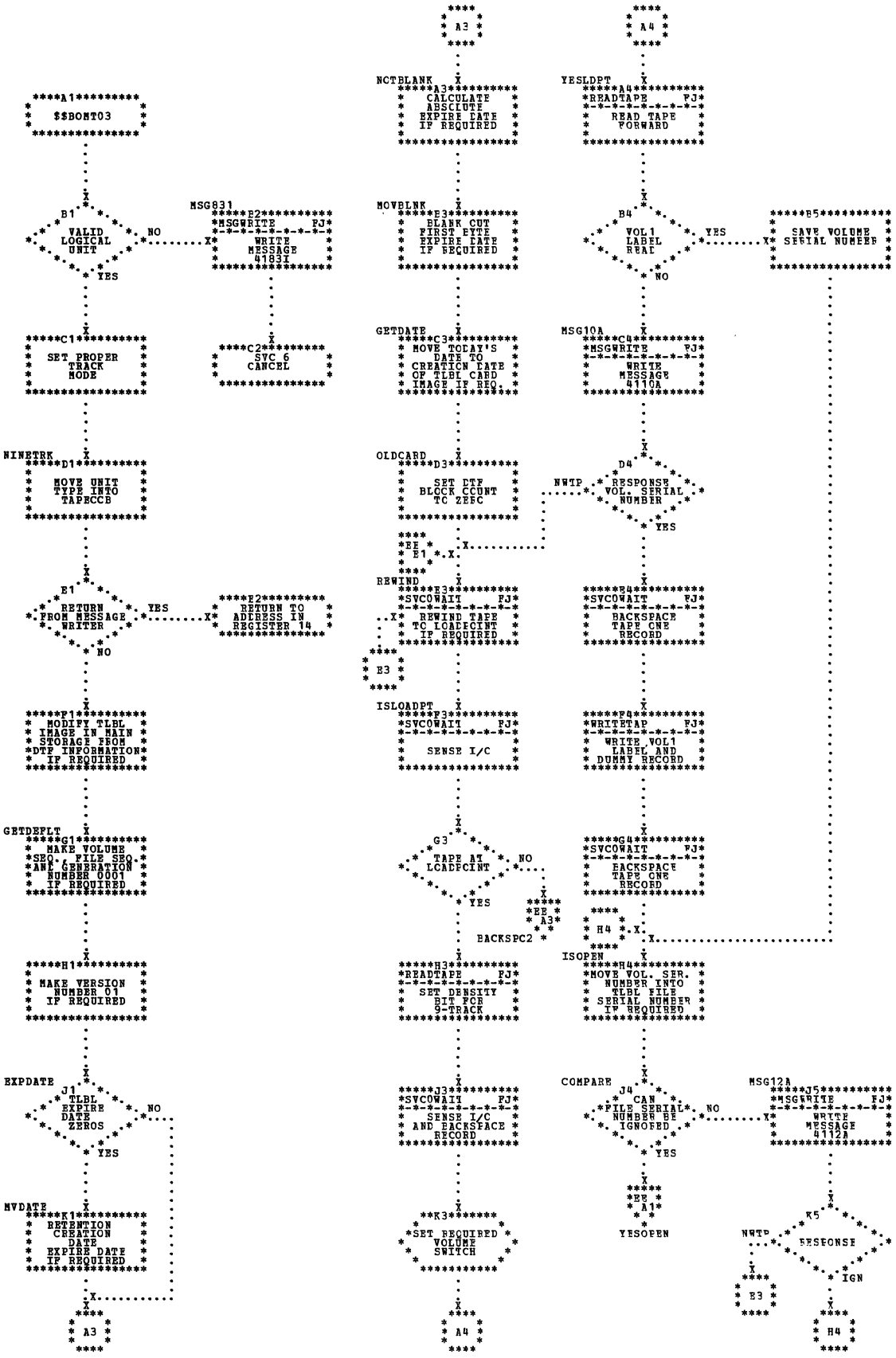
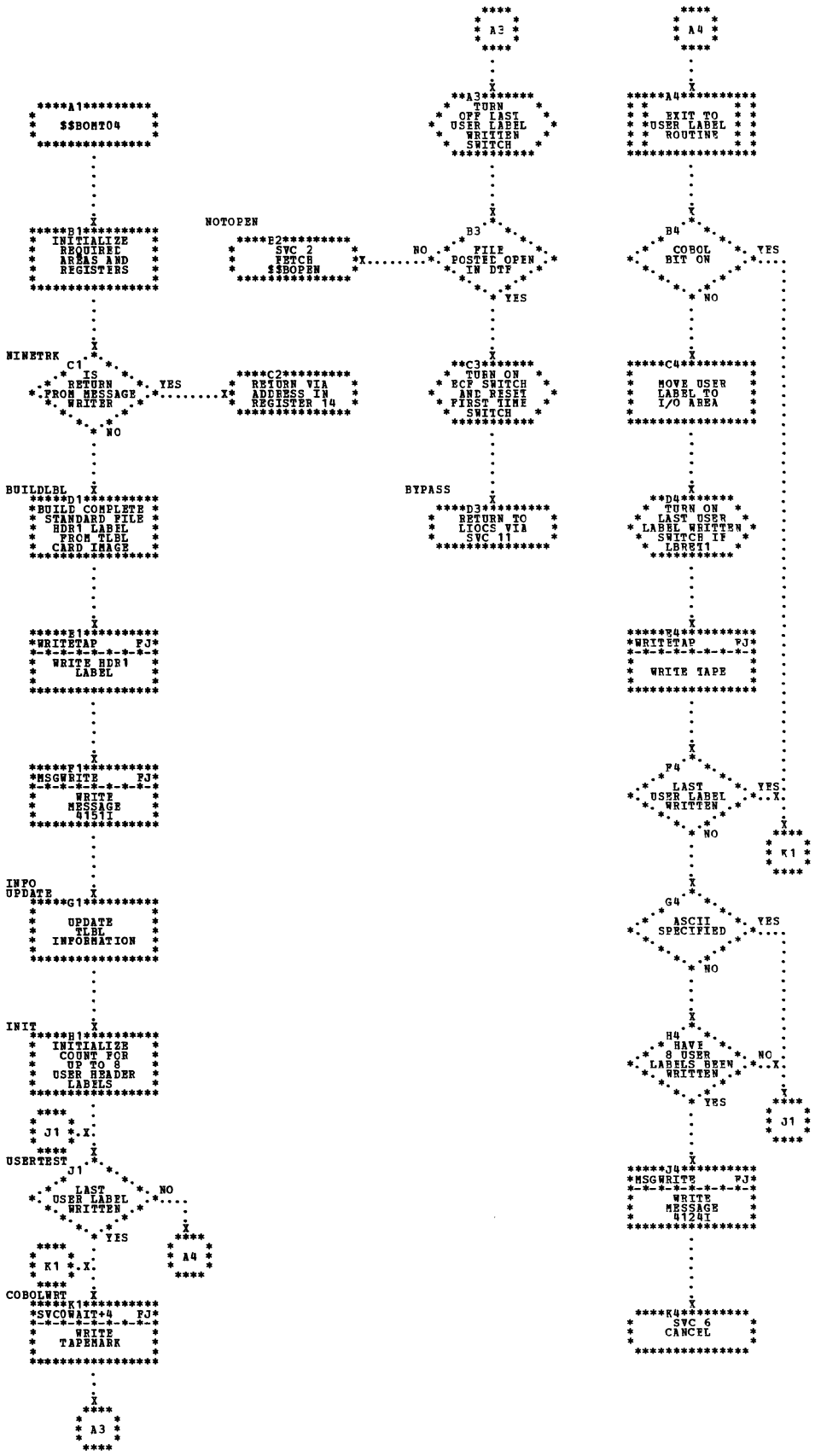


Chart EF. $$BOMT04: Open Output Standard Labels, Forward, Phase 2



## Chart EH. $$BOMT06: Open Workfiles

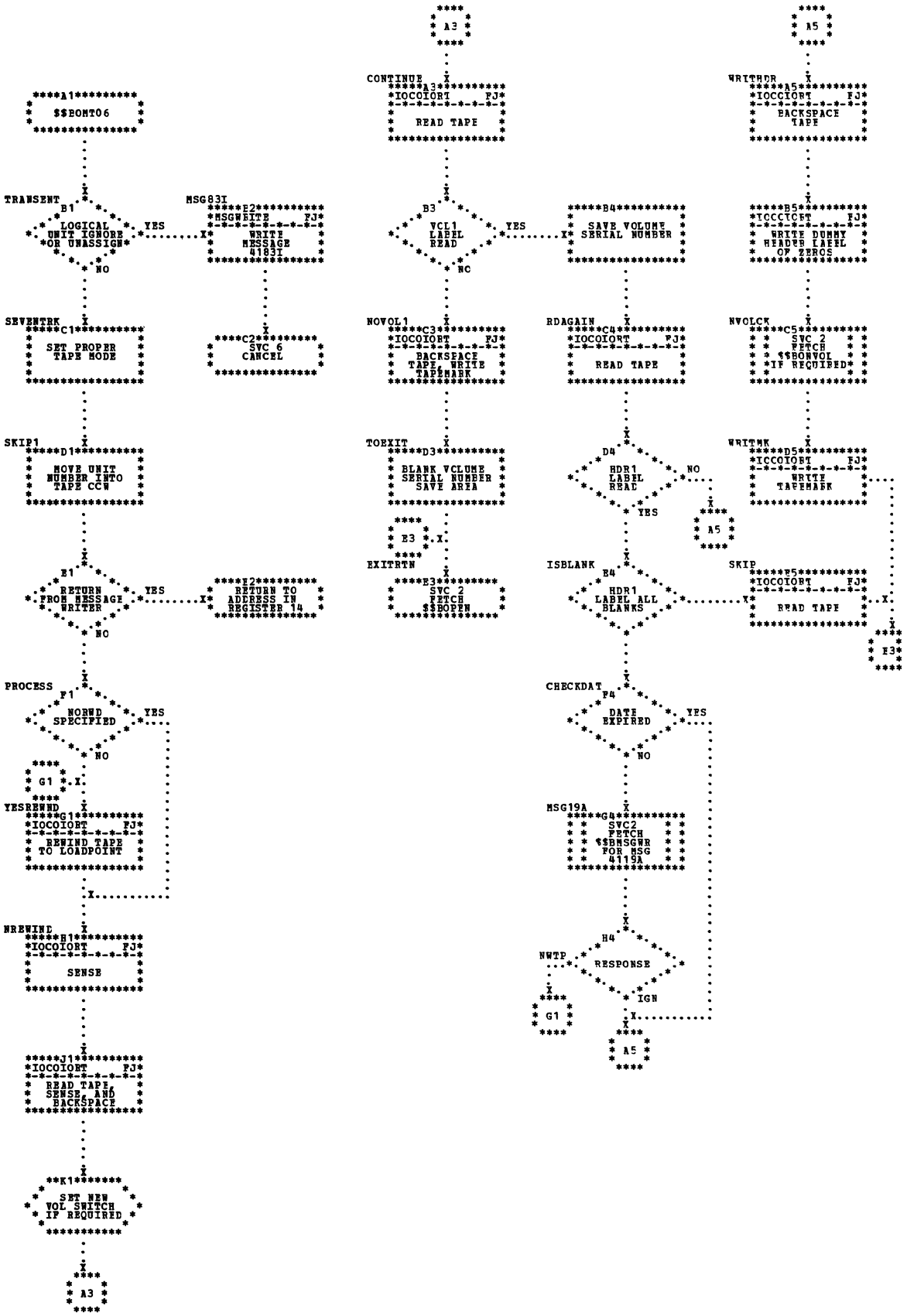
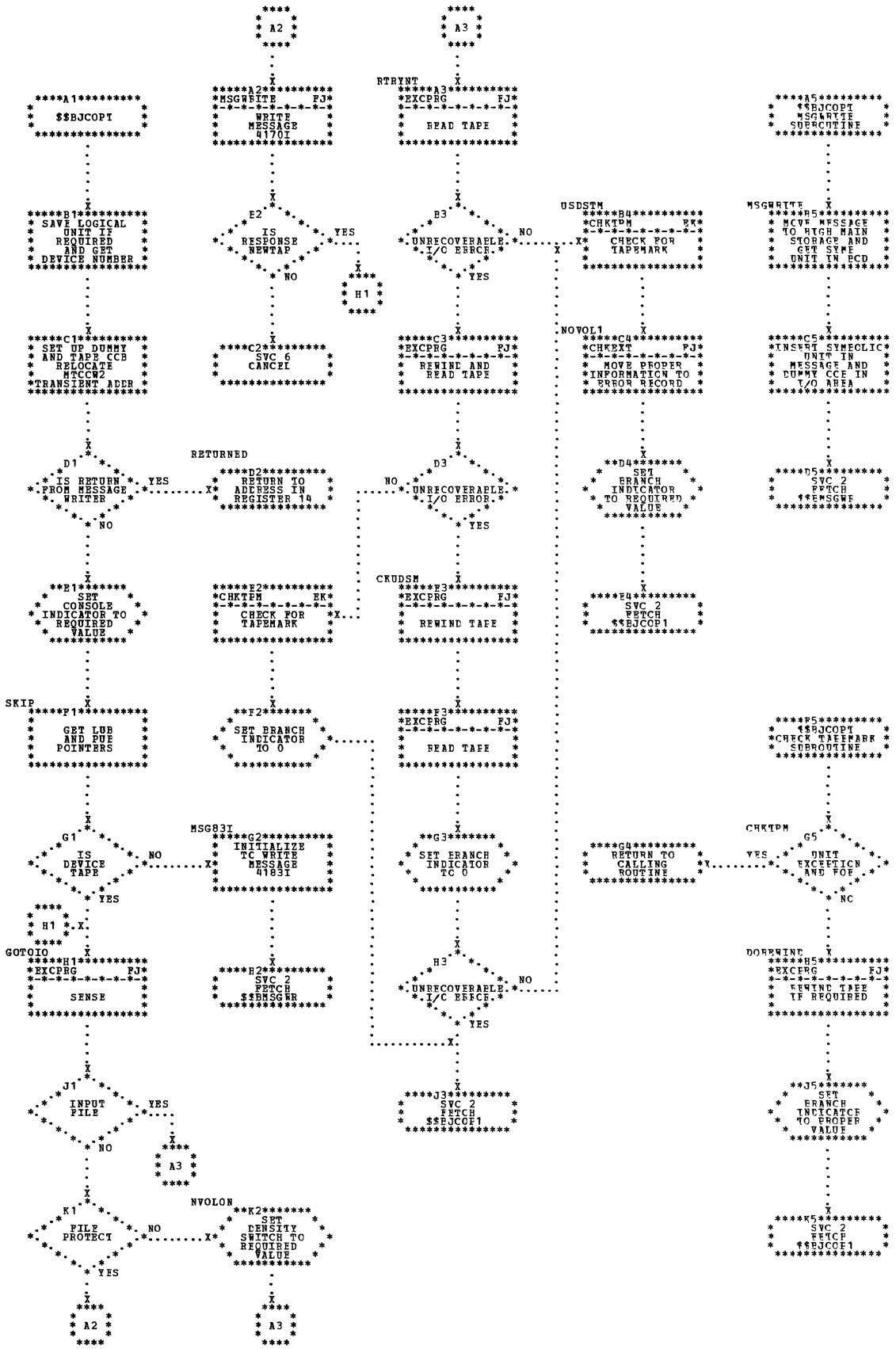
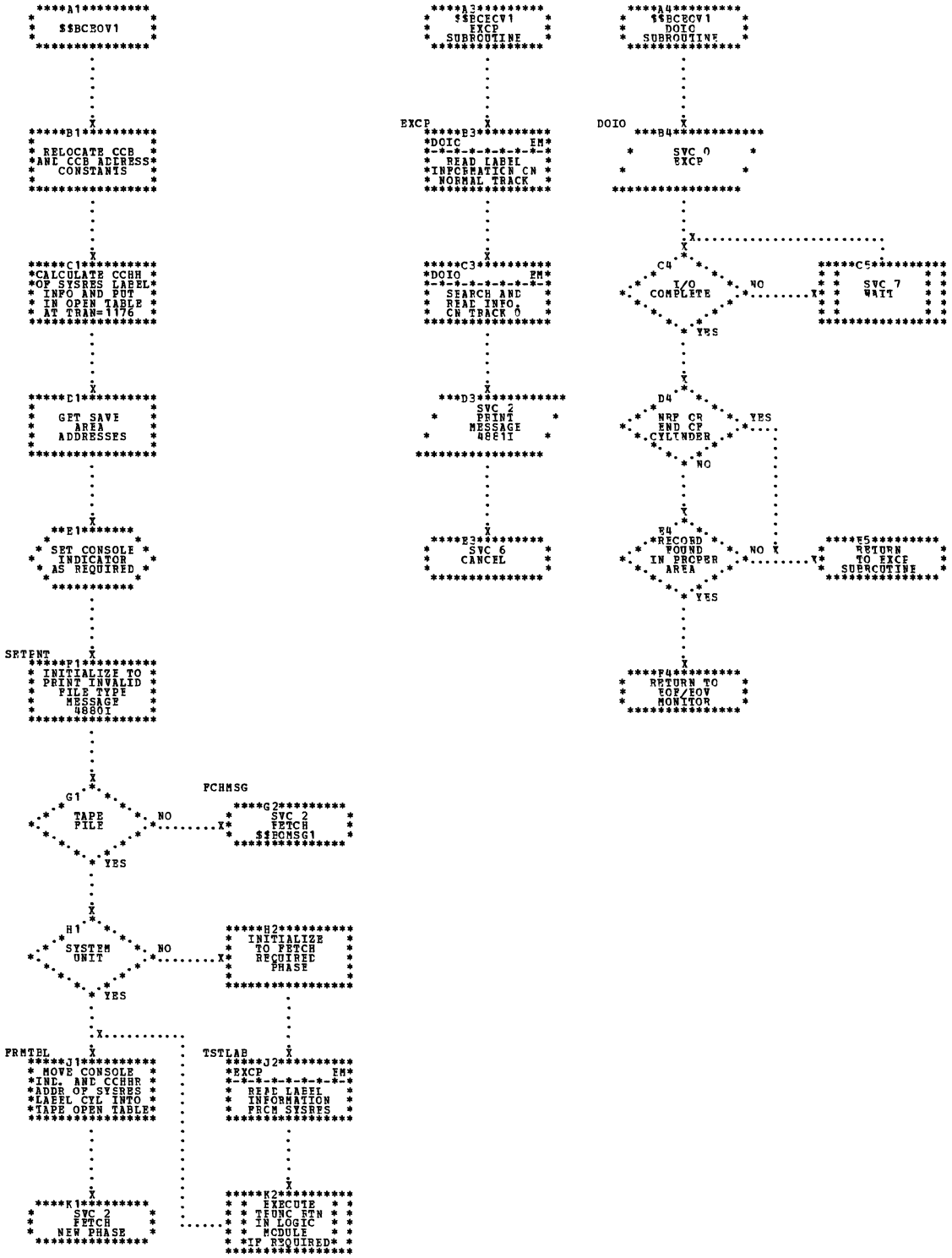


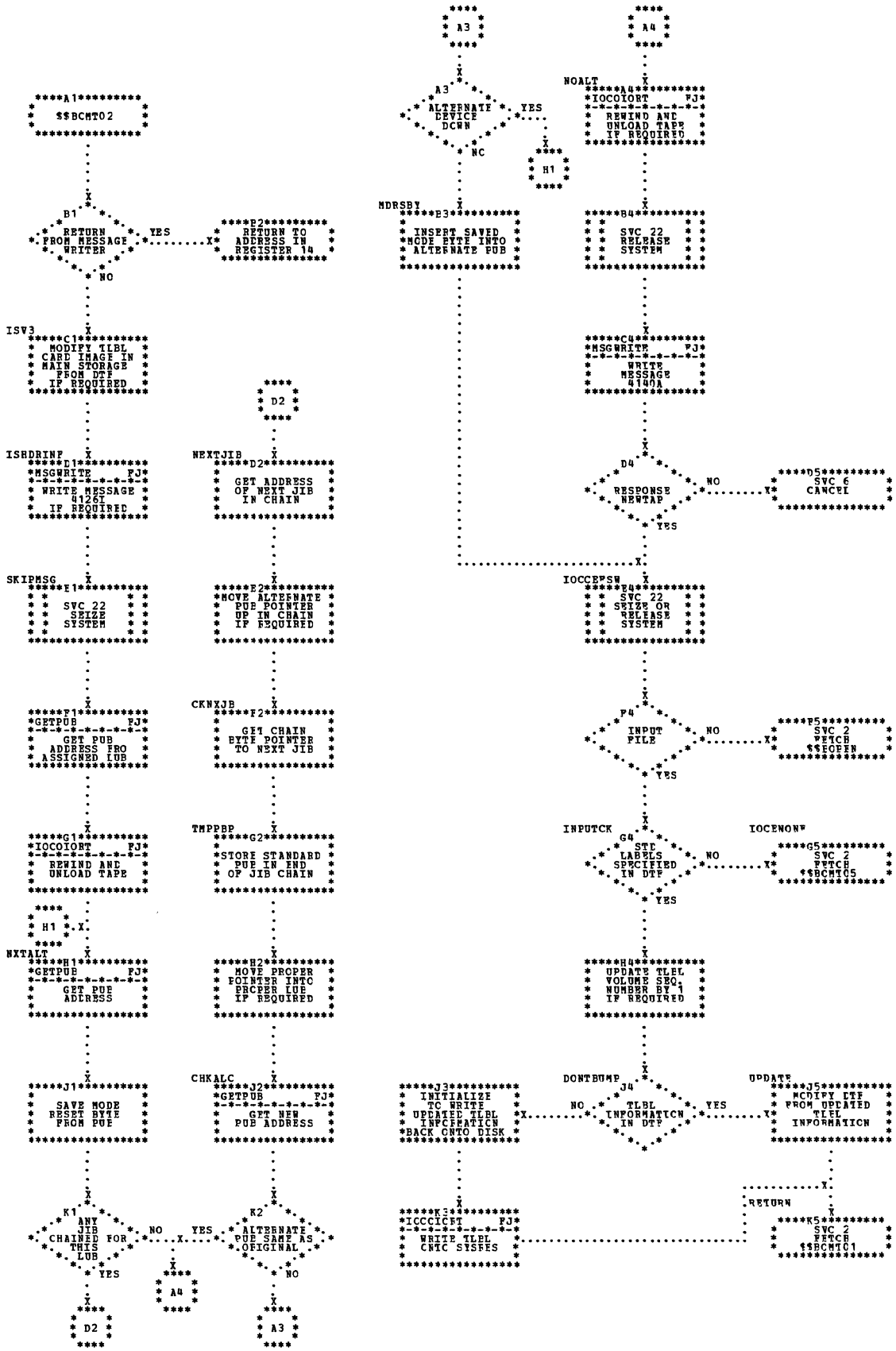
Chart EK. $$BJCCPT: Job Ccntrcl Tape Open Routine, Phase 1



## Chart EM. $$BCECV1: EOF/ECV Monitor



## Chart FB. $$BCMT02: Tape Close, Alternate Switching for EOV



## Chart FD. $$BCMT04: Tape Close, EOV Cutput Fcrward

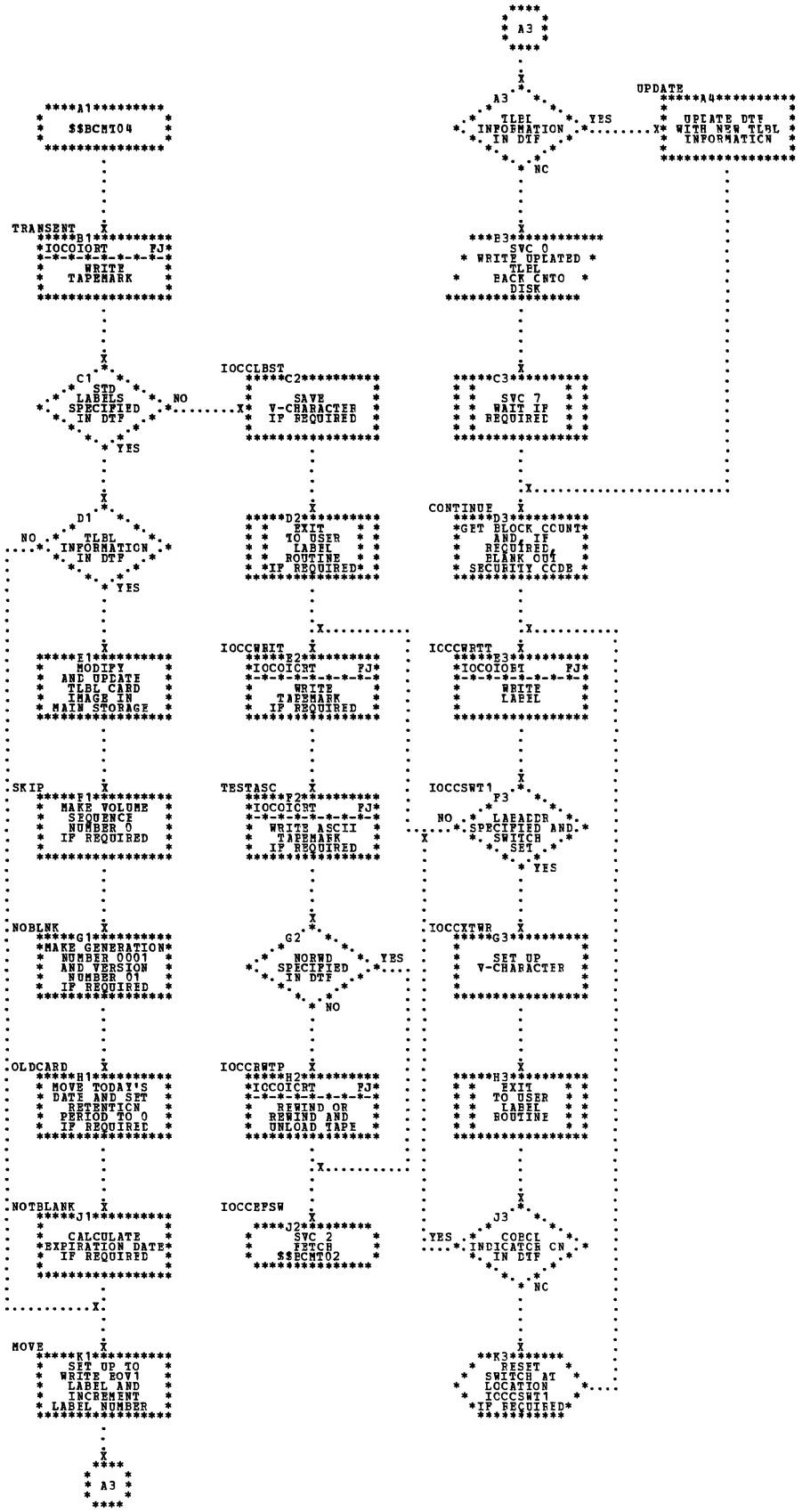


Chart FF. $$BCMT05: Close Standard, Nonstandard, and Unlabeled Files, All Types Except Workfiles (2 of 2)

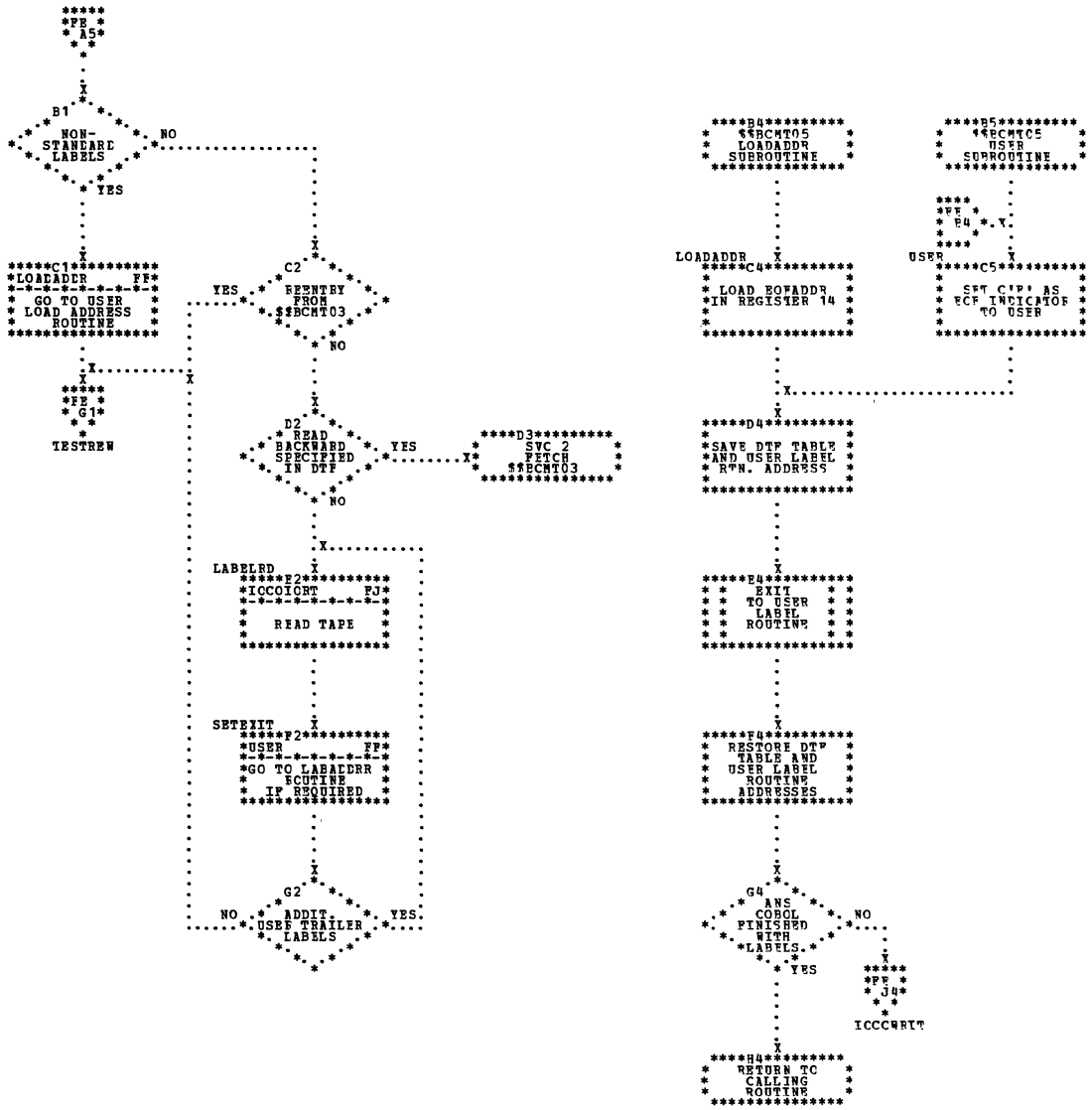


Chart FH. $$BCMT07: Tape Close, Alternate Switching for System Units

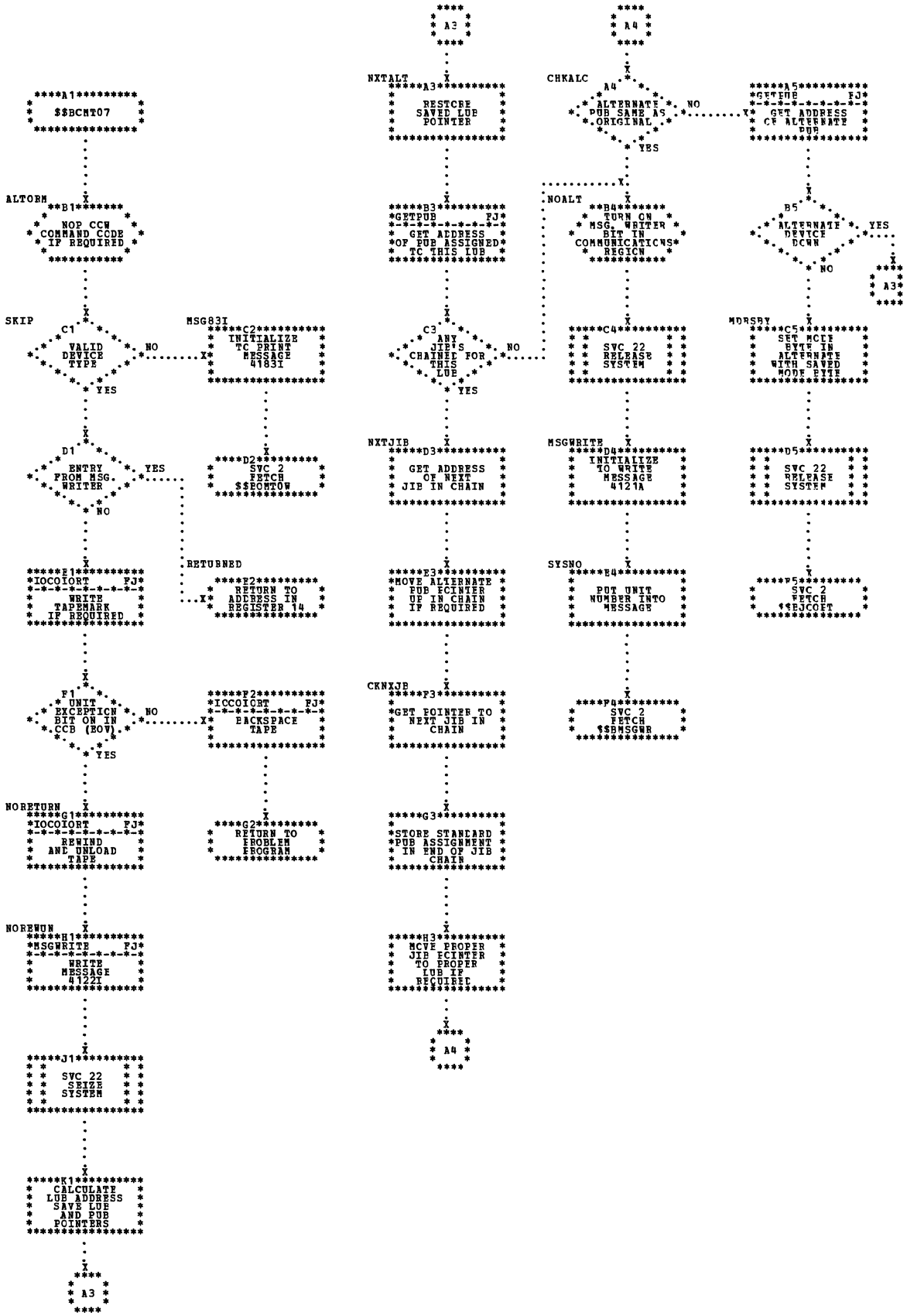


Chart FK. $$BOMT0M and $$BCMT0W: Tape Open Message Writers

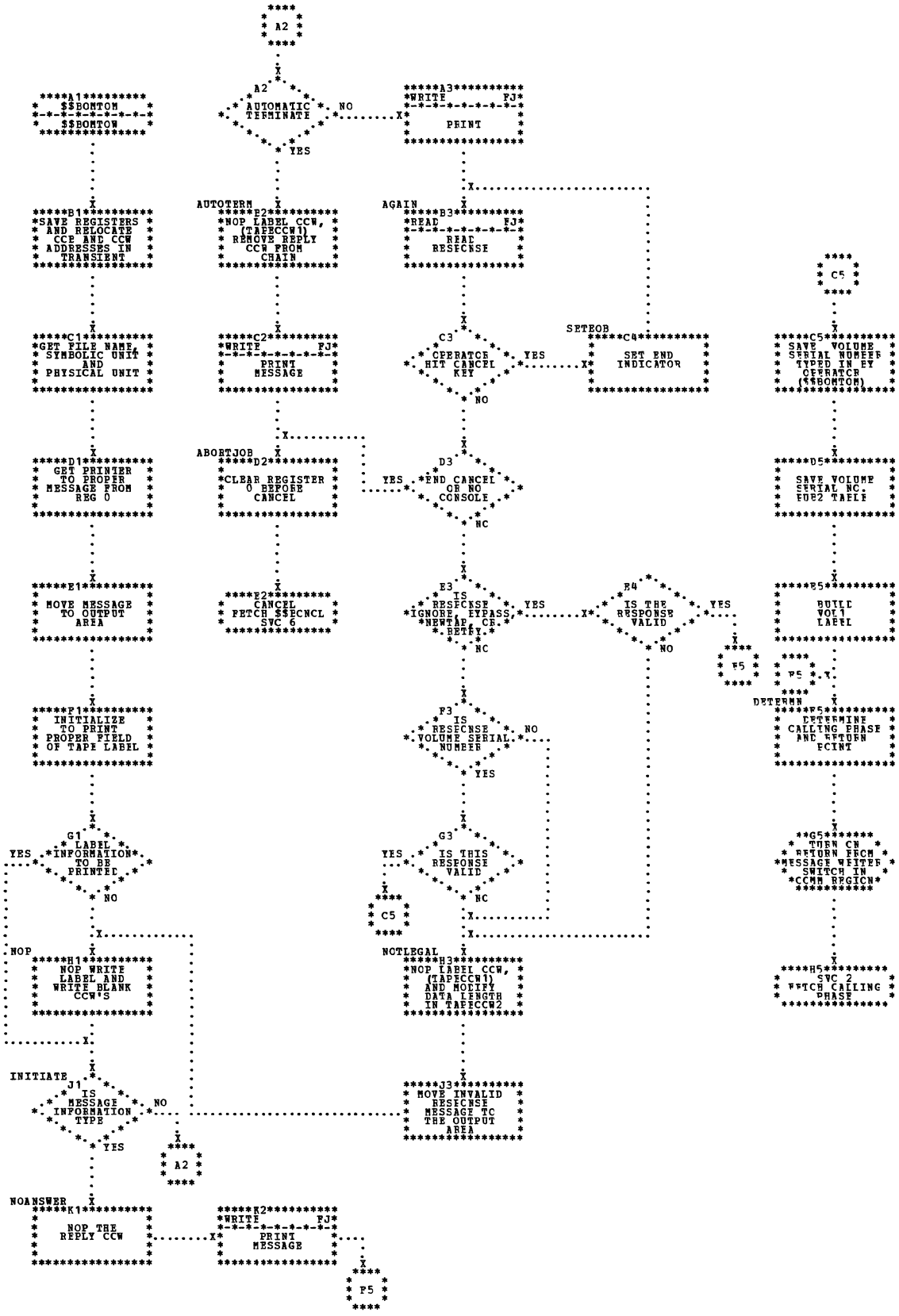


Chart FM. $$BCNVOL: Standard Volume Label Rewriter

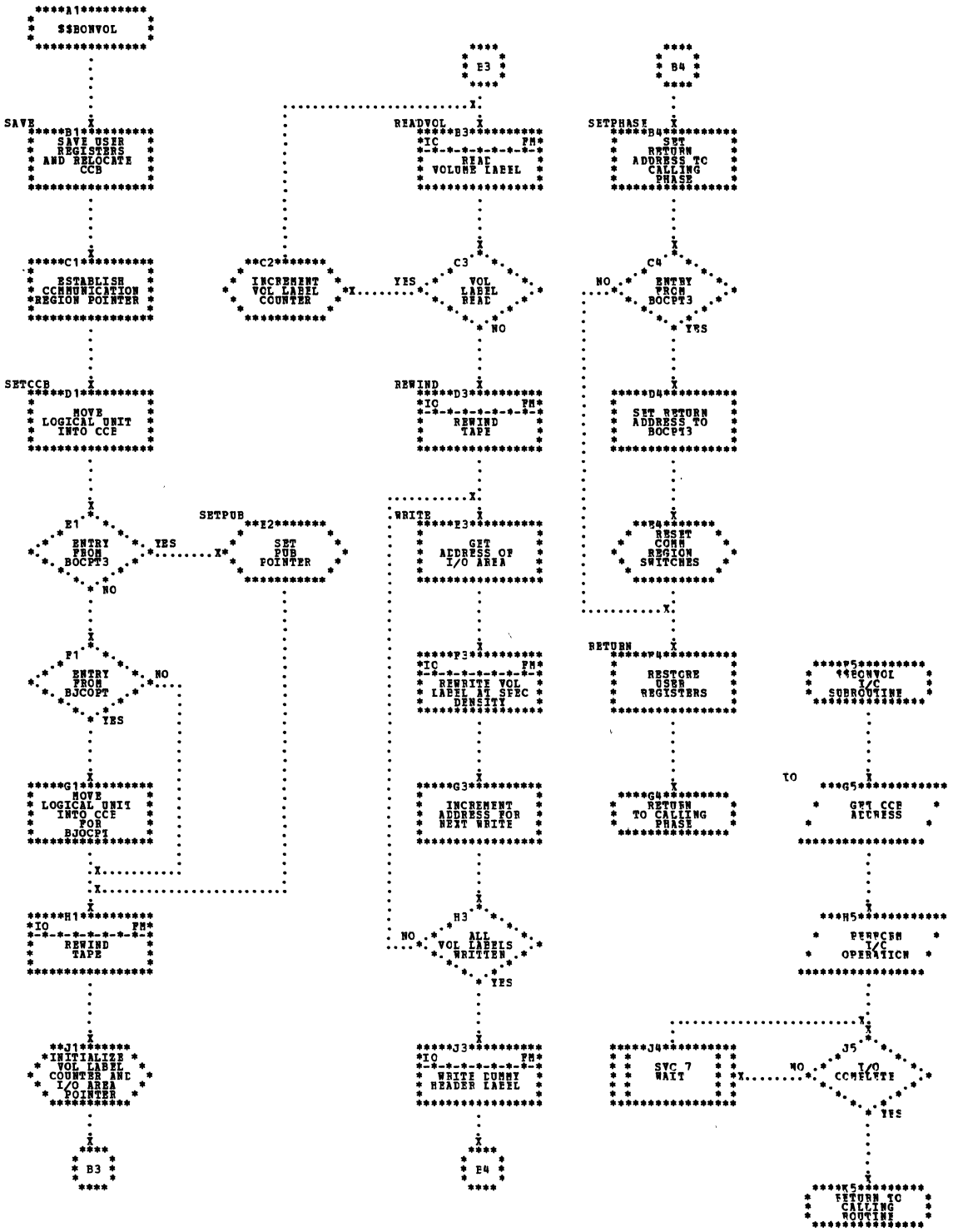


Chart GB. SDMODFI: GET Macro With Truncaticn (2 cf 2)

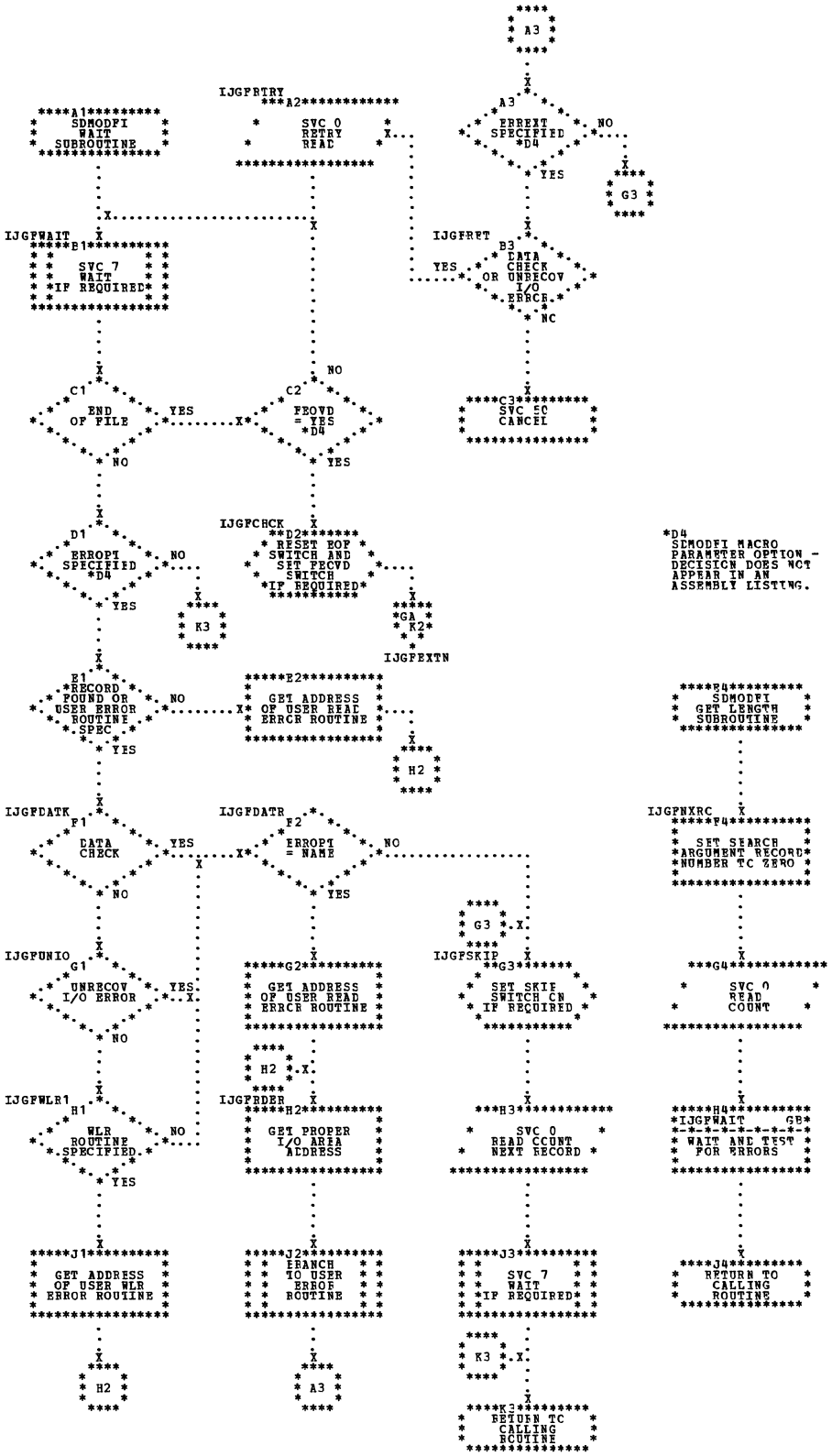


Chart GD. SDMCDFO: PUT Macro With Truncation

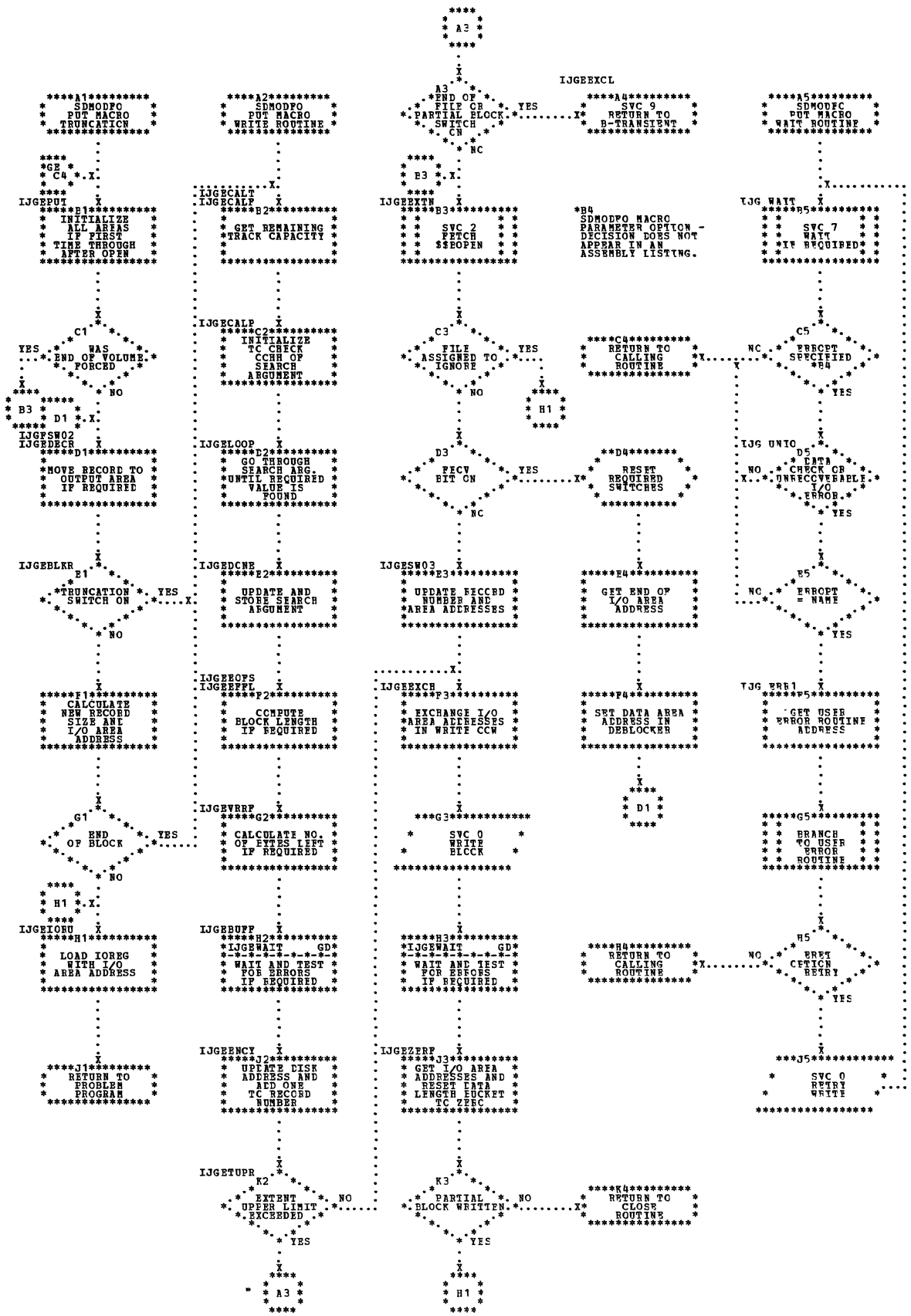


Chart GF. SDMODFO: PUT Macro Without Truncation

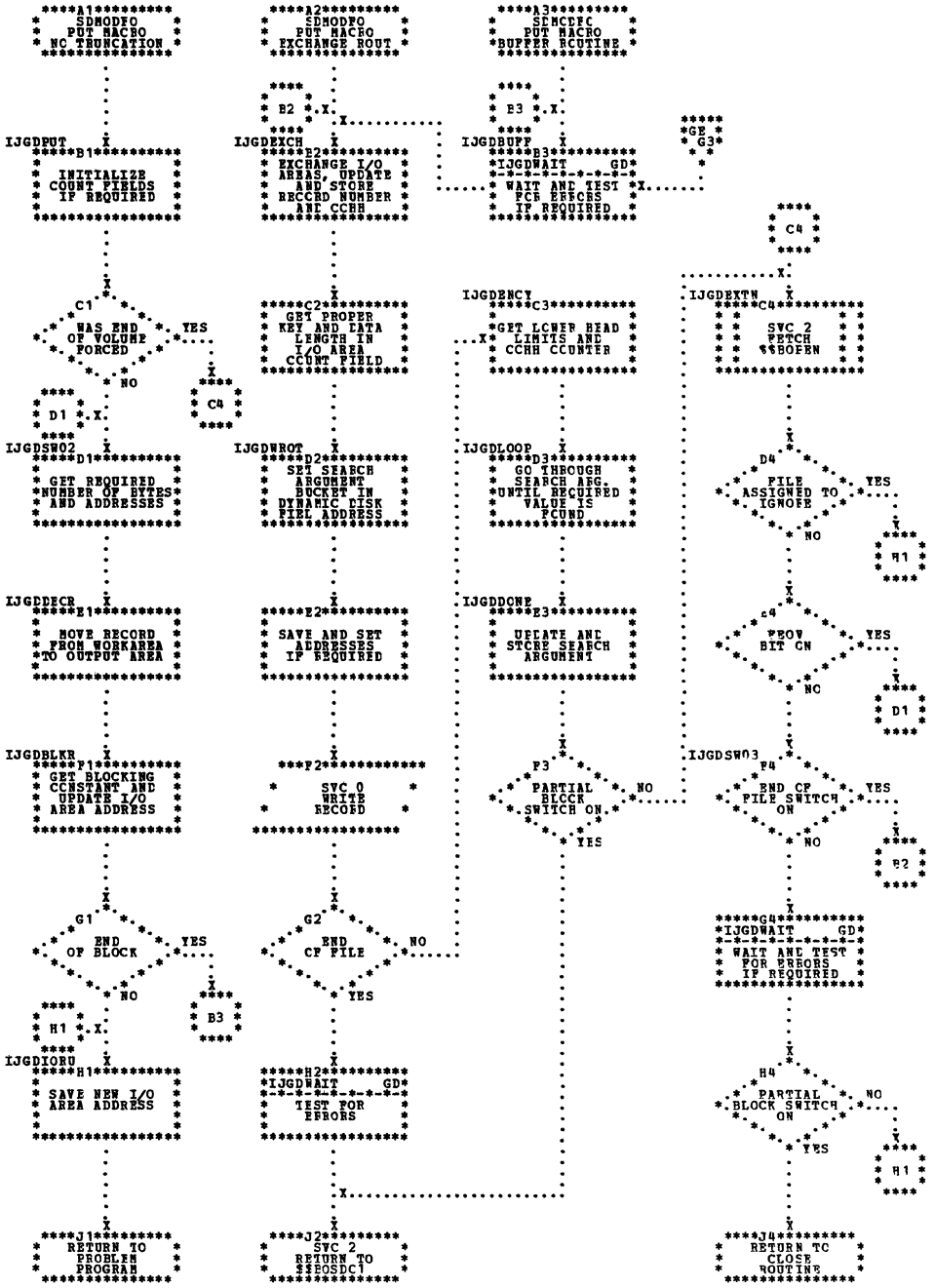


Chart GH. SDMODFU: GET Macro With Truncaticn (2 cf 3)

