File Number S360-30 (DOS)
Order No. GY24-5088-4

Program Logic

# DOS LIOCS Volume 3
# SAM and DAM for DASD

## Program Numbers
SDMOD 360N-IO-455
DAMOD 360N-IO-454

This reference publication is one of four Program Logic Manuals that describe the internal logic of the Logical IOCS (Input/Output Control System) programs for the IBM System/360 Disk Operating System. The four related Program Logic Manuals are listed below.

Note: Although titles of some DOS publications have been simplified, the change does not affect the contents of the publications.

Volume 1: Introduction, GY24-5020.

Volume 2: Unit Record, Magnetic Tape, and Device Independent Files, GY24-5087.

Volume 3: SAM and DAM for DASD, GY24-5088.

Volume 4: ISFMS GY24-5089.

This manual is intended for use by persons involved in program maintenance and by system programmers who are altering the program design. Program logic information is not necessary for the operation of the programs described. Therefore, distribution is limited to those with maintenance and alteration responsibilities.

Effective use of this publication requires an understanding of IBM System/360 operation and the Disk Operating System Assembler language and its associated macro definition language. Reference publications for this information are listed in the Preface.

For the titles and abstracts of other related publications, refer to the IBM System/360 and System/370 Bibliography, GA22-6822.

DOS Release 25

Fifth Edition (April 1971)

This publication was formerly titled IBM System/360 Disk Operating System Logical IOCS Volume 3: Sequential and Direct Access DASD Files. Although titles of some DOS publications (including this one) have been simplified, the change does not affect the contents of the publications.

This edition applies to Release 25 of IBM System/360 Disk Operating System and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM systems, consult the latest System/360 and System/370 SRL Newsletter, GN20-0360, for the editions that are applicable and current.

This edition is a major revision of, and obsoletes, GY24-5088-3 and Technical Newsletter GN24-5421.

## Summary of Amendments

The changes to this manual reflect the program modifications concerning data set security and improved forced end of volume for disk. Support is also included for the IBM 2319 Disk Storage Facility. The flowchart symbols used in this manual conform with the American National Standards Institute, Inc. flowcharting standards. See Appendix F for an explanation of the symbols. Miscellaneous maintenance changes are also included.

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

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Department G60, P.O. Box 6 Endicott, New York, 13760.

This manual consists of four sections:

- General Information.
- Sequential Access DASD Files.
- Direct Access DASD Files.
- Detailed Flowcharts.

The first section supplies general information pertinent to both sequential access DASD files and direct access DASD files, including LIOCS extensions for Asynchronous Processing, DASD label processing, and common logical transient phases.

The second section supplies descriptions of the declarative macros and DTF tables, descriptions of the imperative macros, and the initialization and termination procedures for sequential access DASD files.

The third section supplies information about direct access DASD files, including descriptions of the declarative macros and DTF tables, discussion of the referencing methods and addressing systems, descriptions of the imperative macros, and initialization and termination procedures.

The fourth section contains the detailed flowcharts of the first three sections.

## PREREQUISITE PUBLICATIONS

Note: Although titles of some DOS publications have been simplified, the change does not affect the contents of the publications.

- IBM System/360 Principles of Operation, GA22-6821.
- DOS Data Management Concepts, GC24-3427.
- DOS Supervisor and I/O Macros, GC24-5037.
- DOS System Control and Service, GC24-5036.
- IBM System/360 Disk and Tape Operating Systems, Assembler Specifications, GC24-3414.

## RELATED PUBLICATIONS

- Introduction to DOS Logic, GY24-5017. GY24-5017.
- DOS IPL and Job Control, GY24-5086.
- DOS Supervisor and Related Transients, GY24-5151.
- DOS Logical Transients, GY24-5152.
- DOS System Service Programs, GY24-5153.
- DOS Librarian, GY24-5079.
- DOS Linkage Editor, GY24-5080.
- IBM System/360 Disk Operating System, Basic Telecommunications Access Method, Program Logic Manual, GY30-5001.
- IBM System/360 Disk Operating System, Queued Telecommunications Access Method, Program Logic Manual, GY30-5002.
- DOS Operating Guide, GC24-5022.
- DOS Messages, GC24-5074.