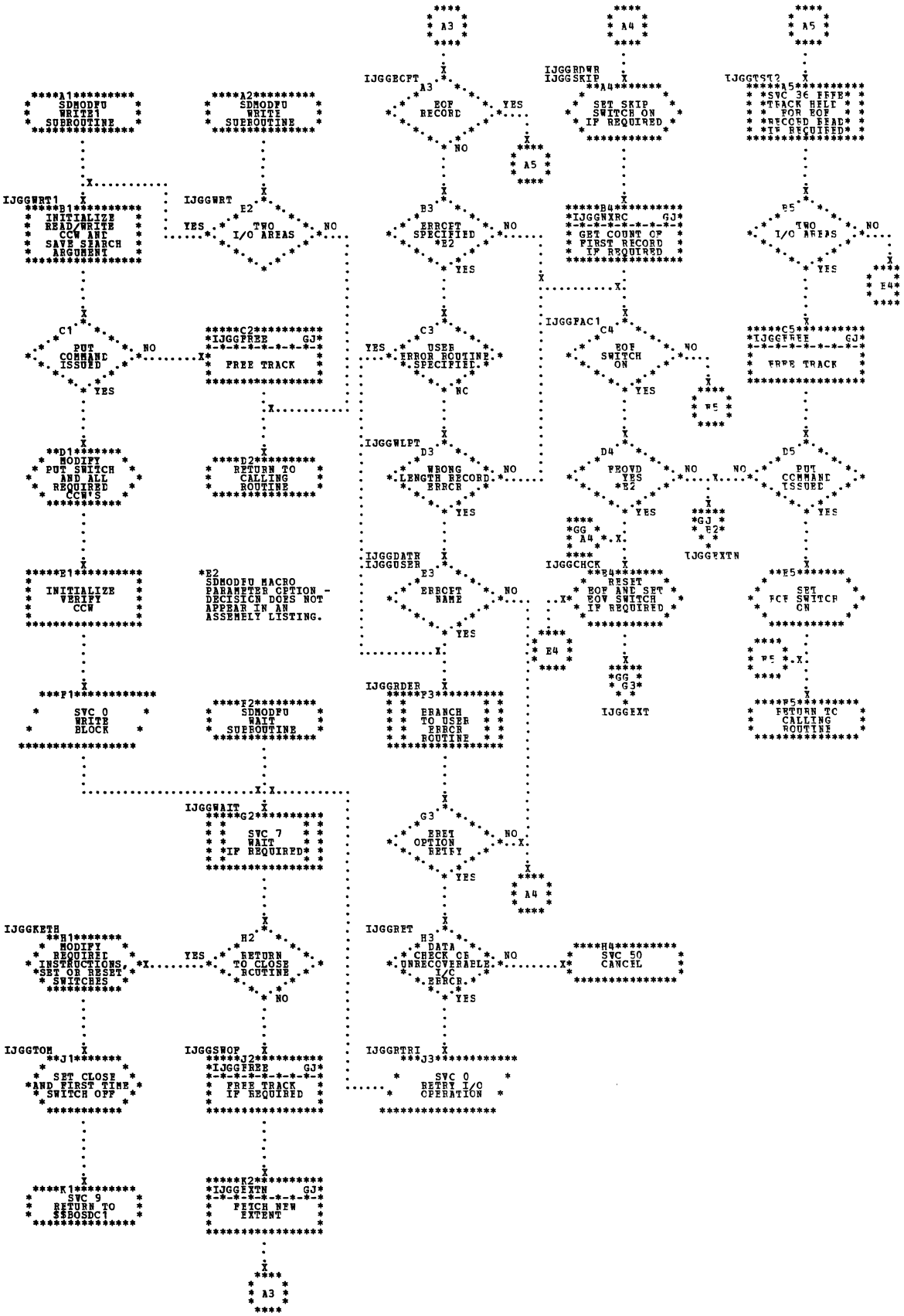


Chart GK. SDMODFU: Without Truncation, GET Macro and CLOSE Routine (1 of 3)

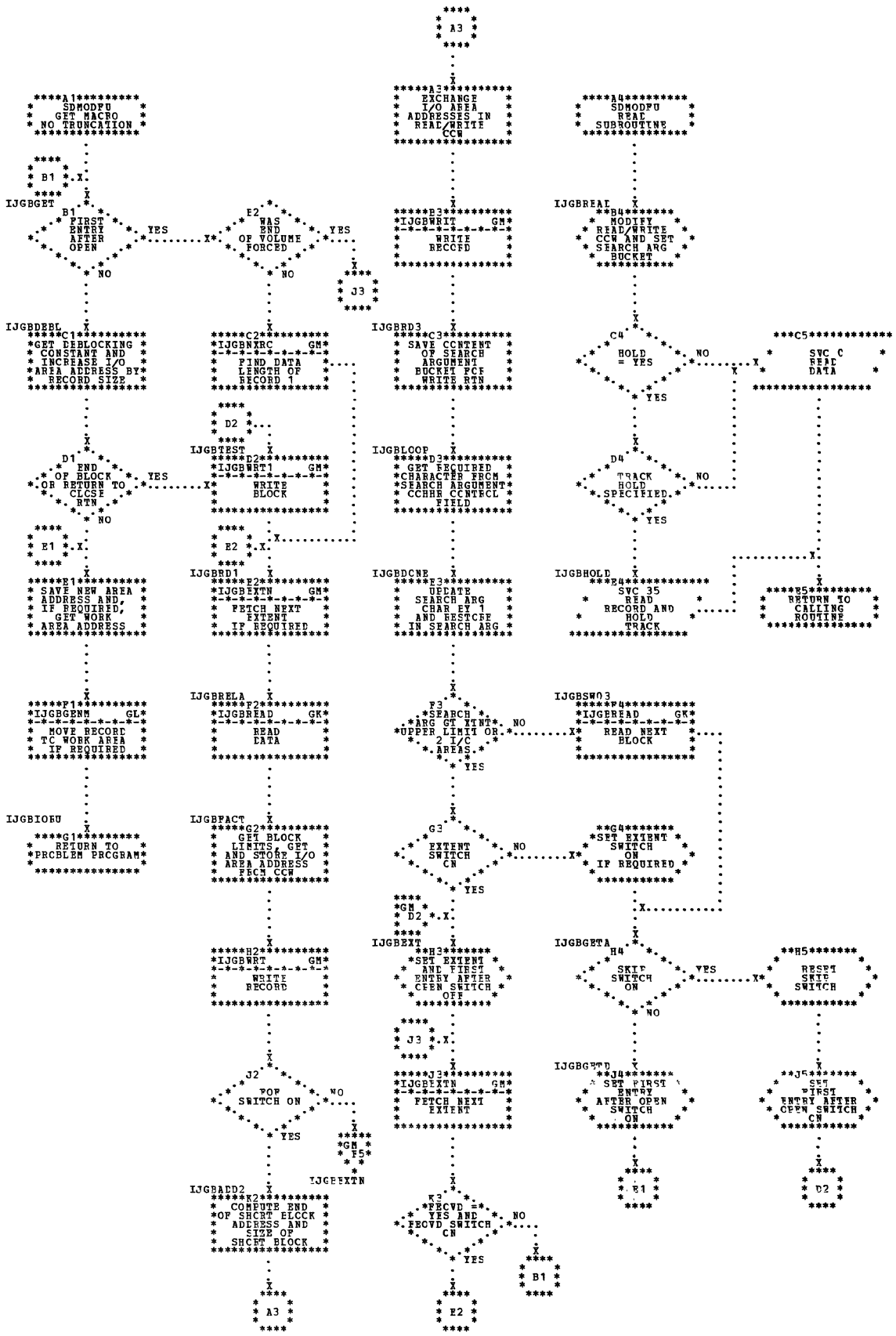


Chart GM. SDMODFU: Without Truncation, GET Macro and CLOSE Routine (3 of 3)

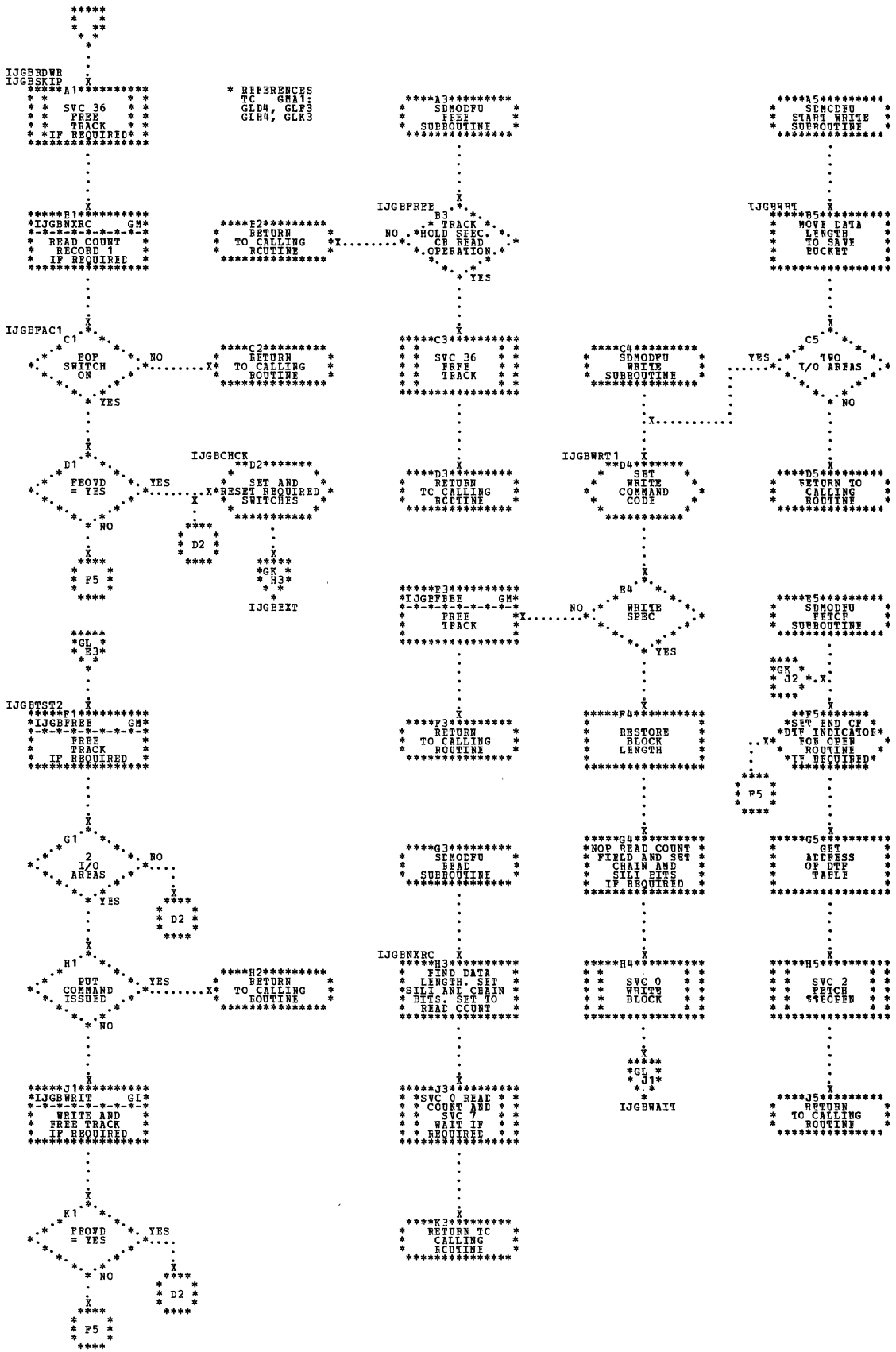


Chart HA. SDMODVI: GET Macro (1 cf 3)

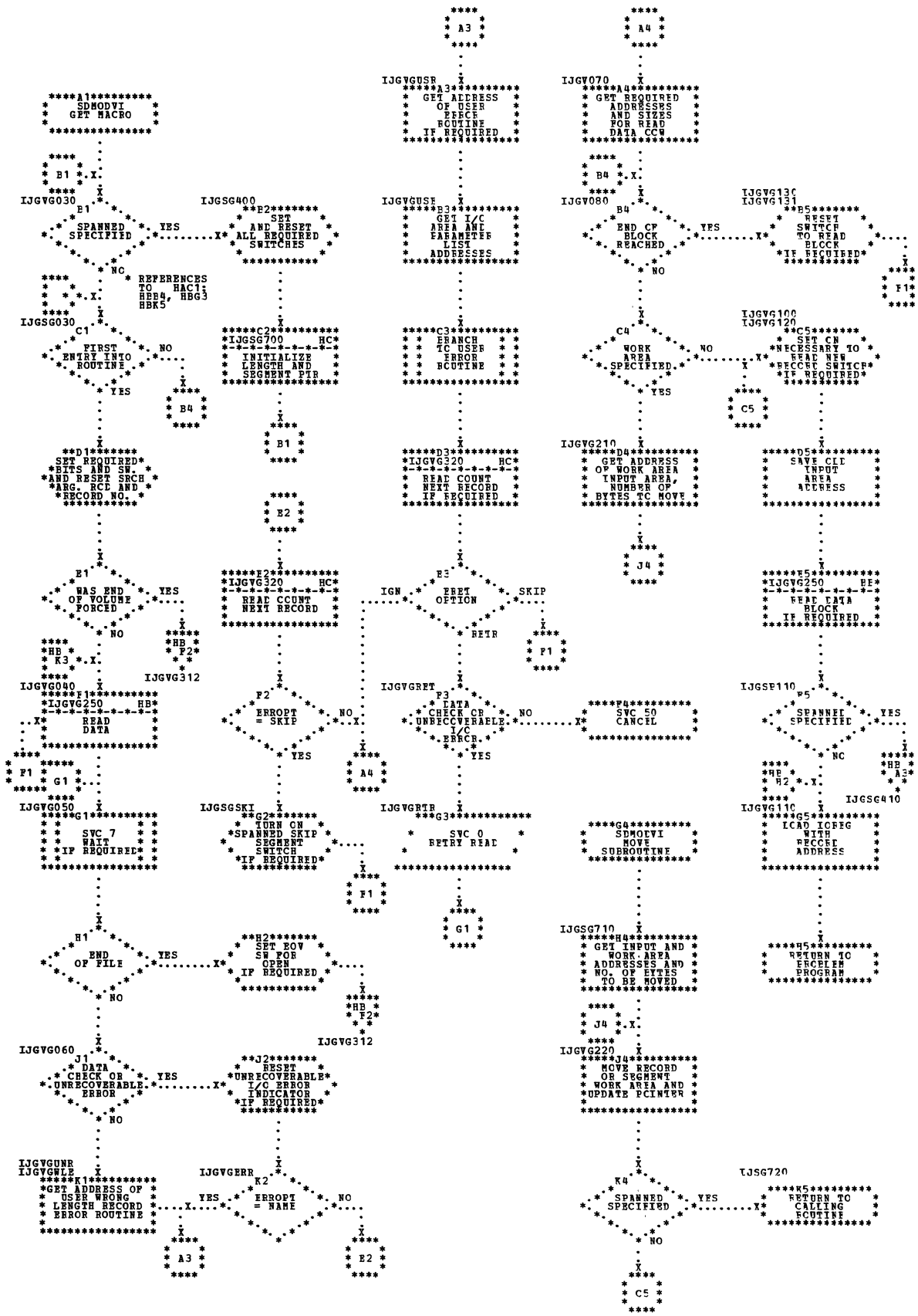


**Chart HC. SDMODVI: GET Macro (3 cf 3)**

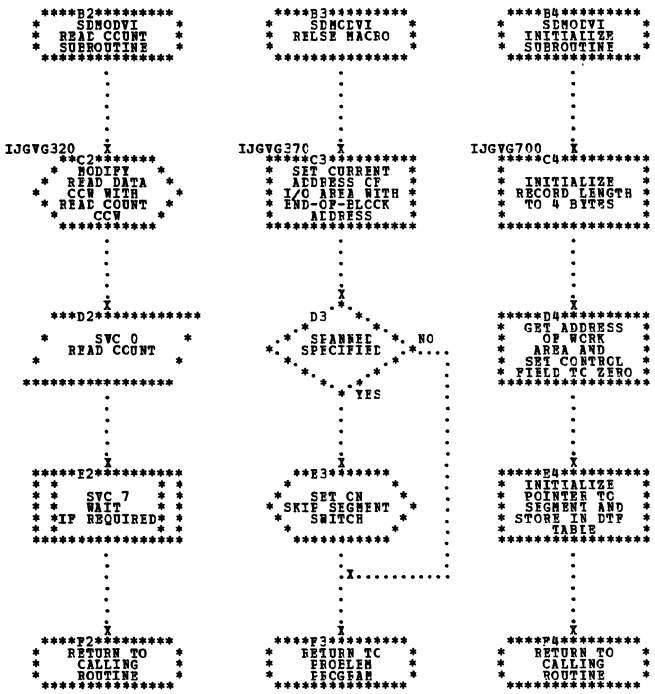


Chart HE. SDMODVO: PUT Macro (2 cf 4)

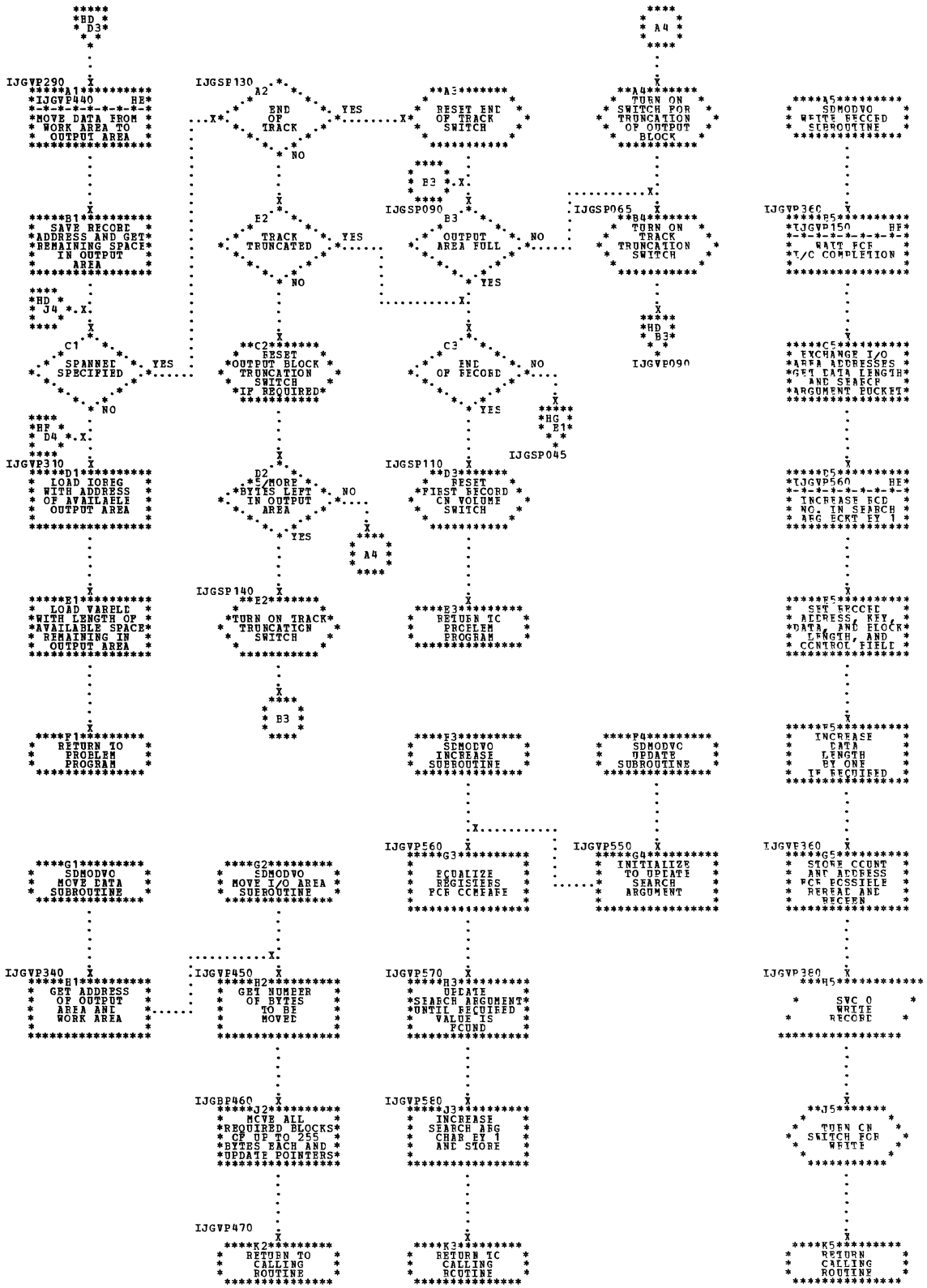


Chart HG. SDMODVO: PUT Macro (4 of 4)

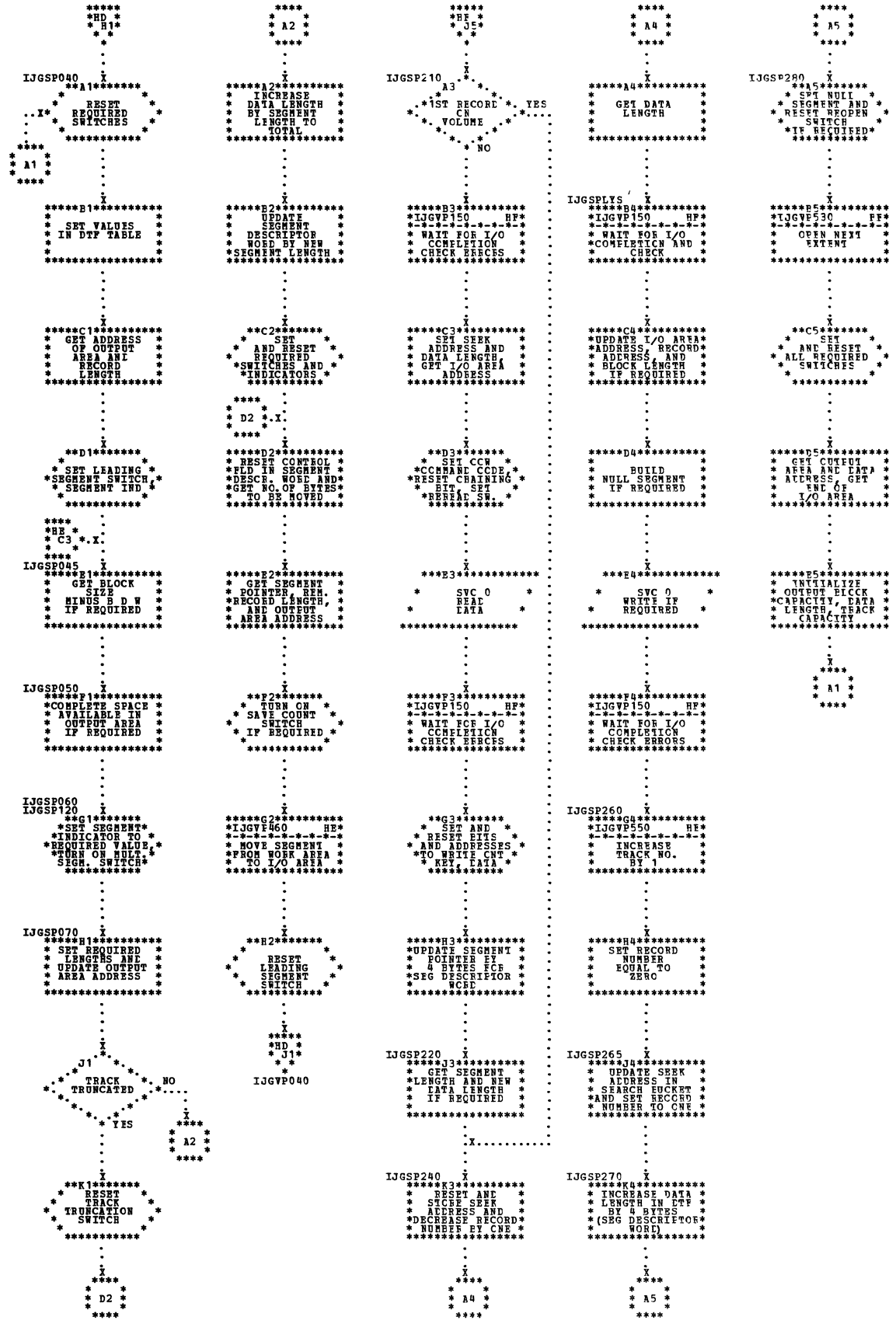


Chart HJ. SDMODVU: GET Macro (1 cf 4)

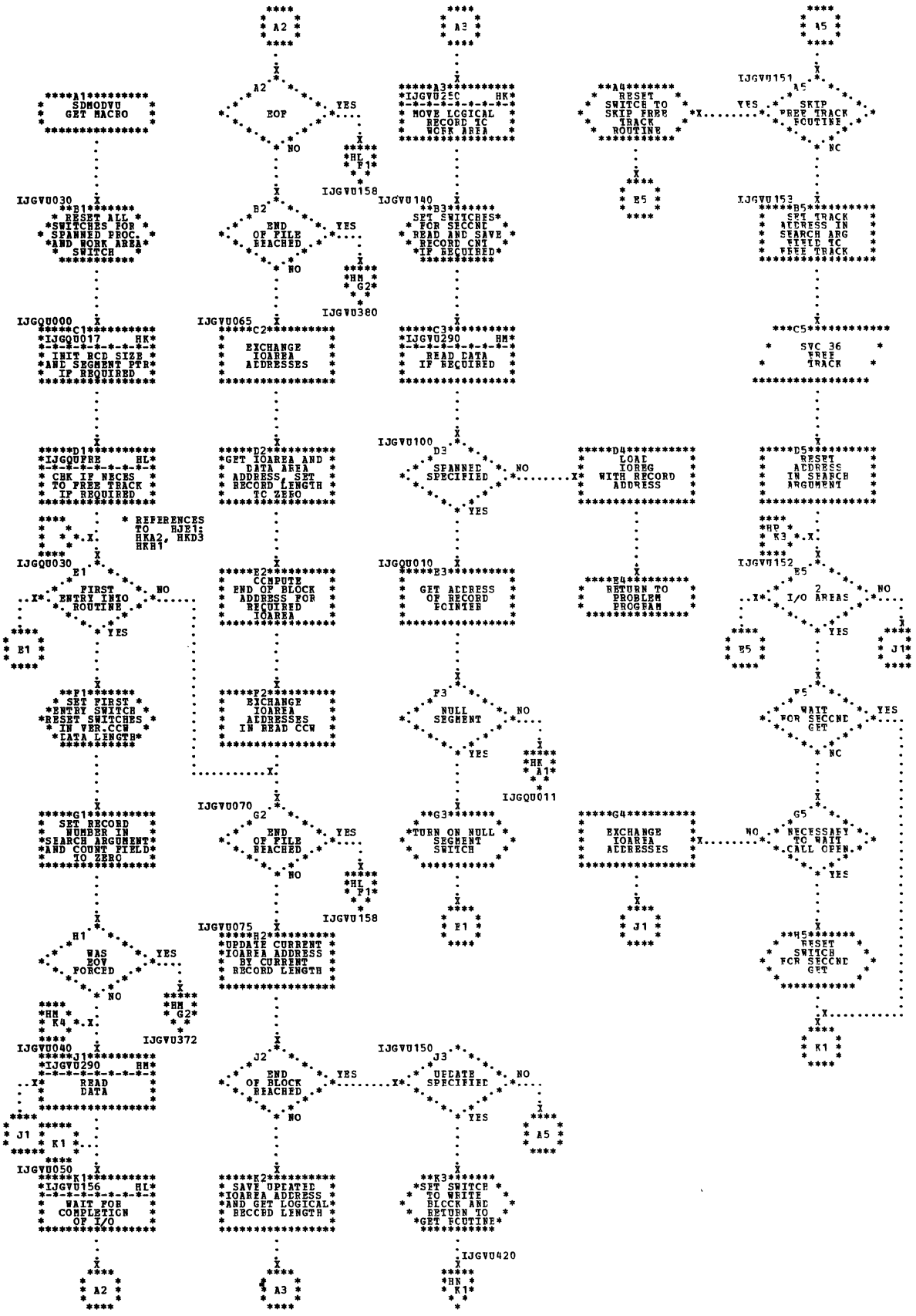


Chart HL. SDMODVU: GET Macro (3 cf 4)

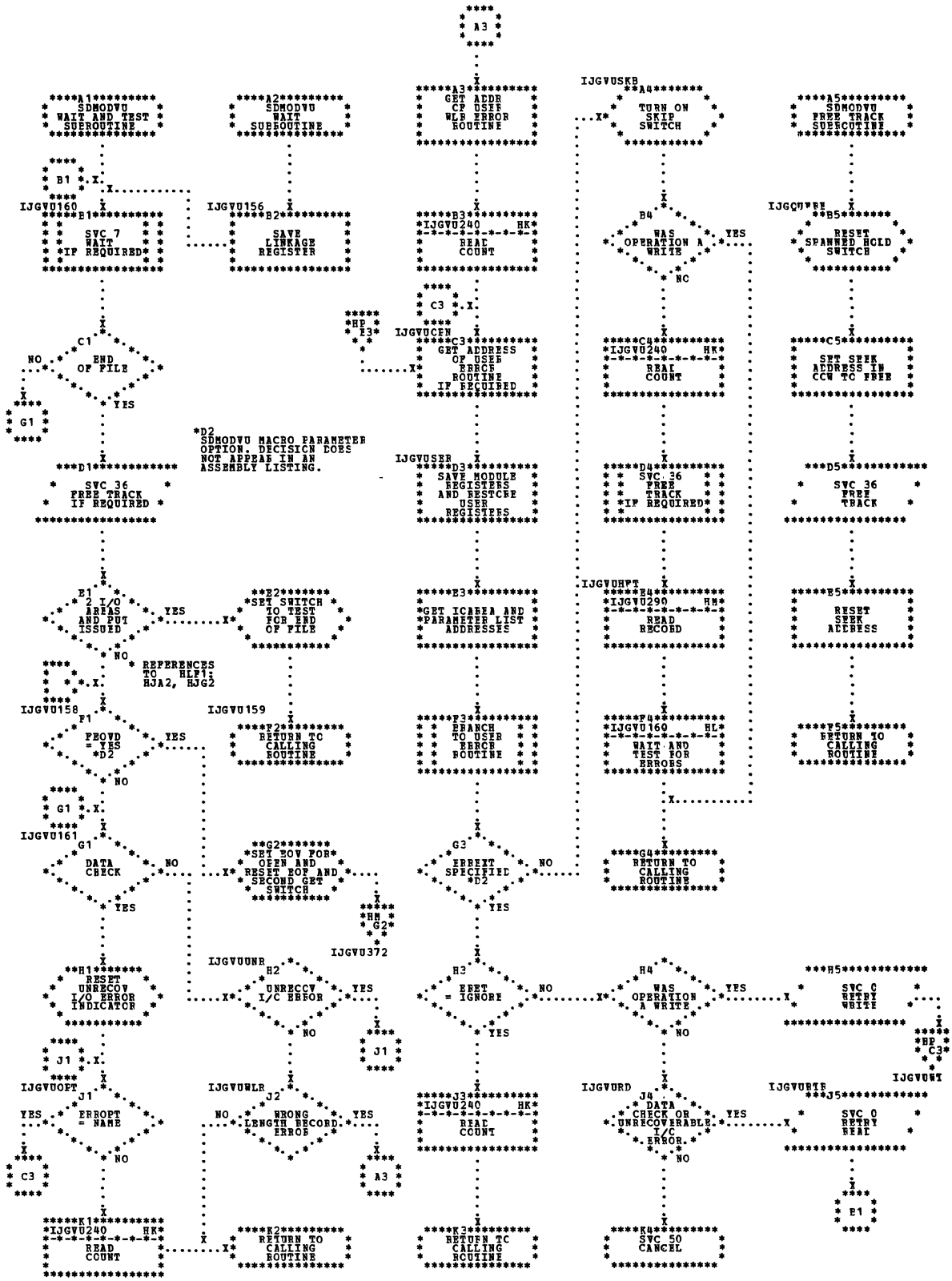


Chart HN. SDMODVU: PUT Macro (1 of 3)

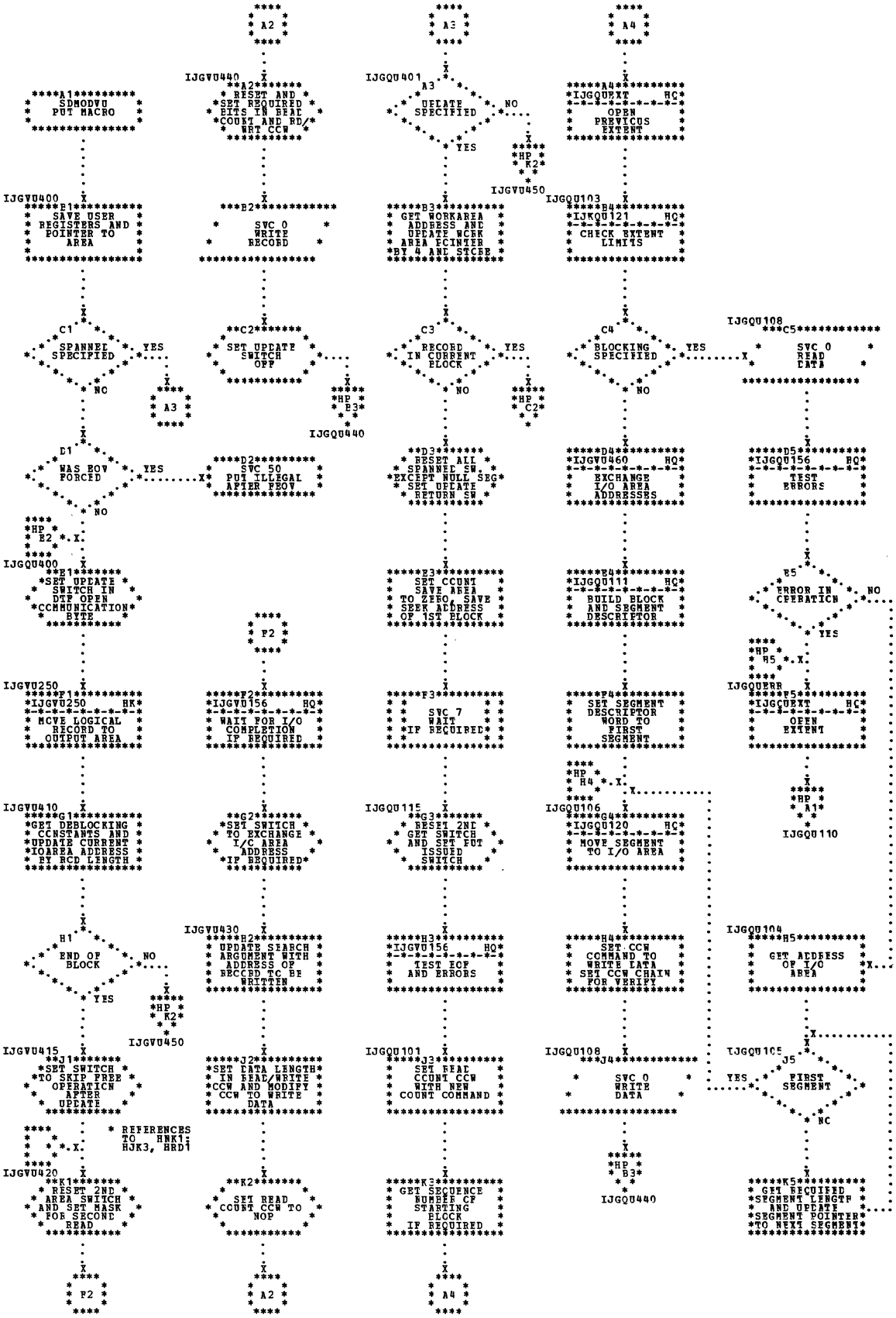


Chart HQ. SDMODVU: PUT Macro (3 cf 3)

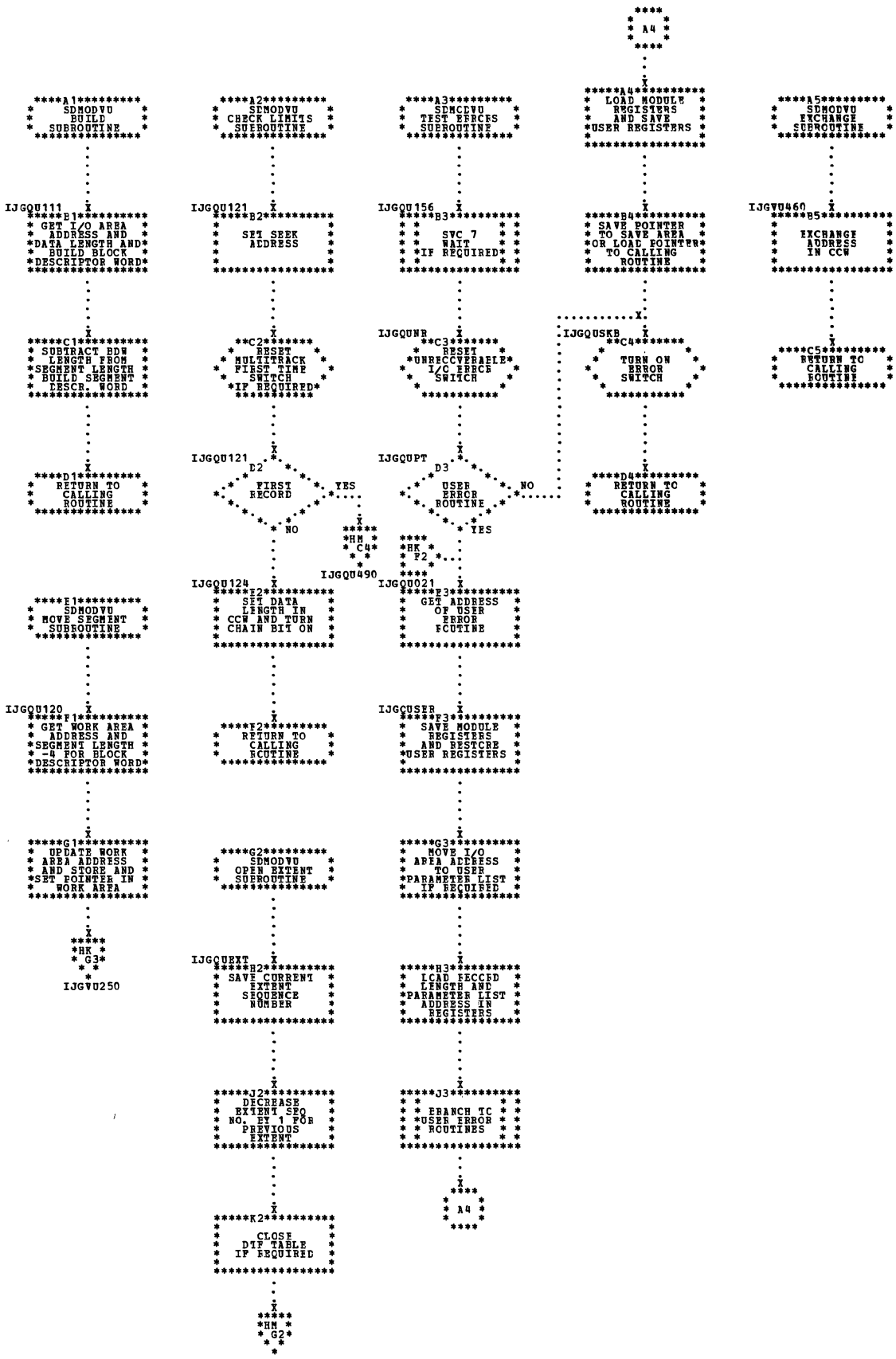


Chart JA. SDMODUI: GET Macro (1 of 2)

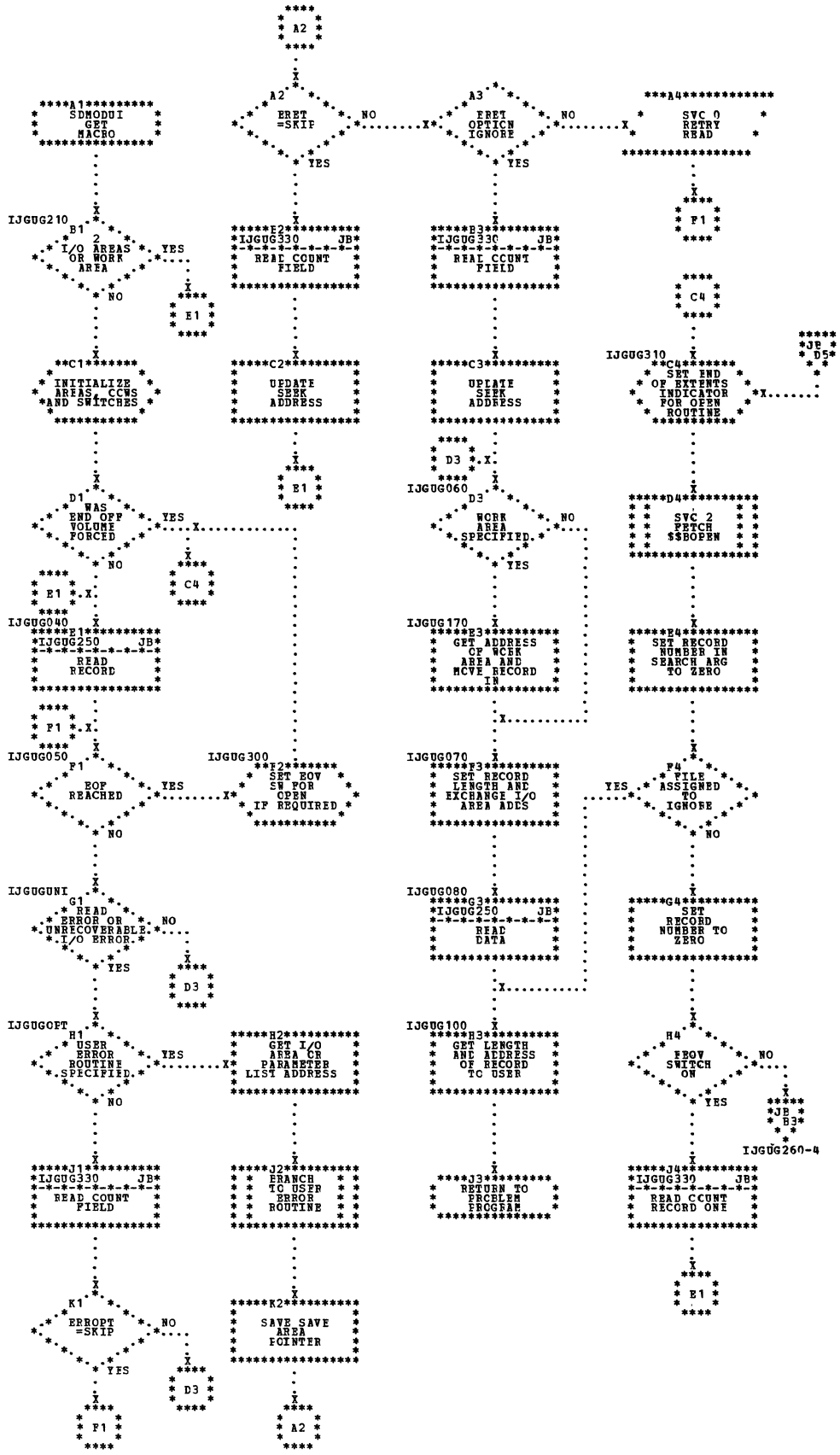


Chart JC. SDMODUO: PUT Macro (1 of 2)

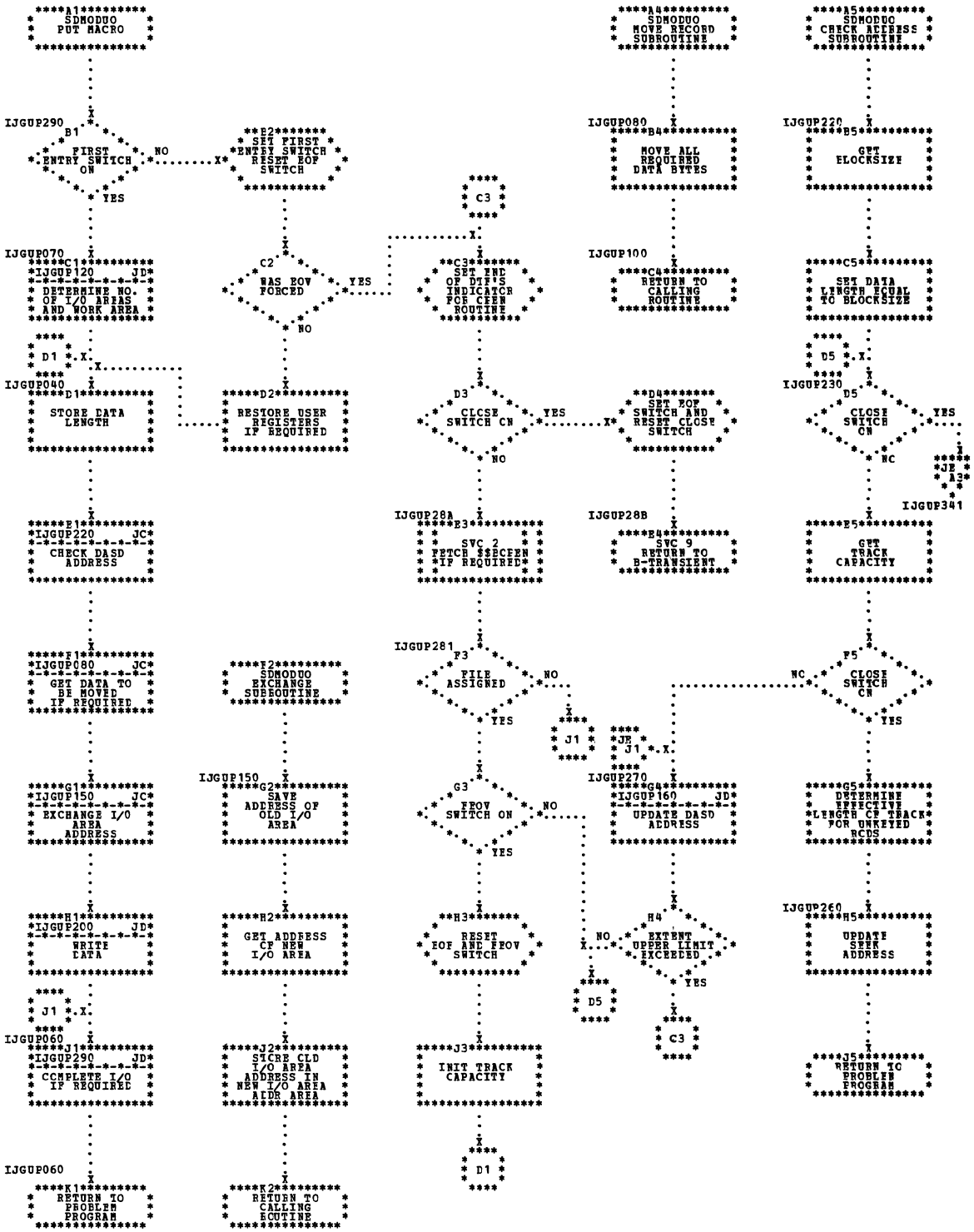


Chart JE. SDMODUO: Close Rcutine

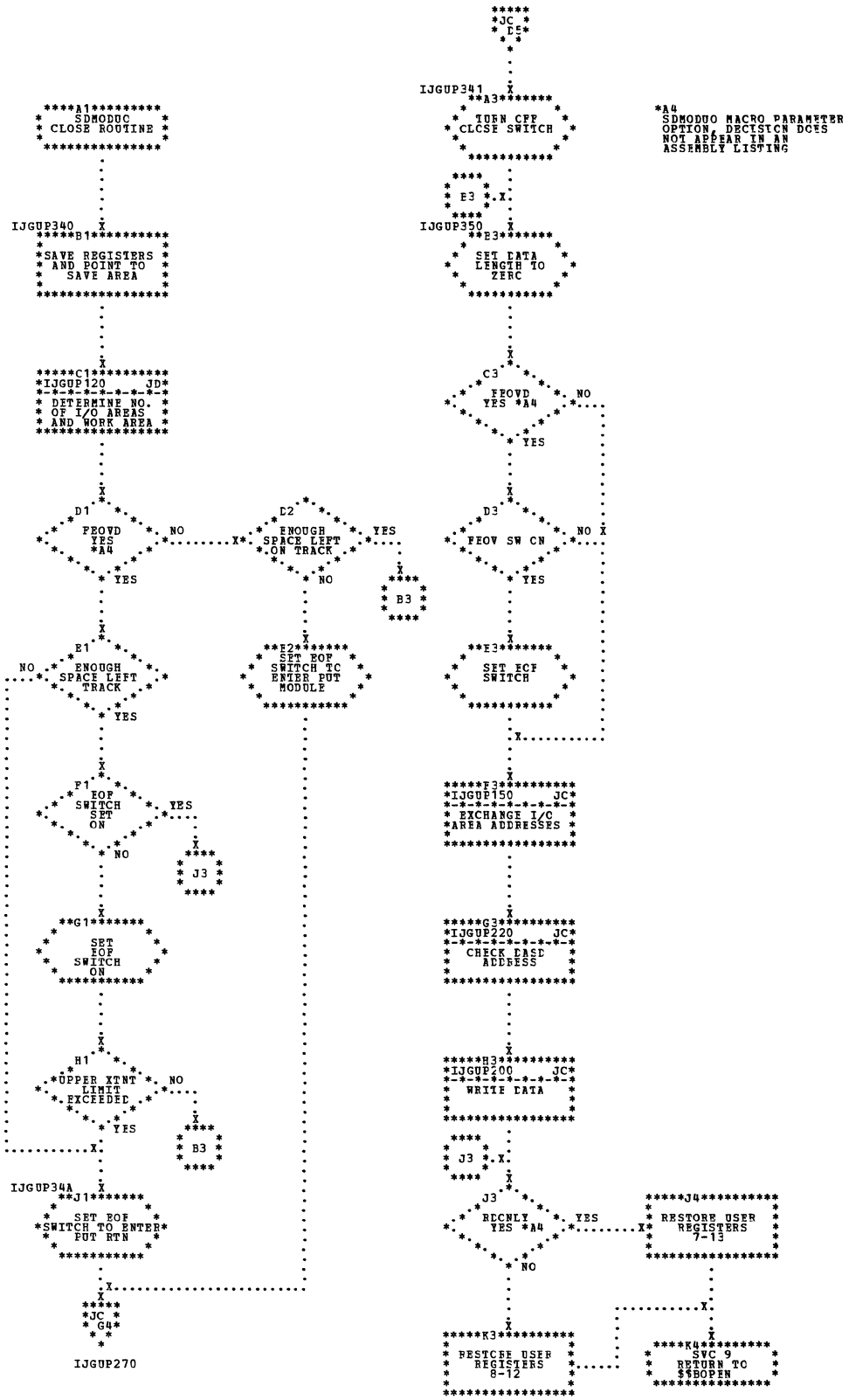
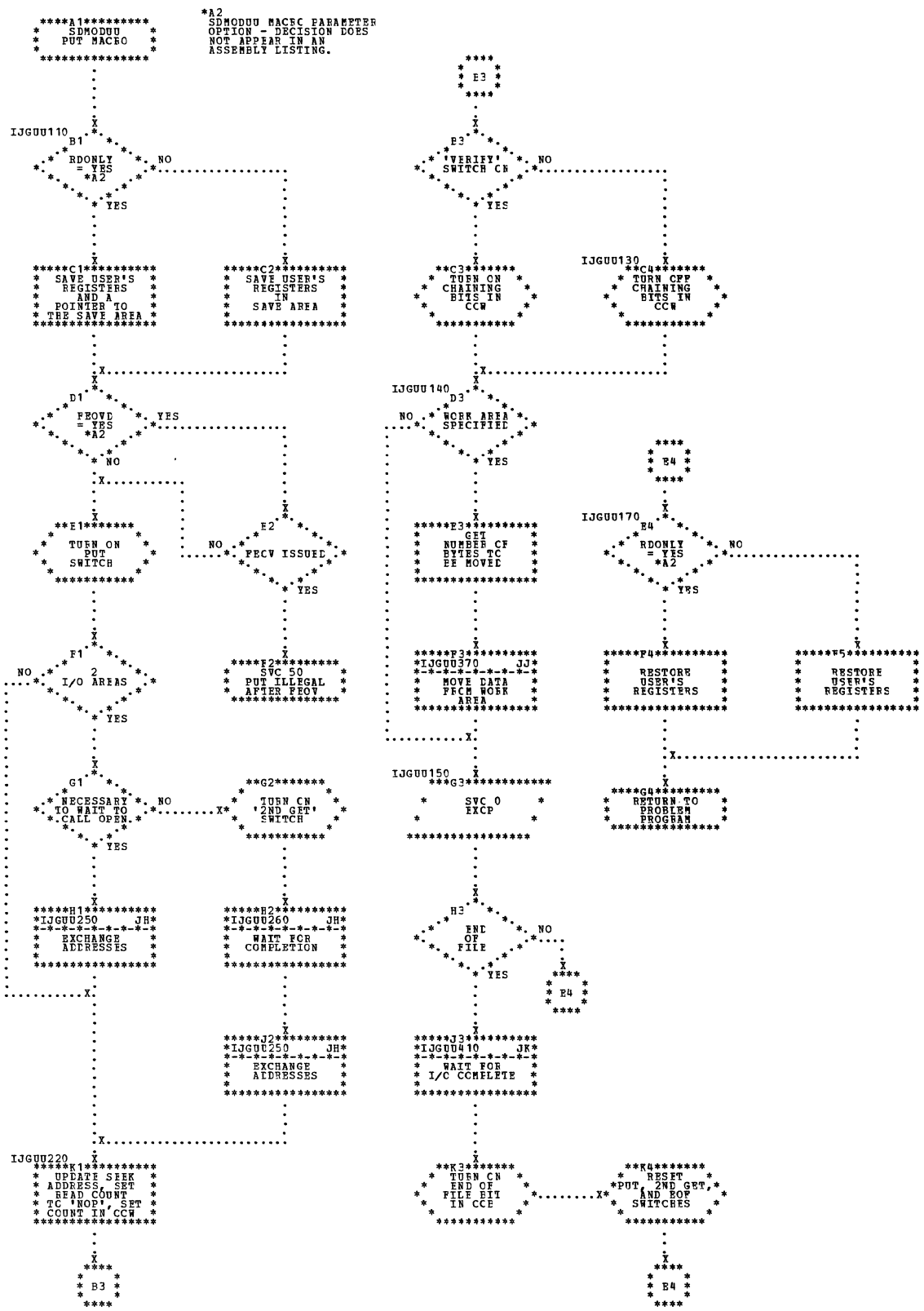
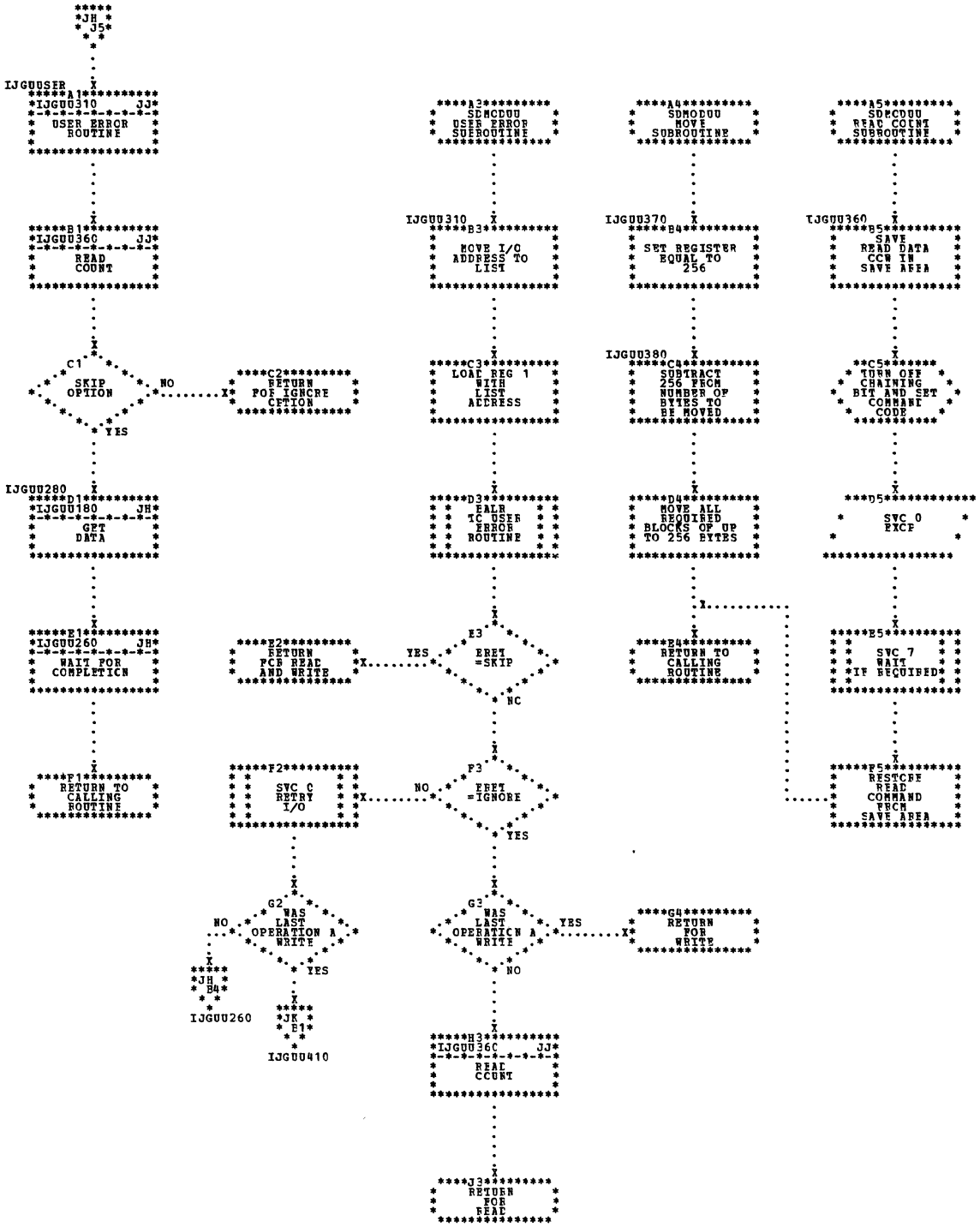


Chart JG. SDMCDUU: PUT Macro



## Chart JJ. SDMODUU: Subroutines (2 of 3)



## Chart KA. SDMCDW: READ Macro

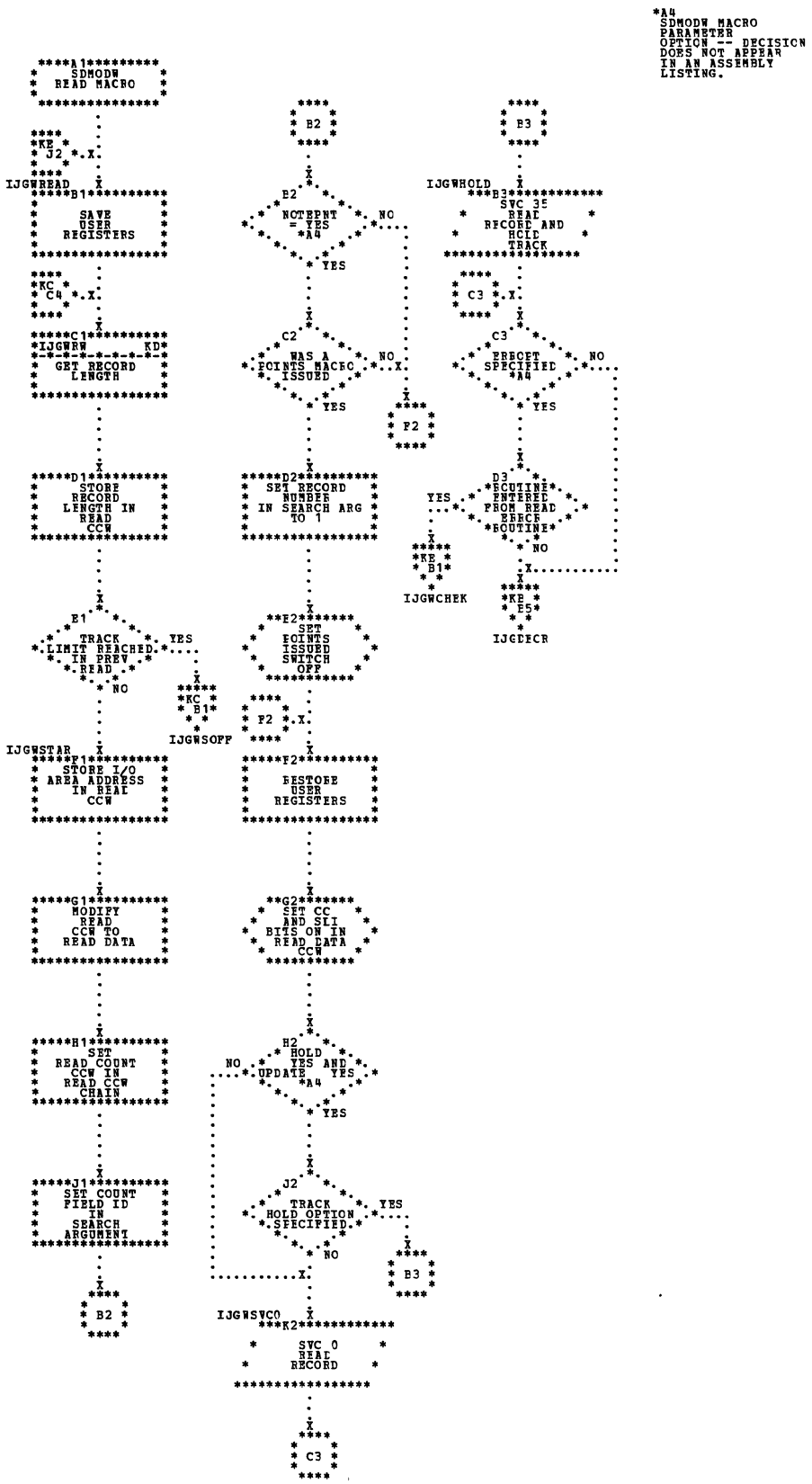
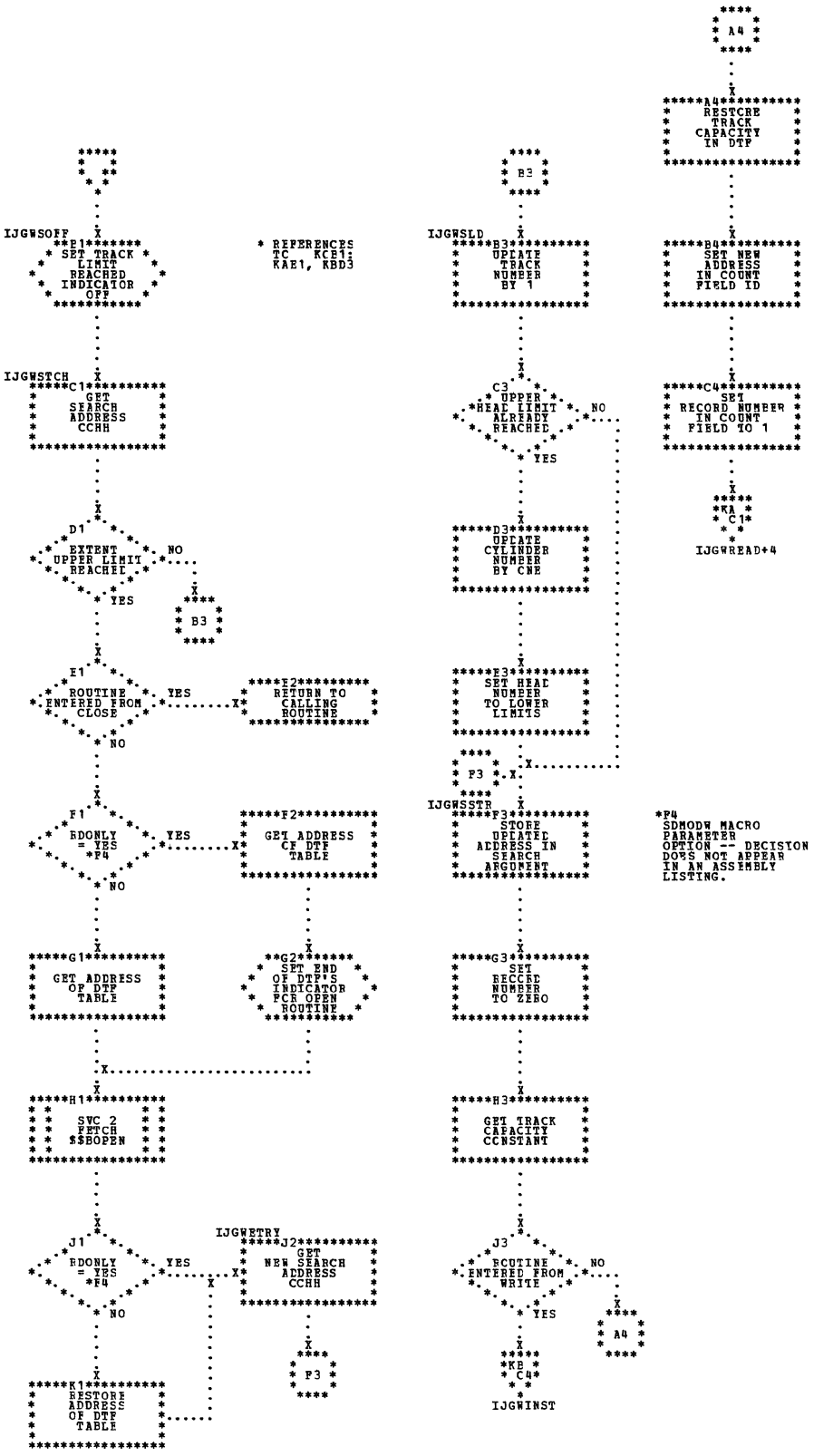


Chart KC. SDMODW: READ, WRITE Macros, Common Routine



## Chart KE. SDMODW: CHECK Macro

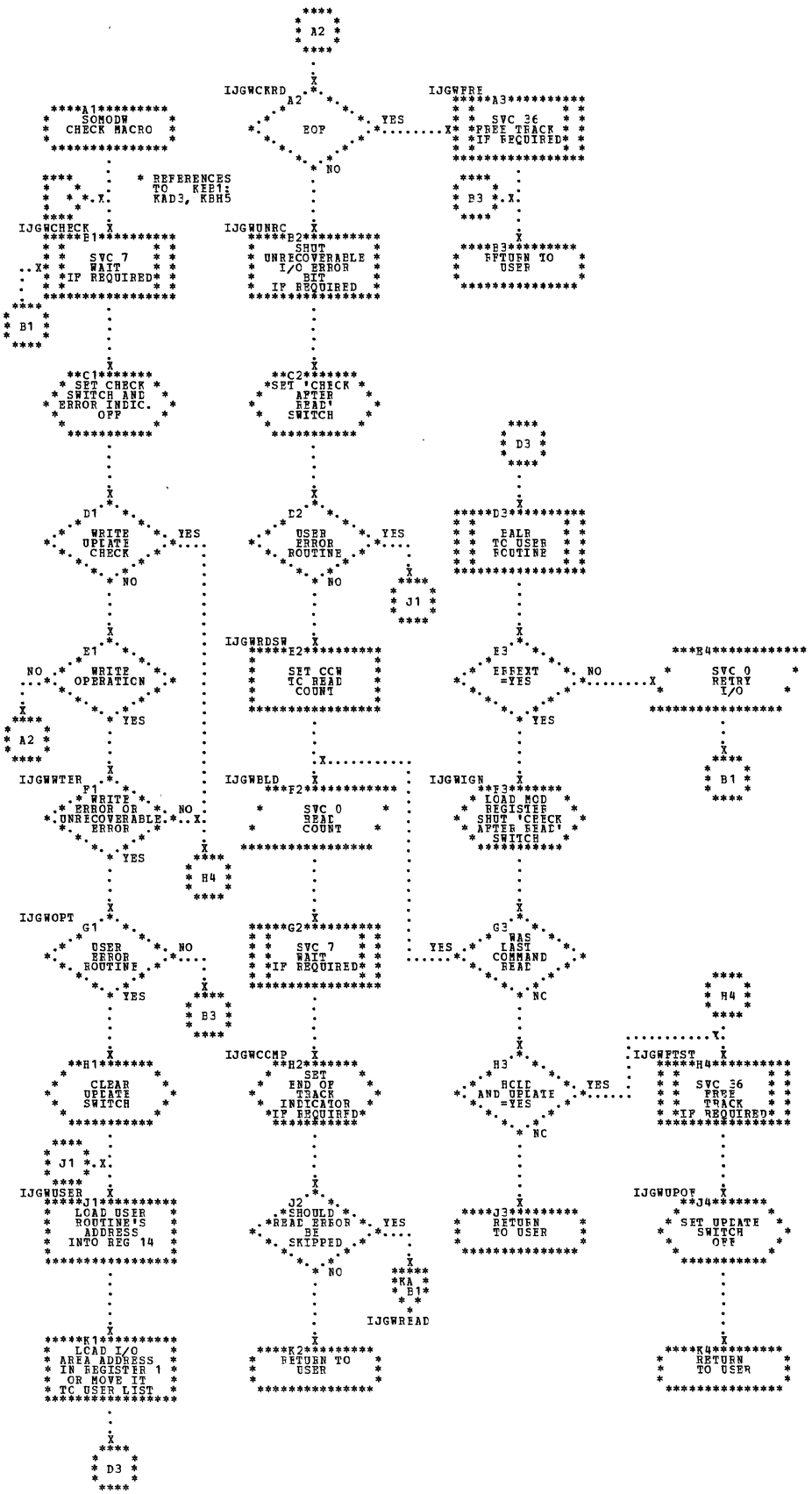
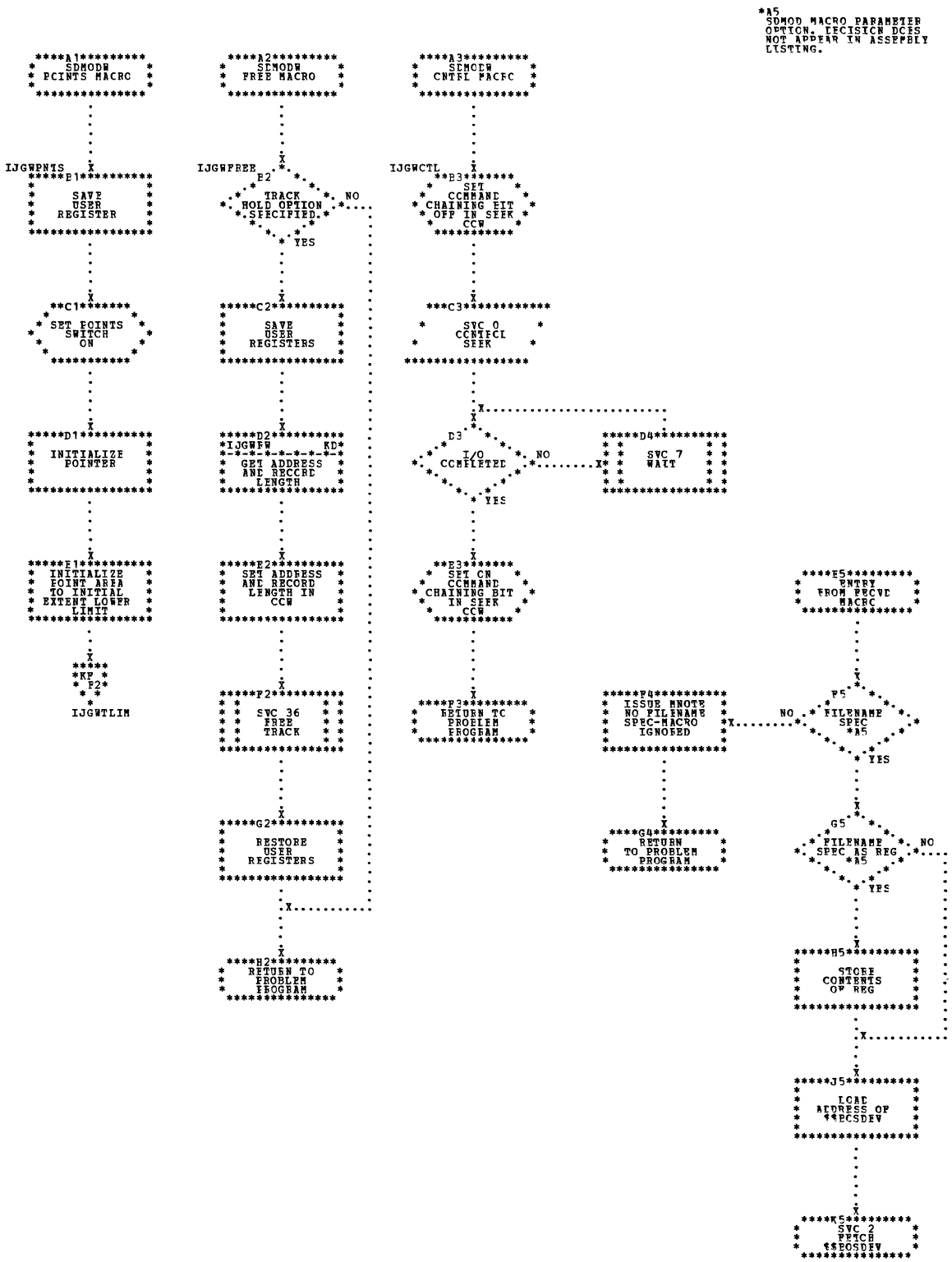


Chart KG. SDMODW: POINTS, FREE, CNTRI Macrcs, and SDMCD: FEOVD Macrc



## Chart LB. $$BCSD01: SD Open, DLBL Extents

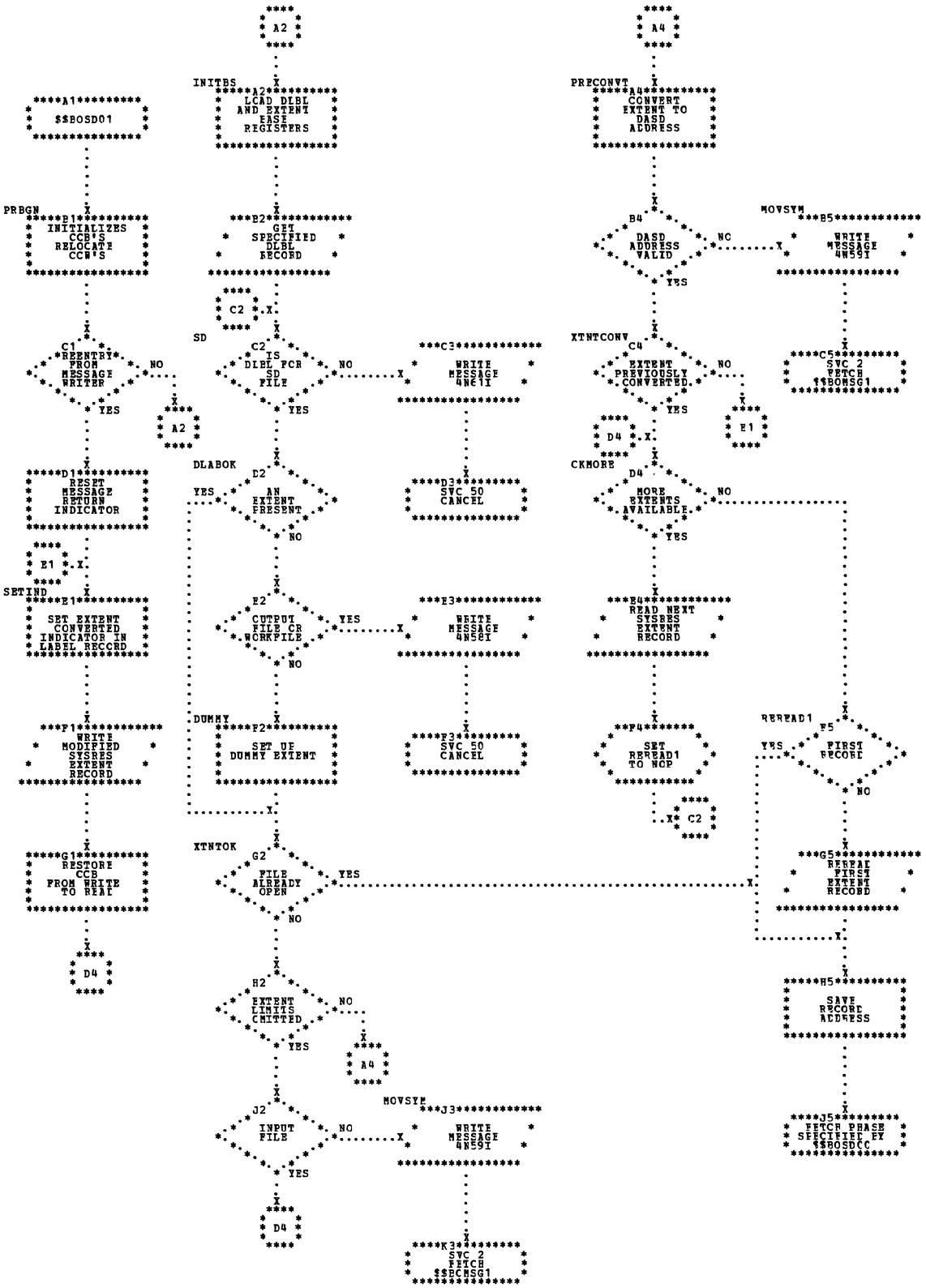


Chart LD. $$BOSD02: SD Open, DLBL Extents for 3340 (2 of 2)

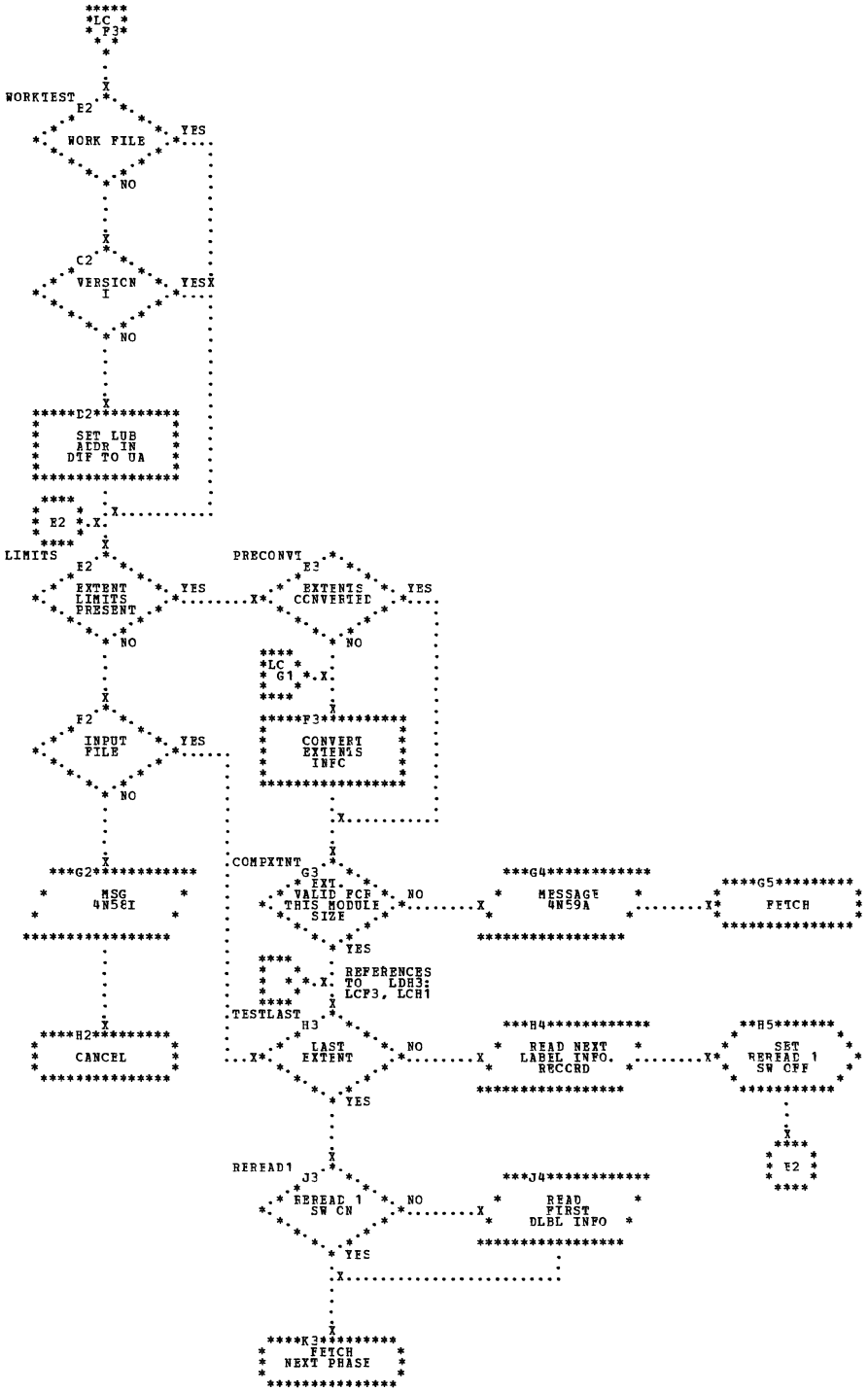


Chart LF. $$BOSDI1: SD Open Input, DLBL Extents (2 of 2)

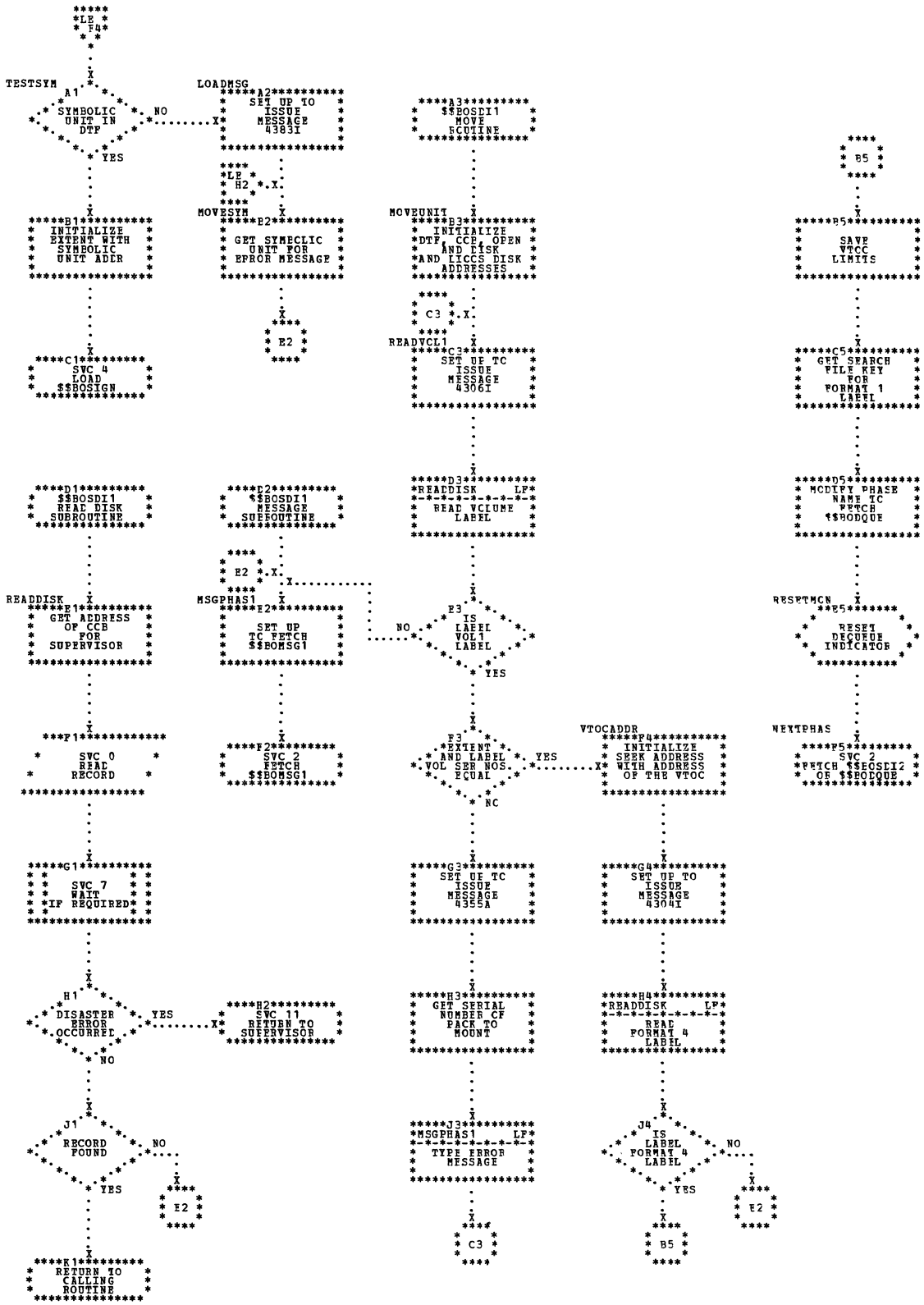


Chart LH. $$BOSDI3: SD Open Input, User Labels

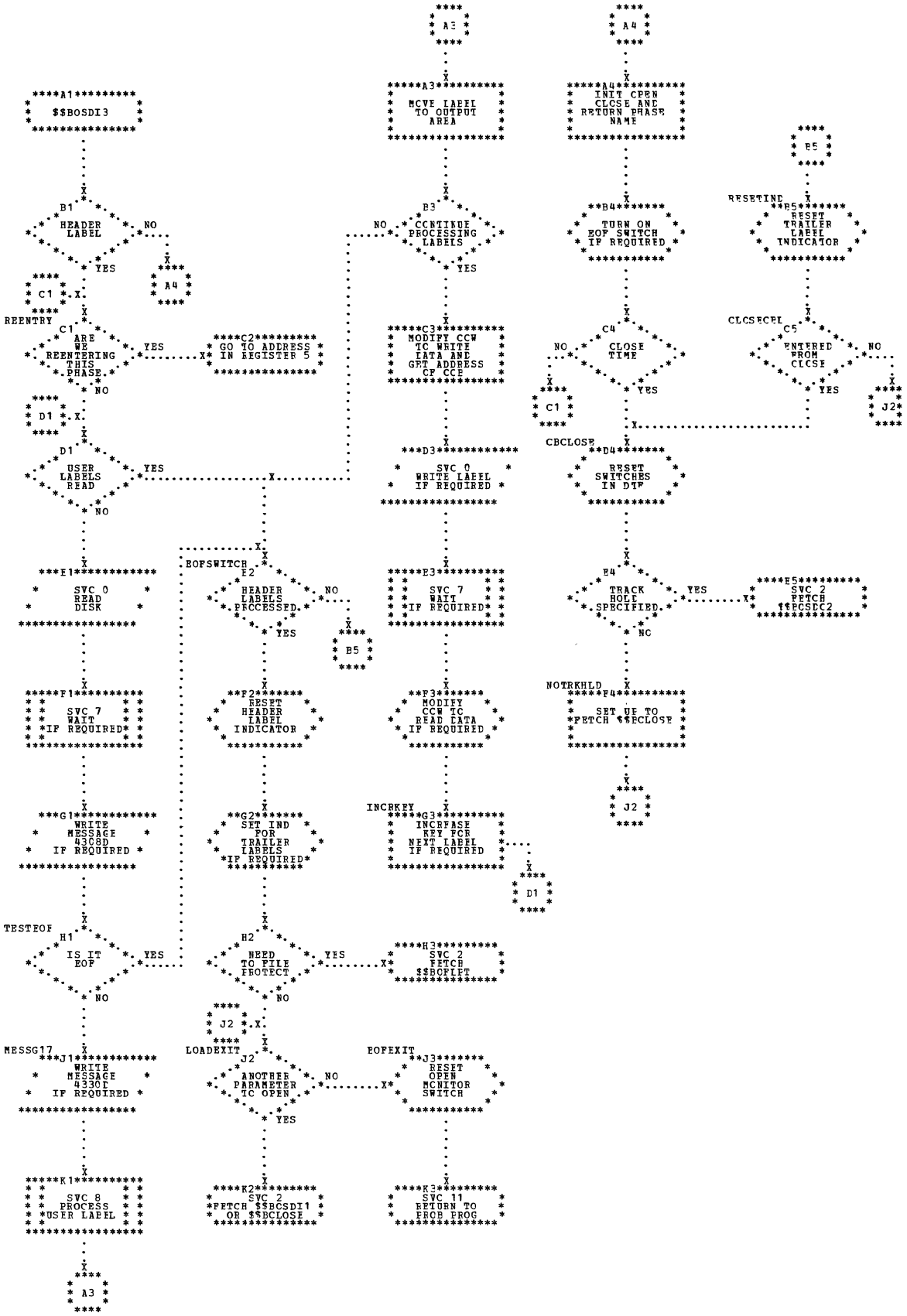


Chart LK. $$BOSDI5: SD Open Input, Post DTF Block

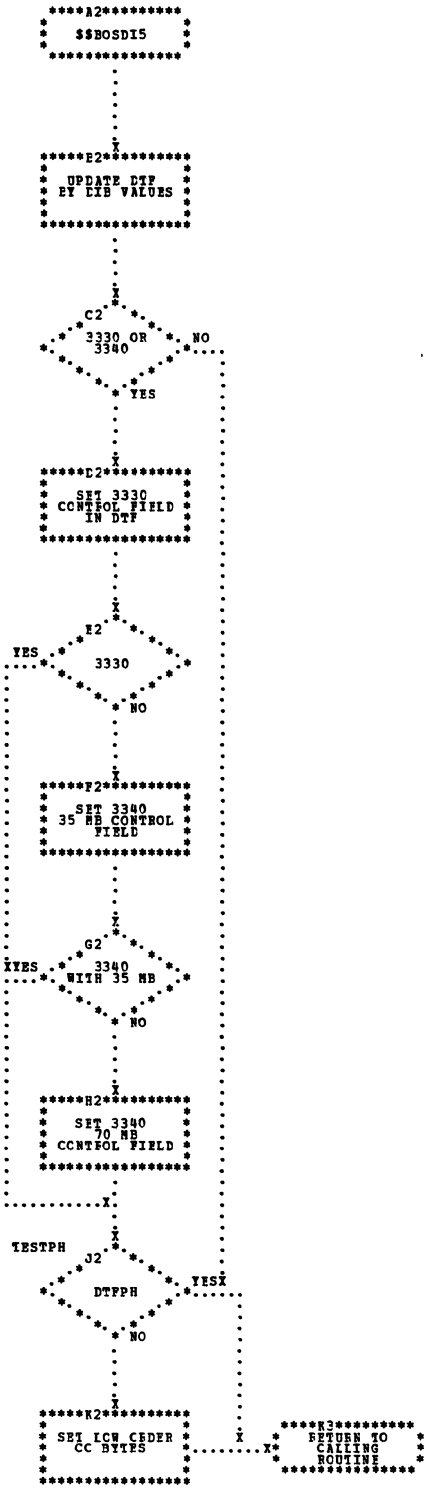


Chart LM. $$BOSDO1: SD Open Output, Contrcl (2 cf 2)

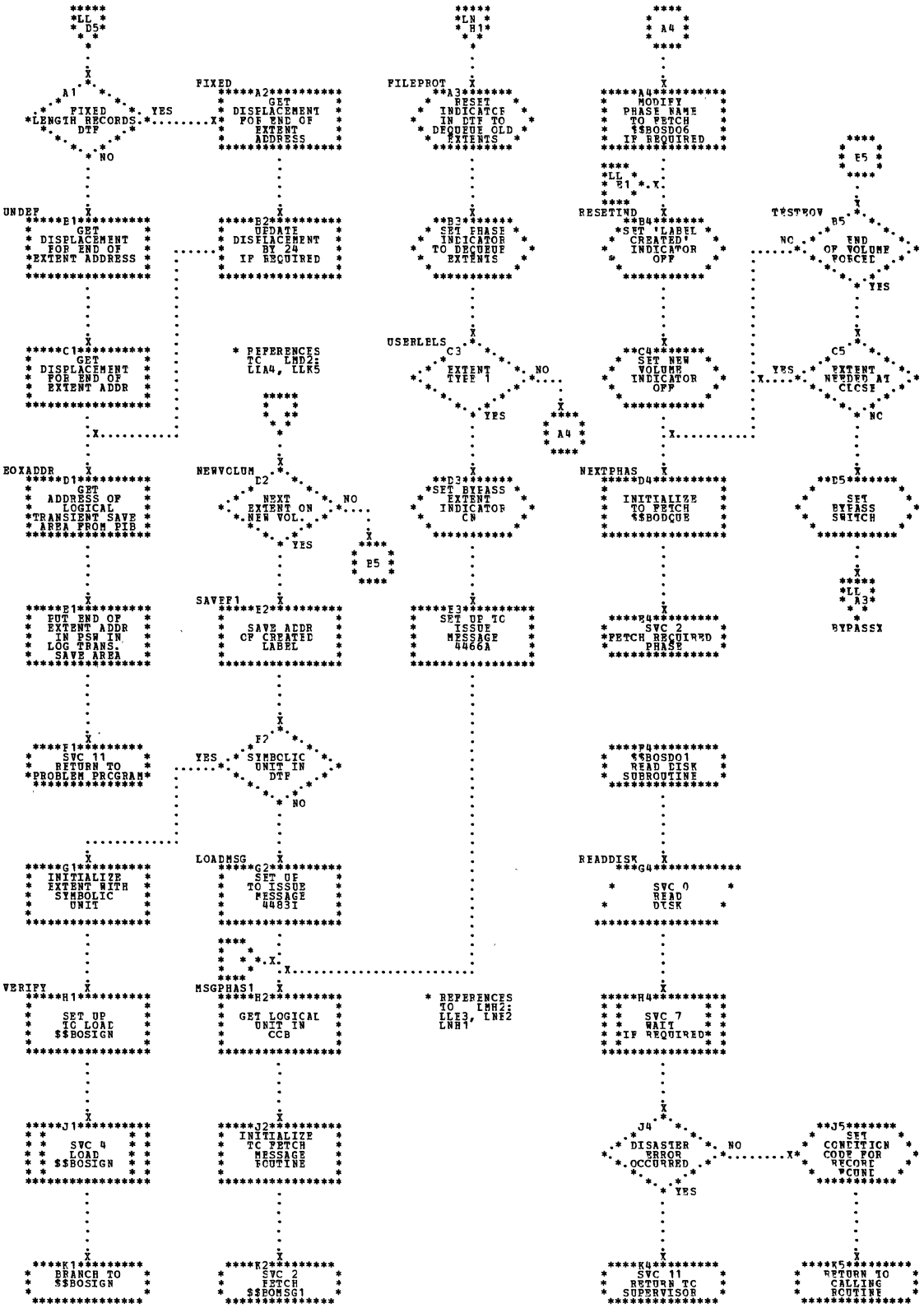


Chart LO. $$BOSDO2: SD Open Output, Volume Label

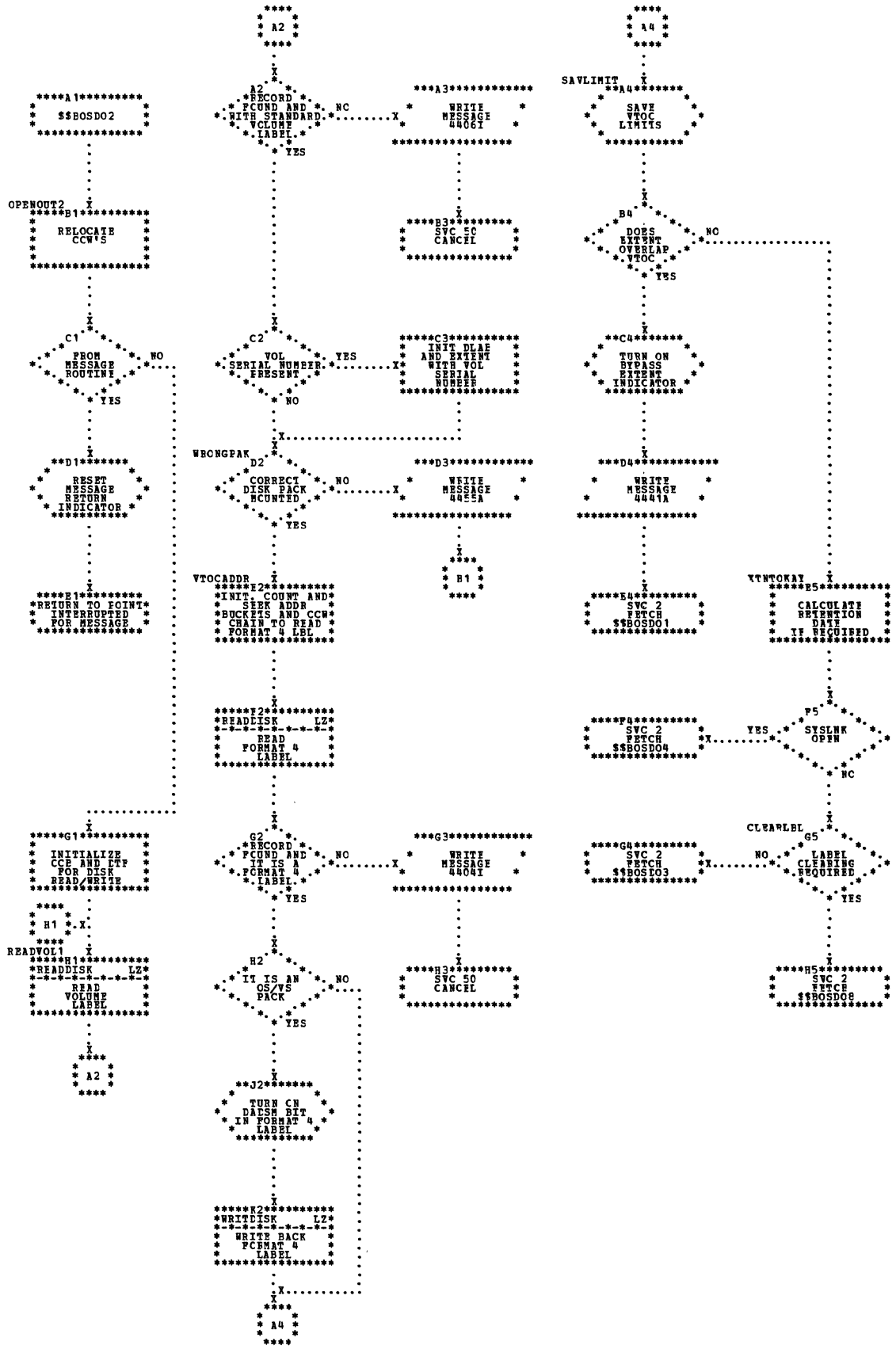


Chart LQ. $$BOSDO3: SD Open Output, Extent Overlap (2 of 2)