FIGURES

# CHARTS

This volume of DOS LIOCS Program Logic Manual provides detailed information on the logical IOCS support of DASD (Direct Access Storage Device) files processed by the Sequential Access Method, and by the Direct Access Method. It is intended for use by trained maintenance personnel experienced in the use of the Logical Input/Output Control System (LIOCS) for file processing.

Beyond brief introductory descriptions of the two DASD file processing methods covered, this volume does not contain information of a general nature. If the reader requires basic knowledge or a review of the general concept and function of Logical IOCS, he should refer to Volume 1 of DOS LIOCS listed on the front cover of this volume.

This volume contains information on all the logical IOCS items (modules, DTF tables, imperative macros, declarative macros, open and close routines, etc.) required for the two processing methods. The only exceptions are certain common and special purpose routines that cannot be related to any specific file. These routines, namely the open and close monitors, the open routines for self-relocating programs, and the Checkpoint/Restart routines are described in detail in Volume 1.

The first section of this volume contains general information which is pertinent to both sequential access DASD files and direct access DASD files. This information includes:

- LIOCS extensions for Asynchronous Processing.
- DASD label processing.
- Logical transient phases that provide special functions.

The next section supplies detailed information on sequential access DASD files. The information includes:

- Descriptions of record formats and main storage areas.
- Descriptions of the declarative macros DTFSD, DTFPH, and SDMODxx, and DTF tables.
- Discussions of imperative LIOCS macros (GET, PUT, READ, etc.) used with sequential DASD files.
- Open and close logical transients.

The third section supplies detailed information concerning LIOCS support of direct access files. The information includes:

- Descriptions of the declarative macros DTFDA, DTFPH, and DAMOD.
- Discussion of the referencing methods and addressing systems used by the Direct Access Method.
- Discussion of imperative LIOCS macros (READ, WRITE, WAITF, etc.) used with direct access files.
- Description of direct access channel program builder routine.
- Open and close logical transients.

The last section contains the detailed flowcharts of the imperative logical IOCS macros supported by the data handling logic modules discussed in this manual. It also contains flowcharts of the logical transient routines 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 charts contain decision blocks to illustrate the logic (i.e., coding) generated for certain module generation macro parameter options. These decisions do not appear in an assembly listing, but they do determine the contents of a particular module at module generation time. If an assembly listing is not available for a specific logic module, a listing of the source statements used to generate the module can be obtained from microfiche cards (Appendix D).

# GENERAL INFORMATION

This section includes general information that is applicable to both sequential access DASD files and direct access DASD files. The areas covered include the LIOCS extensions for the Asynchronous Processing function, DASD label processing, and several logical transients that provide special functions.

## ASYNCHRONOUS PROCESSING EXTENSIONS

Asynchronous Processing extensions for Logical IOCS consist of six functions:

- OPEN/IGN
- Sequential Disk End-of-Extent
- Relative Addressing
- Trailer Label Processing
- Reentrant Modules
- Track Hold

The OPEN/IGN function and Sequential Disk End-of-Extent function are American National Standard COBOL requirements, and are provided mainly for American National Standard COBOL use. These functions are documented within this manual, but are not covered in the general discussion.

The Relative Addressing and Trailer Label Processing functions apply only to direct access files. Discussion and documentation of these functions are found in the section, Direct Access Files.

Reentrant Modules and Track hold functions apply to both sequential access DASD files and direct access files. A general discussion of both functions is included in the following text.

## REENTRANT MODULES

A reentrant module is a logic module that can be asynchronously used, or shared, by more than one file. A module is made reentrant by including the parameter RDONLY=YES in the operand of the module generation macro (SDMODxx or DAMOD macro instruction), and the DTFSD/DTFDA macro.

The RDONLY (read-only) parameter assures that the generated logic module is never modified, regardless of the processing requirements of any file(s) using the module. The reentrant feature is implemented through the establishment of unique save areas, one for each DTF using the module. Each save area must be 72 bytes long and aligned on a doubleword boundary. A task must provide the address of the save area associated with the DTF in register 13 before issuing an imperative macro and entering the logic module.