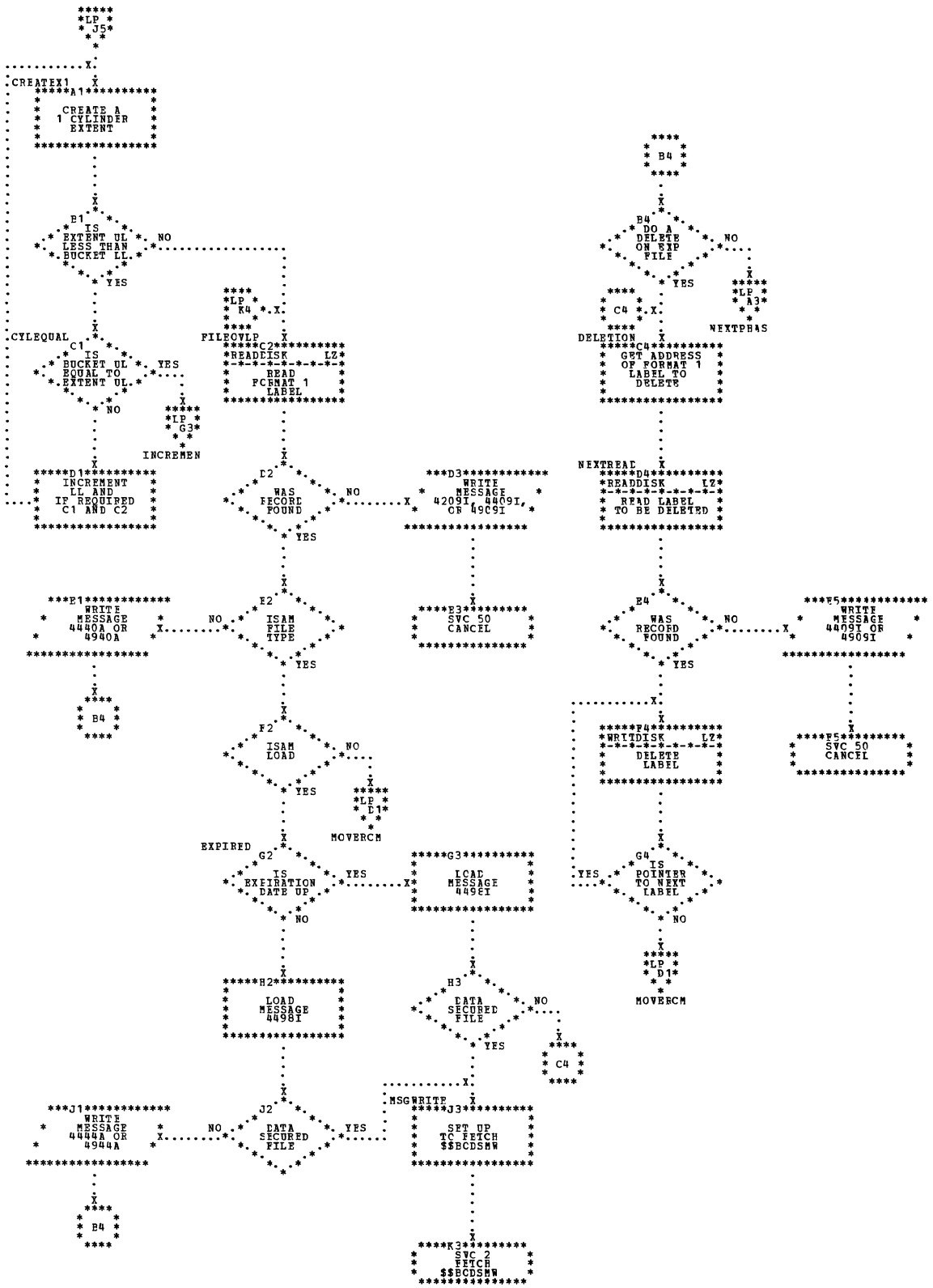


Chart LS. $$BOSDO5: SD Open Output, Format 3 Label (1 of 2)

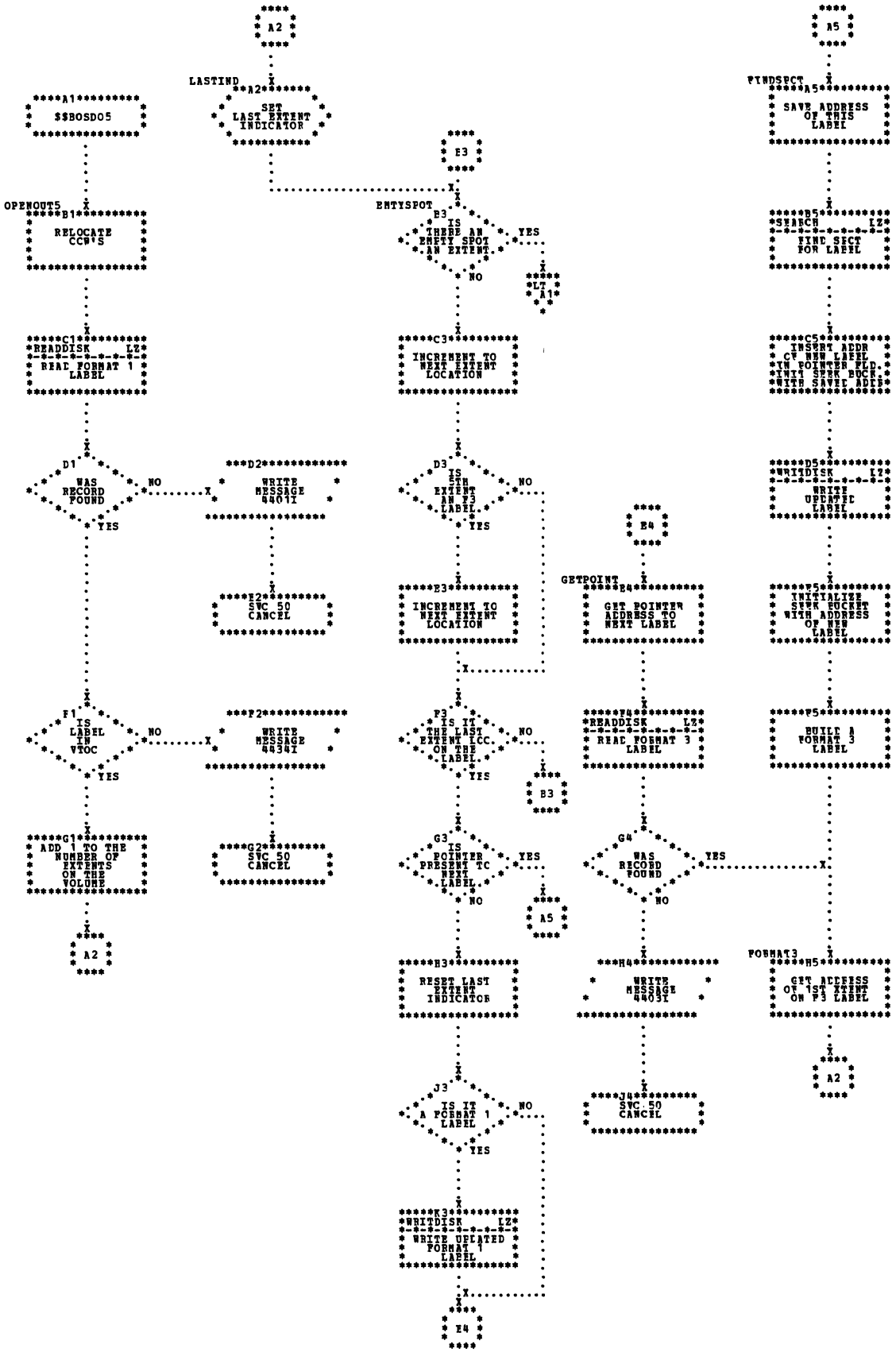


Chart LU. $$BOSDO6: SD Open Output, User Labels (1 of 2)

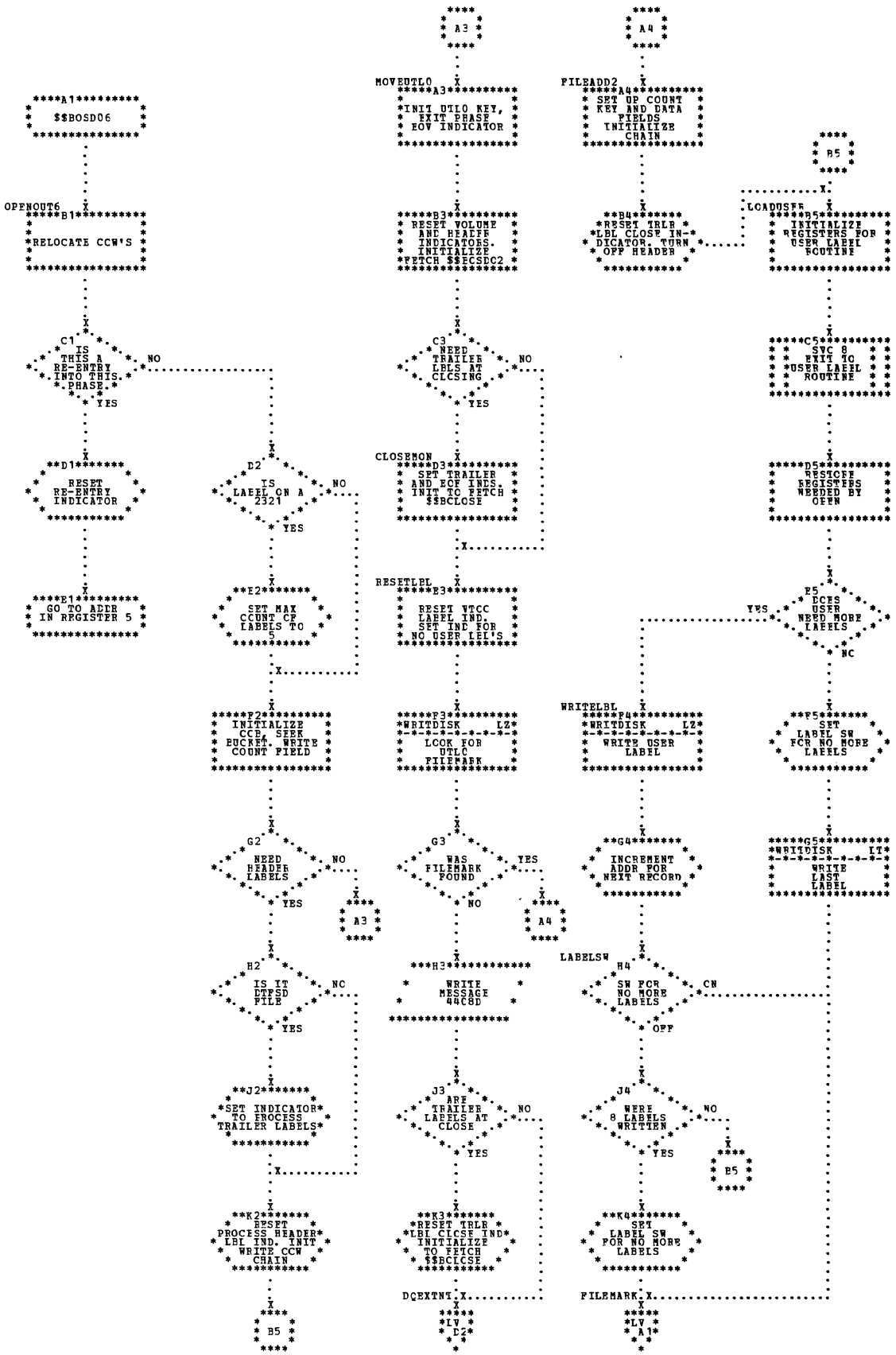


Chart LW. $$BOSDO7: SD Open Output, Extents from Console

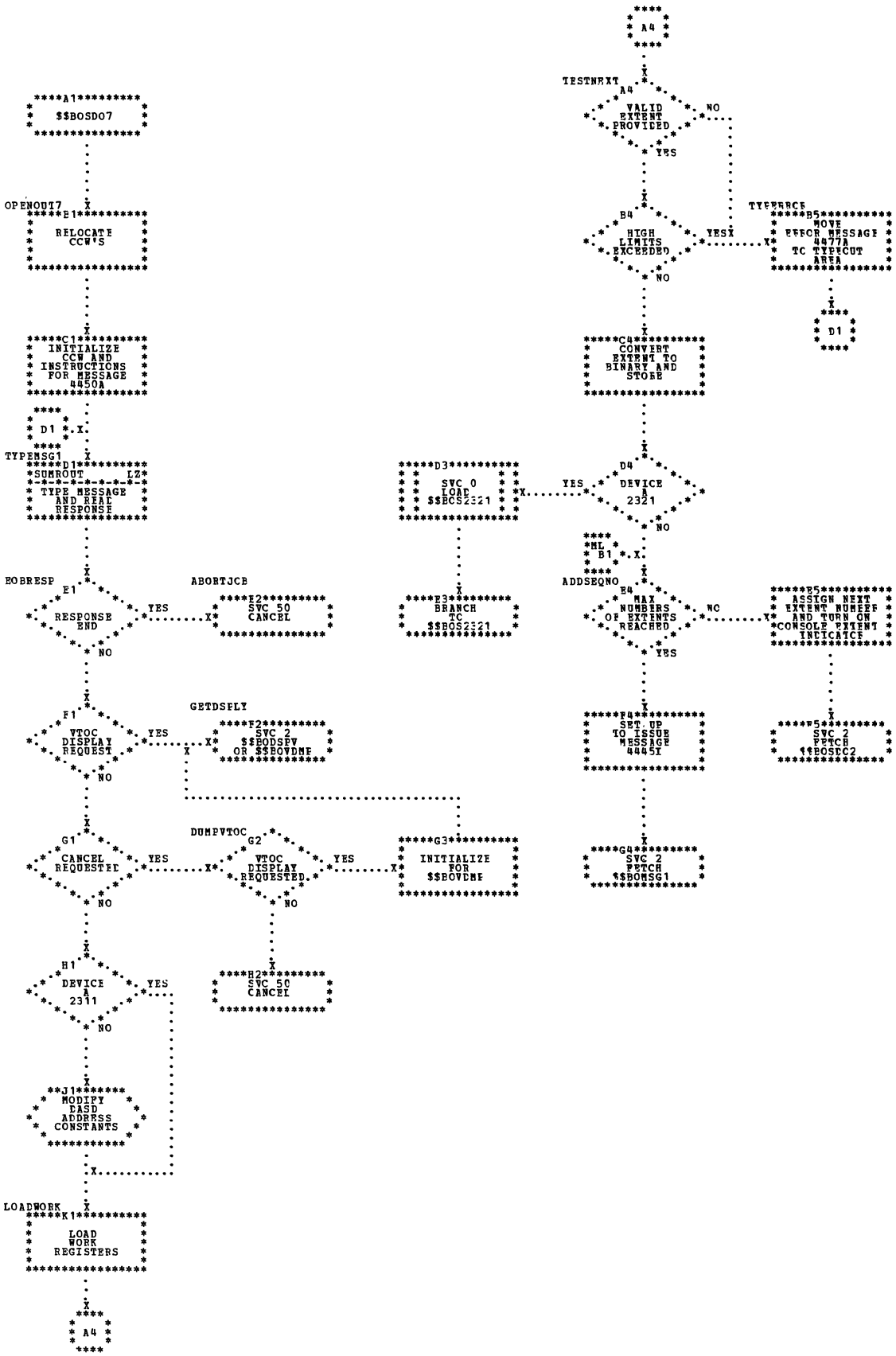


Chart LY. $$BOSD09: SD Open Output, Extent to DTF

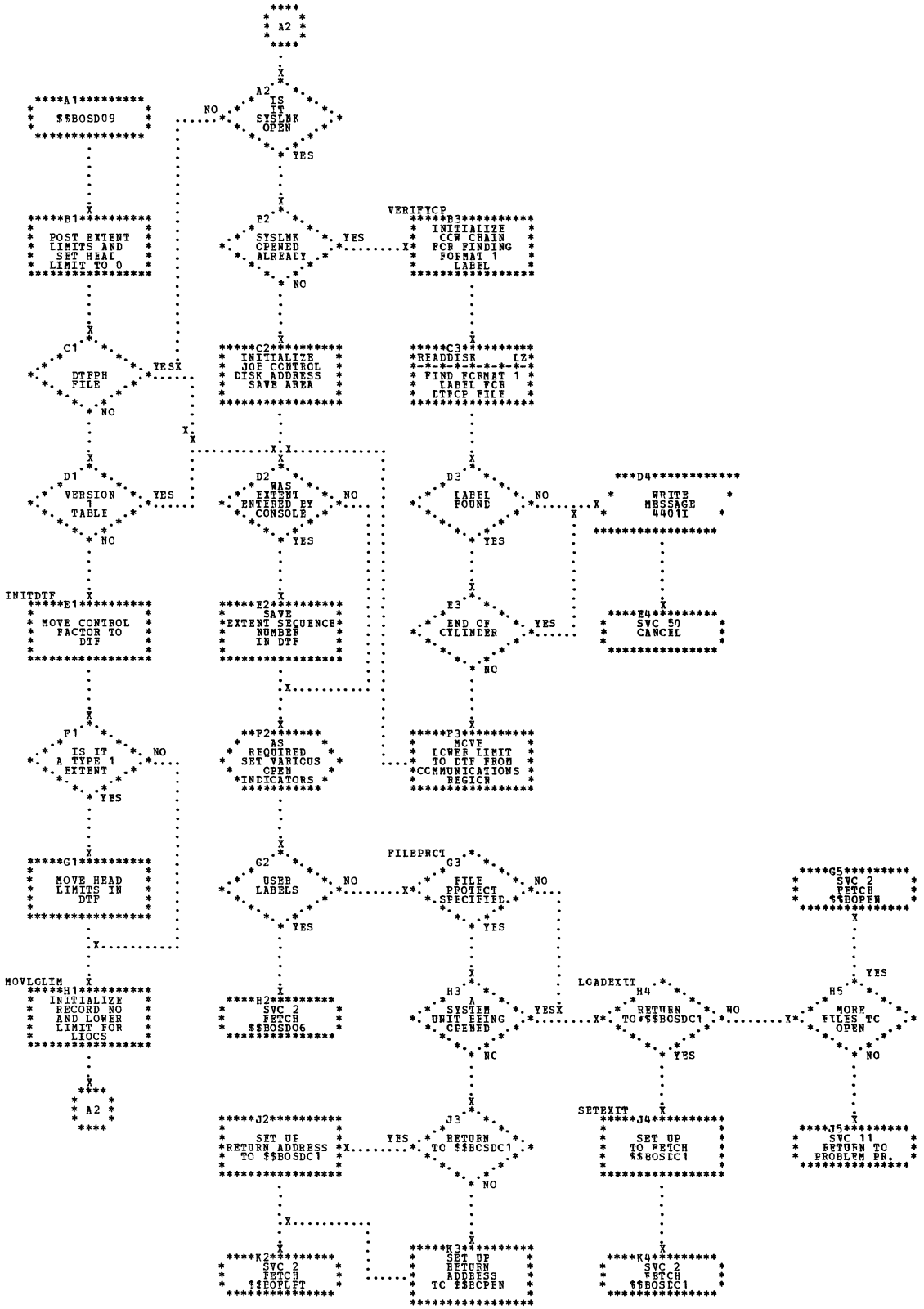


Chart MA. $$BOSDW1: SD Open Work File, Volume Label (1 of 2)

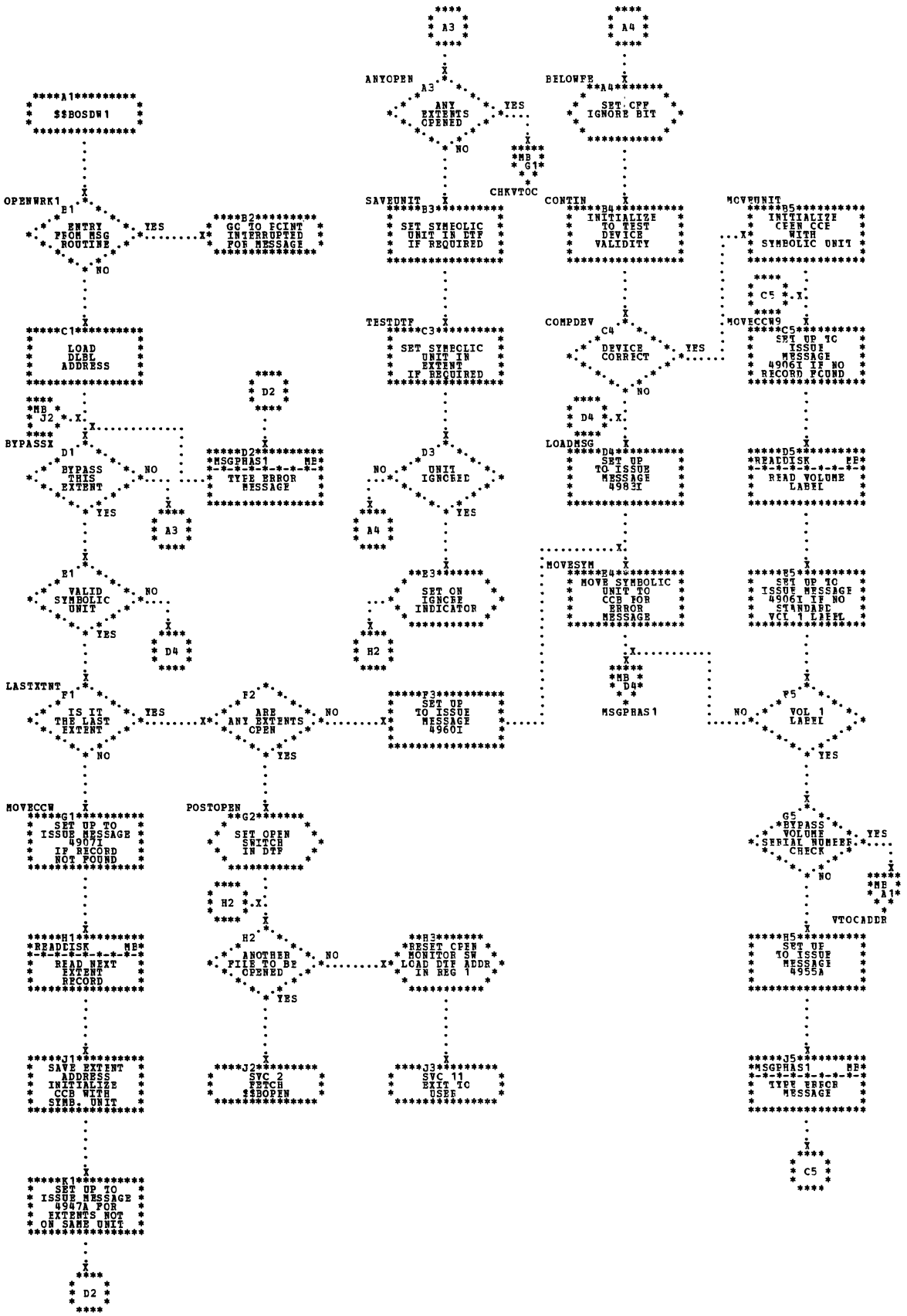


Chart MC. $$BOSDW2: SD Open Work File, File Label (1 of 2)

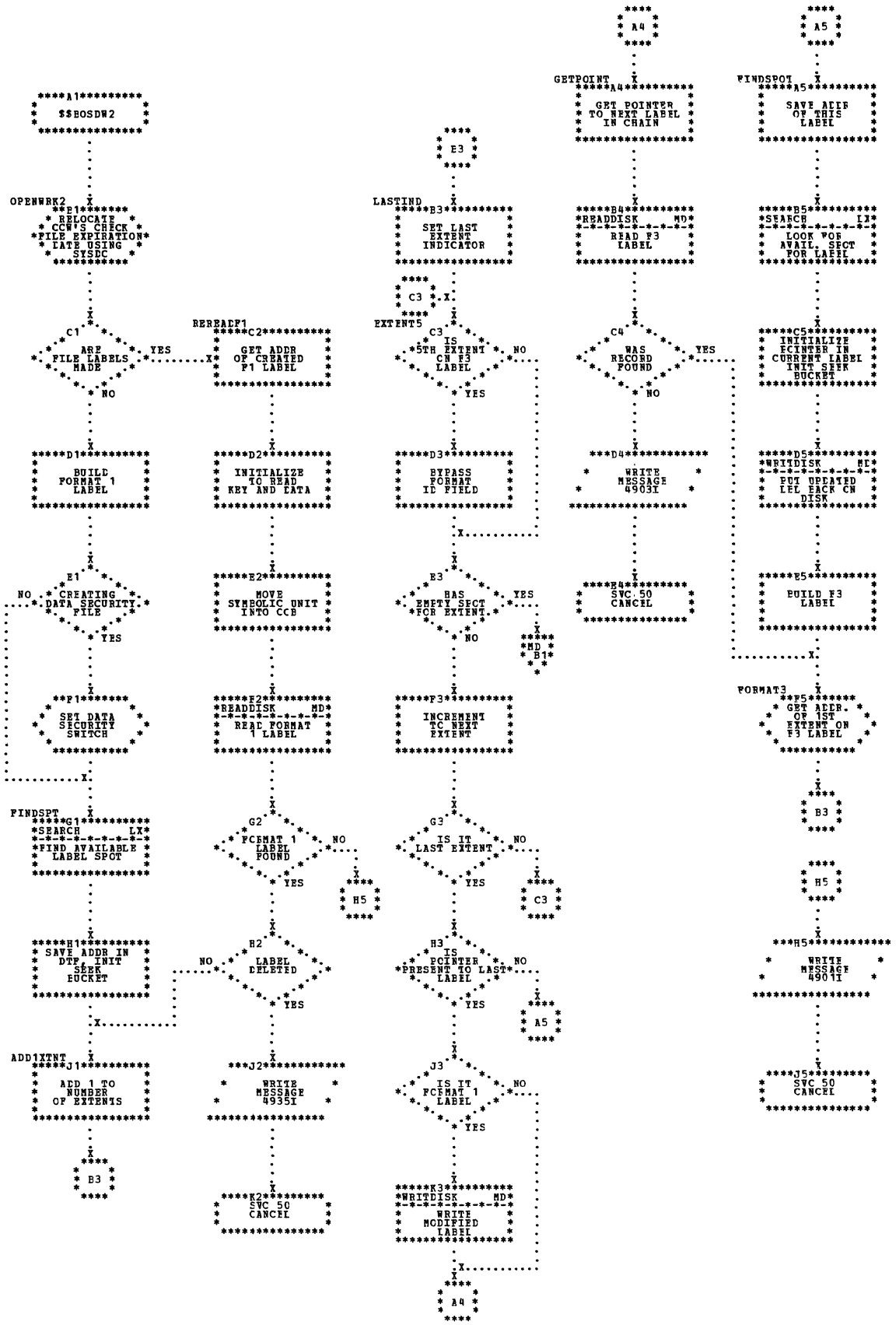


Chart ME. $$BOSDW3: SD Open Wcrk File, Extent tc DTF

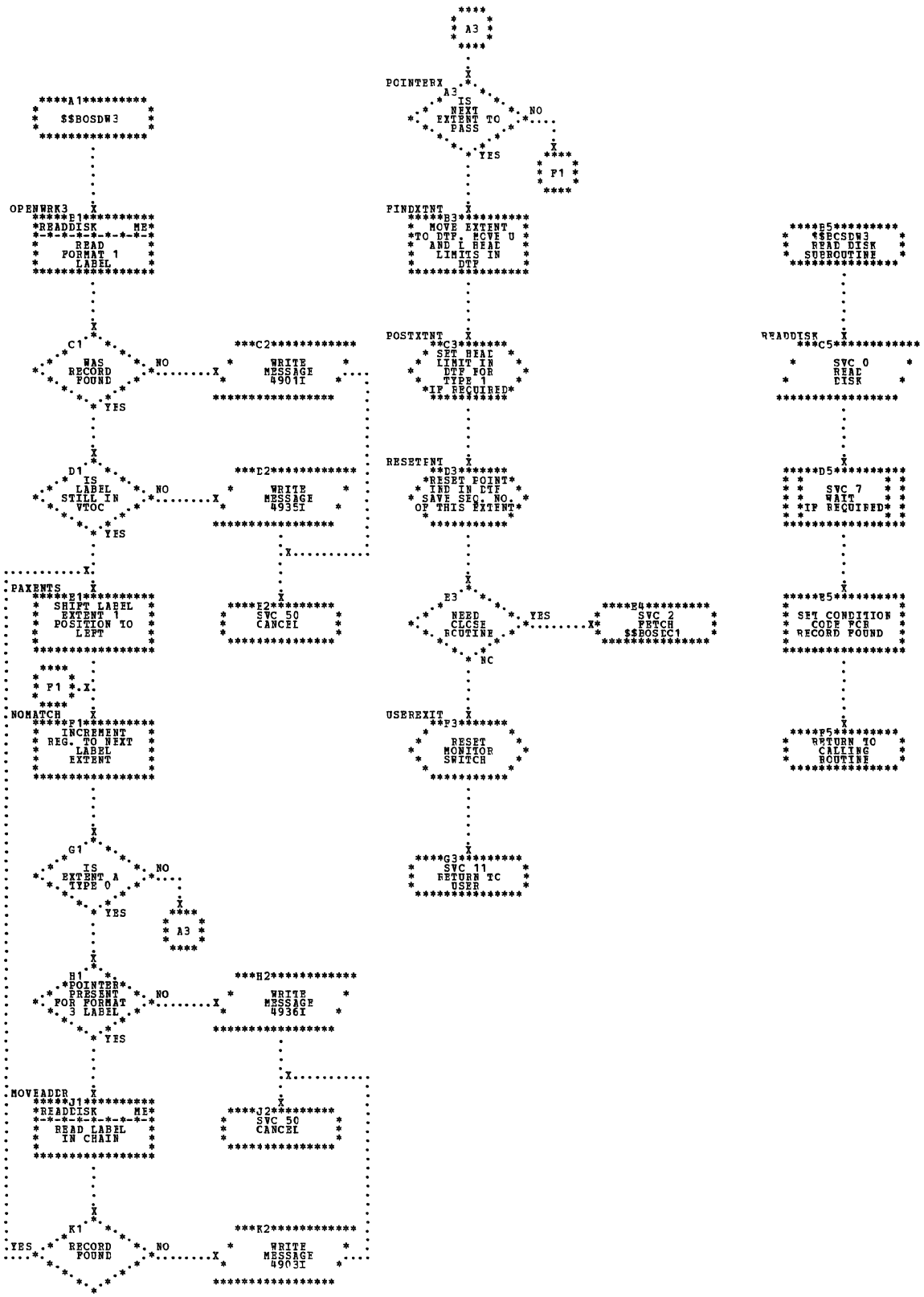


Chart MG. $$BCSDC1: SD Clcse Input and Output (2 of 2)

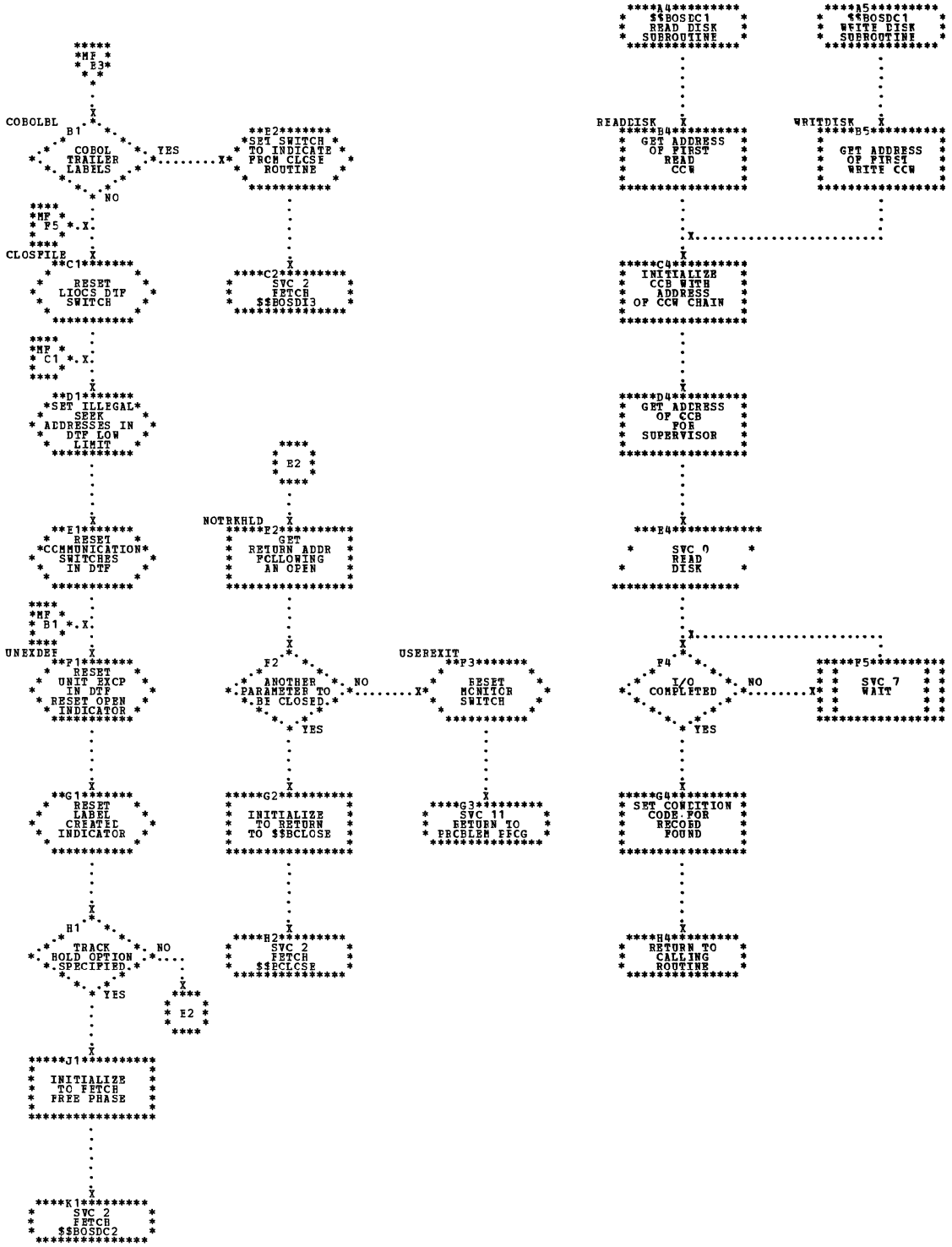


Chart MJ. $$BODQUE: Dequeue Extent JIBs

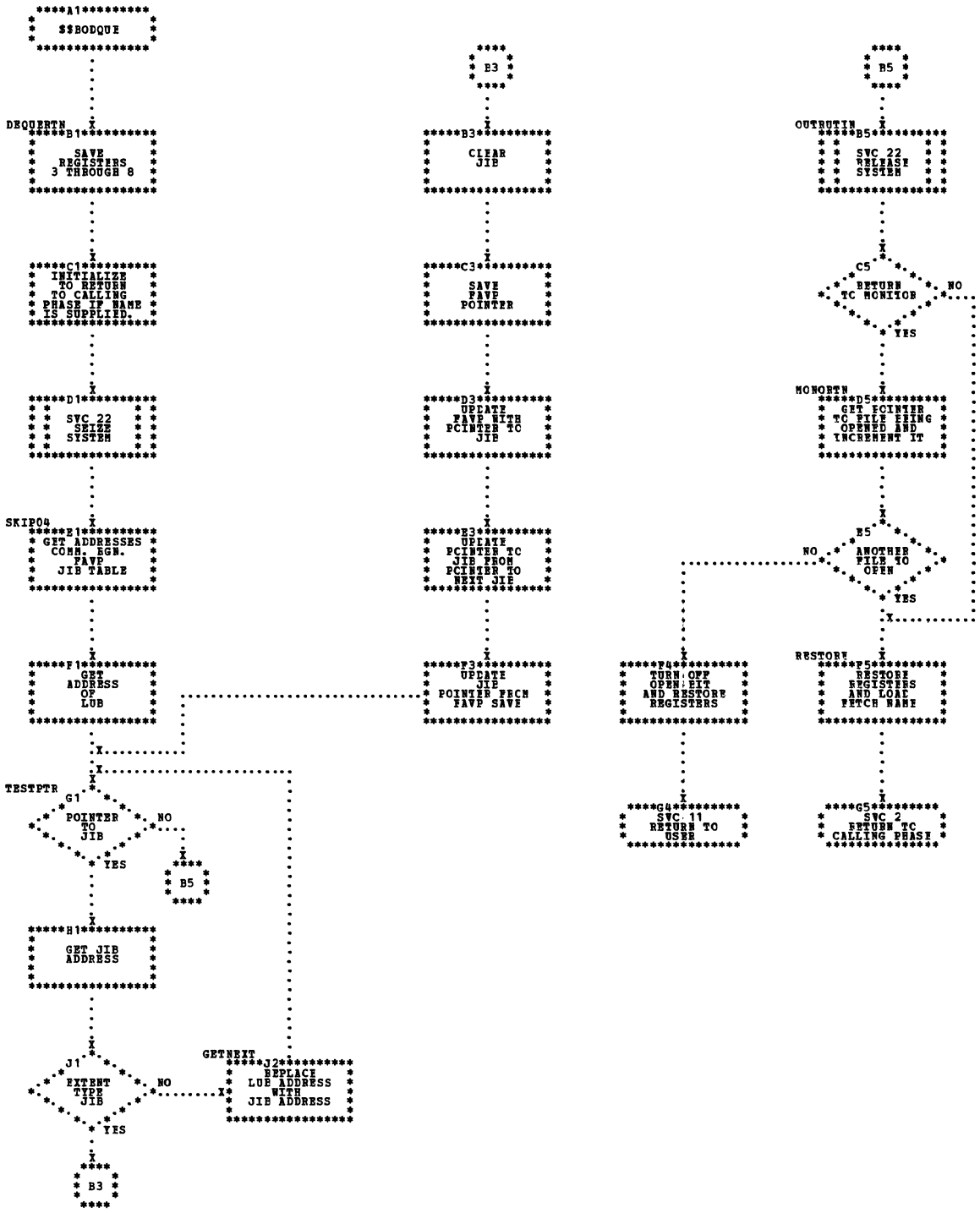


Chart ML. $$BC2321: SD Open Output, 2321 Extents from Console

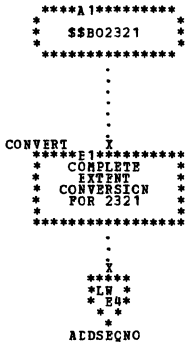


Chart NB.  CPMOD:  GET Macrc, Cne I/O Area cr ICPTR=YES

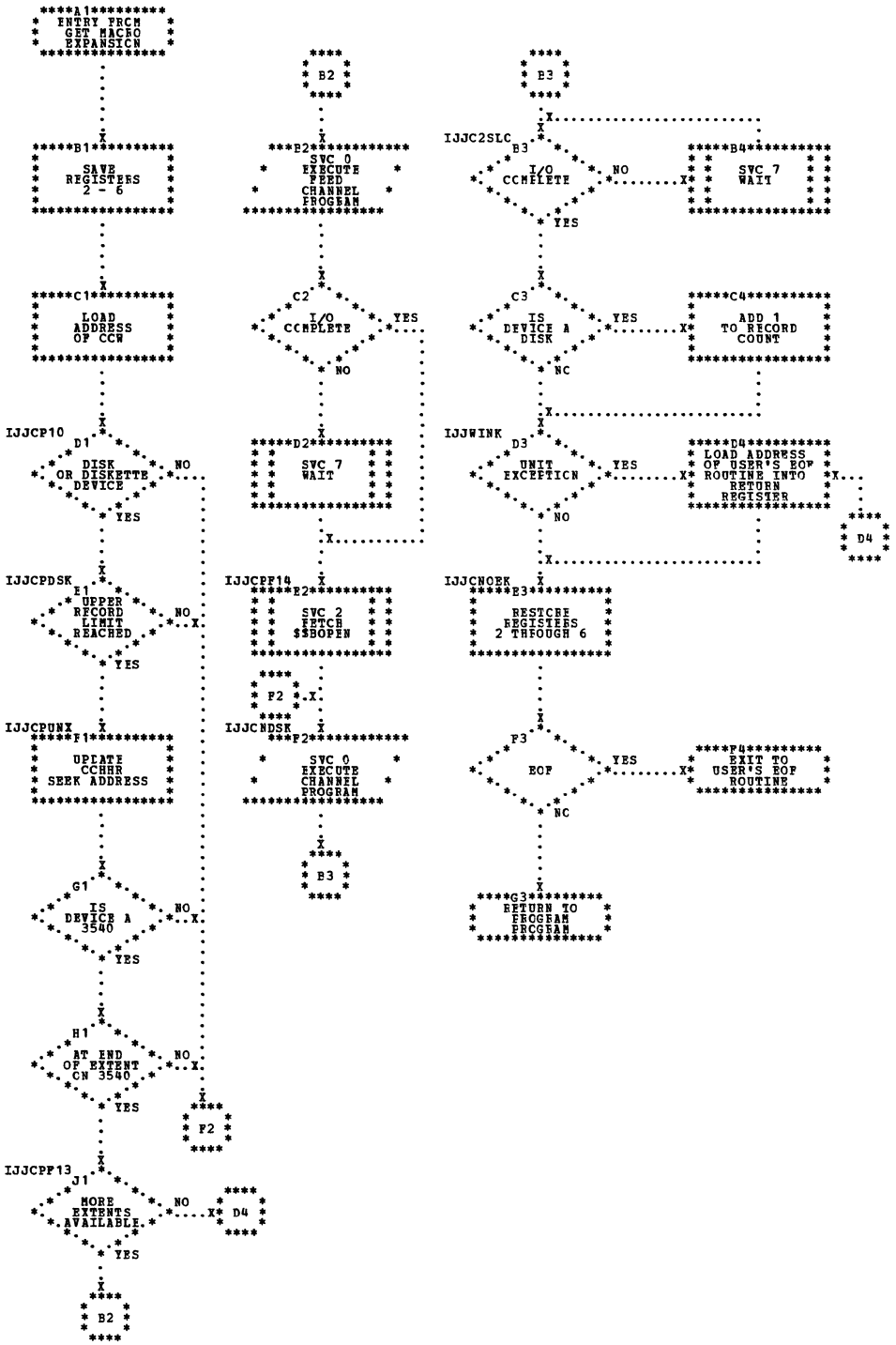


Chart ND. CPMCD: PUT Macrc, Twc I/O Areas (2 of 2)

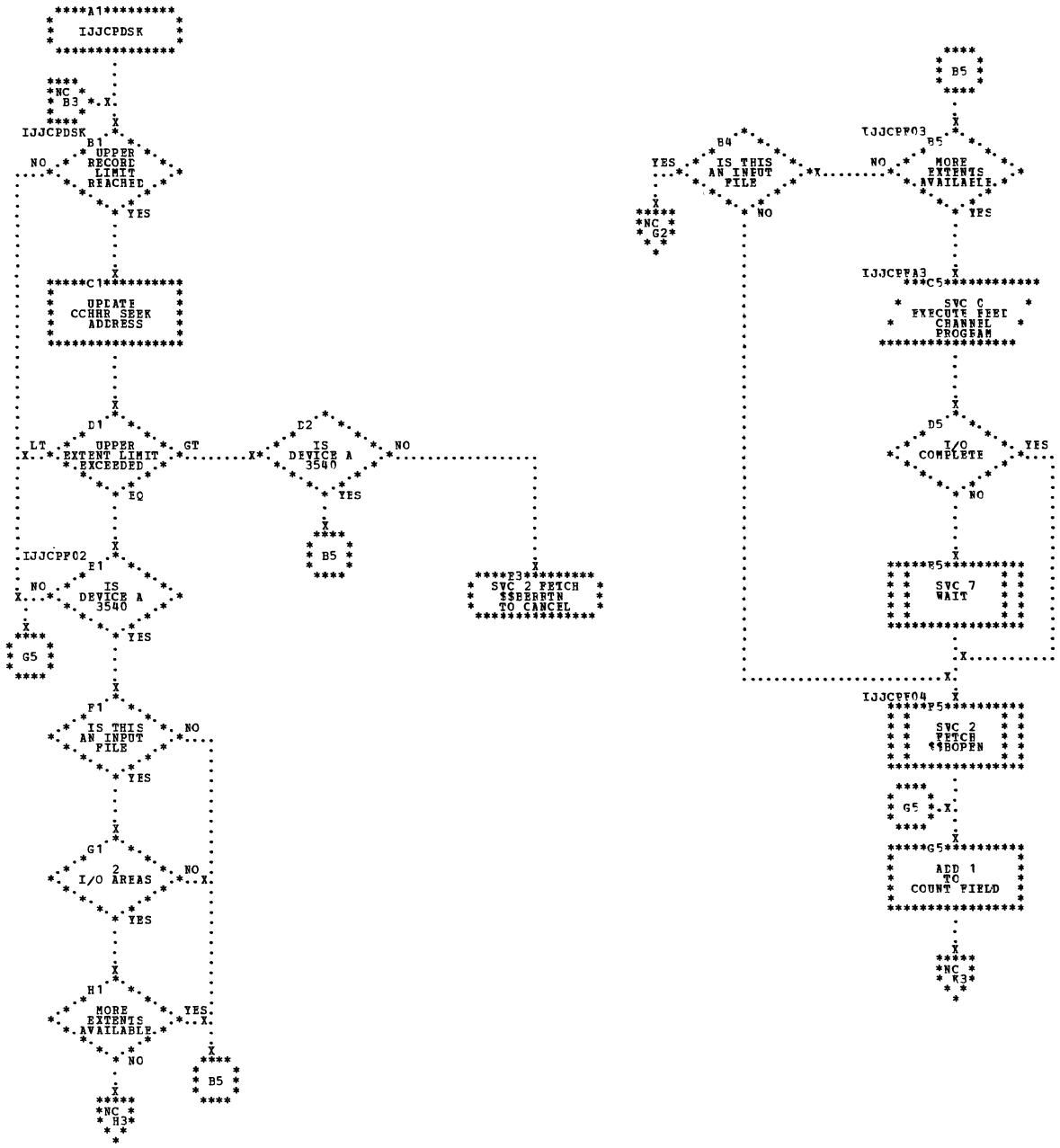


Chart NF. CPMOD: PUT Macrc, Cne I/O Area (2 cf 2)

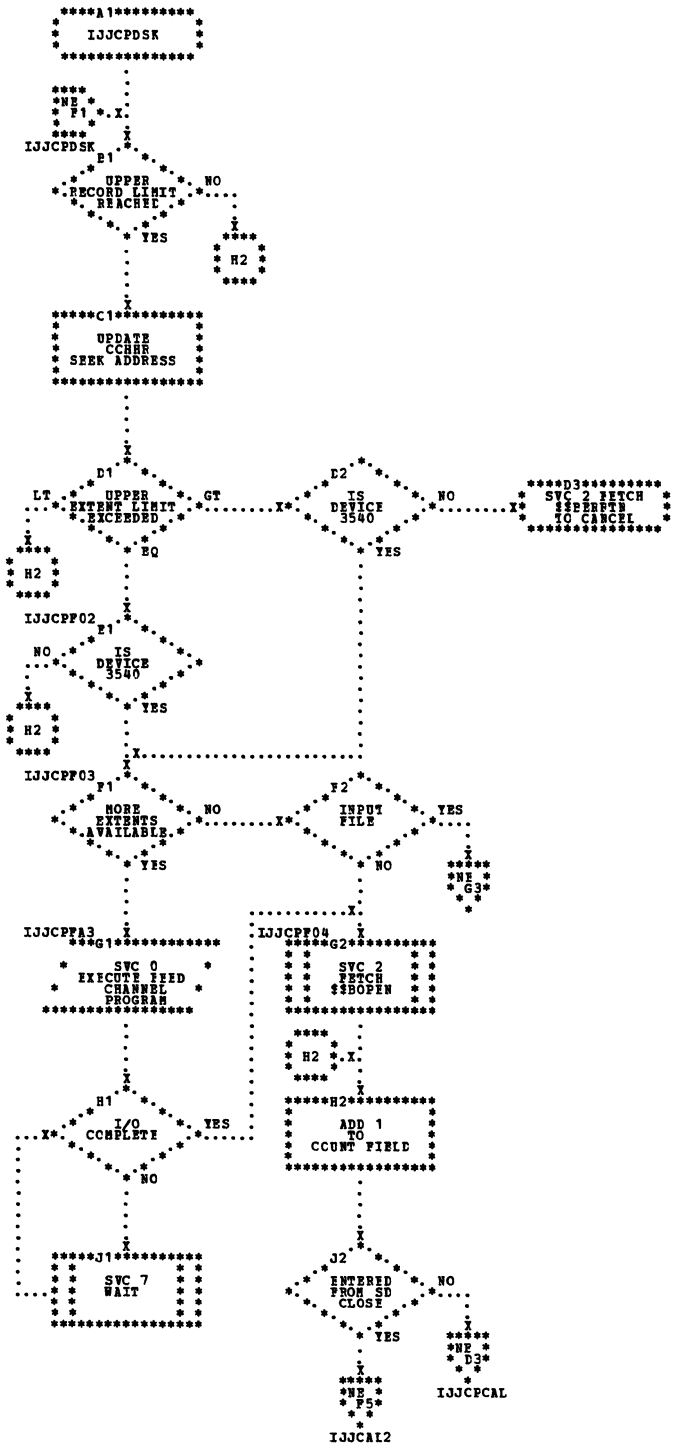


Chart NH. CPMOD: PUT Macrc, IOPTR=YES (2 of 2)

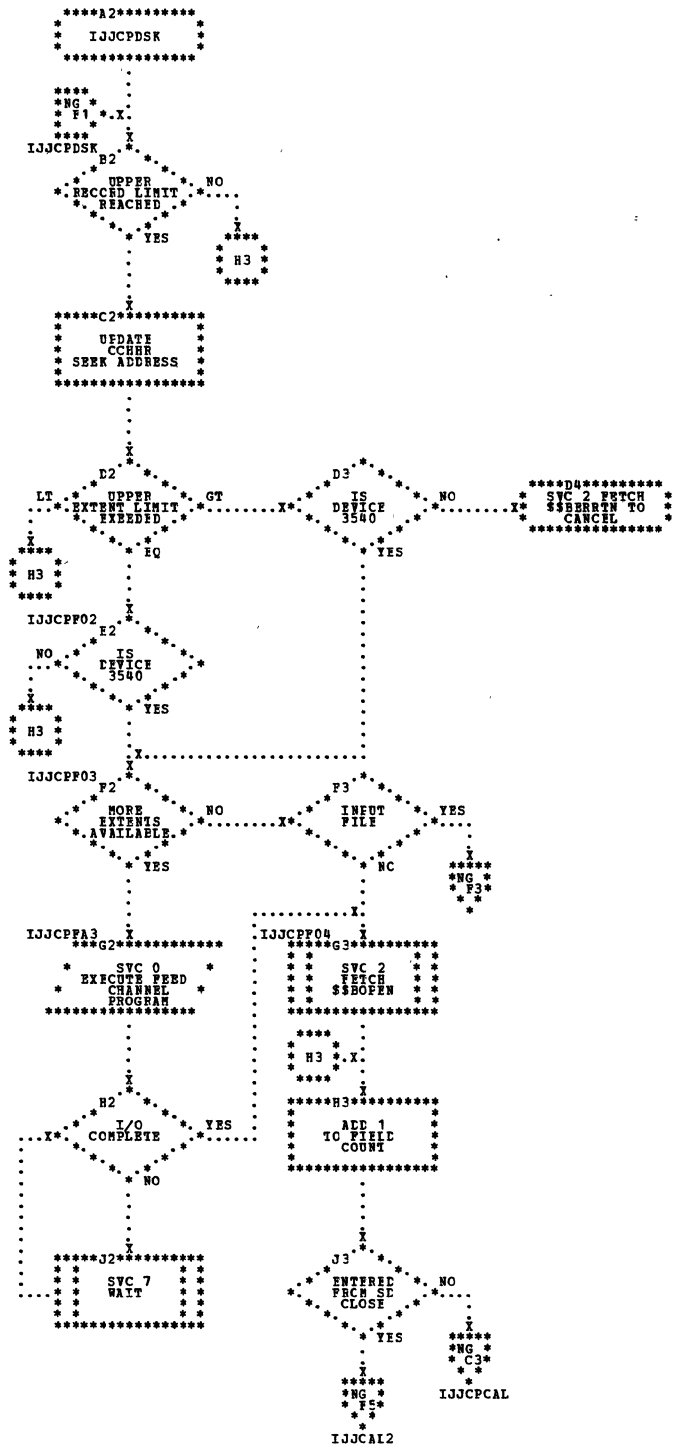


Chart PB. DIMOD: GET Macrc, Twc I/O Areas

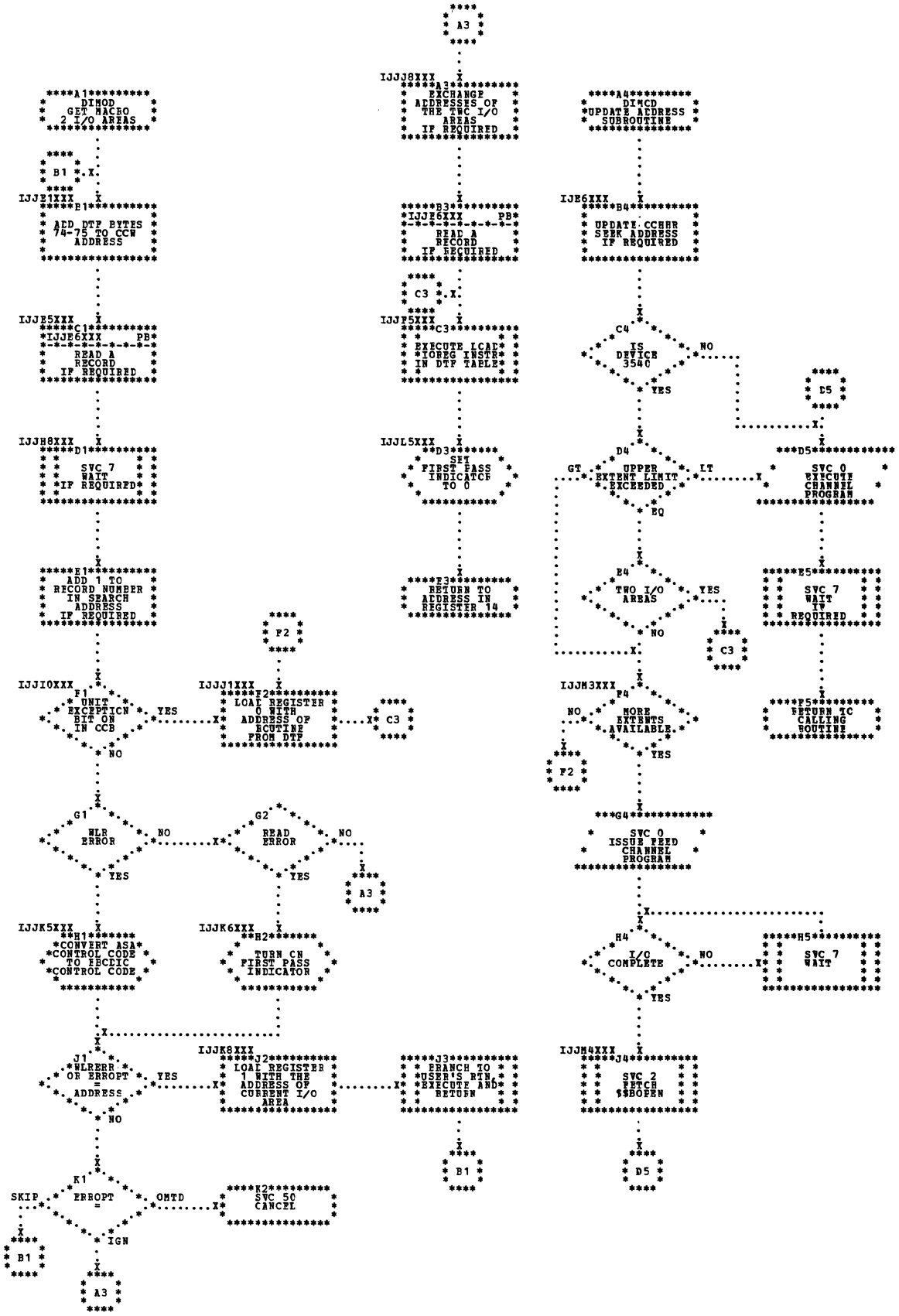


Chart PD. DIMCD: PUT Macrc, One I/O Area (2 cf 2)

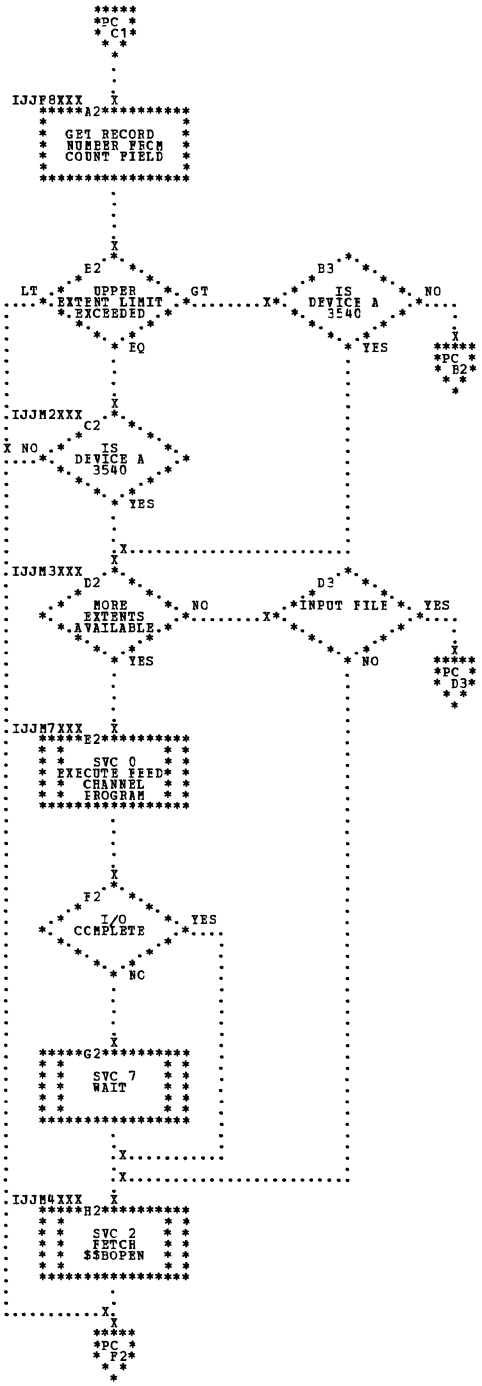


Chart PF. DIMOD: PUT Macrc, Twc I/O Areas (2 cf 2)

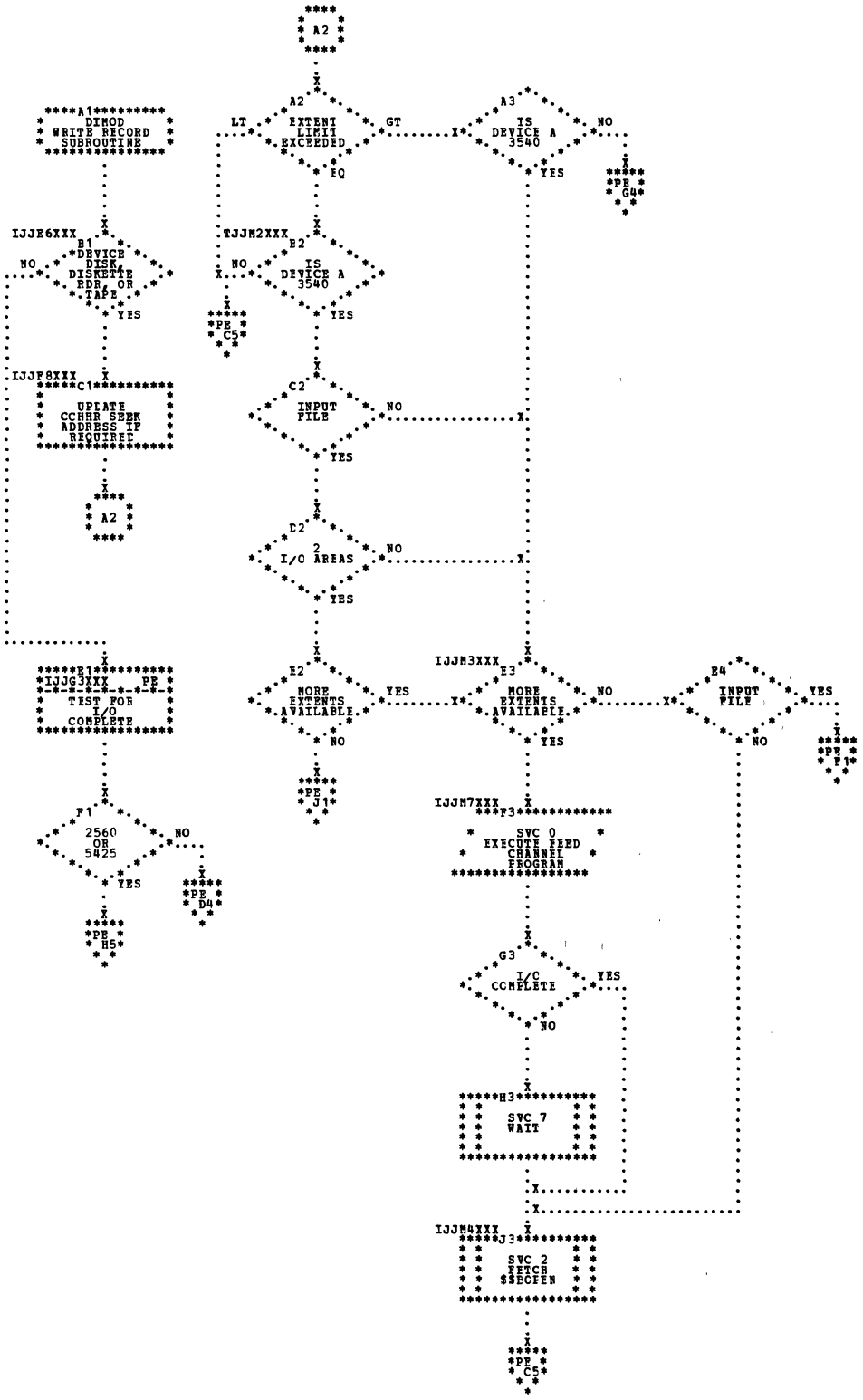


Chart PH. $$BERPTP: 1018 Punch-Tape Punch Errcr Reccvery Rcutine (1 cf 2)

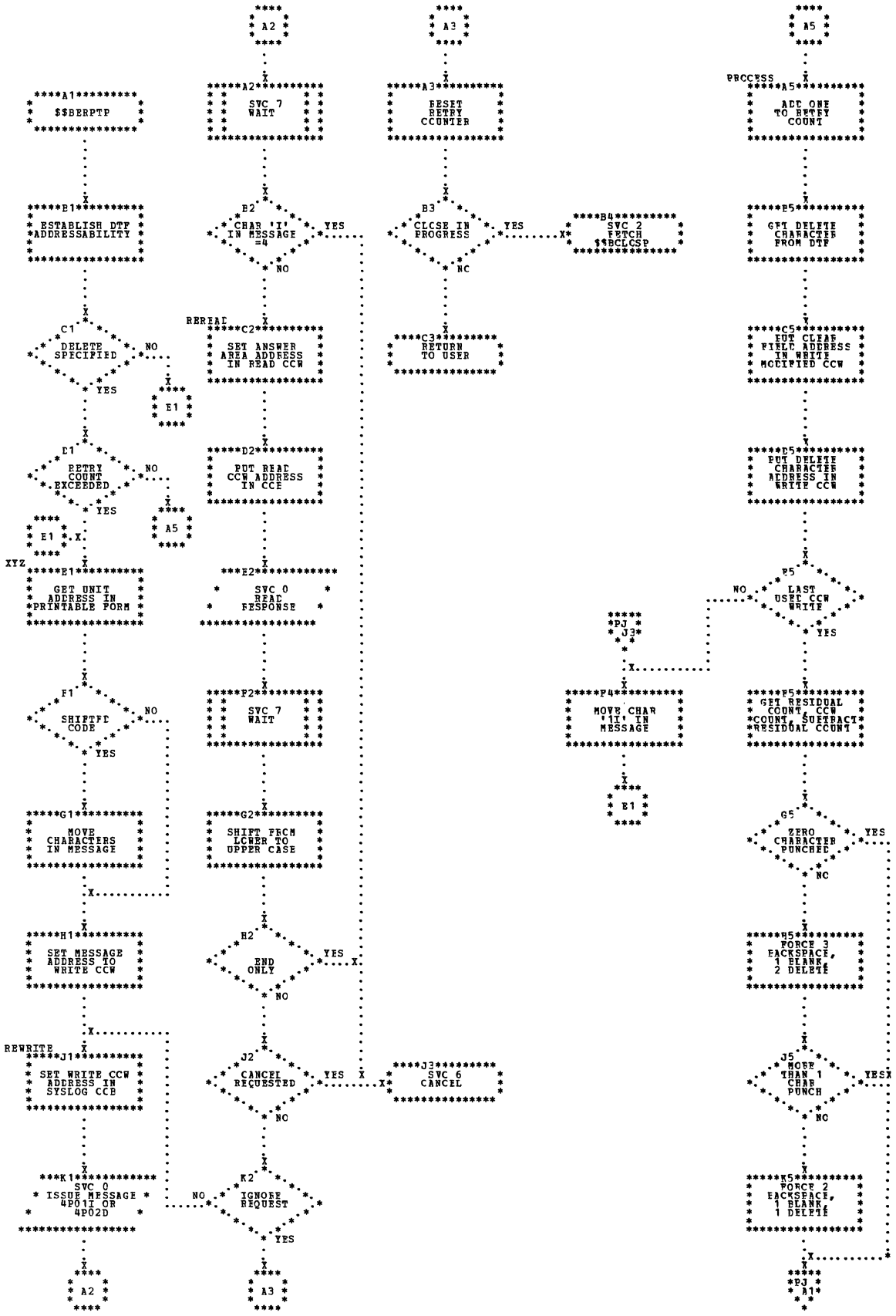
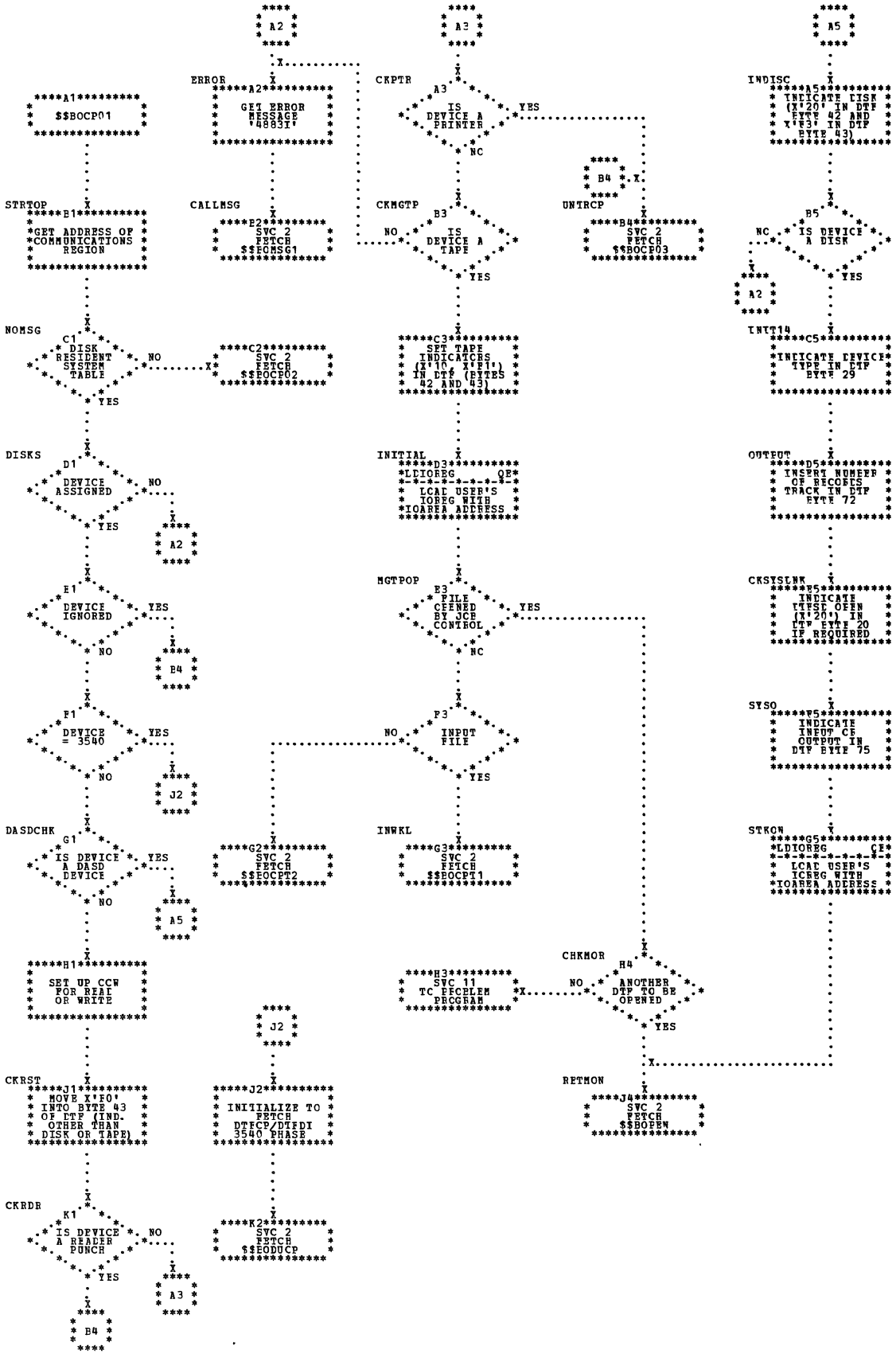


Chart QA. $$BCCP01: Open Device Independent Files, Phase 1



## Chart QC. $$BOCP03: Open Device Independent Files, Phase 3

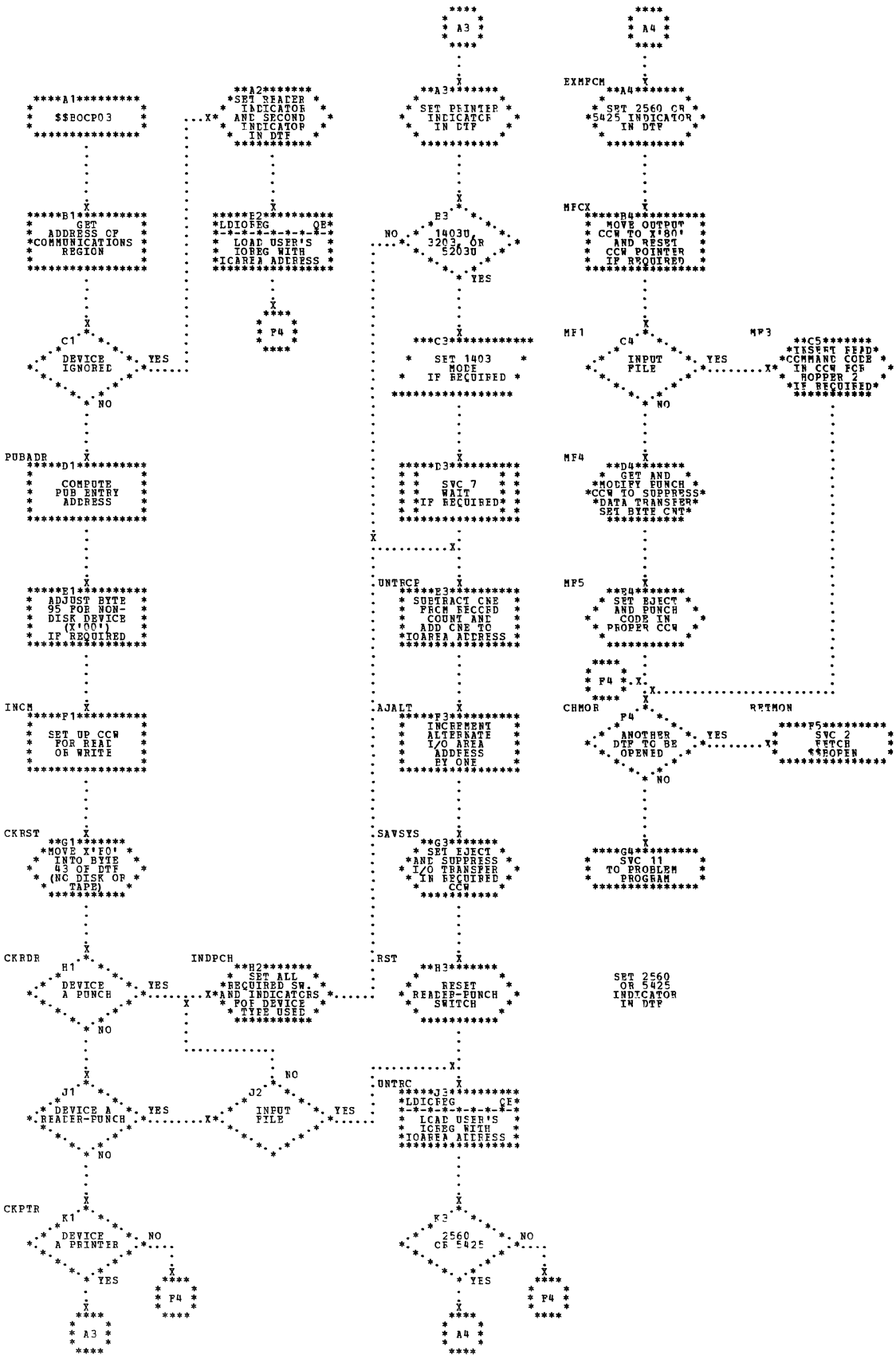
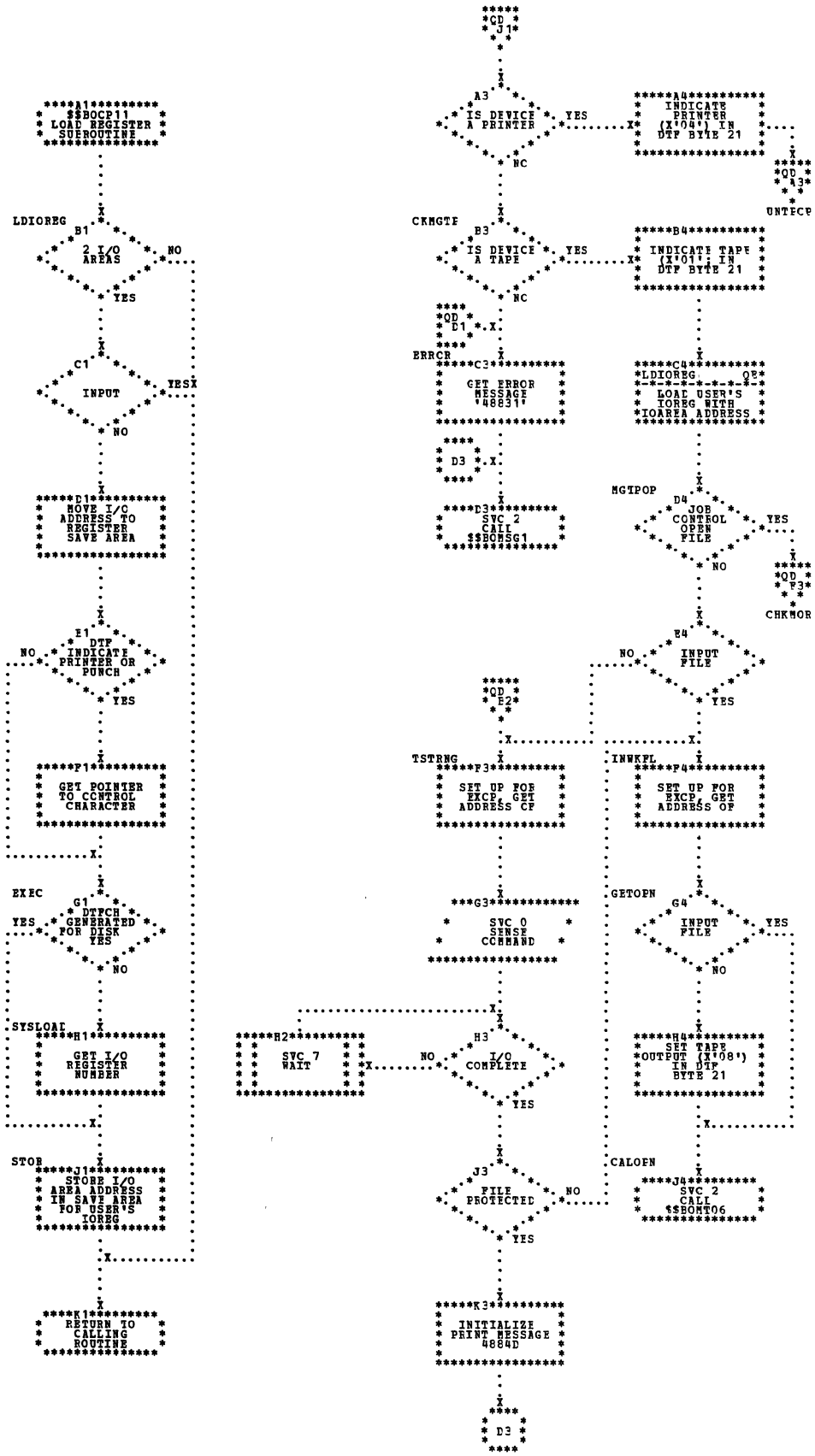


Chart QE. $$BOCP11: Open DTFCP (Version 1 Only), Phase 1 (2 of 2)



## Chart RA. $$BOCPT1: Open DTFCP and DTFDI Input Tape

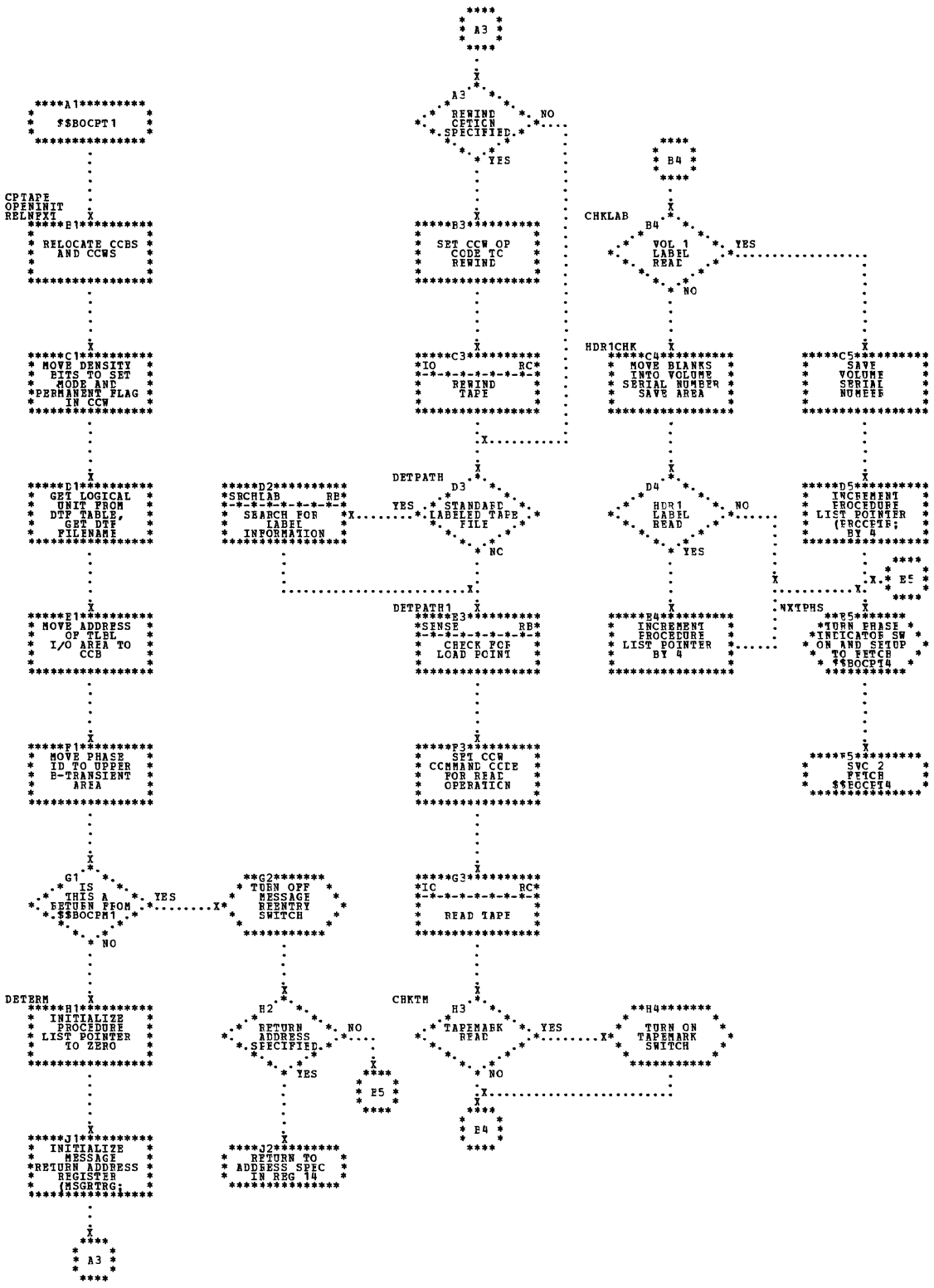


Chart RC. $$BCCPT1: Subroutines for Open DTFCP and DTFDI Input Tape (2 of 2)

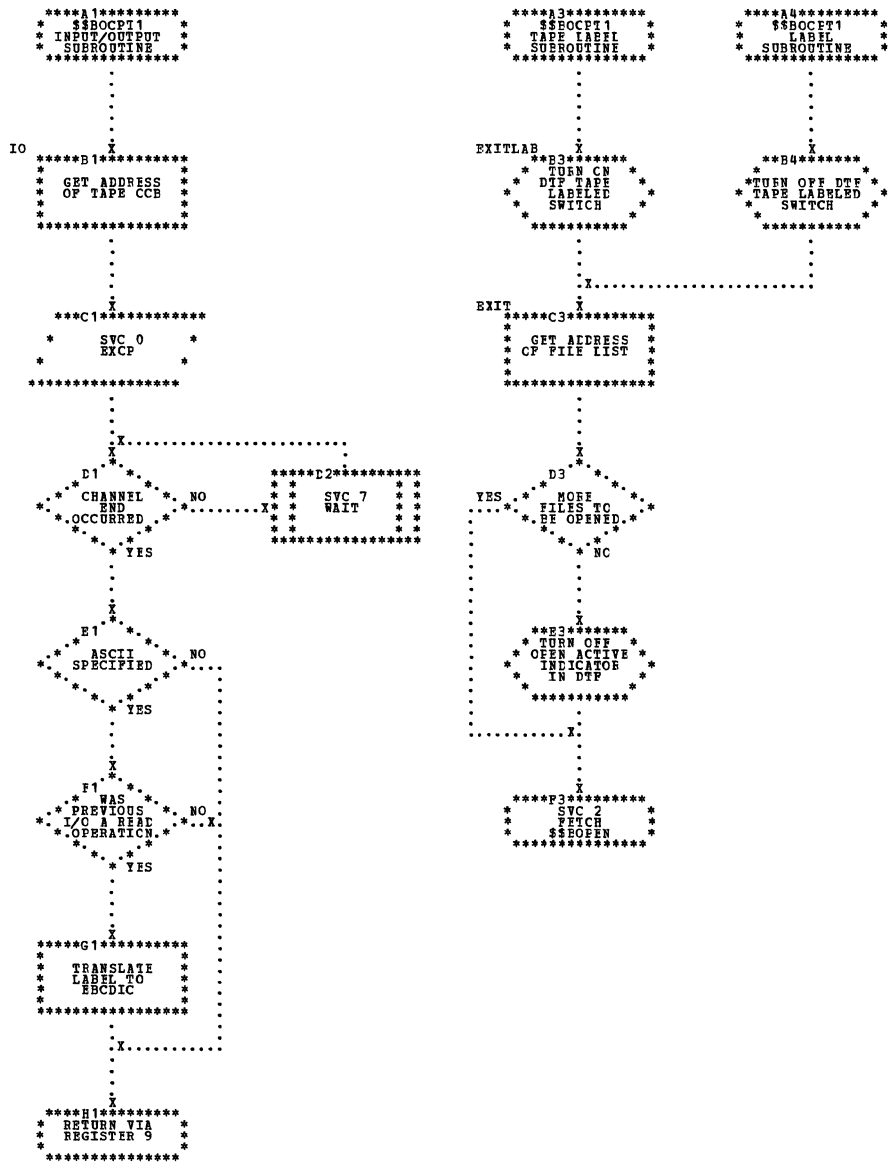
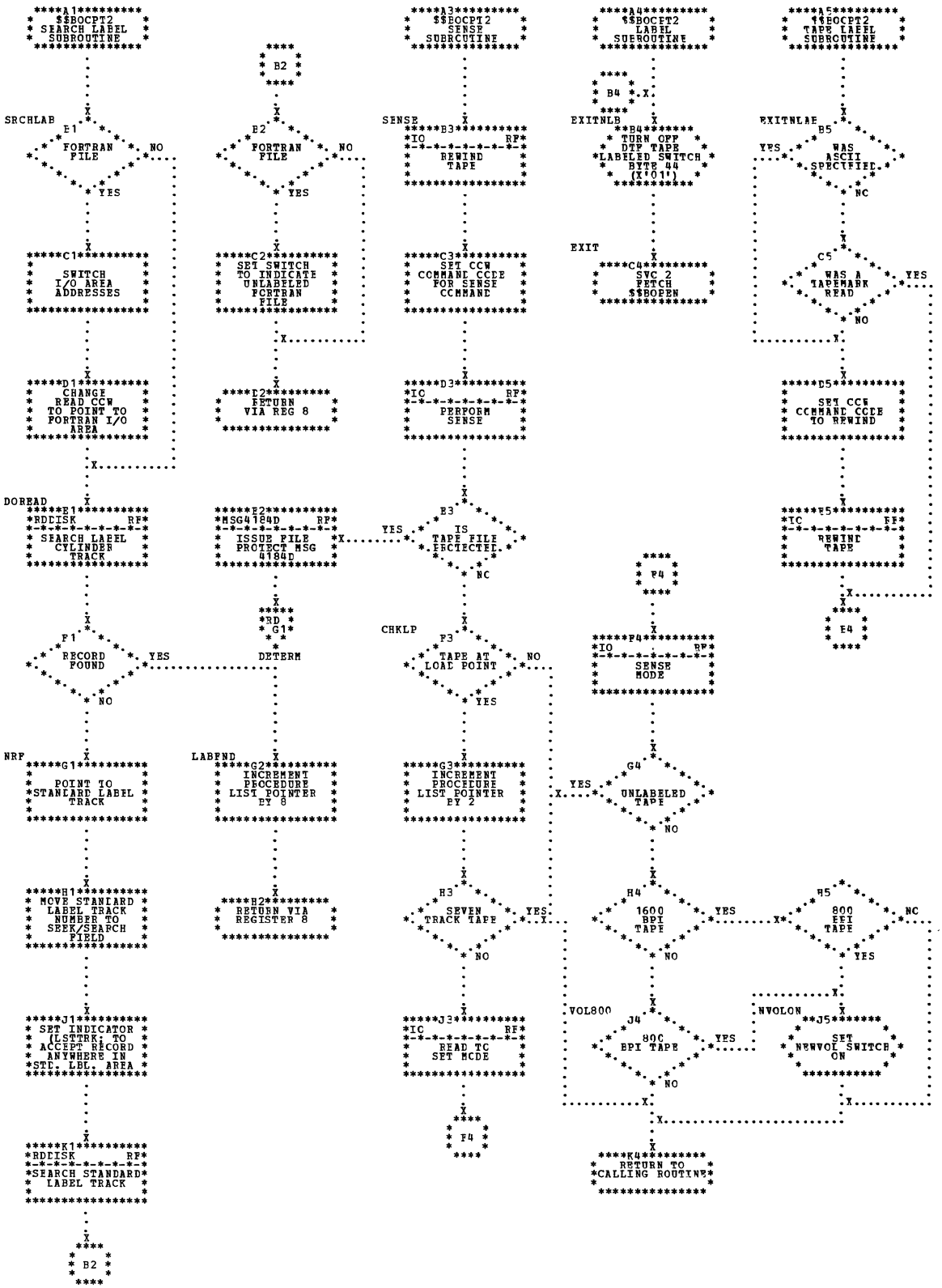


Chart RE. $$BOCPT2: Subrcutines fcr Cpen DTFCP and DTFDI Output Tape (1 cf 2)



## Chart RG. $$BOCPT3: Open DTFCP and DTFDI Labeled Output Tape

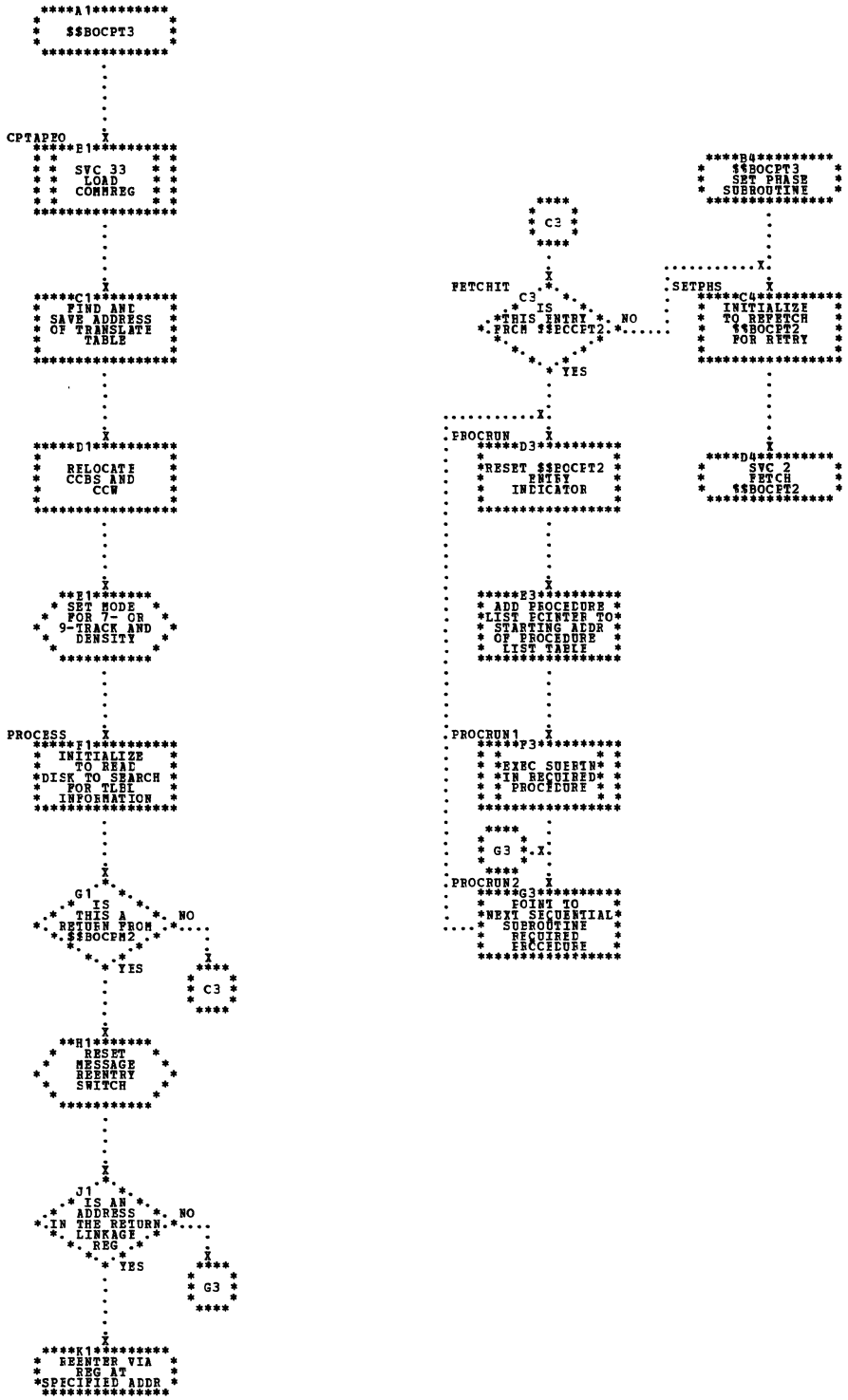


Chart RJ. $$BOCPT3: Subrcutines for Open DTFCP and DTFDI Labeled Output Tape (2 of 2)

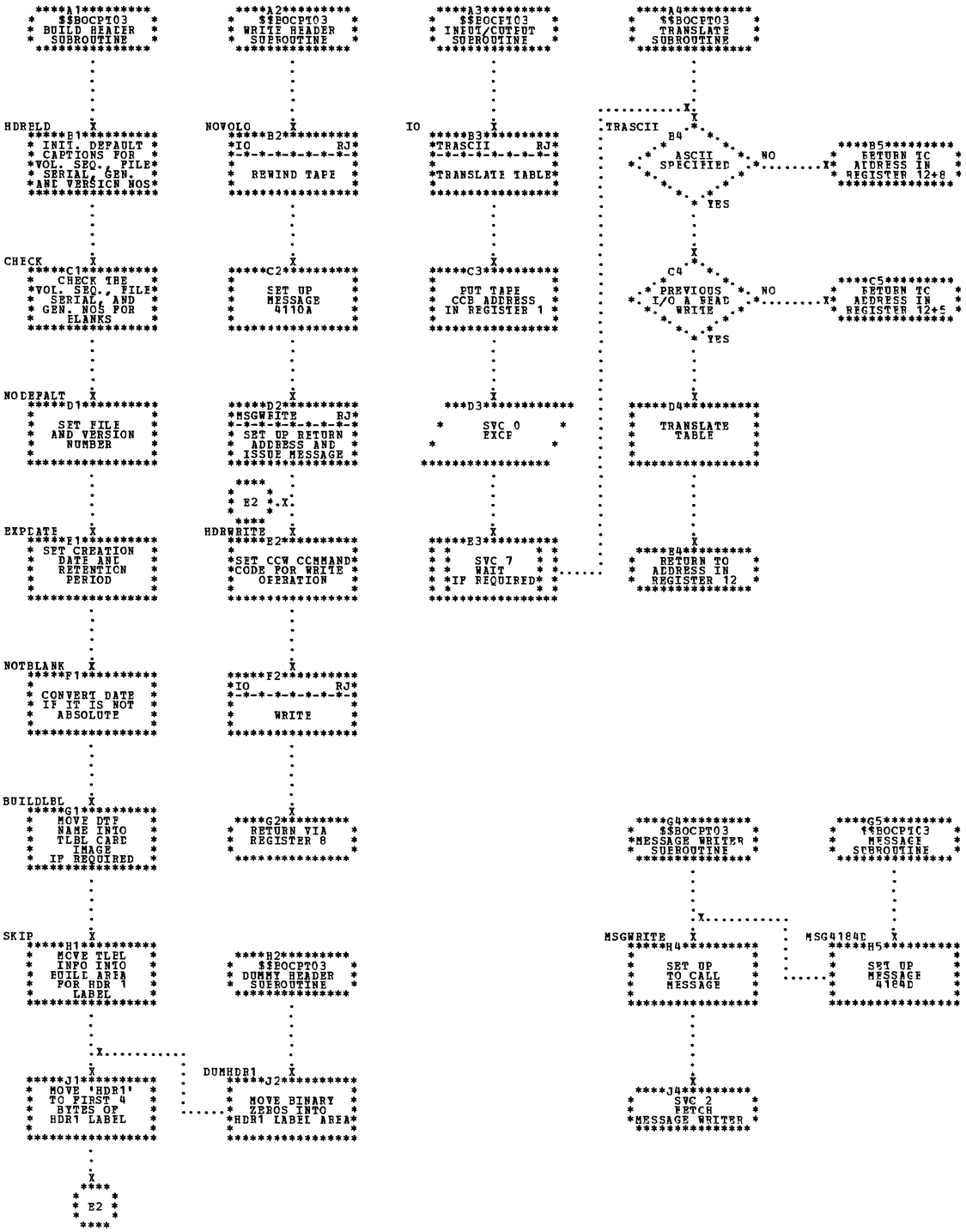


Chart RL. $$BOCPT4: Subroutines for Open DTFCP and DTFDI Labeled Input Tape (1 of 2)

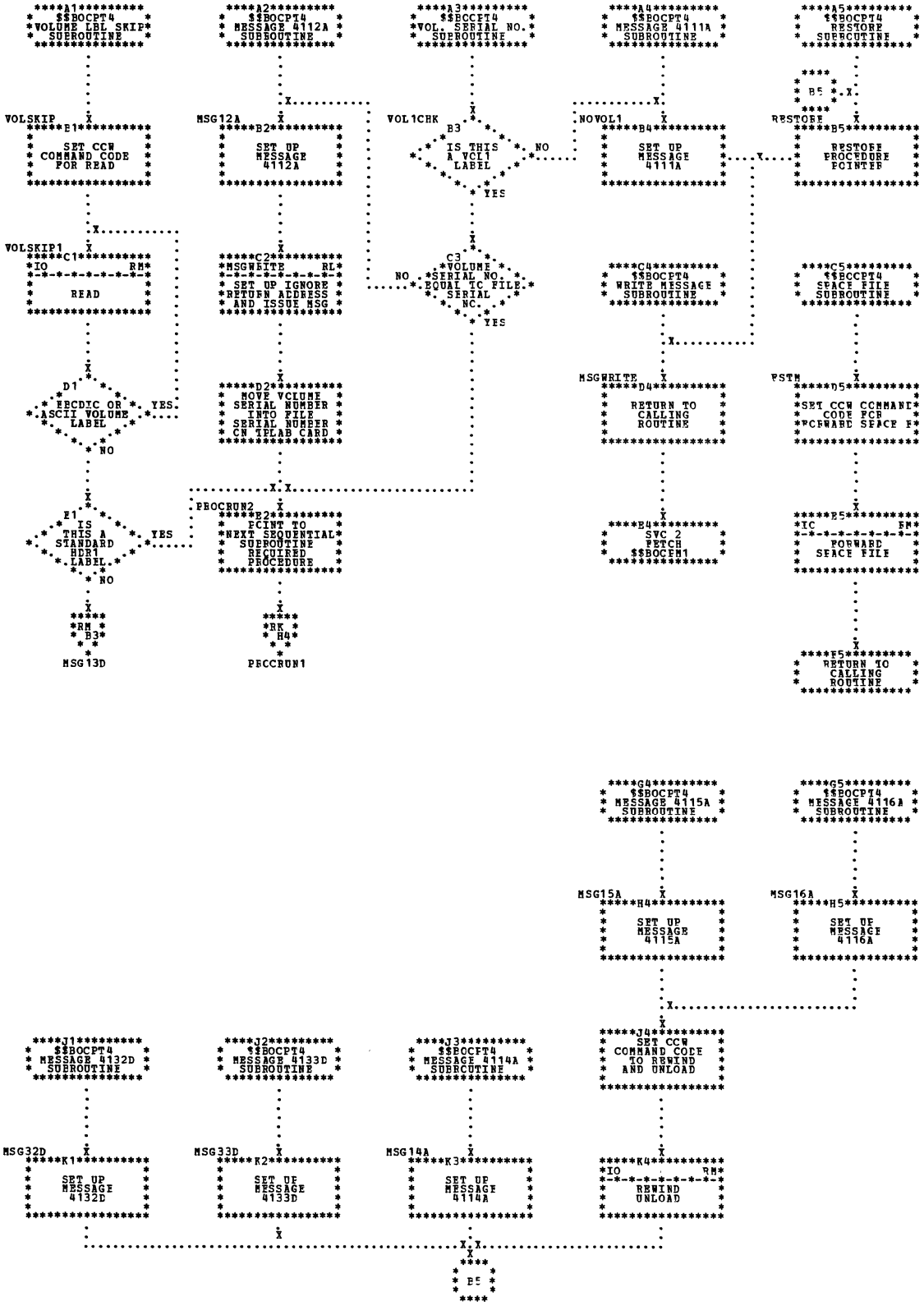


Chart SA. $$BCCPT1: Close DTFCP and DTFDI Tape Files (1 of 2)