## TRACK HOLD FUNCTION

The track hold function provides DASD track protection when the parameter HOLD=YES is specified in the operand of the module generation macro (SDMODxx/DAMOD) and the DTFSD/DTFDA macro. If a task has previously accessed a DASD track and is currently modifying a record from that track, DASD track protection prevents another task in main storage from accessing that track. The task attempting to access the held track is put in the wait state until the track has been released. For direct access, the problem program must issue the FREE macro to release a track held on READ operation. The module automatically holds and releases all tracks for WRITE operations. For sequential DASD, the problem program releases the track by issuing the FREE macro, if work files have been specified with the UPDATE=YES parameter included, and if the record is not updated. If the record is updated, the module automatically releases the track when the record is written.

For fixed-length, undefined-length and blocked variable-length SD (Sequential DASD) files, the next GET macro that actually causes an I/O operation releases the track.

Exception: If blocked variable-length records are specified and a PUT macro is issued for the last record in a block, the PUT macro releases the track. The PUT macro also releases the track if unblocked variable-length records are specified.

The track hold function is applicable to three situations:

1. Sequential DASD update files (data).

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

3. Direct access files.

For more information concerning the track hold function, refer to DOS Supervisor and I/O Macros, listed in the Preface.

## DASD LABEL PROCESSING

Before a DASD file can be processed by logical IOCS, the file must be opened to permit transfer of data. The function of the open routines is to check the DASD labels identifying the file. This is accomplished by comparing the information from the actual file labels in the Volume Table of Contents (VTOC), with the label information in the SYSRES label information cylinder. Figure 1 illustrates the format of the VTOC. Job control stores label information, supplied by the user in job control cards, in the SYSRES label information cylinder.

Figure 2 illustrates the format of this stored information as it appears in both the label information cylinder and main storage.

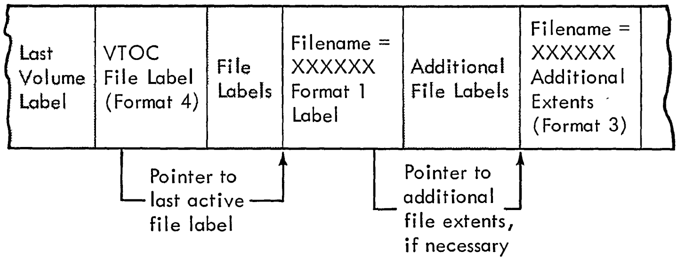

Last Volume Label
VTOC File Label (Format 4)
File Labels
Filename = XXXXXX Format 1 Label
Additional File Labels
Filename = XXXXXX Additional Extents (Format 3)
Pointer to last active file label
Pointer to additional file extents, if necessary

Figure 1. Volume Table of Contents (VTOC) for a Nonresident Disk Pack

Note: To simplify creation of DASD files and label processing, Version 3 makes it possible for the user to identify a particular file through the use of two job control statements, // DLBL and // EXTENT, (instead of the three statements, // VOL, // DLAB, and // XTENT required by previous versions). The user, however, is not obligated to change any job control statements already in use because job control handles both forms. Further references made in this manual to the new // DLBL and // EXTENT job control statement also apply to the // VOL, // DLAB, and // XTENT statements

The standard DASD labels processed by DOS logical IOCS are discussed and illustrated in Appendix G. A more complete discussion of DASD labels is contained in Volume 1.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | | 19 | Field |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DLBL- EXTENT Indicator | Filename | DA/IS Switch | File ID | Format ID | File Serial Number | Volume Seq. No. | Creation Date | Expiration Date | Reserved | Open Code | System Code | Volume Serial Number | EXTENT Type | EXTENT Seq. No. | Extent Lower Limit | Extent Upper Limit | Logical (Symbolic) Unit Address | 2321 Lower Cell | 2321 Upper Cell | Another Extent if DA or ISFMS |
| 1 | 7 | 1 | 44 | 1 | 6 | 2 | 3 | 3 | 2 | 1 | 13 | 6 | 1 | 1 | 4 | 4 | 2 | 1 | 1 | Bytes |
| 0 | 1 | 8 | 9 | 53 | 54 | 60 | 62 | 65 | 68 | 70 | 71 | 84 | 90 | 91 | 92 | 96 | 100 | 102 | 103 | Displacement |