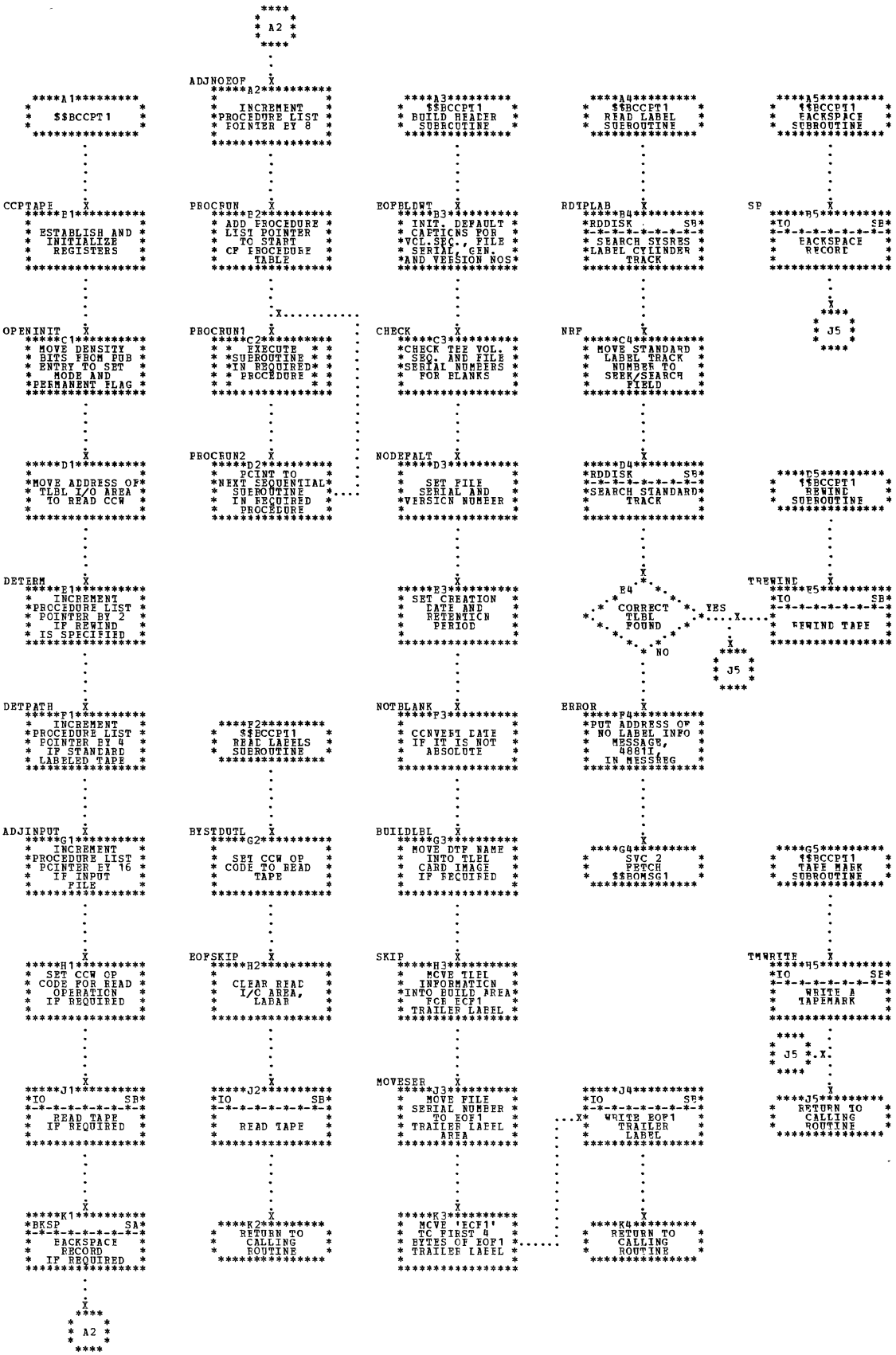


Chart SC. $$BCLCSP: Punch File Close (1 cf 2)

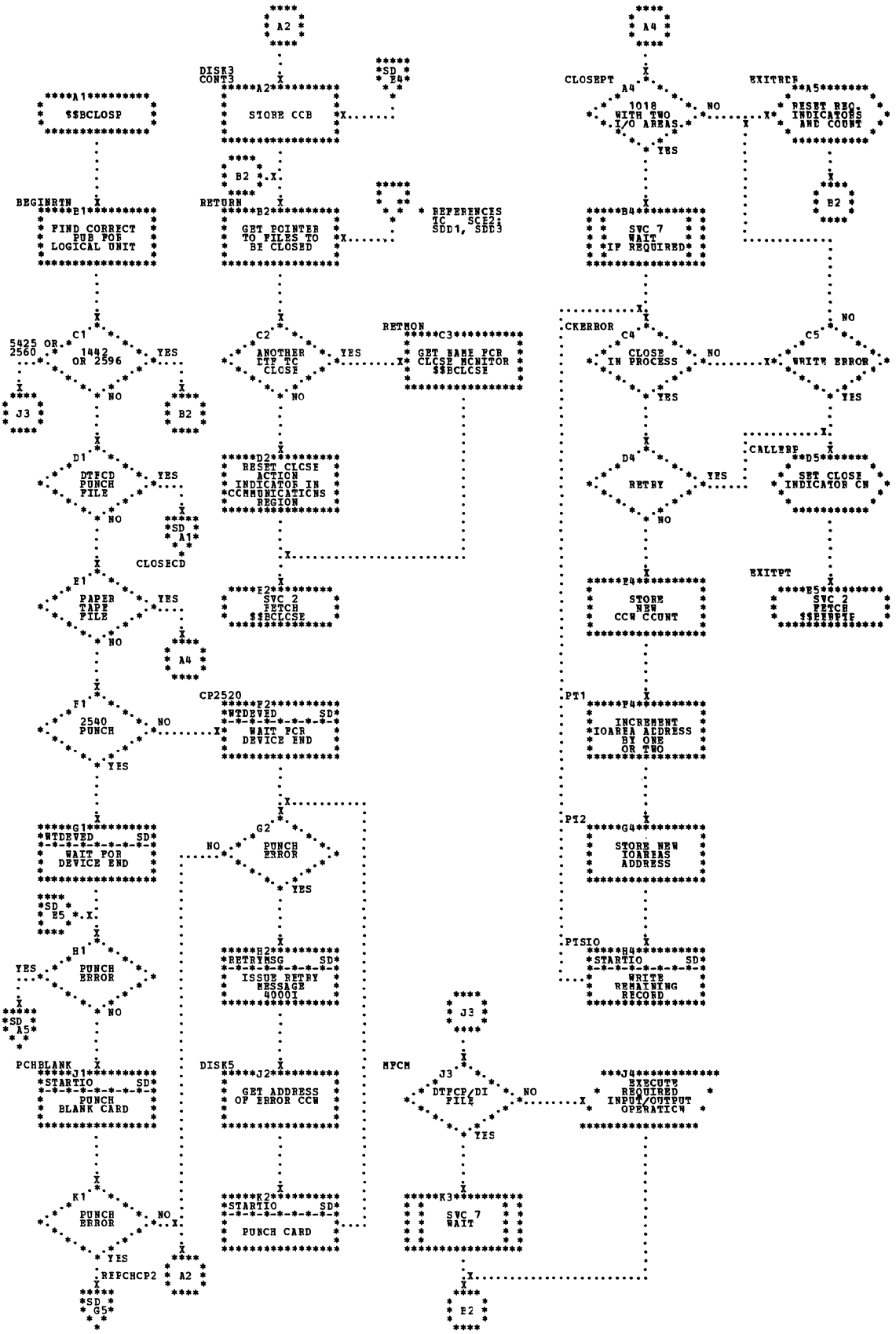
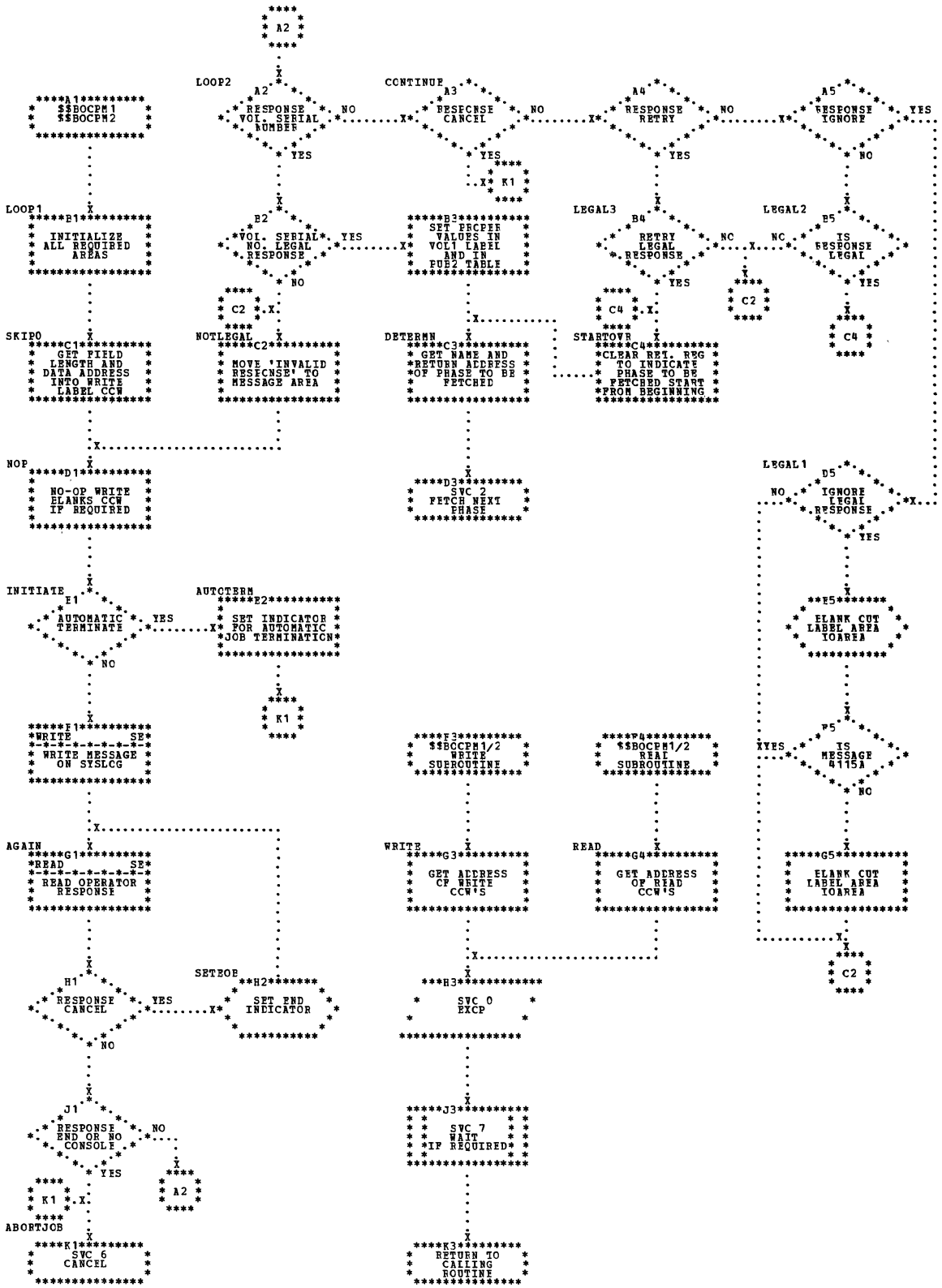
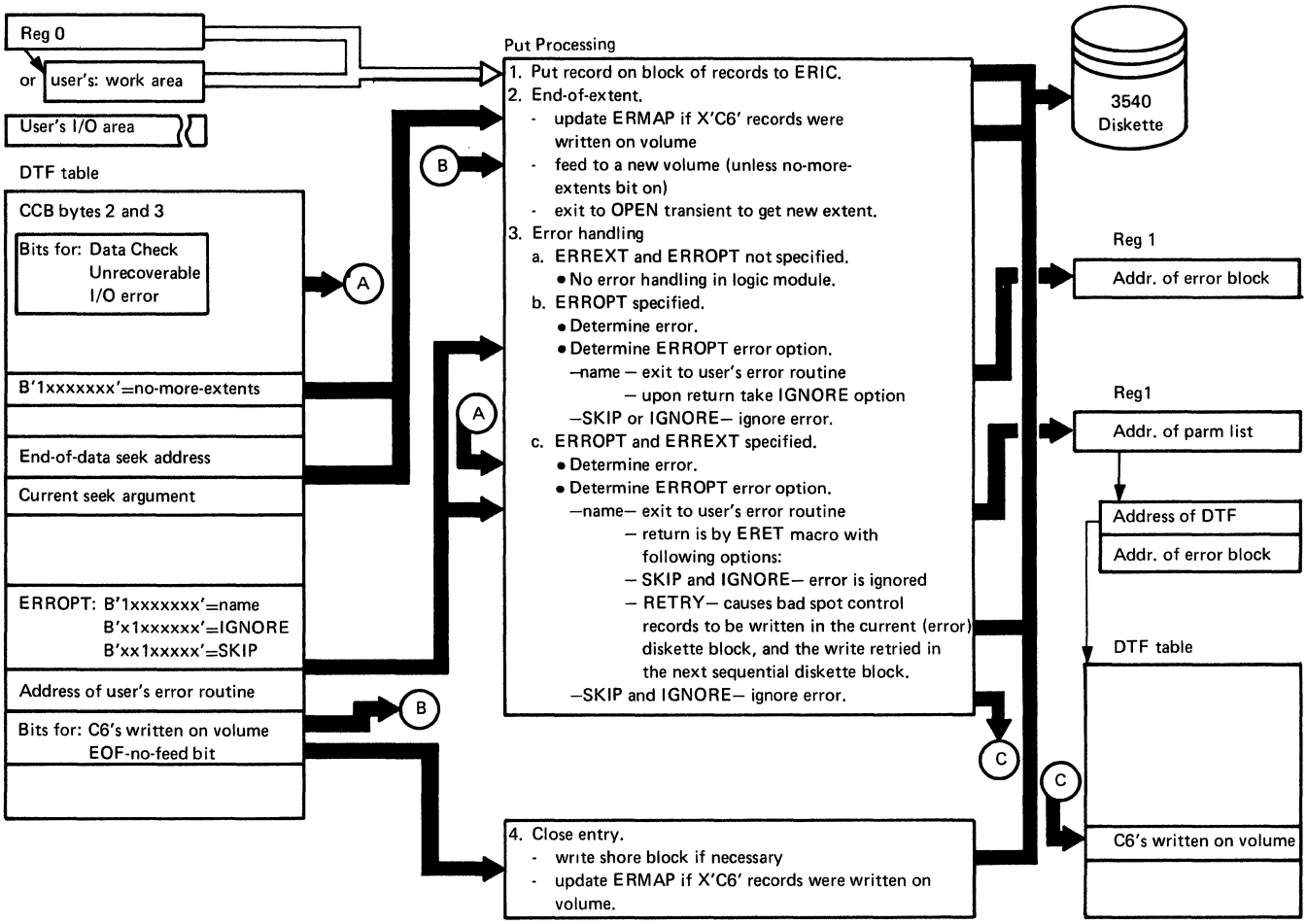


Chart SE. $$BOCPM1 and $$BOCPM2: DTFCP/DTFDI Message Writers



## Chart UB. DUMODFO: Put and Close Routines



## Chart UD. Transition through OPEN

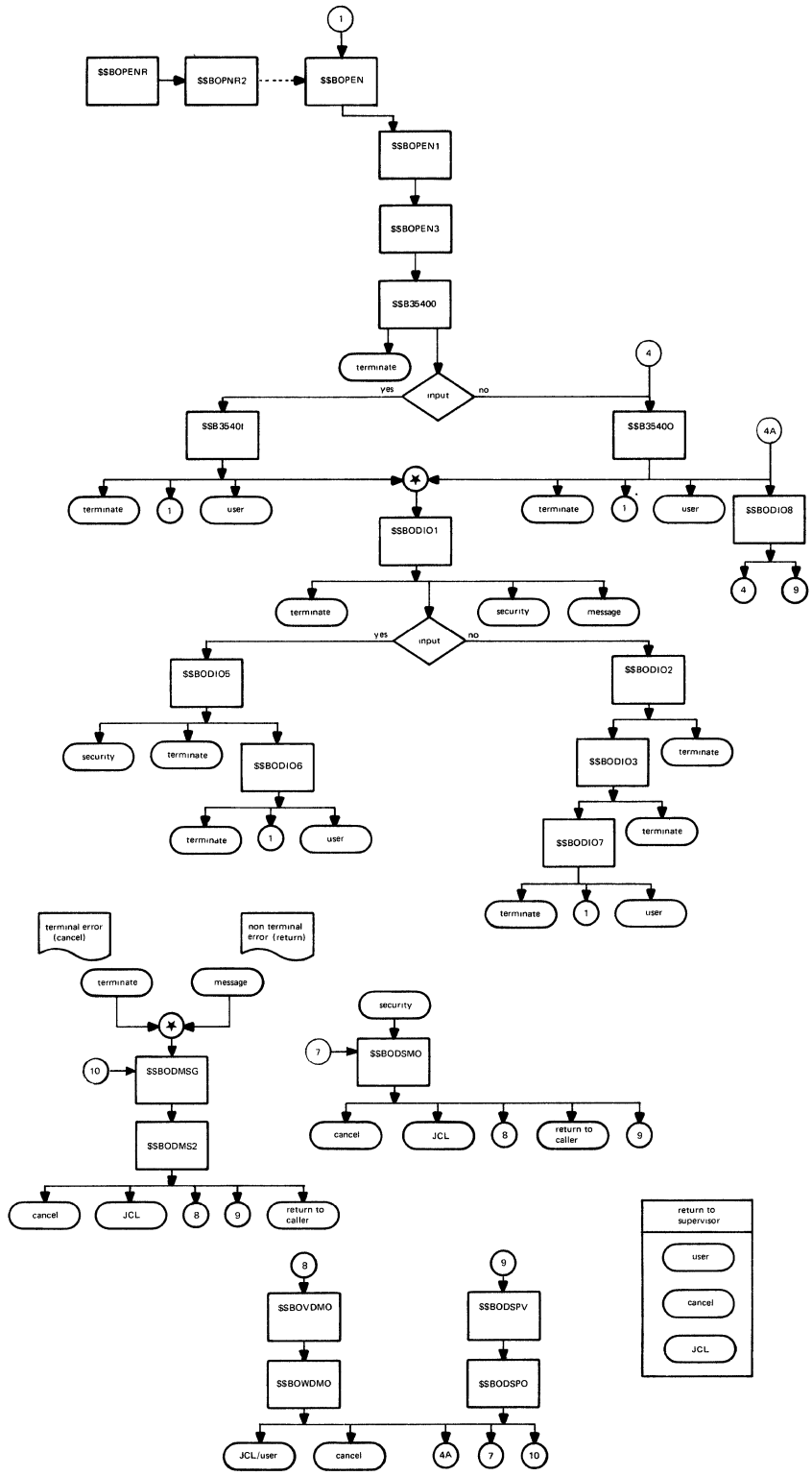


Chart UF. $$B35400: Diskette Open, Initialization (Part 2 of 3)

**Input** **Processing** **Output**

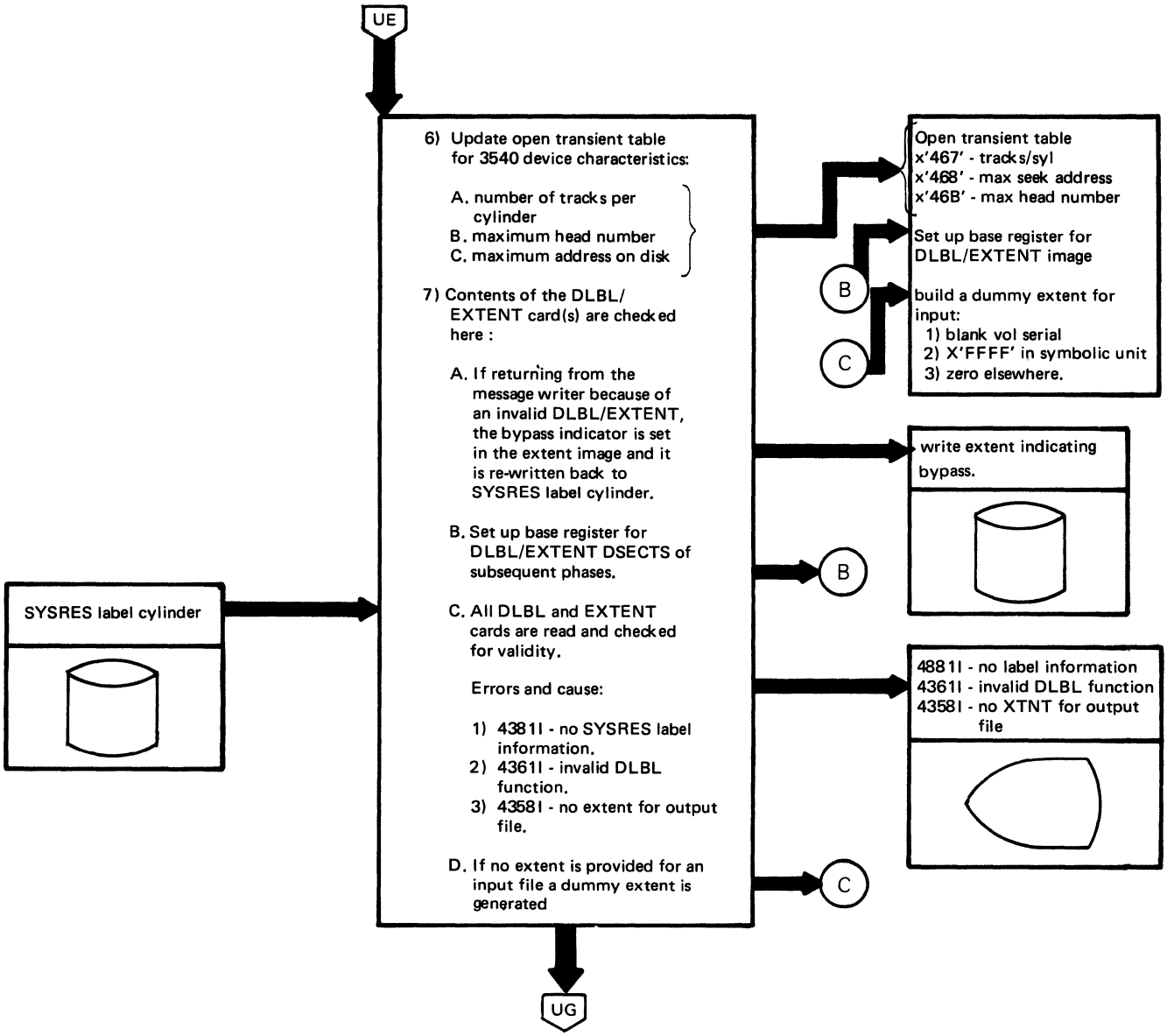


Chart VA. $$B3540I: Diskette Open Input (Part 1 of 3)

**Input** **Processing** **Output**

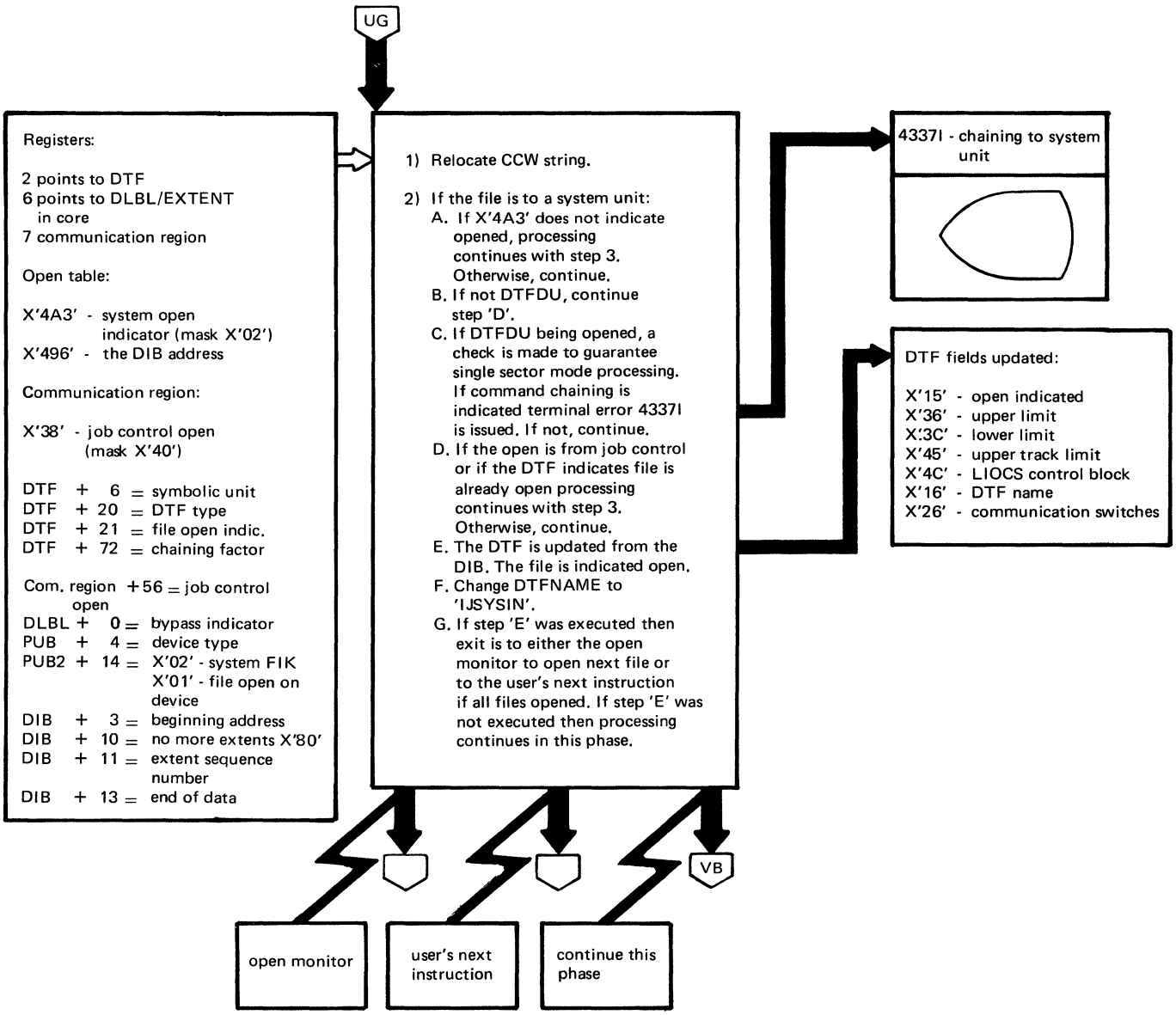


Chart VC. $$B3540I: Diskette Open Input (Part 3 of 3)

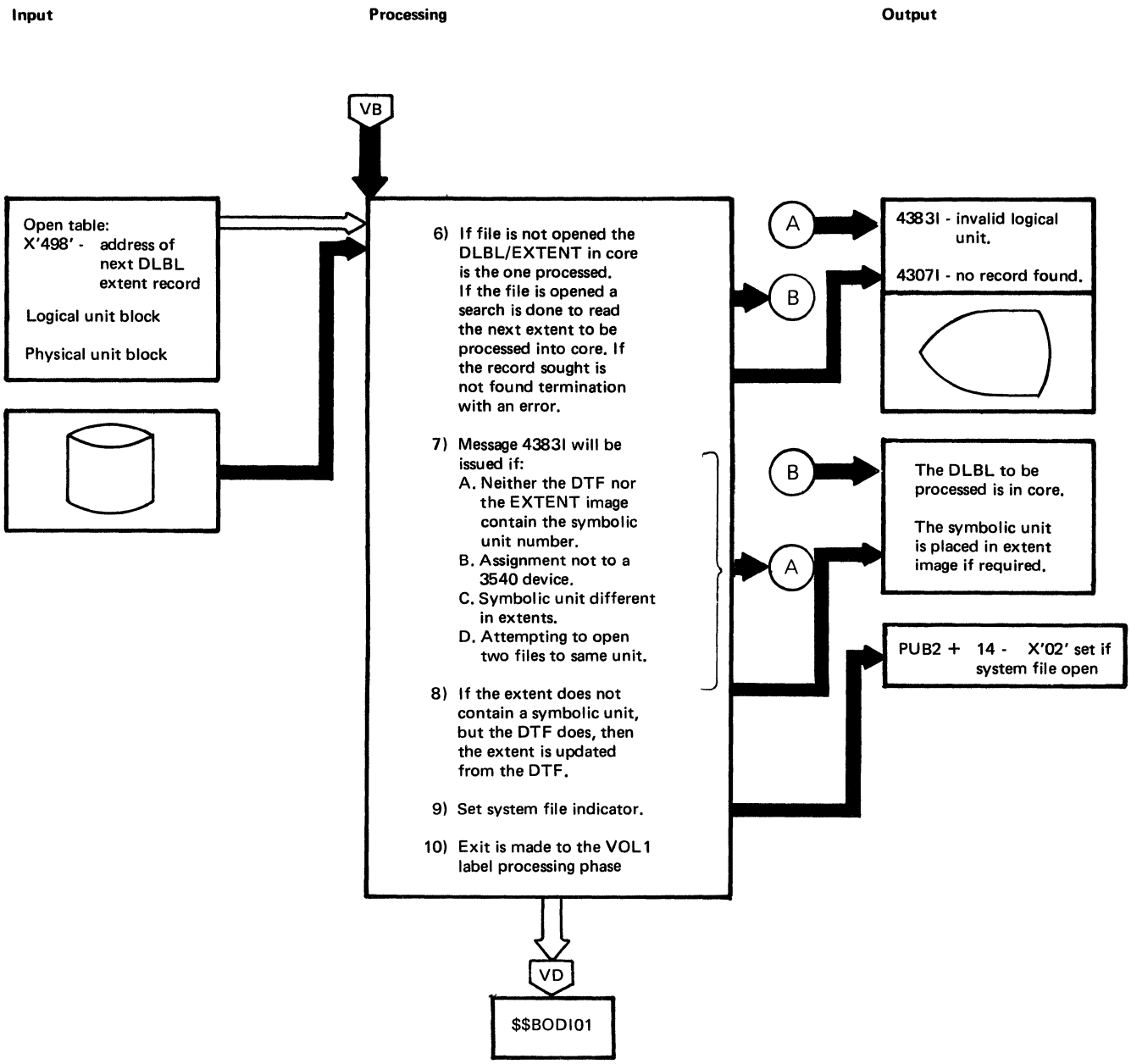


Chart VE. $$BODIO1: Diskette Volume Label Prccessor (Part 2 of 2)

**Input** **Processing** **Output**

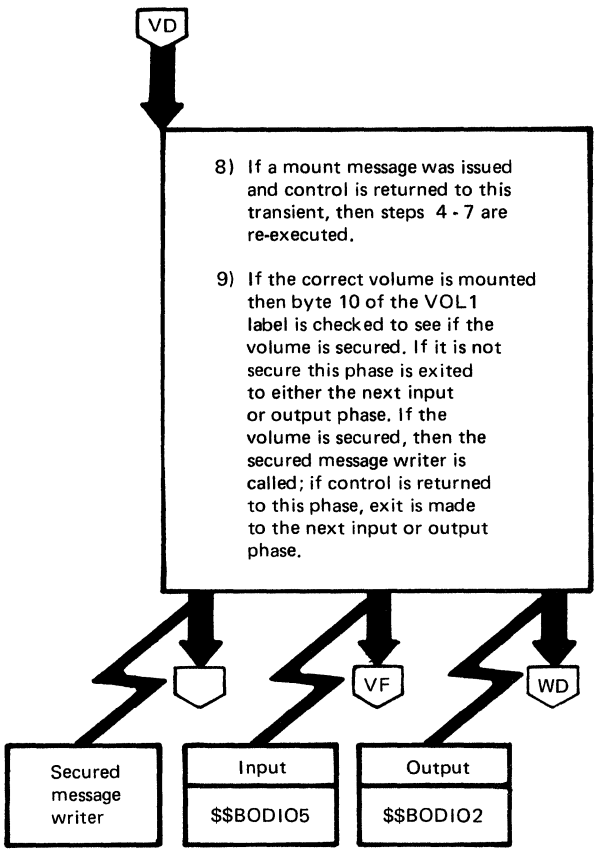


Chart VG. $$BODIO5: Diskette Open Input, HDR1 Label Processor (Part 2 of 5)

**Input** **Processing** **Output**

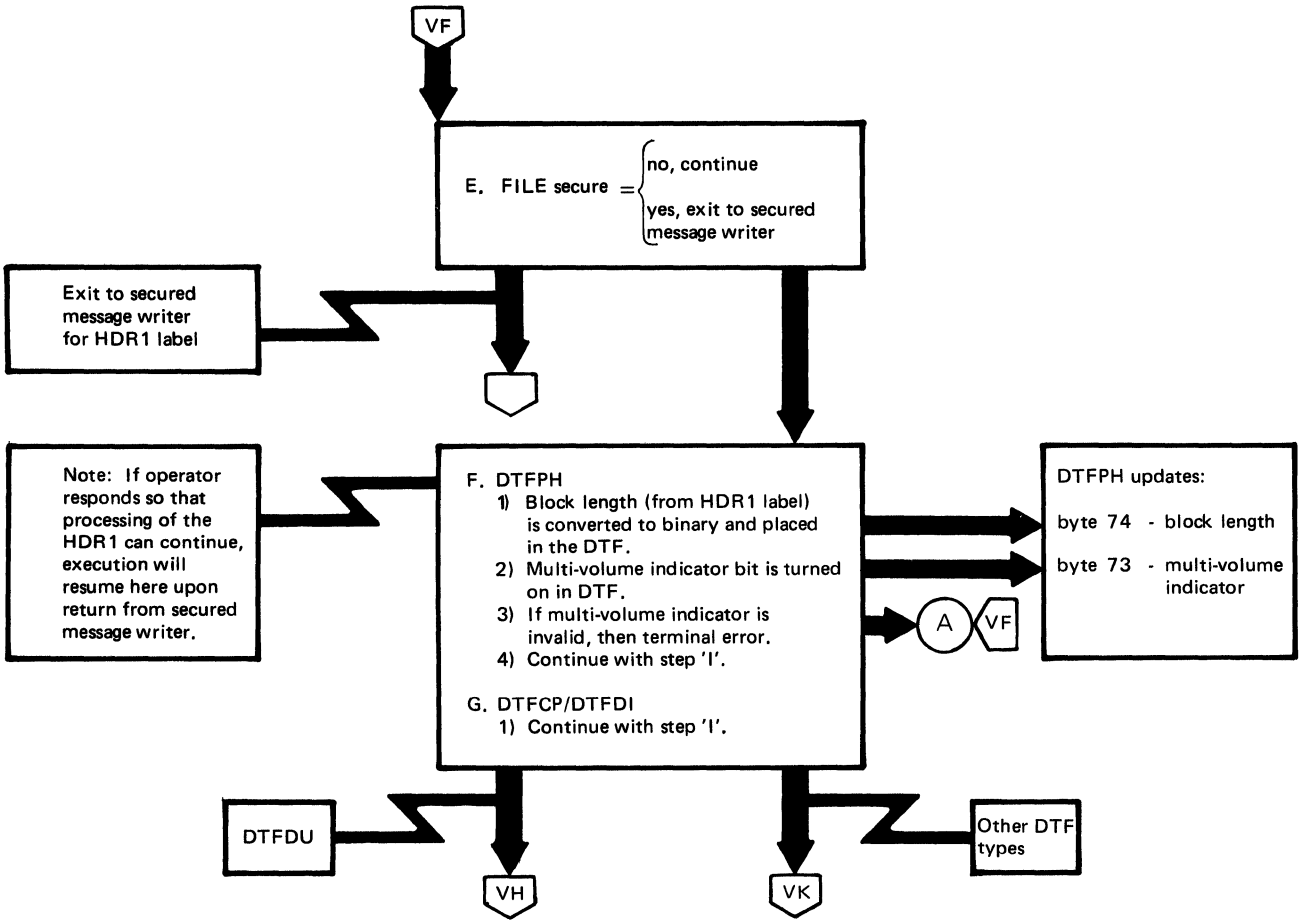


Chart VJ. $$BCDIO5: Diskette Open Input, HDR1 Label Processor (Part 4 of 5)

**Input** **Processing** **Output**

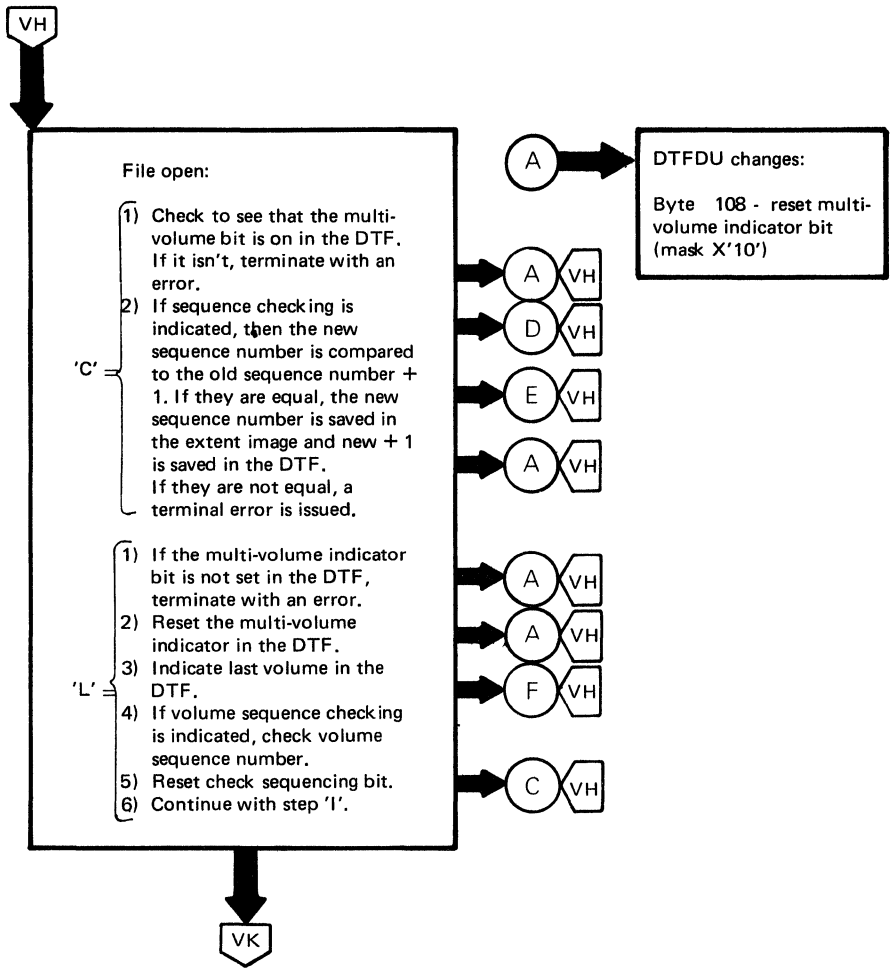
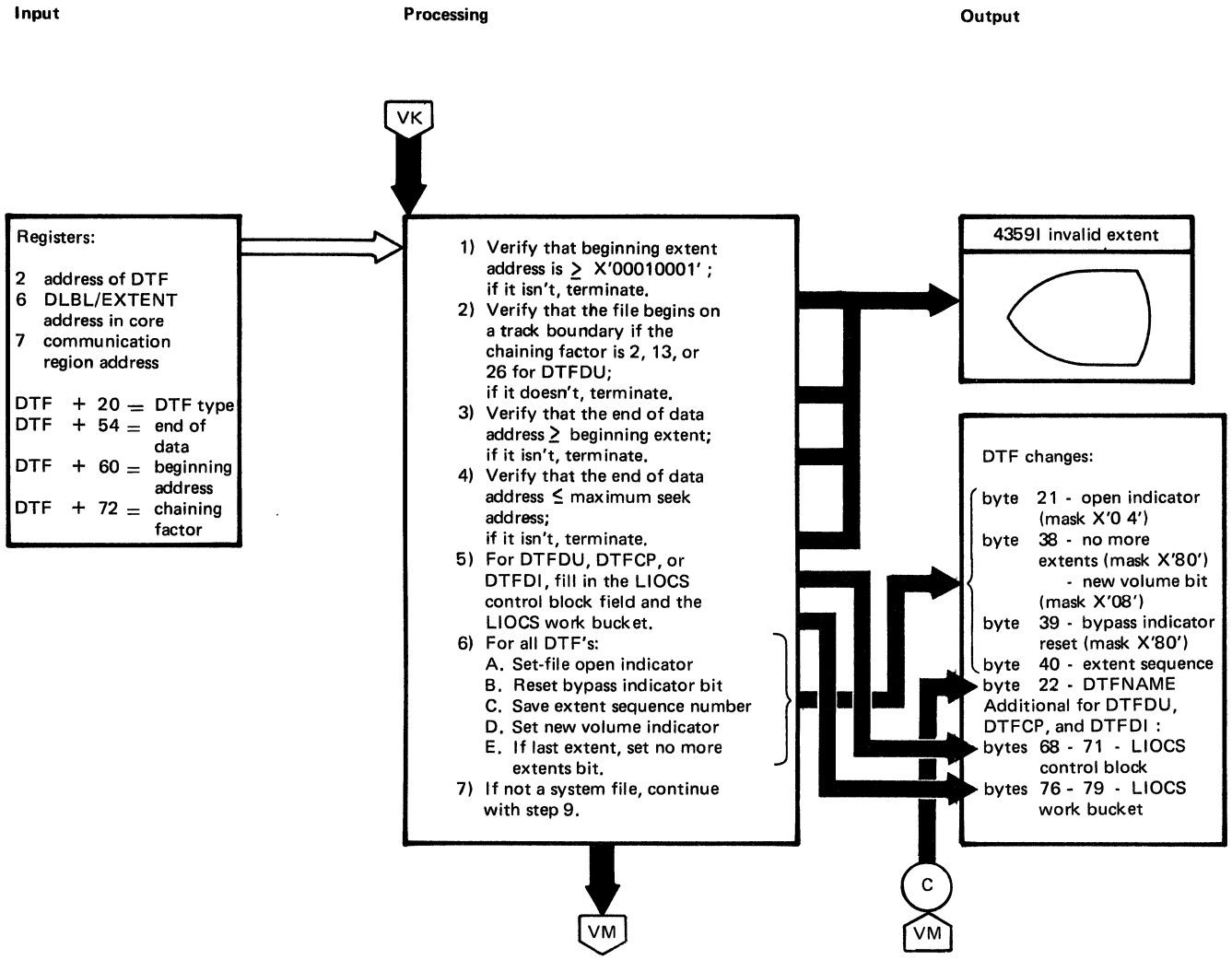


Chart VL. $$BODIO6: Diskette Open Input, Initialize DTF Table (Part 1 of 2)



## Chart WA. $$B3540O: Diskette Open Output (Part 1 of 3)

**Input** **Processing** **Output**

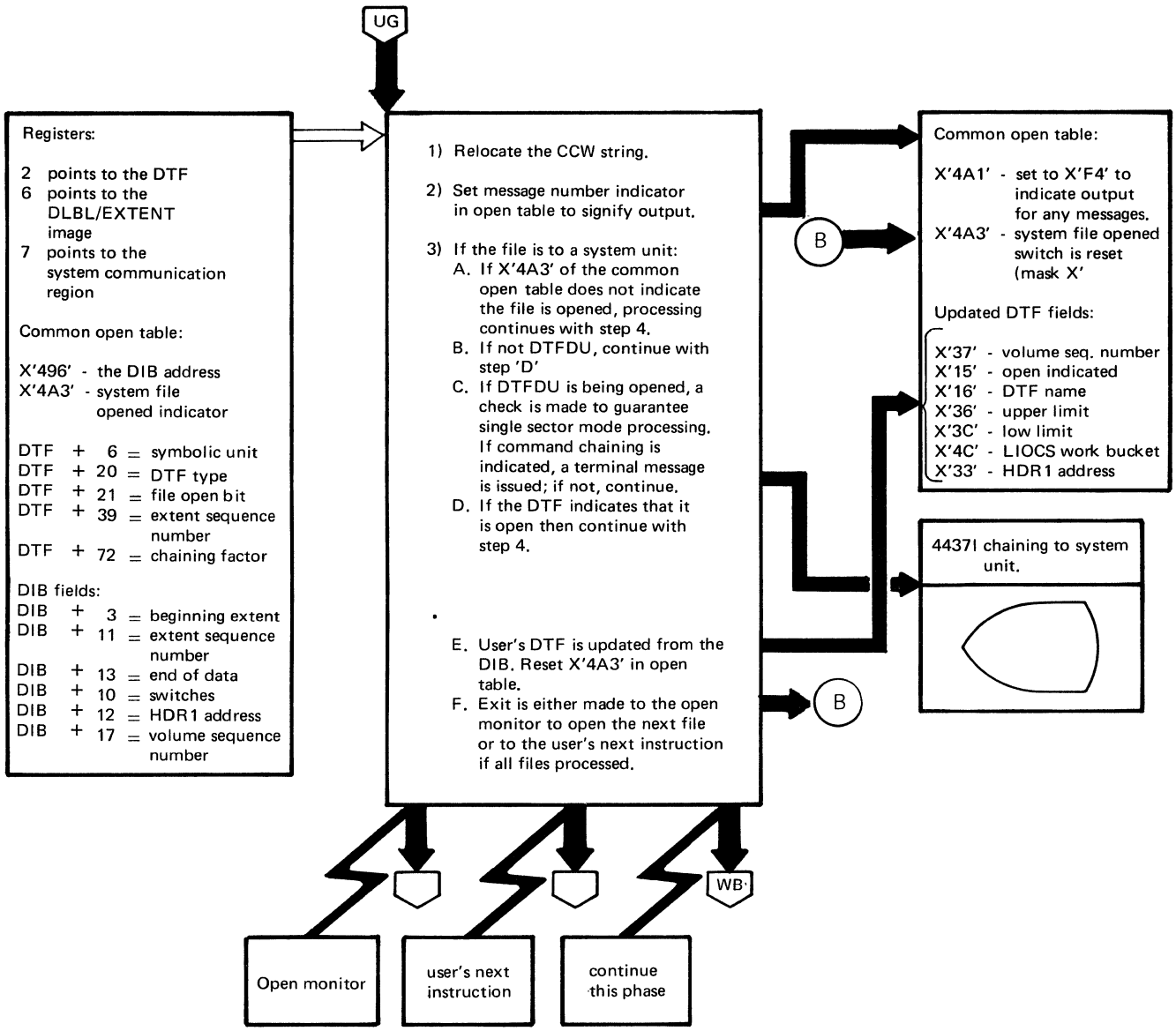


Chart WC. $$B3540O: Diskette Cpen Output (Part 3 of 3)

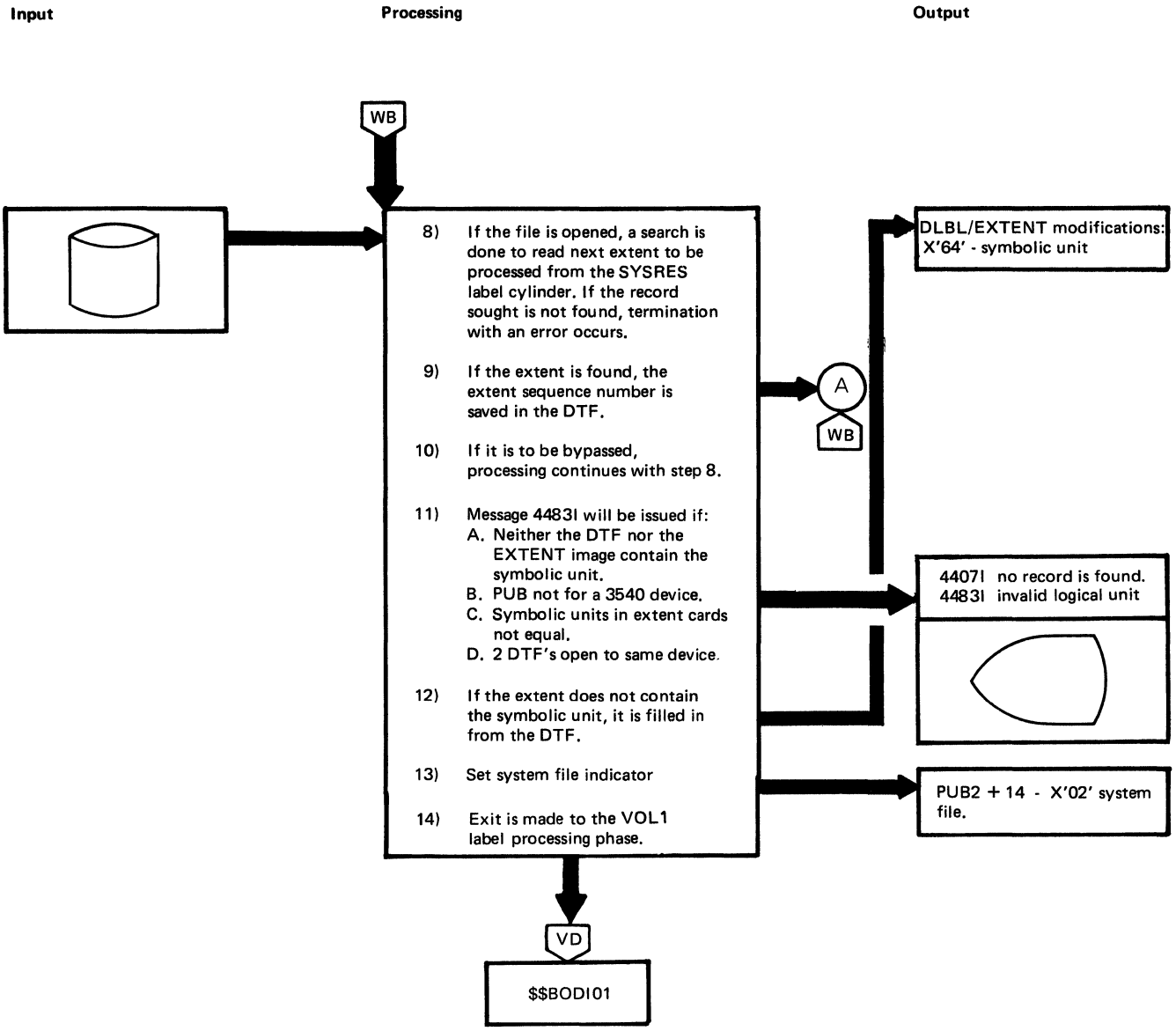


Chart WE. $$BODIO2: Diskette Open Output, Determine Extents (Part 2 of 3)

**Input** **Processing** **Output**

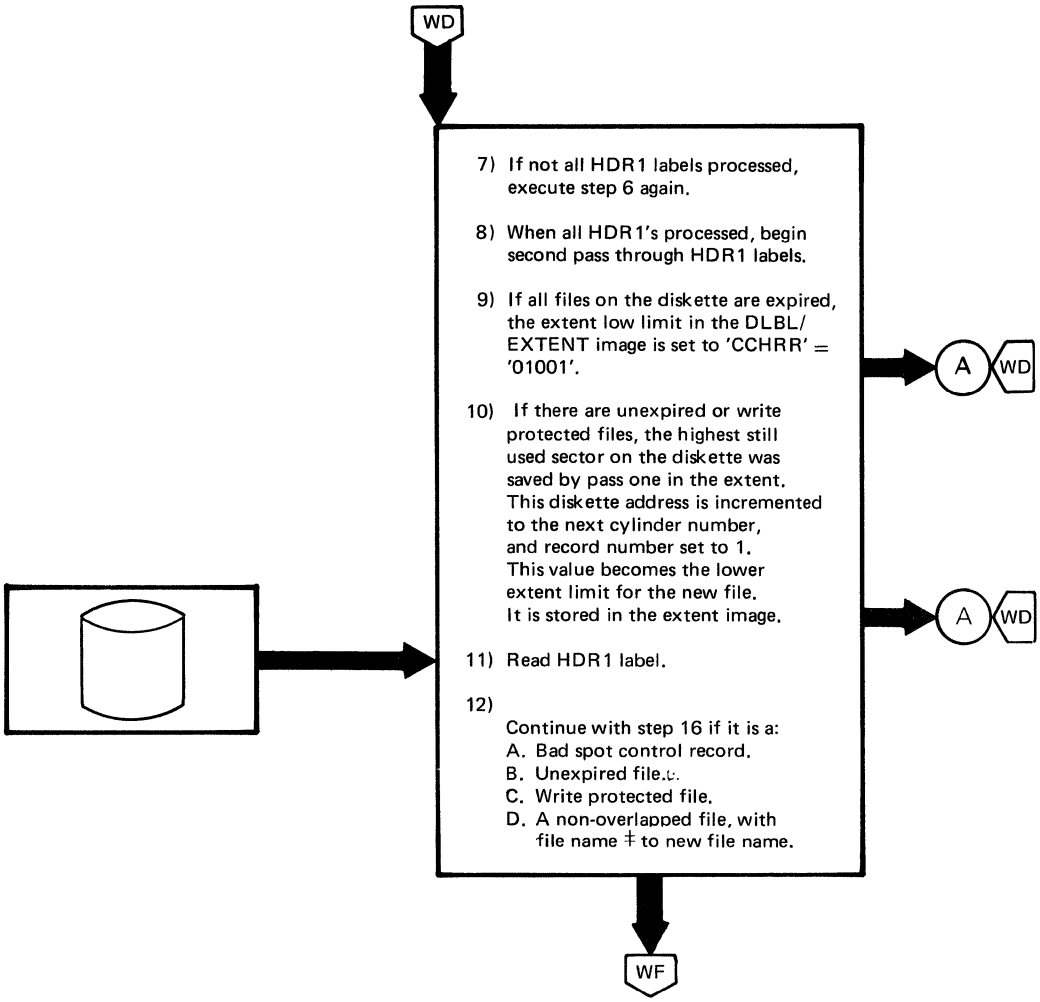


Chart WG. $$BODIO3: Diskette Open Output, Create/Write New HDR1 Label (Part 1 of 3)

**Input** **Processing** **Output**

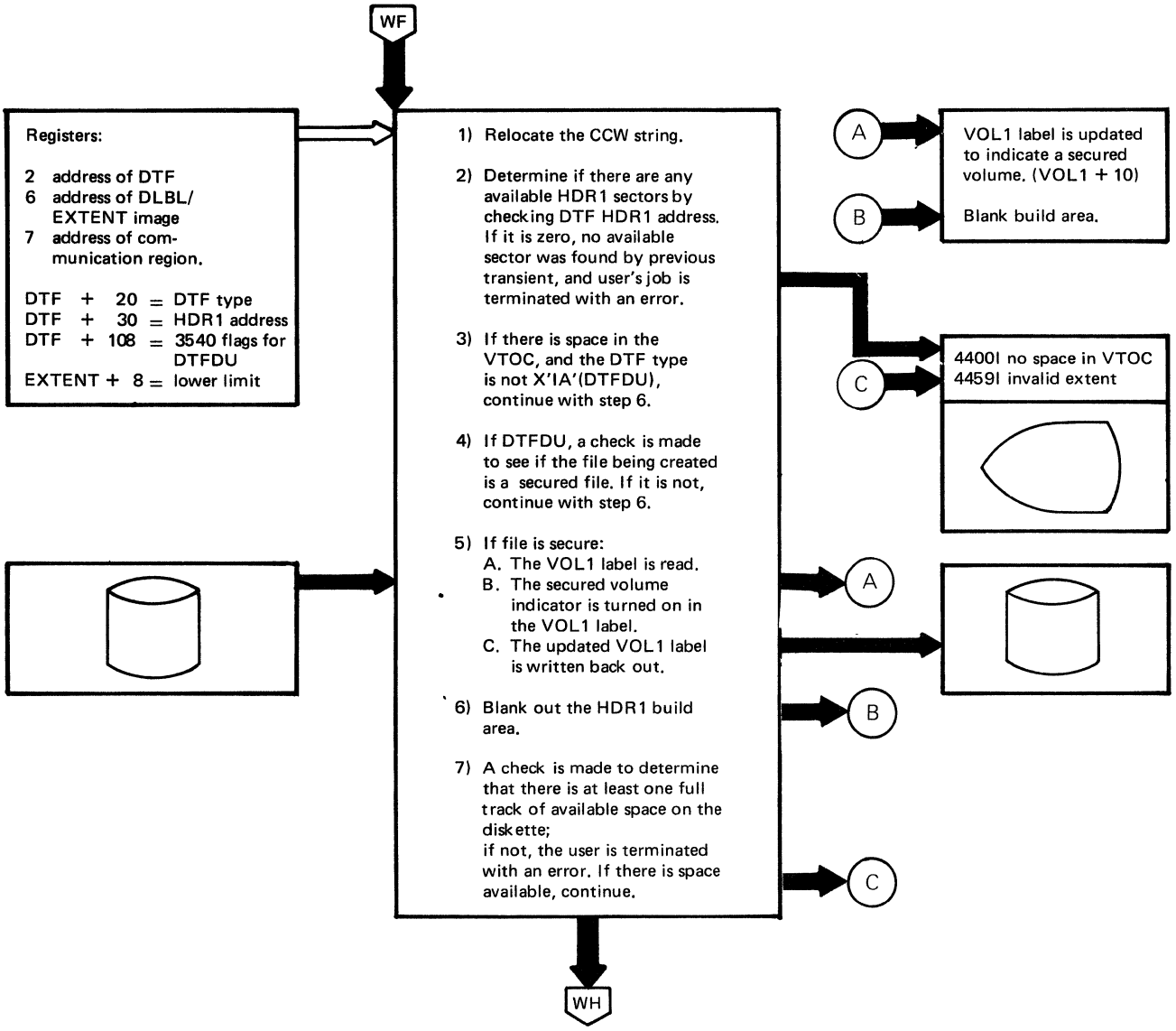


Chart WJ. $$BODIO3: Diskette Open Output, Create/Write New HDR1 Label (Part 3 of 3)

**Input** **Processing** **Output**

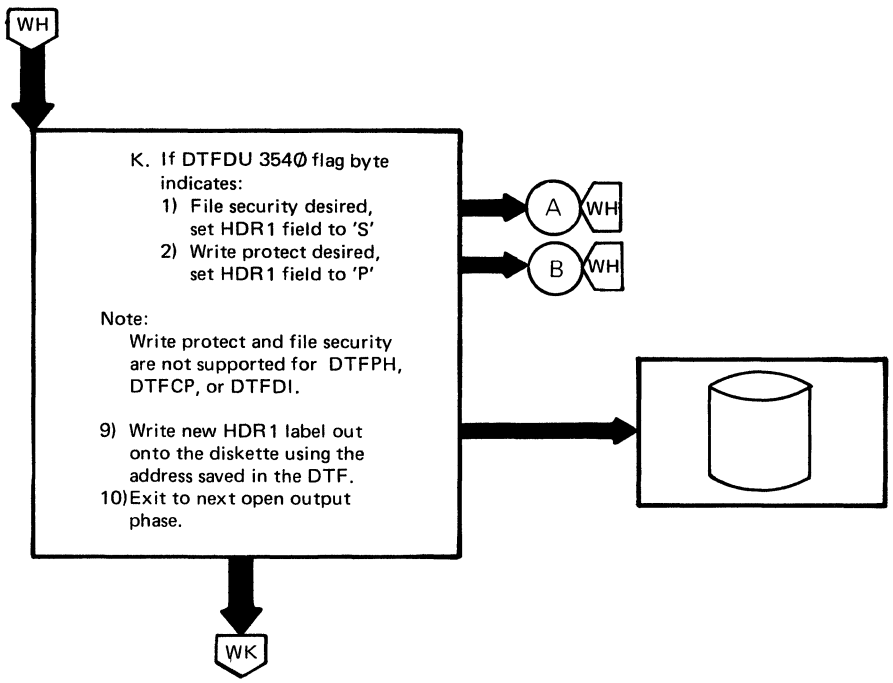
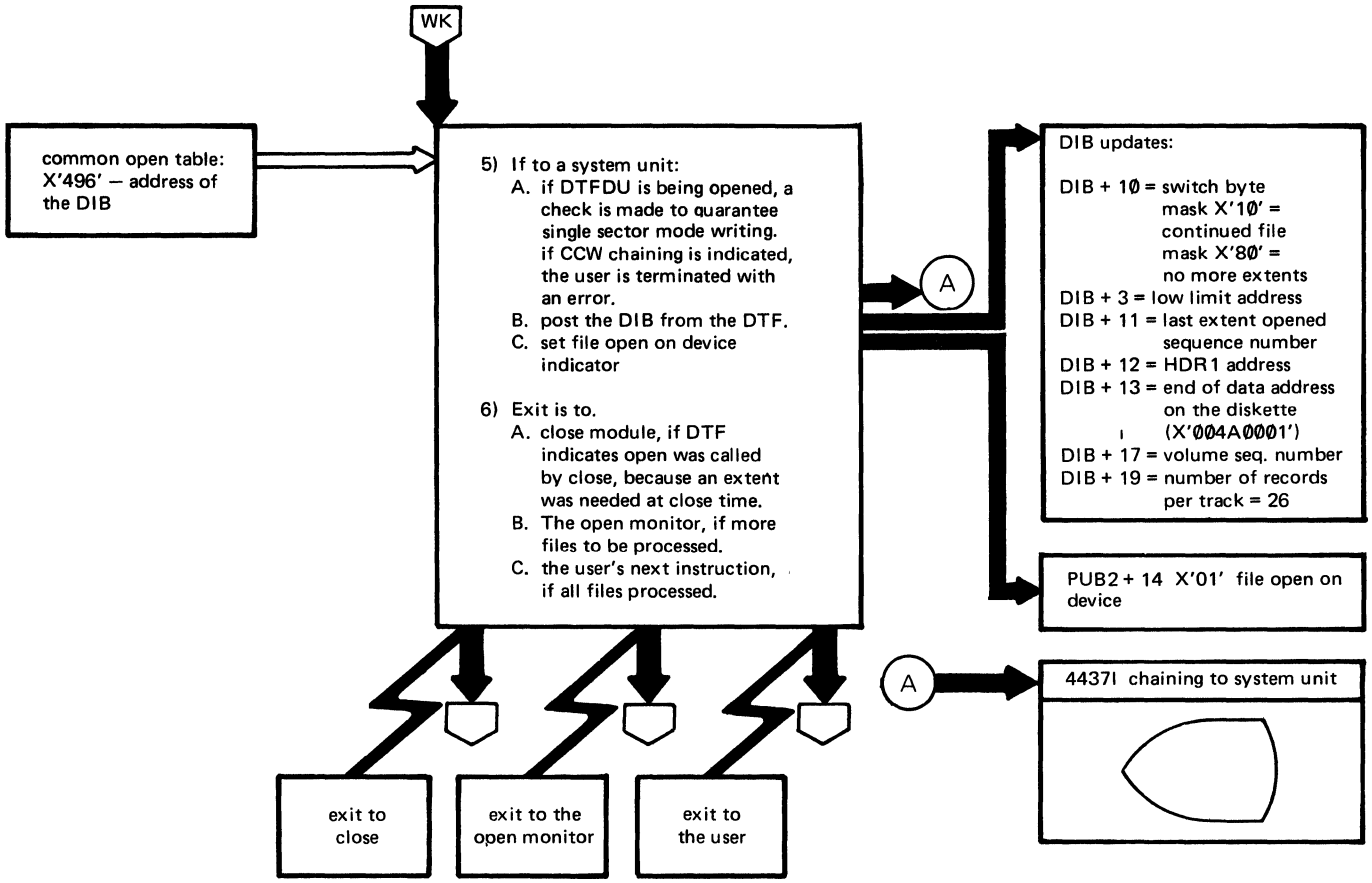


Chart WL. $$BODIO7: Diskette Open Output, Initialize DTF Table (Part 2 of 2)

**Input** **Processing** **Output**



## Chart WN. $$BCDUCP: Diskette DTFCP/DTFDI Open (Part 2 cf 2)

**Input** **Processing** **Output**

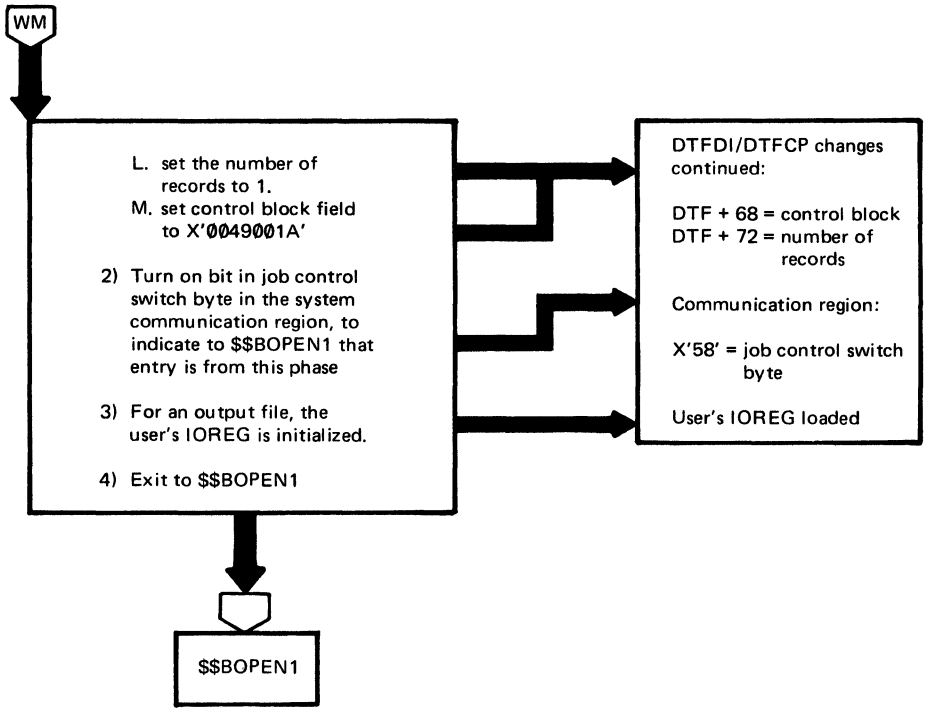


Chart WQ. $$BODIO8: Diskette Cpen Output, Extents frcm Console (Part 2 of 2)

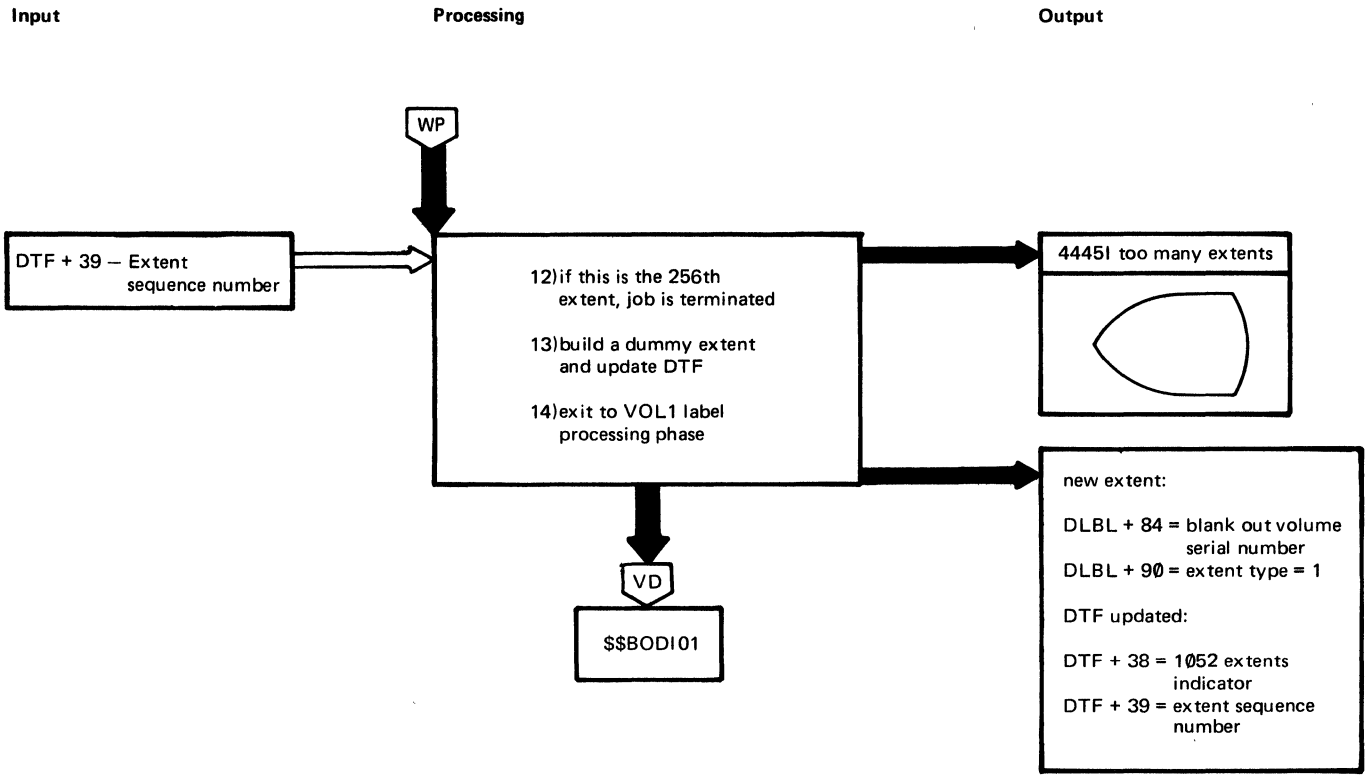
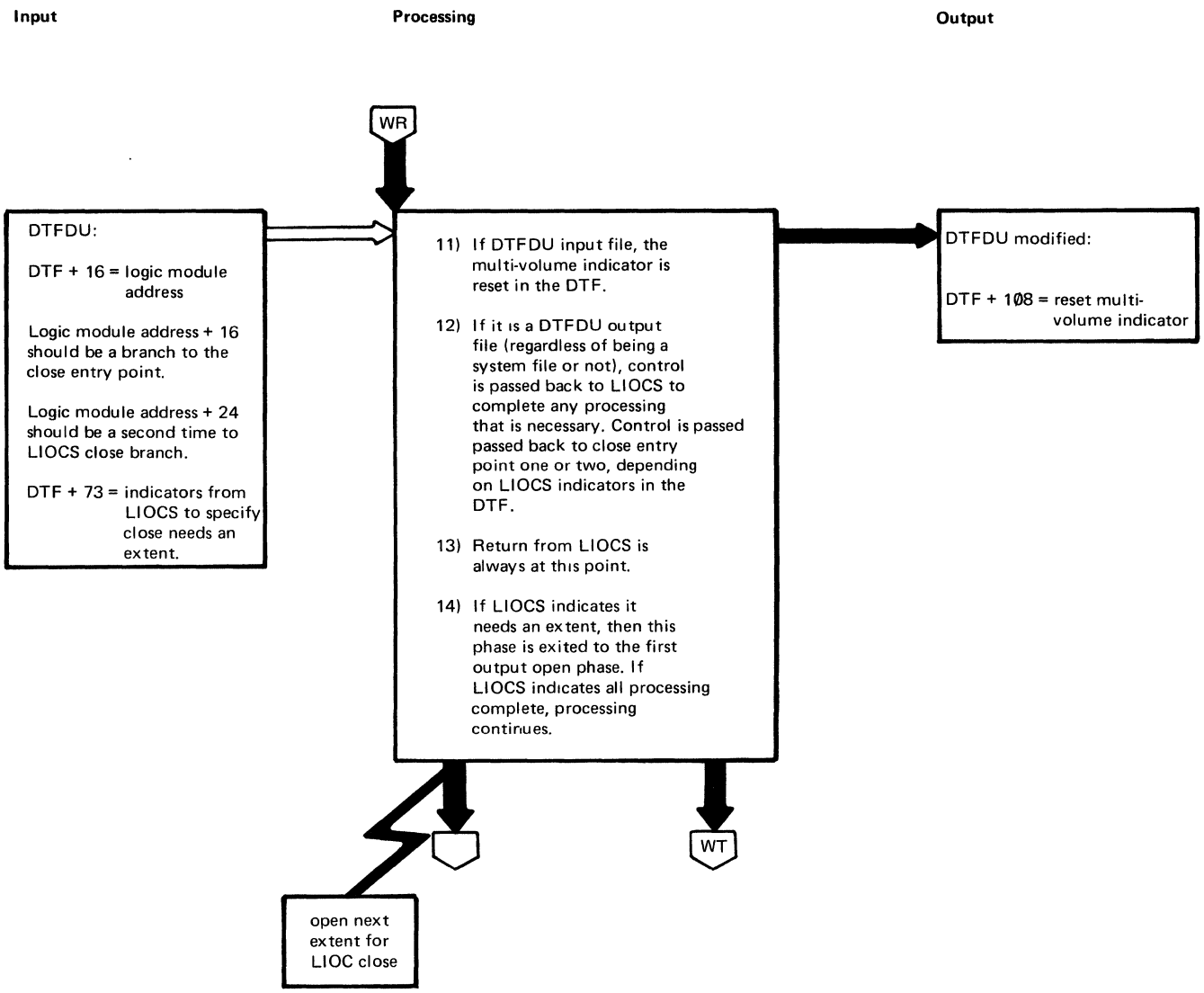
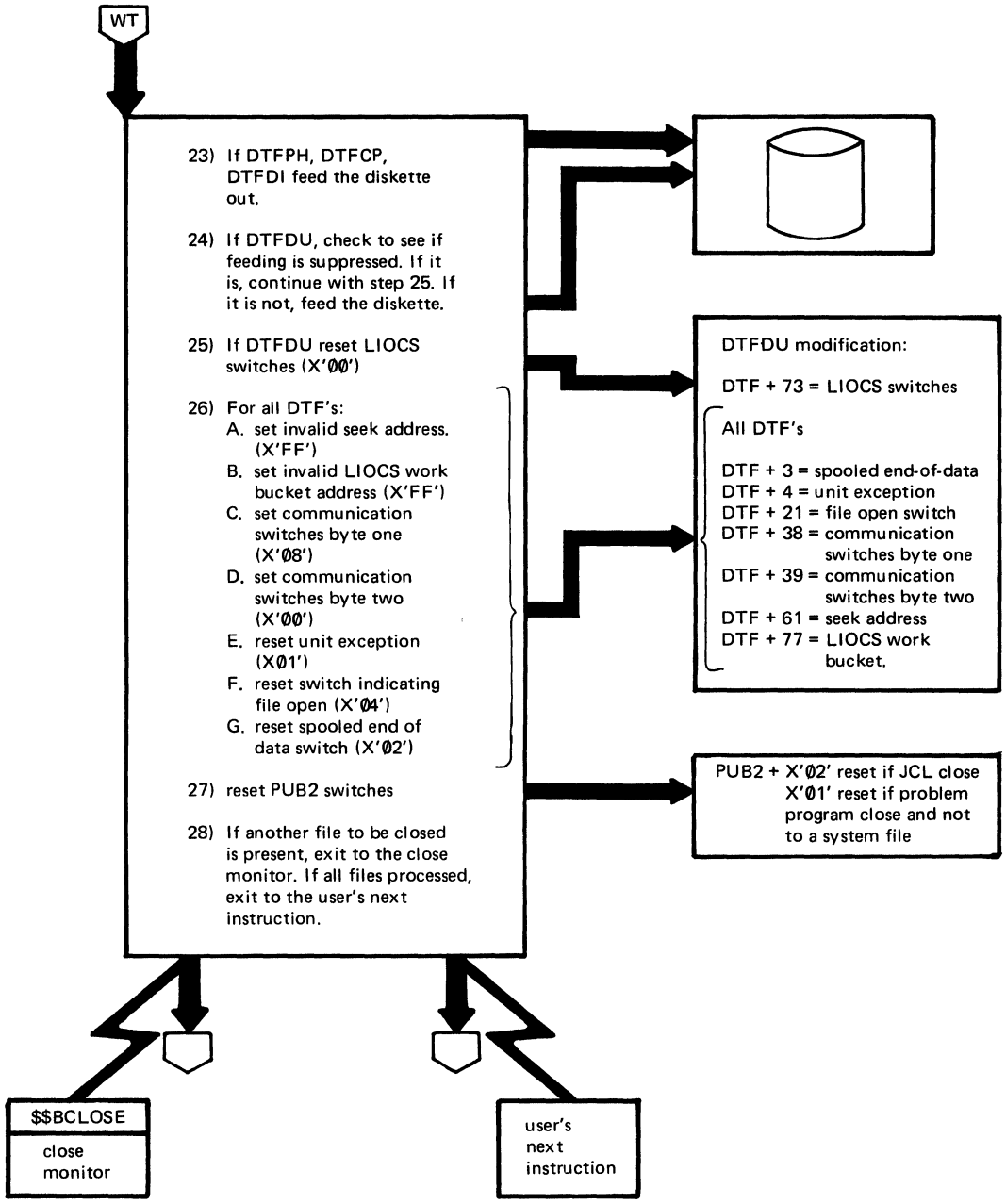


Chart WS. $$BODIO4: Diskette Clcse (Part 2 of 4)



## Chart WU. $$BODIO4: Diskette Close (Part 4 of 4)

**Input** **Processing** **Output**



| Label | Phase | Location |
|---|---|---|
| .BK2 | PRMOD | CFA4 |
| .BK3 | PRMOD | CFB4 |
| .BK4 | PRMOD | CGJ5 |
| .BK5 | PRMOD | CGJ4 |
| .MF114 | CDMOD | AMH2 |
| .P19 | PRMOD | CFA5 |
| .P21 | PRMOD | CGA1 |
| .P22 | PRMOD | CGH1 |
| .P23 | PRMOD | CFF5 |
| .P24 | PRMOD | CGF1 |
| .P27 | PRMOD | CGD1 |
| .P28 | PRMOD | CGK1 |
| .P30 | PRMOD | CGG4 |
| .P36 | PRMOD | CGA5 |
| .P37 | PRMOD | CGE5 |
| .P38 | PRMOD | CEB2 |
| .P38C | PRMOD | CDG3 |
| .P41 | PRMOD | CGH2 |
| .P42 | PRMOD | CGC4 |
| .P44 | PRMOD | CDE3 |
| .P45 | PRMOD | CGK4 |
| .P46 | PRMOD | CEE2 |
| .P47 | PRMOD | CEH5 |
| .P50 | PRMOD | CFE4 |
| .P51 | PRMOD | CFC4 |
| .P53 | PRMOD | CFH2 |
| .P53A | PRMOD | CFJ2 |
| .P54 | PRMOD | CFK2 |
| .P54A | PRMOD | CFE3 |
| .P55 | PRMOD | CGC1 |
| .P56 | PRMOD | CGJ1 |
| .P66 | PRMOD | CFK4 |
| .P67 | PRMOD | CFB5 |
| .P76 | PRMOD | CGD4 |
| .P77 | PRMOD | CGF4 |
| .P79C | PRMOD | CEG2 |
| .P81 | PRMOD | CEB5 |
| ABORTJOB | $$BOMT07 | FKD2 |
| ABORTJOB | $$BMSGWR | FLJ3 |
| ABORTJOB | $$BOSD07 | LWE2 |
| ABORTJOB | SD | LZE5 |
| ABORTJOB | $$BCCPM1 | SEK1 |
| ABORTNOT | $$BOSDW1 | MBG2 |
| ABOVEFE | $$BOUR01 | ABC2 |
| ADDSEQNO | $$BOSD07 | LWE4 |
| ADD1XTNT | $$BOSDW2 | MCJ1 |
| ADJINPUT | $$BCCPT1 | SAG1 |
| ADJNOEOF | $$BCCPT1 | SAA2 |
| AFTERSDR | $$BOMT02 | ECD3 |
| AGAIN | $$BOMT07 | FKB3 |
| AGAIN | $$BMSGWR | FLG4 |
| AGAIN | $$BCCPM1 | SEG1 |
| AJALT | $$BOCP03 | QCF3 |
| AJALT | $$BOCP11 | QDB3 |
| ALLBYPS | $$BOSD01 | LLE3 |
| ALTORM | $$BOMT07 | FHB1 |
| ANYOPEN | $$BOSDW1 | MAA3 |
| AUTOTERM | $$BOMT07 | FKB2 |
| AUTOTERM | $$BMSGWR | FLG3 |
| AUTOTERM | $$BCCPM1 | SEE2 |
| BACKSPC2 | $$BOMT03 | EEA3 |
| BEGINRTN | $$BERRTN | PGB2 |
| BEGINRTN | $$BCLOSP | SCB1 |
| BELOWFE | $$BOUR01 | ABC1 |
| BELOWFE | $$BOSIGN | LNF1 |
| BELOWFE | $$BOSDW1 | MAA4 |
| BKSP | $$BOCPT3 | RHD3 |
| BKSPACE | MTMOD | FJE2 |
| BRSWIT | $$BOSD00 | LAE4 |
| BUILDF1 | $$BOSD04 | LRA2 |
| BUILDLBL | $$BOMT04 | EFD1 |
| BUILDLBL | $$BOCPT3 | RJG1 |
| BUILDLBL | $$BCCPT1 | SAG3 |
| BYPASS | $$BOMT04 | EFD3 |
| BYPASSED | $$BOSDI1 | LEB4 |
| BYPASSX | $$BOSDI1 | LED2 |
| BYPASSX | $$BOSD01 | LLA3 |
| BYPASSX | $$BOSDW1 | MAD1 |
| BYSTDUTL | $$BCCPT1 | SAG2 |
| CALLERP | $$BCLOSP | SCD5 |
| CALLMSG | $$BOCP01 | QAB2 |
| CALLMSG1 | $$BOCP02 | QBE2 |
| CALLMSG1 | $$BOCP12 | QFF3 |
| CALOPN | $$BOCP11 | QEJ4 |
| CALOPN1 | $$BOCP02 | QBK5 |
| CALOPN1 | $$BOCP12 | QFK4 |
| CBCLOSE | $$BOSDI3 | LHD4 |
| CBDUMP | $$BJCOP1 | ELJ3 |
| CCPTAPE | $$BCCPT1 | SAB1 |
| CD25202 | $$BCLOSP | SDC1 |
| CD25402 | $$BCLOSP | SDB2 |
| CHECK | $$BOMP01 | BED5 |
| CHECK | $$BOMT02 | ECB3 |
| CHECK | $$BOSDI1 | LEG5 |
| CHECK | $$BOCPT3 | RJC1 |
| CHECK | $$BCCPT1 | SAC3 |
| CHECKDAT | $$BOMT06 | EHF4 |
| CHECKID | $$BOMT01 | EAH4 |
| CHECKID | $$BOMT02 | ECE3 |
| CHECKLL | $$BOSD08 | LXE4 |
| CHECKXNT | $$BOSD08 | LXG4 |
| CHEKXTNT | $$BOSD03 | LPH4 |
| CHKALC | $$BCMT02 | FBJ2 |
| CHKALC | $$BCMT07 | FHA4 |
| CHKFORM | $$BOMPCE | ACE1 |
| CHKIPT | $$BJCOP1 | ELG1 |
| CHKLAB | $$BOCPT1 | PAB4 |
| CHKLAB | $$BOCPT2 | PDD3 |
| CHKLP | $$BOCPT1 | RBF5 |
| CHKLP | $$BOCPT2 | REF3 |
| CHKMOR | $$BOCP01 | QAH4 |
| CHKMOR | $$BOCP02 | QBG1 |
| CHKMOR | $$BOCP11 | QDF3 |
| CHKMOR | $$BOCP12 | QFG1 |
| CHKNUMB | $$BOMPCE | ADC1 |
| CHKPCH | $$BOUP01 | AAB5 |
| CHKPRTR | $$BOUP01 | AAA4 |
| CHKRDR | $$BOUP01 | AAB1 |
| CHKTM | $$BOCPT1 | RAH3 |
| CHKTM | $$BOCPT2 | PDC3 |

| Label | Phase | Location |
|---|---|---|
| EXIT | $$BOCPT1 | RCC3 |
| EXIT | $$BOCPT2 | REC4 |
| EXIT | $$BOCPT3 | RHH3 |
| EXIT | $$BOCPT4 | RMH3 |
| EXIT | $$BCCPT1 | SBE4 |
| EXITCLOS | $$BCMT06 | FGJ3 |
| EXITLAB | $$BOCPT1 | RCB3 |
| EXITLAB | $$BOCPT3 | RHG3 |
| EXITLAB | $$BOCPT4 | RMG3 |
| EXITMON | $$BCCPT1 | SBF4 |
| EXITNLAB | $$BOCPT2 | REB5 |
| EXITNLAB | $$BOCPT3 | RHG5 |
| EXITNLAB | $$BOCPT4 | RMD4 |
| EXITNLB | $$BOCPT2 | REB4 |
| EXITNLB | $$BOCPT3 | RHG4 |
| EXITNLB | $$BOCPT4 | RME4 |
| EXITPT | $$BCLOSP | SCE5 |
| EXITRDR | $$BCLOSP | SCA5 |
| EXITRTN | $$BOMT06 | EHE3 |
| EXITUSER | $$BOSDC2 | MHJ4 |
| EXMFCM | $$BOCP03 | QCA4 |
| EXPDATE | $$BOMT03 | EDJ1 |
| EXPDATE | $$BOCPT3 | RJE1 |
| EXPIRCHK | $$BOCPT3 | RHG1 |
| EXPIRED | $$BOSDO3 | LQG2 |
| EXPIRED | $$BOSDO8 | LXF3 |
| EXTENT5 | $$BOSDW2 | MCC3 |
| FCHMSG | $$BCEOV1 | EMG2 |
| FCHMSG | $$BOSDO0 | LAJ2 |
| FEOVRET | $$BOSDO5 | LTG4 |
| FETCHALT | $$BCMT01 | FAK2 |
| FETCHIT | $$BOCPT3 | RGC3 |
| FETCHIT | $$BOCPT4 | RKH1 |
| FETCH07 | $$BOMT01 | EAK3 |
| FETCH07 | $$BOMT01 | EBK2 |
| FILEADD2 | $$BOSDO6 | LUA4 |
| FILEMARK | $$BOSDO6 | LVA1 |
| FILEOVLP | $$BOSDO3 | LQC2 |
| FILEPROT | $$BOSDI4 | LJB2 |
| FILEPROT | $$BOSDO1 | LMA3 |
| FILEPROT | $$BOSDO5 | LTH4 |
| FILEPROT | $$BOSDO9 | LYG3 |
| FILETYPE | $$BOSDC1 | MFB3 |
| FINDLAST | $$BOSDO1 | LLD4 |
| FINDSPOT | $$BOSDO4 | LRD2 |
| FINDSPOT | $$BOSDO5 | LSA5 |
| FINDSPOT | $$BOSDW2 | MCA5 |
| FINDSPT | $$BOSDW2 | MCG1 |
| FINDXTNT | $$BOSDW3 | MEB3 |
| FINISH | $$BOMRCE | ADH4 |
| FIRSTCD | $$BOMRCE | ACJ1 |
| FIXED | $$BOSDO1 | LMA2 |
| FORMAT3 | $$BCSDO5 | LSH5 |
| FORMAT3 | $$BOSDW2 | MCF5 |
| FORTUNL | $$BOCPT3 | RHC4 |
| FRMTBL | $$BCEOV1 | EMJ1 |
| FRSTINST | $$BOMR01 | BEB1 |
| FRSTINST | $$BOMR01 | BEB5 |
| FRSTINST | $$BMMR20 | BFB2 |
| FSTM | $$BOCPT4 | RLD5 |
| GETDATE | $$BOMT03 | EDC3 |
| GETDEFLT | $$BCMT03 | EDG1 |
| GETDSPLY | $$BOSDO7 | LWF2 |
| GETNEXT | $$BOSDI1 | LEC5 |
| GETNEXT | $$BODQUE | MJJ2 |
| GETOPN | $$BOCP02 | QBK4 |
| GETOPN | $$BOCP11 | QEG4 |
| GETPOINT | $$BOSDO5 | LSE4 |
| GETPOINT | $$BOSDW2 | MCA4 |
| GETPUB | $$BOUR01 | AAK2 |
| GETPUB | MTMOD | FJG5 |
| GOSENS | $$BOCPT2 | RDF5 |
| GOTOIO | $$BJCOPT | EKH1 |
| HDRBLD | $$BOCPT3 | RJB1 |
| HDRCHK | $$BOCPT4 | RMB1 |
| HDRCHKX | $$BOCPT4 | RMD2 |
| HDRCHK2 | $$BOCPT4 | RMC1 |
| HDRMSG | $$BOCPT4 | RMG2 |
| HDRWRITE | $$BOCPT3 | RJE2 |
| HDR1CHK | $$BOCPT1 | RAC4 |
| HEADERSW | $$BOSDO6 | LVC1 |
| IFOPENFD | $$BOSDO3 | LPC3 |
| IGNORE | $$BOUR01 | ABD3 |
| IJBWCONT | SDMODW | KDK3 |
| IJCBXXXX | DTFCN | AQC3 |
| IJCEXXX | CDMOD | AKG4 |
| IJCMBXXX | CDMOD | AED5 |
| IJCMBXXX | CDMOD | AEE5 |
| IJCMCXXX | CDMOD | AGB1 |
| IJCMDXXX | CDMOD | ALB4 |
| IJCMDXXX | CDMOD | ALB5 |
| IJCMEXXX | CDMOD | AGD3 |
| IJCMRXXX | CDMOD | ALC4 |
| IJCPKXXX | CDMOD | AEC4 |
| IJCPNXXX | CDMOD | ALD1 |
| IJCPYXXX | CDMOD | AKA4 |
| IJCPZXXX | CDMOD | AJF3 |
| IJCRAXXX | CDMOD | AFC2 |
| IJCRBXXX | CDMOD | AFF1 |
| IJCRFXXX | CDMOD | AGA5 |
| IJCRTXXX | CDMOD | AJH3 |
| IJCRVXXX | CDMOD | AFE1 |
| IJCR9XXX | CDMOD | AED2 |
| IJCR9XXX | CDMOD | AED3 |
| IJCXXXXX | CDMOD | AHB4 |
| IJC00XXX | CDMOD | AFB4 |
| IJC30XXX | CDMOD | AHA1 |
| IJC30XXX | CDMOD | AMA1 |
| IJC33XXX | CDMOD | AME1 |
| IJC34XXX | CDMOD | AHD1 |
| IJC34XXX | CDMOD | AMG1 |
| IJC35XXX | CDMOD | AME4 |
| IJC36XXX | CDMOD | AHE1 |
| IJC38XXX | CDMOD | AHH2 |
| IJC41XXX | CDMOD | AHB3 |
| IJC42XXX | CDMOD | AHD2 |
| IJC43XXX | CDMOD | AHE3 |
| IJC44XXX | CDMOD | AHH3 |
| IJC50XXX | CDMOD | AME3 |
| IJC96XXX | CDMOD | AHD3 |
| IJC96XXX | CDMOD | AHG1 |
| IJC97XXX | CDMOD | AHE4 |
| IJC98XXX | CDMOD | AHC5 |
| IJDPKXXX | PRMOD | CDD3 |
| IJDPKXXX | PRMOD | CDF3 |
| IJDPMXXX | PRMOD | CED2 |
| IJDPRXXX | PRMOD | CFG2 |
| IJDPUXXX | PRMOD | CGB2 |

| Label | Phase | Location |
|---|---|---|
| IJF RCK | MTMOD | DVB1 |
| IJF REL | MTMOD | DJH1 |
| IJF REL | MTMOD | DJH3 |
| IJF REL | MTMOD | DJJ2 |
| IJF REST | MTMOD | DWF2 |
| IJF SKI | MTMOD | DPE3 |
| IJF SKIP | MTMOD | DPJ2 |
| IJF SKIP | MTMOD | DSH3 |
| IJF SKIP | MTMOD | DUA1 |
| IJF TEST | MTMOD | DDH1 |
| IJF TR | MTMOD | DVB4 |
| IJF TRI | MTMOD | DVD4 |
| IJF TRR | MTMOD | DVE4 |
| IJF TRU | MTMOD | DJB3 |
| IJF TRU | MTMOD | DJB5 |
| IJF VER1 | MTMOD | DVF2 |
| IJF VER2 | MTMOD | DVH1 |
| IJF WMV | MTMOD | DWD2 |
| IJF WORK | MTMOD | DWB1 |
| IJF WORK | MTMOD | DWK1 |
| IJF WRK | MTMOD | DWB2 |
| IJF WRK1 | MTMOD | DWB3 |
| IJF WSR | MTMOD | DWE2 |
| IJF YSOT | MTMOD | DPB1 |
| IJFFBCKW | MTMOD | DLD4 |
| IJFFCON | MTMOD | DLJ3 |
| IJFFCSVC | MTMOD | DNG3 |
| IJFFIGN | MTMOD | DLG4 |
| IJFFRCK | MTMOD | DNB1 |
| IJFFRCT | MTMOD | DLC3 |
| IJFFSTD | MTMOD | DLF3 |
| IJFFTET | MTMOD | DLG3 |
| IJFFVRS1 | MTMOD | DNF1 |
| IJFFVRS2 | MTMOD | DNE3 |
| IJFNOPK | MTMOD | DTB3 |
| IJFNRDWR | MTMOD | DWB4 |
| IJFRDCBN | MTMOD | DDF4 |
| IJFREX2 | MTMOD | DQG5 |
| IJFRNLCO | MTMOD | DGD3 |
| IJFRNOPD | MTMOD | DDF3 |
| IJFRNOPR | MTMOD | DQF5 |
| IJFRYSPD | MTMOD | DDE5 |
| IJFSCNTS | MTMOD | DBA5 |
| IJFSCNTT | MTMOD | DEE3 |
| IJFSCNTU | MTMOD | DEH3 |
| IJFSCNTV | MTMOD | DKB2 |
| IJFSEOVS | MTMOD | DRC3 |
| IJFSEOVT | MTMOD | DRH3 |
| IJFSEOVU | MTMOD | DRJ3 |
| IJFSEOVV | MTMOD | DRK3 |
| IJFSEOVW | MTMOD | DKB3 |
| IJFSGETS | MTMOD | DDB2 |
| IJFSGETT | MTMOD | DDC1 |
| IJFSGETU | MTMOD | DEA1 |
| IJFSGETV | MTMOD | DEA2 |
| IJFSMOVS | MTMOD | DEJ2 |
| IJFSMOVT | MTMOD | DEA3 |
| IJFSPUTC | MTMOD | DGC3 |
| IJFSPUTS | MTMOD | DHB1 |
| IJFSPUTT | MTMOD | DHC1 |
| IJFSPUTU | MTMOD | DHF4 |
| IJFSPUTV | MTMOD | DHD3 |
| IJFSSHF1 | MTMOD | DEH1 |
| IJFSSHF2 | MTMOD | DEH2 |
| IJFSSHIF | MTMOD | DEE1 |
| IJFSSQER | MTMOD | DEA4 |

| Label | Phase | Location |
|---|---|---|
| IJFWCHEK | MTMOD | DAB1 |
| IJFWCTL | MTMOD | DBH3 |
| IJFWEXCP | MTMOD | DWF5 |
| IJFWEXT | MTMOD | DFJ2 |
| IJFWFND | MTMOD | DFF2 |
| IJFWFSF | MTMOD | DWB5 |
| IJFWIGNR | MTMOD | DAG5 |
| IJFWNOTE | MTMOD | DFJ5 |
| IJFWNRXT | MTMOD | DAD3 |
| IJFWOUT | MTMOD | DAE2 |
| IJFWPNTR | MTMOD | DFB2 |
| IJFWPNTS | MTMOD | DFB3 |
| IJFWPNTW | MTMOD | DFB1 |
| IJFWRIT | MTMOD | DJB2 |
| IJFWSKIP | MTMOD | DAH4 |
| IJFWSTOR | MTMOD | DWE4 |
| IJFWTEST | MTMOD | DFE1 |
| IJFXFOR | MTMOD | DMG1 |
| IJFXFORW | MTMOD | DMA2 |
| IJFXFWD | MTMOD | DKD4 |
| IJFXIGN | MTMOD | DMC3 |
| IJFXLOOP | MTMOD | DME1 |
| IJFXLTST | MTMOD | DMD1 |
| IJFXPAD | MTMOD | DMF1 |
| IJFXRCT | MTMOD | DLC5 |
| IJFXRST | MTMOD | DKG4 |
| IJFXTDCH | MTMOD | DMA3 |
| IJFXTRA | MTMOD | DKB4 |
| IJG ERR1 | SDMODFO | GDF5 |
| IJG UNIO | SDMODFO | GDD5 |
| IJG WAIT | SDMODFO | GDB5 |
| IJGBADD2 | SDMODFU | GKK2 |
| IJGBCHCK | SDMODFU | GMD2 |
| IJGBDATK | SDMODFU | GLH3 |
| IJGBDEBL | SDMODFU | GKC1 |
| IJGBDECL | SDMODFU | GLC5 |
| IJGBDECR | SDMODFU | GLA4 |
| IJGBDONE | SDMODFU | GKE3 |
| IJGBEOFT | SDMODFU | GLE3 |
| IJGBEXT | SDMODFU | GKH3 |
| IJGBFACT | SDMODFU | GKG2 |
| IJGBFAC1 | SDMODFU | GMC1 |
| IJGBFREE | SDMODFU | GMB3 |
| IJGBGENM | SDMODFU | GLB5 |
| IJGBGET | SDMODFU | GKB1 |
| IJGBGETA | SDMODFU | GKH4 |
| IJGBGETD | SDMODFU | GKJ4 |
| IJGBHOLD | SDMODFU | GKE4 |
| IJGBIORU | SDMODFU | GKG1 |
| IJGBKETH | SDMODFU | GLG2 |
| IJGBLOOP | SDMODFU | GKD3 |
| IJGBNXRC | SDMODFU | GMH3 |
| IJGBPUT | SDMODFU | GNB1 |
| IJGBP460 | SDMODVO | HEJ2 |
| IJGBRDER | SDMODFU | GLF4 |
| IJGBRD1 | SDMODFU | GKE2 |
| IJGBRD3 | SDMODFU | GKC3 |
| IJGBREAD | SDMODFU | GKB4 |
| IJGBRELA | SDMODFU | GKF2 |
| IJGBRESI | SDMODFU | GLD5 |
| IJGBRET | SDMODFU | GLJ4 |
| IJGBRTRY | SDMODFU | GLJ5 |
| IJGBSKIP | SDMODFU | GMA1 |
| IJGBSWOF | SDMODFU | GLA3 |
| IJGBSWO3 | SDMODFU | GKF4 |
| IJGBTEST | SDMODFU | GKD2 |

| Label | Phase | Location |
|---|---|---|
| IJGGSW03 | SDMODFU | GGG4 |
| IJGGSW04 | SDMODFU | GGD4 |
| IJGGSW06 | SDMODFU | GGH4 |
| IJGGTEST | SDMODFU | GGE1 |
| IJGGTOM | SDMODFU | GHJ1 |
| IJGGTST2 | SDMODFU | GHA5 |
| IJGGUSER | SDMODFU | GHE3 |
| IJGGWAIT | SDMODFU | GHG2 |
| IJGGWLPT | SDMODFU | GHD3 |
| IJGGWRT | SDMODFU | GHB2 |
| IJGGWRT1 | SDMODFU | GHB1 |
| IJGQUERR | SDMODVU | HNF5 |
| IJGQUEXT | SDMODVU | HQH2 |
| IJGQUFRE | SDMODVU | HLB5 |
| IJGQUNR | SDMODVU | HQC3 |
| IJGQUPT | SDMODVU | HQD3 |
| IJGQUSER | SDMODVU | HQF3 |
| IJGQUSKB | SDMODVU | HQC4 |
| IJGQU000 | SDMODVU | HJC1 |
| IJGQU010 | SDMODVU | HJE3 |
| IJGQU011 | SDMODVU | HKA1 |
| IJGQU012 | SDMODVU | HKD1 |
| IJGQU013 | SDMODVU | HKG1 |
| IJGQU015 | SDMODVU | HKA2 |
| IJGQU016 | SDMODVU | HKC1 |
| IJGQU017 | SDMODVU | HKG5 |
| IJGQU018 | SDMODVU | HKG4 |
| IJGQU020 | SDMODVU | HKF2 |
| IJGQU021 | SDMODVU | HQE3 |
| IJGQU023 | SDMODVU | HKC3 |
| IJGQU024 | SDMODVU | HKD2 |
| IJGQU030 | SDMODVU | HJE1 |
| IJGQU10K | SDMODVU | HPD4 |
| IJGQU101 | SDMODVU | HNJ3 |
| IJGQU103 | SDMODVU | HNB4 |
| IJGQU104 | SDMODVU | HNH5 |
| IJGQU105 | SDMODVU | HNJ5 |
| IJGQU106 | SDMODVU | HNG4 |
| IJGQU107 | SDMODVU | HPA4 |
| IJGQU108 | SDMODVU | HNC5 |
| IJGQU108 | SDMODVU | HNJ4 |
| IJGQU109 | SDMODVU | HPF5 |
| IJGQU110 | SDMODVU | HPA1 |
| IJGQU111 | SDMODVU | HQB1 |
| IJGQU112 | SDMODVU | HPE1 |
| IJGQU114 | SDMODVU | HPB2 |
| IJGQU115 | SDMODVU | HNG3 |
| IJGQU120 | SDMODVU | HQF1 |
| IJGQU121 | SDMODVU | HQB2 |
| IJGQU121 | SDMODVU | HQD2 |
| IJGQU123 | SDMODVU | HMA5 |
| IJGQU124 | SDMODVU | HQE2 |
| IJGQU156 | SDMODVU | HQB3 |
| IJGQU400 | SDMODVU | HNE1 |
| IJGQU401 | SDMODVU | HNA3 |
| IJGQU440 | SDMODVU | HPB3 |
| IJGQU490 | SDMODVU | HMC4 |
| IJGRELS | SDMODFU | GNB4 |
| IJGSGSKI | SDMODVI | HAG2 |
| IJGSG010 | SDMODVI | HBE4 |
| IJGSG030 | SDMODVI | HAC1 |
| IJGSG400 | SDMODVI | HAB2 |
| IJGSG410 | SDMODVI | HBA3 |
| IJGSG420 | SDMODVI | HBC3 |
| IJGSG430 | SDMODVI | HBE3 |
| IJGSG430 | SDMODVI | HBF3 |
| IJGSG500 | SDMODVI | HBB4 |
| IJGSG600 | SDMODVI | HBD3 |
| IJGSG710 | SDMODVI | HAH4 |
| IJGSG800 | SDMODVI | HBD5 |
| IJGSG910 | SDMODVI | HBK4 |
| IJGSPLYS | SDMODVO | HGB4 |
| IJGSP030 | SDMODVO | HDH1 |
| IJGSP040 | SDMODVO | HDC2 |
| IJGSP040 | SDMODVO | HGA1 |
| IJGSP045 | SDMODVO | HGE1 |
| IJGSP050 | SDMODVO | HGF1 |
| IJGSP065 | SDMODVO | HEB4 |
| IJGSP070 | SDMODVO | HGH1 |
| IJGSP090 | SDMODVO | HEB3 |
| IJGSP110 | SDMODVI | HAF5 |
| IJGSP110 | SDMODVO | HED3 |
| IJGSP120 | SDMODVO | HGG1 |
| IJGSP130 | SDMODVO | HEA2 |
| IJGSP140 | SDMODVO | HEE2 |
| IJGSP150 | SDMODVO | HFF2 |
| IJGSP200 | SDMODVO | HFD5 |
| IJGSP202 | SDMODVO | HFF5 |
| IJGSP203 | SDMODVO | HFG5 |
| IJGSP205 | SDMODVO | HFJ5 |
| IJGSP210 | SDMODVO | HGA3 |
| IJGSP220 | SDMODVO | HGJ3 |
| IJGSP240 | SDMODVO | HGK3 |
| IJGSP260 | SDMODVO | HGG4 |
| IJGSP265 | SDMODVO | HGJ4 |
| IJGSP270 | SDMODVO | HGK4 |
| IJGSP280 | SDMODVO | HGA5 |
| IJGUGOPT | SDMODUI | JAH1 |
| IJGUGUNI | SDMODUI | JAG1 |
| IJGUG040 | SDMODUI | JAE1 |
| IJGUG050 | SDMODUI | JAF1 |
| IJGUG060 | SDMODUI | JAD3 |
| IJGUG070 | SDMODUI | JAF3 |
| IJGUG080 | SDMODUI | JAG3 |
| IJGUG100 | SDMODUI | JAH3 |
| IJGUG170 | SDMODUI | JAE3 |
| IJGUG210 | SDMODUI | JAB1 |
| IJGUG220 | SDMODUI | JBC4 |
| IJGUG240 | SDMODUI | JBC5 |
| IJGUG250 | SDMODUI | JBB3 |
| IJGUG260 | SDMODUI | JBC3 |
| IJGUG261 | SDMODUI | JBF3 |
| IJGUG290 | SDMODUI | JBH3 |
| IJGUG300 | SDMODUI | JAF2 |
| IJGUG310 | SDMODUI | JAC4 |
| IJGUG320 | SDMODUI | JBA4 |
| IJGUG330 | SDMODUI | JBB1 |
| IJGUPUNI | SDMODUO | JDD1 |
| IJGUP040 | SDMODUO | JCD1 |
| IJGUP060 | SDMODUO | JCJ1 |
| IJGUP060 | SDMODUO | JCK1 |
| IJGUP070 | SDMODUO | JCC1 |
| IJGUP080 | SDMODUO | JCB4 |
| IJGUP100 | SDMODUO | JCC4 |
| IJGUP120 | SDMODUO | JDB2 |
| IJGUP120 | SDMODUO | JDC4 |
| IJGUP130 | SDMODUO | JDC2 |
| IJGUP150 | SDMODUO | JCG2 |
| IJGUP160 | SDMODUO | JDB5 |
| IJGUP200 | SDMODUO | JDB3 |
| IJGUP220 | SDMODUO | JCB5 |
| IJGUP230 | SDMODUO | JCD5 |

| Label | Phase | Location |
|---|---|---|
| IJGVU040 | SDMODVU | HJJ1 |
| IJGVU050 | SDMODVU | HJK1 |
| IJGVU065 | SDMODVU | HJC2 |
| IJGVU070 | SDMODVU | HJG2 |
| IJGVU075 | SDMODVU | HJH2 |
| IJGVU100 | SDMODVU | HJD3 |
| IJGVU140 | SDMODVU | HJB3 |
| IJGVU150 | SDMODVU | HJJ3 |
| IJGVU151 | SDMODVU | HJA5 |
| IJGVU152 | SDMODVU | HJE5 |
| IJGVU153 | SDMODVU | HJB5 |
| IJGVU156 | SDMODVU | HLB2 |
| IJGVU158 | SDMODVU | HLF1 |
| IJGVU159 | SDMODVU | HLF2 |
| IJGVU160 | SDMODVU | HLB1 |
| IJGVU161 | SDMODVU | HLG1 |
| IJGVU220 | SDMODVU | HKB4 |
| IJGVU240 | SDMODVU | HKB5 |
| IJGVU250 | SDMODVU | HKG3 |
| IJGVU250 | SDMODVU | HNF1 |
| IJGVU260 | SDMODVU | HKH3 |
| IJGVU270 | SDMODVU | HKJ3 |
| IJGVU290 | SDMODVU | HMB1 |
| IJGVU292 | SDMODVU | HMC1 |
| IJGVU300 | SDMODVU | HMD1 |
| IJGVU310 | SDMODVU | HMF1 |
| IJGVU330 | SDMODVU | HMJ1 |
| IJGVU340 | SDMODVU | HMD3 |
| IJGVU360 | SDMODVU | HMA4 |
| IJGVU370 | SDMODVU | HMF2 |
| IJGVU380 | SDMODVU | HMG2 |
| IJGVU400 | SDMODVU | HNB1 |
| IJGVU410 | SDMODVU | HNG1 |
| IJGVU415 | SDMODVU | HNJ1 |
| IJGVU420 | SDMODVU | HNK1 |
| IJGVU430 | SDMODVU | HNH2 |
| IJGVU440 | SDMODVU | HNA2 |
| IJGVU442 | SDMODVU | HPH3 |
| IJGVU444 | SDMODVU | HPJ3 |
| IJGVU450 | SDMODVU | HPK2 |
| IJGVU460 | SDMODVU | HQB5 |
| IJGVU470 | SDMODVU | HPJ2 |
| IJGVU490 | SDMODVU | HMD4 |
| IJGVU510 | SDMODVU | HME4 |
| IJGVU520 | SDMODVU | HRB1 |
| IJGVC70 | SDMODVI | HAA4 |
| IJGVC80 | SDMODVI | HAB4 |
| IJGWBLD | SDMODW | KEF2 |
| IJGWCHECK | SDMODW | KEB1 |
| IJGWCKRD | SDMODW | KEA2 |
| IJGWCLOS | SDMODW | KDB3 |
| IJGWCOMP | SDMODW | KEH2 |
| IJGWCTL | SDMODW | KGB3 |
| IJGWDECR | SDMODW | KBB5 |
| IJGWETRY | SDMODW | KCJ2 |
| IJGWFOPN | SDMODW | KFH2 |
| IJGWFRE | SDMODW | KEA3 |
| IJGWFREE | SDMODW | KGB2 |
| IJGWFTST | SDMODW | KEH4 |
| IJGWHOLD | SDMODW | KAB3 |
| IJGWIGN | SDMODW | KEF3 |
| IJGWINST | SDMODW | KBC4 |
| IJGWLCTY | SDMODW | KFG1 |
| IJGWLOAD | SDMODW | KBD2 |
| IJGWLREG | SDMODW | KDH1 |
| IJGWNOTE | SDMODW | KFB1 |
| IJGWOPT | SDMODW | KEG1 |
| IJGWPASS | SDMODW | KFF3 |
| IJGWPNT | SDMODW | KFB2 |
| IJGWPNTR | SDMODW | KFB3 |
| IJGWPNTS | SDMODW | KGB1 |
| IJGWRDSW | SDMODW | KEE2 |
| IJGWREAD | SDMODW | KAB1 |
| IJGWRITE | SDMODW | KBB1 |
| IJGWRW | SDMODW | KDB1 |
| IJGWSLD | SDMODW | KCB3 |
| IJGWSOFF | SDMODW | KCB1 |
| IJGWSSET | SDMODW | KBG4 |
| IJGWSSTR | SDMODW | KCF3 |
| IJGWSTAR | SDMODW | KAF1 |
| IJGWSTCH | SDMODW | KCC1 |
| IJGWSVCO | SDMODW | KAK2 |
| IJGWTEST | SDMODW | KBC5 |
| IJGWTLIM | SDMODW | KFF2 |
| IJGWUND | SDMODW | KDE2 |
| IJGWUNRC | SDMODW | KEB2 |
| IJGWUPOF | SDMODW | KEJ4 |
| IJGWUSER | SDMODW | KEJ1 |
| IJGWWRTE | SDMODW | KBC1 |
| IJGWWTER | SDMODW | KEF1 |
| IJGWWXIT | SDMODW | KBG5 |
| IJG1XXX | DIMOD | PCF1 |
| IJJASCNQ | CPMOD | NEE1 |
| IJJASCN1 | CPMOD | NEG5 |
| IJJASCN2 | CPMOD | NEK5 |
| IJJCAL2 | CPMOD | NCE5 |
| IJJCAL2 | CPMOD | NEF5 |
| IJJCAL2 | CPMOD | NGF5 |
| IJJCEXEX | CPMOD | NCJ1 |
| IJJCFOND | CPMOD | NCD3 |
| IJJCFOND | CPMOD | NEH1 |
| IJJCFOND | CPMOD | NGG1 |
| IJJCNDSK | CPMOD | NBF2 |
| IJJCNOEK | CPMOD | NBE3 |
| IJJCPCAL | CPMOD | NCK4 |
| IJJCPCAL | CPMOD | NED3 |
| IJJCPCAL | CPMOD | NGC3 |
| IJJCPDSK | CPMOD | NBE1 |
| IJJCPDSK | CPMOD | NDB1 |
| IJJCPDSK | CPMOD | NFB1 |
| IJJCPDSK | CPMOD | NHB2 |
| IJJCPEND | CPMOD | NAJ2 |
| IJJCPEOV | CPMOD | NCE2 |
| IJJCPFA3 | CPMOD | NDC5 |
| IJJCPFA3 | CPMOD | NFG1 |
| IJJCPFA3 | CPMOD | NHG2 |
| IJJCPF02 | CPMOD | NDE1 |
| IJJCPF02 | CPMOD | NFE1 |
| IJJCPF02 | CPMOD | NHE2 |
| IJJCPF03 | CPMOD | NDB5 |
| IJJCPF03 | CPMOD | NFF1 |
| IJJCPF03 | CPMOD | NHF2 |
| IJJCPF04 | CPMOD | NDF5 |
| IJJCPF04 | CPMOD | NFG2 |
| IJJCPF04 | CPMOD | NHG3 |
| IJJCPF13 | CPMOD | NAH4 |
| IJJCPF13 | CPMOD | NBJ1 |
| IJJCPF14 | CPMOD | NAF5 |
| IJJCPF14 | CPMOD | NBE2 |
| IJJCPGP | CPMOD | NAB2 |
| IJJCPGP | CPMOD | NCC1 |
| IJJCPGP | CPMOD | NEC1 |

| Label | Phase | Location |
|---|---|---|
| IJME3XXX | ORMOD | BMA4 |
| IJME4 | DRMOD | CAD3 |
| IJME4XXX | ORMOD | BMC4 |
| IJME5 | DRMOD | CAE3 |
| IJME6 | DRMOD | CAF3 |
| IJME7 | DRMOD | CAG3 |
| IJME8 | DRMOD | CAE4 |
| IJME9 | DRMOD | CAG4 |
| IJMGBXXX | ORMOD | BJE2 |
| IJMGOXXX | ORMOD | BSE3 |
| IJMM2XXX | ORMOD | BSF4 |
| IJMM3XXX | ORMOD | BSF3 |
| IJMM4XXX | ORMOD | BSG3 |
| IJMOGXXX | ORMOD | BHF1 |
| IJMRD | DRMOD | CAB5 |
| IJMRDXXX | ORMOD | BPB2 |
| IJMRGXXX | ORMOD | BTA2 |
| IJMRLXXX | ORMOD | BRF2 |
| IJMRTXXX | ORMOD | BSK5 |
| IJMRTXXX | ORMOD | BTA4 |
| IJMRUXXX | ORMOD | BTA3 |
| IJMRXXXX | ORMOD | BTC4 |
| IJMRZXXX | ORMOD | BRF1 |
| IJMR1XXX | ORMOD | BRG2 |
| IJMR3XXX | ORMOD | BRH2 |
| IJMR4XXX | ORMOD | BRH3 |
| IJMR5XXX | ORMOD | BRB1 |
| IJMR8XXX | ORMOD | BRA2 |
| IJMR9XXX | ORMOD | BRD2 |
| IJMSAXXX | ORMOD | BJJ1 |
| IJMSAXXX | ORMOD | BUB2 |
| IJMSBXXX | ORMOD | BHB4 |
| IJMSBXXX | ORMOD | BJF3 |
| IJMSBXXX | ORMOD | BUA3 |
| IJMSCXXX | ORMOD | BHG4 |
| IJMSCXXX | ORMOD | BJJ3 |
| IJMSCXXX | ORMOD | BUE3 |
| IJMSKXXX | ORMOD | BGB3 |
| IJMSO | DRMOD | CBB1 |
| IJMSQXXX | ORMOD | BJC4 |
| IJMSQXXX | ORMOD | BMF5 |
| IJMSQXXX | ORMOD | BSA3 |
| IJMS0 | DRMOD | CBC1 |
| IJMS1 | DRMOD | CBD1 |
| IJMS2 | DRMOD | CBF1 |
| IJMS3 | DRMOD | CBH1 |
| IJMS5 | DRMOD | CBD2 |
| IJMS6 | DRMOD | CBA3 |
| IJMS7 | DRMOD | CBG3 |
| IJMS8 | DRMOD | CBH3 |
| IJMS9 | DRMOD | CBJ3 |
| IJMTAXXX | ORMOD | BKB1 |
| IJMTAXXX | ORMOD | BNB1 |
| IJMTAXXX | ORMOD | BSB4 |
| IJMTBXXX | ORMOD | BKF1 |
| IJMTBXXX | ORMOD | BNF1 |
| IJMTBXXX | ORMOD | BSH3 |
| IJMTCXXX | ORMOD | BKJ1 |
| IJMTCXXX | ORMOD | BNJ1 |
| IJMTDXXX | ORMOD | BKK1 |
| IJMTDXXX | ORMOD | BNK1 |
| IJMTEXXX | ORMOD | BKA2 |
| IJMTEXXX | ORMOD | BNB3 |
| IJMTFXXX | ORMOD | BKC2 |
| IJMTFXXX | ORMOD | BNC4 |
| IJMTFXXX | ORMOD | BTE1 |
| IJMTIXXX | ORMOD | BLJ5 |
| IJMTJXXX | ORMOD | BJF1 |
| IJMTKXXX | ORMOD | BJC1 |
| IJMTLXXX | ORMOD | BJD1 |
| IJMTMXXX | ORMOD | BKB2 |
| IJMTNXXX | ORMOD | BLC3 |
| IJMTOXXX | ORMOD | BJH2 |
| IJMTOXXX | ORMOD | BLG3 |
| IJMTOXXX | ORMOD | BME4 |
| IJMTOXXX | ORMOD | BSE1 |
| IJMT1XXX | ORMOD | BPC3 |
| IJMT2XXX | ORMOD | BPD3 |
| IJMT3XXX | ORMOD | BPE3 |
| IJMT5XXX | ORMOD | BGH4 |
| IJMT6XXX | ORMOD | BGE3 |
| IJMT7XXX | ORMOD | BGD4 |
| IJMT8XXX | ORMOD | BMD4 |
| IJMT8XXX | ORMOD | BSD1 |
| IJMT9XXX | ORMOD | BKA1 |
| IJMT9XXX | ORMOD | BNA1 |
| IJMT9XXX | ORMOD | BSB3 |
| IJMUAXXX | ORMOD | BTK1 |
| IJMUBXXX | ORMOD | BTJ1 |
| IJMUCXXX | ORMOD | BTC3 |
| IJMUSXXX | ORMOD | BTH1 |
| IJMUTXXX | ORMOD | BTD3 |
| IJMUUXXX | ORMOD | BTE3 |
| IJMUVXXX | ORMOD | BTF3 |
| IJMUWXXX | ORMOD | BTG3 |
| IJMWEXXX | ORMOD | BSJ3 |
| IJMWT | DRMOD | CBB5 |
| IJMWTXXX | ORMOD | BSB1 |
| IJMWWXXX | ORMOD | BSK3 |
| IJMW1 | DRMOD | CBC5 |
| IJMW2 | DRMOD | CBD5 |
| IJMW3 | DRMOD | CBF5 |
| IJMZBXXX | ORMOD | BLD5 |
| IJMZCXXX | ORMOD | BLB5 |
| IJMZDXXX | ORMOD | BLF5 |
| IJMZEXXX | ORMOD | BLB4 |
| IJMZJXXX | ORMOD | BLH1 |
| IJMZKXXX | ORMOD | BLJ1 |
| IJMZLXXX | ORMOD | BLK2 |
| IJMZQXXX | ORMOD | BGF5 |
| IJMZQXXX | ORMOD | BGJ5 |
| IJM10XXX | ORMOD | BSJ1 |
| IJM2MXXX | ORMOD | BHC3 |
| IJOPEN | $$BOSD04 | LPE1 |
| IJPTXXXX | DTFCN | APC3 |
| IJSG720 | SDMODVI | HAK5 |
| IJUADONE | MRMOD | BAF2 |
| IJUAGAIN | MRMOD | BDD3 |
| IJUCALLW | MRMOD | BAG5 |
| IJUCHECK | MRMOD | BAH1 |
| IJUCHEK | MRMOD | BAB1 |
| IJUCKMES | MRMOD | BBD3 |
| IJUCLBUF | MRMOD | BAG2 |
| IJUCLOEX | MRMOD | BAJ3 |
| IJUDISDL | MRMOD | BDC1 |
| IJUDISEN | MRMOD | BDB1 |
| IJUENDTS | MRMOD | BBB1 |
| IJUENDWT | MRMOD | BDG2 |
| IJUENGCK | MRMOD | BAH3 |
| IJUENLST | MRMOD | BDG5 |
| IJUEXIT | MRMOD | BDH3 |
| IJUFEED | MRMOD | BAC4 |

| Label | Phase | Location |
|---|---|---|
| ISDONE | $$BOMT02 | ECB4 |
| ISHDRINF | $$BCMT02 | FBD1 |
| ISLOADPT | $$BOMT01 | EAH1 |
| ISLOADPT | $$BCMT03 | EDF3 |
| ISLOADPT | $$BCMT05 | EGH1 |
| ISOPEN | $$BOMT03 | EDH4 |
| ISOPEN | $$BOMT07 | EJF3 |
| ISREWIND | $$BOMT01 | EAF1 |
| ISV3 | $$BCMT02 | FBC1 |
| LABELCHK | $$BOMT01 | EBF1 |
| LABELCHK | $$BCMT02 | ECC4 |
| LABELCK | $$BCMT03 | FCA3 |
| LABELRD | $$BCMT05 | FFE2 |
| LABELSPC | $$BCMT05 | FEB5 |
| LABELSW | $$BOSD06 | LUH4 |
| LABFND | $$BOCPT1 | RBD2 |
| LABFND | $$BOCPT2 | REG2 |
| LABLD | $$BOCPT2 | RDF3 |
| LAF1XTNT | $$BOSDI2 | LGC3 |
| LASTCARD | $$BOMRCE | ADA4 |
| LASTIND | $$BOSD05 | LSA2 |
| LASTIND | $$BOSDW2 | MCB3 |
| LASTOPN | $$BOUR01 | ABE3 |
| LASTOPN | $$BOMRCE | ADJ4 |
| LASTSW | $$BCSD01 | LLJ4 |
| LASTVOL | $$BCSDI1 | LEA4 |
| LASTXTNT | $$BCSD05 | LTG5 |
| LASTXTNT | $$BOSDW1 | MAF1 |
| LASTXTNT | $$BCSDW2 | MDH1 |
| LDIOREG | $$BOCP02 | QBG3 |
| LDIOREG | $$BOCP11 | QEB1 |
| LEGAL1 | $$BOCPM1 | SED5 |
| LEGAL2 | $$BOCPM1 | SEB5 |
| LEGAL3 | $$BOCPM1 | SEB4 |
| LFCOMP | $$BCCPT4 | RMG1 |
| LIMITCHK | $$BOSD02 | LCG5 |
| LIMITS | $$BOSD02 | LDE2 |
| LOADADDR | $$BCMT05 | FFC4 |
| LOADEXIT | $$BOSDI3 | LHJ2 |
| LOADEXIT | $$BOSD05 | LTJ4 |
| LOADEXIT | $$BOSD06 | LVG1 |
| LOADEXIT | $$BCSD09 | LYH4 |
| LOADINP | $$BCSD00 | LAD2 |
| LOADIO | $$BOSDEV | MKD3 |
| LOADMSG | $$BOSDI1 | LFA2 |
| LOADMSG | $$BOSD01 | LMG2 |
| LOADMSG | $$BOSDW1 | MAD4 |
| LOADREG0 | $$BOSD06 | LVF2 |
| LOADUSER | $$BCSD06 | LUB5 |
| LOADWORK | $$BOSDI1 | LEE3 |
| LOADWORK | $$BOSD07 | LWK1 |
| LOOP | $$BOMT07 | EJB3 |
| LOOP | $$BERPTP | PJF1 |
| LOOP1 | $$BCCPM1 | SEB1 |
| LOOP2 | $$BOCPM1 | SEA2 |
| MDRSBY | $$BCMT02 | FBB3 |
| MDRSBY | $$BCMT07 | FHC5 |
| MESSG17 | $$BCSDI3 | LHJ1 |
| MESSG47 | $$BOSDI1 | LEH2 |
| MFCM | $$BCLOSP | SCJ3 |
| MFCX | $$BOCP03 | QCB4 |
| MF1 | $$BOCP03 | QCC4 |
| MF3 | $$BCCP03 | QCC5 |
| MF4 | $$BCCP03 | QCD4 |
| MF5 | $$BOCP03 | QCE4 |
| MGTPOP | $$BOCP01 | QAE3 |
| MGTPOP | $$BOCP02 | QBE4 |
| MGTPOP | $$BOCP11 | QED4 |
| MGTPOP | $$BOCP12 | QFE4 |
| MODE9 | $$BJCOP1 | ELE3 |
| MODIFY3 | $$BOSDI1 | LEJ3 |
| MONORTN | $$BODQUE | MJD5 |
| MORXTNTS | $$BOSDI1 | LEE2 |
| MOVBLNK | $$BOMT03 | EDB3 |
| MOVBLNK | $$BCMT05 | FFH3 |
| MOVE | $$BOMT03 | EEC2 |
| MOVE | $$BCMT04 | FDK1 |
| MOVE | $$BCMT05 | FEJ3 |
| MOVEADDR | $$BOSDW3 | MEJ1 |
| MOVECCW | $$BOSDI2 | LGF4 |
| MOVECCW | $$BOSDW1 | MAG1 |
| MOVECCW9 | $$BOSDW1 | MAC5 |
| MOVERCM | $$BOSD03 | LPD1 |
| MOVERCM | $$BOSD08 | LXG1 |
| MOVESER | $$BCCPT1 | SAJ3 |
| MOVESYM | $$BOSDI1 | LFB2 |
| MOVESYM | $$BOSDW1 | MAE4 |
| MOVEUNIT | $$BOMT01 | EAD1 |
| MOVEUNIT | $$BOSDI1 | LFB3 |
| MOVEUNIT | $$BOSDW1 | MAB5 |
| MOVEUNIT | $$BOSDC1 | MFH3 |
| MOVEUTL0 | $$BOSD06 | LUA3 |
| MOVLOLIM | $$BOSD09 | LYH1 |
| MOVSYM | $$BOSD01 | LBB5 |
| MOVSYM | $$BOSD01 | LBJ3 |
| MSGEXIT | $$BOSD03 | LPC1 |
| MSGEXIT | $$BOCPT4 | PMD3 |
| MSGPHAS1 | $$BOSDI1 | LFE2 |
| MSGPHAS1 | $$BOSDI2 | LGJ2 |
| MSGPHAS1 | $$BOSD01 | LMH2 |
| MSGPHAS1 | $$BOSDW1 | MBD4 |
| MSGWRITE | $$BJCOPT | EKB5 |
| MSGWRITE | $$BCMT07 | FHD4 |
| MSGWRITE | MTMOD | FJJ1 |
| MSGWRITE | $$BOSD03 | LQJ3 |
| MSGWRITE | $$BOCPT2 | RFB3 |
| MSGWRITE | $$BOCPT3 | RJH4 |
| MSGWRITE | $$BOCPT4 | RLD4 |
| MSG10A | $$BOMT03 | EDC4 |
| MSG11A | $$BOMT01 | EAH2 |
| MSG12A | $$BOMT01 | EAB3 |
| MSG12A | $$BOMT03 | EDJ5 |
| MSG12A | $$BOCPT3 | RHB2 |
| MSG12A | $$BOCPT4 | PLB2 |
| MSG13D | $$BOMT01 | EAG3 |
| MSG13D | $$BOCPT4 | RMB3 |
| MSG14A | $$BOMT01 | EBC2 |
| MSG14A | $$BOCPT4 | RLK3 |
| MSG15A | $$BOMT01 | EAB5 |
| MSG15A | $$BOCPT4 | RLH4 |
| MSG16A | $$BOMT01 | EAE5 |
| MSG16A | $$BOCPT4 | RLH5 |
| MSG17D | $$BOMT02 | ECG2 |
| MSG18D | $$BOMT02 | ECF3 |
| MSG19A | $$BOMT03 | EED1 |
| MSG19A | $$BOMT06 | EHG4 |
| MSG19A | $$BOCPT3 | PHH1 |
| MSG23D | $$BOMT02 | ECA5 |
| MSG25D | $$BOMT05 | EGH2 |
| MSG30A | $$BCMT01 | FAH5 |

| Label | Phase | Location |
|---|---|---|
| PCHBLNK | $$BCLOSP | SDD4 |
| POINTERX | $$BOSDW3 | MEA3 |
| POSTDIB | $$BOSDI1 | LEE1 |
| POSTDIB | $$BOSDO1 | LLF1 |
| POSTEDX | $$BOSDEV | MKG3 |
| POSTOPEN | $$BOSDI4 | LJH1 |
| POSTOPEN | $$BOSDW1 | MAG2 |
| POSTXTNT | $$BOSDI2 | LGF2 |
| POSTXTNT | $$BOSDI4 | LJB1 |
| POSTXTNT | $$BOSDW3 | MEC3 |
| PPRT | $$BOCPO2 | QBD3 |
| PRBGN | $$BOSDO1 | LBB1 |
| PRECONVT | $$BOSDO1 | LBA4 |
| PRECONVT | $$BOSDO2 | LDE3 |
| PRINTIN | $$BOCPO2 | QBA5 |
| PROCESS | $$BOMT06 | EHF1 |
| PROCESS | $$BERPTP | PHA5 |
| PROCESS | $$BOCPT3 | RGF1 |
| PROCRUN | $$BOCPT3 | RGD3 |
| PROCRUN | $$BOCPT4 | RKF4 |
| PROCRUN | $$BCCPT1 | SAB2 |
| PROCRUN1 | $$BOCPT3 | RGF3 |
| PROCRUN1 | $$BOCPT4 | RKH4 |
| PROCRUN1 | $$BCCPT1 | SAC2 |
| PROCRUN2 | $$BOCPT3 | RGG3 |
| PROCRUN2 | $$BOCPT4 | RKJ4 |
| PROCRUN2 | $$BOCPT4 | RLE2 |
| PROCRUN2 | $$BCCPT1 | SAD2 |
| PTSIO | $$BCLOSP | SCH4 |
| PT1 | $$BCLOSP | SCF4 |
| PT2 | $$BCLOSP | SCG4 |
| PUBADR | $$BOCPO3 | QCD1 |
| PUNCHBLK | $$BCLOSP | SDC3 |
| PVTLIBRY | $$BOSDC1 | MFD1 |
| P50 | CDMOD | AKC1 |
| QTAMFLPT | $$BOSDI4 | LJB3 |
| QTAMOPEN | $$BOSDI4 | LJC2 |
| RCOPEN1 | $$BOSDC1 | MFF2 |
| RDAGAIN | $$BOMT06 | EHC4 |
| RDDISK | $$BOCPT1 | RBB3 |
| RDDISK | $$BOCPT2 | RFB1 |
| RDDISK | $$BCCPT1 | SBB1 |
| RDFRWRD | $$BCORO1 | CCG1 |
| RDTPLAB | $$BCCPT1 | SAB4 |
| READ | $$BOMT05 | EGF1 |
| READ | MTMOD | FJB3 |
| READ | $$BOCPM1 | SEG4 |
| READAGAN | $$BOMT02 | ECH3 |
| READBK | $$BOMT02 | ECH1 |
| READDISK | $$BOSDI1 | LFE1 |
| READDISK | $$BOSDI2 | LGG5 |
| READDISK | $$BOSDO1 | LMG4 |
| READDISK | SD | LZB2 |
| READDISK | $$BOSDW1 | MBB2 |
| READDISK | $$BOSDW2 | MDB3 |
| READDISK | $$BOSDW3 | MEC5 |
| READDISK | $$BOSDC1 | MGB4 |
| READLABL | $$BOMT01 | EBG1 |
| READTAPE | $$BOMT01 | EAE3 |
| READTAPE | MTMOD | FJB2 |
| READVOL | $$BONVOL | FMB3 |
| READVOL1 | $$BOSDI1 | LFC3 |
| READVOL1 | $$BOSDO2 | LOH1 |
| READXTNT | $$BOSDO1 | LLF4 |

| Label | Phase | Location |
|---|---|---|
| READ3TM | $$BOMT01 | EAJ5 |
| READ3TM | $$BOMT01 | EBB3 |
| REENTRY | $$BOSDI2 | LGB2 |
| REENTRY | $$BOSDI3 | LHC1 |
| REGSAVE | $$BOSDEV | MKF1 |
| RELNEXT | $$BOCPT1 | RAB1 |
| RELNEXT | $$BOCPT2 | RDC2 |
| RELNEXT | $$BOCPT4 | RKC2 |
| RELOCATE | $$BOMT01 | EAB1 |
| REOPEN1 | $$BOSDC1 | MFH5 |
| REPCHCP1 | $$BCLOSP | SDA5 |
| REPCHCP2 | $$BCLOSP | SDG5 |
| REPUNCH | $$BERPTN | PGE2 |
| REREAD | $$BERPTP | PHC2 |
| REREADF1 | $$BOSDW2 | MCC2 |
| REREAD1 | $$BOSDO1 | LBF5 |
| REREAD1 | $$BOSDO2 | LDJ3 |
| RESET | $$BOMT07 | EJC3 |
| RESETCP | $$BOSDO0 | LAF2 |
| RESETEOF | $$BOSDI1 | LEC3 |
| RESETIND | $$BOSDI3 | LHB5 |
| RESETIND | $$BOSDO1 | LMB4 |
| RESETLBL | $$BOSDO6 | LUE3 |
| RESETMON | $$BOSDI1 | LFE5 |
| RESETPNT | $$BOSDW3 | MED3 |
| RESET9 | $$BOSDC1 | MFH2 |
| RESTORE | $$BODQUE | MJF5 |
| RESTORE | $$BOCPT4 | RLB5 |
| RETMON | $$BOSDI1 | LEF1 |
| RETMON | $$BOSDO1 | LLG1 |
| RETMON | $$BOCPO1 | QAJ4 |
| RETMON | $$BOCPO2 | QBH1 |
| RETMON | $$BOCPO3 | QCF5 |
| RETMON | $$BOCP11 | QDE4 |
| RETMON | $$BCLOSP | SCC3 |
| RETRNPT | $$BJCOP1 | ELB2 |
| RETRNPT | $$BJCOP1 | ELJ1 |
| RETRYMSG | $$BERPTN | PGK2 |
| RETRYMSG | $$BCLOSP | SDH1 |
| RETURN | $$BCMT02 | FBK5 |
| RETURN | $$BONVOL | FMF4 |
| RETURN | $$BOCPT1 | RBK4 |
| RETURN | $$BCLOSP | SCB2 |
| RETURNED | $$BOMT01 | EAD2 |
| RETURNED | $$BJCOPT | EKD2 |
| RETURNED | $$BCMT07 | FHE2 |
| REWIND | $$BOMT01 | EAG1 |
| REWIND | $$BOMT03 | EDE3 |
| REWIND | $$BCMT06 | FGH4 |
| REWIND | $$BONVOL | FMD3 |
| REWINDIT | $$BOMT05 | EGD1 |
| REWINDT | $$BCMT06 | FGF3 |
| REWRITE | $$BERPTP | PHJ1 |
| RST | $$BOCPO3 | QCH3 |
| RST | $$BOCP11 | QDD3 |
| RTRYNT | $$BJCOPT | EKA3 |
| SAVDIB | $$BOSDO0 | LAB4 |
| SAVE | $$BONVOL | FMB1 |
| SAVECON | $$BOSDEV | MKB1 |
| SAVEF1 | $$BOSDO1 | LME2 |
| SAVENUMB | $$BOMRCE | ADE3 |
| SAVESEQ | $$BOSDI2 | LGG1 |
| SAVESEQNO | $$BOSDO1 | LLJ5 |
| SAVEUNIT | $$BOSDW1 | MAB3 |
| SAVLIMIT | $$BOSDO2 | LOA4 |

| Label | Phase | Location |
|---|---|---|
| UNDEF | $$BOSD01 | LMB1 |
| UNEXDEF | $$BOSDC1 | MGF1 |
| UNTPCP | $$BOCP11 | QDA3 |
| UNTRC | $$BOCP02 | QBF1 |
| UNTRC | $$BOCP03 | QCJ3 |
| UNTRC | $$BOCP11 | QDE3 |
| UNTRCP | $$BOCP01 | QAB4 |
| UNTRCP | $$BOCP02 | QBC3 |
| UNTRCP | $$BOCP03 | QCE3 |
| UNTRCP | $$BOCP12 | QFF2 |
| UPDATE | $$BOUR01 | ABE4 |
| UPDATE | $$BOMT04 | EFG1 |
| UPDATE | $$BOMT07 | EJD4 |
| UPDATE | $$BOMT02 | FBJ5 |
| UPDATE | $$BOMT04 | FDA4 |
| USDSTM | $$BJCOPT | EKB4 |
| USER | $$BOMT05 | FFC5 |
| USEREXIT | $$BOSDW3 | MEF3 |
| USEREXIT | $$BOSDC1 | MGF3 |
| USEREXIT | $$BOSDEV | MKD2 |
| USEREXIT | $$BOSDEV | MKJ3 |
| USERLBLS | $$BOSD01 | LMC3 |
| USERRTNE | $$BOMT01 | FAF4 |
| USERTEST | $$BOMT04 | EFJ1 |
| USERXTNT | $$BOSDI2 | LGK1 |
| VERIFY | $$BOSD01 | LMH1 |
| VERIFY | $$BOSIGN | LNB1 |
| VERIFYCP | $$BOSD09 | LYB3 |
| VIA1052 | $$BOSD05 | LTA3 |
| VOLSKIP | $$BOCPT3 | RHE2 |
| VOLSKIP | $$BOCPT4 | RLB1 |
| VOLSKIP1 | $$BOCPT3 | RHF2 |
| VOLSKIP1 | $$BOCPT4 | RLC1 |
| VOL1CHK | $$BOCPT3 | RHB1 |
| VOL1CHK | $$BOCPT4 | RLB3 |
| VOL8CO | $$BJCOP1 | ELF4 |
| VOL8CO | $$BOCPT2 | REJ4 |
| VTOCADDR | $$BOSDI1 | LFF4 |
| VTOCADDR | $$BOSD02 | LOE2 |
| VTOCADDR | $$BOSDW1 | MBA1 |
| WHDRS | $$BOCPT1 | RBK1 |
| WORKCLOS | $$BOSDC1 | MFF5 |
| WORKFILE | $$BOMT06 | FGB3 |
| WORKFILE | $$BOSDC1 | MFE1 |
| WORKTEST | $$BOSD02 | LDB2 |
| WRITDISK | $$BOSD06 | LVB4 |
| WRITDISK | SD | LZB3 |
| WRITDISK | $$BOSDW1 | MBB3 |
| WRITDISK | $$BOSDW2 | MDB4 |
| WRITDISK | $$BOSDC1 | MGB5 |
| WRITE | MTMOD | FJB4 |
| WRITE | $$BONVOL | FME3 |
| WRITE | $$BOCPM1 | SEG3 |
| WRITEFL | $$BOSDC1 | MFE4 |
| WRITELBL | $$BOSD06 | LUF4 |
| WRITETAP | MTMOD | FJD2 |
| WRITHDR | $$BOMT06 | EHA5 |
| WRITMK | $$BOMT06 | EHD5 |
| WRITTM | $$BJCOP1 | ELG4 |
| WRONGPAK | $$BOSD02 | LOD2 |
| WTDEVED | $$BCLOSP | SDH3 |
| XTNTCONV | $$BOSD01 | LBC4 |
| XTNTOK | $$BOSD01 | LBG2 |
| XTNTOK | $$BOSD02 | LCD3 |
| XTNTOKAY | $$BOSD02 | LOE5 |
| XYZ | $$BERPTP | PHE1 |
| YESLDPT | $$BOMT03 | EDA4 |
| YESOPEN | $$BOMT03 | EEA1 |
| YESREWND | $$BOMT06 | EHG1 |
| YESTM | $$BOMT01 | EAH3 |

For further detailed information on these messages, see DOS/VS Messages, GC33-5379.

| Message Number | Issuing Routine | Message |
|---|---|---|
| 4000I | $$BERRTN | RETRY |
| 4110A | $$BCMT03<br>$$BOCPT3 | NO VOL1 LBL FOUND TLBL=xxxxxx<br>filename SYSxxx=cuu |
| 4111I | $$BCMT01<br>$$BOCPT4 | NO VOL1 LBL FOUND filename<br>SYSxxx=cuu |
| 4112A | $$BCMT01<br>$$BCMT03<br>$$BCCPT3<br>$$BOCPT4 | VOL SERIAL NO. ERROR TLBL=xxxxxx<br>filename SYSxxx=cuu |
| 4113D | $$BCMT01<br>$$BOCPT4 | NO HDR1 LABEL FOUND filename<br>SYSxxx=cuu |
| 4114A | $$BCMT01<br>$$BOCPT4 | FILE SEQ NO. ERROR filename<br>SYSxxx=cuu |
| 4115A | $$BCMT01<br>$$BOCPT4 | FILE SER. NO. ERROR TLBL=xxxxxx<br>filename SYSxxx=cuu |
| 4116A | $$BCMT01<br>$$BOCPT4 | VOLUME SEQ. NO. ERROR filename<br>SYSxxx=cuu |
| 4117D | $$BCMT02<br>$$BCMT05 | NO TM FOUND ON READBK filename<br>SYSxxx=cuu |
| 4118D | $$BCMT02 | FILE ID ERROR, READBK filename SYSxxx=cuu |
| 4119A | $$BCMT03<br>$$BCMT06<br>$$BCCPT3 | FILE UNEXPIRED filename<br>SYSxxx=cuu |
| 4120I | $$BCMT03 | TAPE POSITIONED WRONG filename SYSxxx=cuu |
| 4121A | $$BCMT07 | NO ALTERN DRIVE ASSGN SYSxxx=cuu |
| 4122I | $$BCMT07 | EOV ENCOUNTERED SYSxxx=cuu |
| 4123D | $$BCMT02 | WRONG POSITN, READBK filename SYSxxx=cuu |
| 4124I | $$BCMT04 | TOO MANY UHL'S filename SYSxxx=cuu |
| 4125D | $$BCMT05 | VOL1 LBL FOUND filename SYSxxx=cuu |
| 4126I | $$BCMT02 | EOV ENCOUNTERED filename SYSxxx=cuu |
| 4127A | $$BCMT05 | EOV WHILE WRITING ECF |
| 4130A | $$BCMT01 | EOF OR EOV INQUIRY filename SYSxxx=cuu |

Figure 53. Message cross reference list (1 of 5).

| Message Number | Issuing Routine | Message |
|---|---|---|
| 4304I | $$BCSDI1 | NO FCRMAT 4 LABEL IN VTCC |
| | | or |
| 4404I | $$BCSDO2 | NO RECORD FOUND |
| 4904I | $$BCSDW1 | |
| 4306I | $$BCSDI1 | NO STANDARD VOL 1 LABEL |
| | | or |
| 4406I | $$BCSDO2 | NO RECORD FCUND |
| 4506I | $$BCSDC1 | |
| 4906I | $$BCSDW1 | |
| 4307I | $$BCSDI1 | NO RECORD FCUND |
| 4407I | $$BCSDO1 | |
| 4907I | $$BCSDW1 | |
| 4308D | $$BCSDI3 | NO UTL0 FILEMARK FOUND |
| | | or |
| 4408D | $$BCSDO6 | NO RECORD FCUND |
| 4209I | $$BCSDO3 | NO RECORD FCUND |
| | $$BCSDO8 | |
| 4409I | $$BCSDO3 | |
| | $$BCSDO3 | |
| | $$BCSDO8 | |
| 4709I | $$BCSDO8 | |
| 4909I | $$BCSDO3 | |
| | $$BCSDO3 | |
| | $$BCSDO8 | |
| 4n20I | $$BCSD00 | TAPE POSITICNED WRONG filename SYSxxx=cuu |
| 4329D | $$B3540I | EXTENTS NCT EXHAUSTED |
| 4330D | $$BCSDI2 | FMT1-LLAE UNEQUAL |
| 4331D | $$BCSDI2 | VOLUME SEQUENCE ERRCR |
| 4332I | $$BCDIO5 | VOLUME SEQUENCE ERROR |
| 4433A | $$BCSDO8 | EQUAL FILE ID IN VTCC |
| 4733A | $$BCSDO8 | |
| 4933A | $$BCSDO8 | |
| 4434I | $$BCSDO5 | CURRENT FILE LBL DELETED |
| 4935I | $$BCSDW2 | DELETED WCRKFILE LABEL |
| | $$BCSDW3 | |
| 4936I | $$BCSDW3 | NO MCRE AVAIL/MATCH XTNT |

Figure 53. Message cross reference list (3 of 5).

| Message Number | Issuing Routine | Message |
|---|---|---|
| 4465I | $$BODIO2 | EQUAL FILE LABEL IN VTOC |
| 4366A | $$BOSDI2 | 1 TRACK USER LBI EXTENT |
| 4466A | $$BCSDO1 | |
| 4477A | $$BOSDO7 | EXTENT ENTRY ERROR--RETRY |
| 4880I | $$BCEOV1 | INVALID FILE TYPE |
| 4881I | $$BCEOV1 | NO LABEL INFORMATION |
| 4383I | $$BCSDI1 | INVALID LOGICAL UNIT |
| 4483I | $$BOSDO1 | |
| 4983I | $$BCSDW1 | |
| 4884D | $$BOCP02<br>$$BOCP11<br>$$BOCP12 | NEED FILE PROTECT RING |
| 4887I | $$BERRTN | SYS FILE EXTENT EXCEEDED |
| 4888I | $$BERRTN | EOF CN SYSTEM FILE |
| 4497I | $$BOSDO3 | OVLAP EXPIRED SECRD FILE |
| 4399D | $$BOSDI2 | DATA SECURED FILE ACCESSED |
| 4MR1I | MRMOD | EXTERNAL INTERRUPT I/O ERROR filename SYSxxx=cuu |
| 4MR2I | MRMOD | SCU NCT CPERATICNAL filename SYSxxx=cuu |
| 4P01I | $$BERPTR | DATA CHECK SYSxxx=cuu |
| 4P02D | $$BERPTR | DATA CHECK SYSxxx=cuu |

Figure 53. Message cross reference list (5 of 5).

CTLCHR=ASA

A control character must appear in each logical record if the ASA option is chosen. If the control character for the printer is not valid, a message is given and the job is canceled. If the control character for the card punch is not V or W, the card is selected into pocket 1. The codes are as follows:

| Code | Interpretation |
|---|---|
| (blank) | Space one line before printing |
| 0 | Space two lines before printing |
| - | Space three lines before printing |
| + | Suppress space before printing |
| 1 | Skip to channel 1 before printing |
| 2 | Skip to channel 2 before printing |
| 3 | Skip to channel 3 before printing |
| 4 | Skip to channel 4 before printing |
| 5 | Skip to channel 5 before printing |
| 6 | Skip to channel 6 before printing |
| 7 | Skip to channel 7 before printing |
| 8 | Skip to channel 8 before printing |
| 9 | Skip to channel 9 before printing |
| A | Skip to channel 10 before printing |
| B | Skip to channel 11 before printing |
| C | Skip to channel 12 before printing |
| V | Select stacker 1 |
| W | Select stacker 2 |

CTLCHR=YES

The control character is the command-code portion of the System/360 Channel Command Word used in printing a line or spacing the forms. If the character is not one of the following characters, unpredictable events will occur.

Stacker Selection on 1442 and 2596

| 8-Bit Code | Punch Combination | Function |
|---|---|---|
| 10000001 | 12,0,1 | Select into stacker 1 |
| 11000001 | 12,1 | Select into stacker 2 |

Pocket Selection on 2540

| | | |
|---|---|---|
| 00000001 | 12,9,1 | Select into pocket 1 |
| 01000001 | 12,0,9,1 | Select into pocket 2 |
| 10000001 | 12,0,1 | Select into pocket 3 |

Stacker Selection on 2520, 3504, 3505, 3525

| | | |
|---|---|---|
| 00000001 | 12,9,1 | Select into stacker 1 |
| 01000001 | 12,0,9,1 | Select into stacker 2 |

Printer Control (not for 3525)

| | | |
|---|---|---|
| 00000001 | 12,9,1 | Write (no automatic space) |
| 00001001 | 12,9,8,1 | Write and space 1 line after printing |
| 00010001 | 11,9,1 | Write and space 2 lines after printing |
| 00011001 | 11,9,8,1 | Write and space 3 lines after printing |
| 10001001 | 12,0,9 | Write and skip to channel 1 after printing |
| 10010001 | 12,11,1 | Write and skip to channel 2 after printing |
| 10011001 | 12,11,9 | Write and skip to channel 3 after printing |
| 10100001 | 11,0,1 | Write and skip to channel 4 after printing |
| 10101001 | 11,0,9 | Write and skip to channel 5 after printing |
| 10110001 | 12,11,0,1 | Write and skip to channel 6 after printing |
| 10111001 | 12,11,0,9 | Write and skip to channel 7 after printing |

E



F

**READER'S COMMENT FORM**

SY33-8560-1

This sheet is for comments and suggestions about this manual. We would appreciate *your* views, favorable or unfavorable, in order to aid us in improving *this* publication. This form will be sent directly to the author's department. Please include your name and address if you wish a reply. Contact your IBM branch office for answers to technical questions about the system or when requesting additional publications. Thank you.

**Your comments* and suggestions:**

*** We would especially appreciate your comments on any of the following topics:**

| | | | | | |
|---|---|---|---|---|---|
| Clarity of the text | Accuracy | Index | Illustrations | Appearance | Paper |
| Organization of the text | Cross-references | Tables | Examples | Printing | Binding |