| Field | Name | Description |
|---|---|---|
| 1. | DLBL-EXTENT | SD<br>Bit 0: 1 = Next extent on a new pack.<br>Bit 1: 1 = Last extent.<br>Bit 2: 1 = Bypass extent.<br>Bit 3: 1 = New volume on same unit.<br>Bit 4: 1 = Extent limits omitted.<br>Bit 5: 1 = Extent converted to DASD address.<br>Bit 6: 1 = No EXTENT/XTENT card.<br>Bit 7: 1 = Unused.<br>DA or ISFMS<br>Number of extents. |
| 2. | Filename | |
| 3. | DA/IS Switch | Bits 0-3: Unused.<br>Bit 4: 1 = Extent limits omitted.<br>Bit 5: 1 = Extent converted to DASD address.<br>Bit 6 & 7: Unused. |
| 4. | File ID | File identifier including generation and version numbers. If field is missing on DLBL card, Filename padded with blanks is inserted. |
| 5. | Format ID | Numeric 1 is inserted. |
| 6. | File Serial No. | Volume serial number from first extent. |
| 7. | Volume Seq. No. | Always initialized to X'0001'. |
| 8. | Creation Date | Initialized with 3 bytes of X'00'. |
| 9. | Expiration Date | If date is in the form YYDDD, it is converted to YDD. If date is in retention period form, 1 to 4 characters, the field is padded with binary zeros. |
| 10. | Reserved | The retention period, if specified is converted to a 2-byte number and inserted in this field. |
| 11. | Open Code | DLBL type:<br>S = Sequential<br>D= Direct Access<br>C or E = Indexed sequential File Management System where:<br>C = Load create function<br>E = Load extend function |
| 12. | System Code | Initialized to contain DOS/360 VER 3. This field is not processed by DOS. |
| 13. | Volume Serial No. | Volume serial number for extent. |
| 14. | Extent Type | Same codes as in Format - 1 label:<br>X'00' = Next three fields do not indicate any extent.<br>X'01' = Prime data area (ISFMS) or consecutive area, etc., (that is the extent containing the user's data records).<br>X'02' = Overflow area of an ISFMS file.<br>X'04' = Cylinder index or master index of an ISFMS file.<br>X'40' = User label track area.<br>X'8n' = Shared cylinder indicator, where n = 1, 2, or 4. |
| 15. | Extent Seq. No. | Number of extent as determined by the extent card sequence. |
| 16. & 17. | Extent Lower and Upper Limits | Before the OPEN, DLBL/EXTENT information is in the relative track form of HHNNT followed by three bytes of binary zeros.<br>HH = Relative (to 0) start address in tracks.<br>NN = Number of tracks.<br>T = 0 or upper track number for split cylinder in SD files.<br>Following an OPEN on DLBL/EXTENT cards, or whenever DLAB/XTENT cards are used, the extent lower and upper limits are each in the CCHH format. |
| 18. | Logical (Symbolic) Unit Address | This 2-byte field identifies the logical unit with the same code as that used in a CCB. The first byte identifies the unit class:<br>X'00' = System Logical Unit<br>X'01' = Programmer Logical Unit<br>The second byte identifies the logical unit within its class.<br>Thus X'0003' denotes SYSLST and X'0103' denotes SYS003. |
| 19. | 2321 Lower Cell<br>2321 Upper Cell | 2321 extent lower and upper cell limit. This 2-byte field contains zeros for 2311/2314/2319 disk. |

Note: For Sequential Disk files, a complete 104-byte block is repeated for each new EXTENT.
For Direct Access and ISFMS files, only fields 13 through 18 are repeated for each EXTENT.

Figure 2. SYSRES DASD Label Information

## COMMONLY USED LOGICAL TRANSIENTS

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

### $$BOFLPT: DASD File-Protect Charts AA-AD

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

Entry:

- From phases $$BOSDI2, $$BOSDW2, $$BOSDO4, $$BOSDO5, or $$BOSDO6 for sequential DASD files.
- From phases $$BODAIN or $$BODAO4 for direct access files.
- From phase $$BOIS07 for indexed sequential files (refer to Volume 4).

Exits:

- To the open monitor, $$BOPEN, if more files are to be opened and a specific phase name is not supplied.
- To phase $$BOQ001 (not documented in this PLM) if the file is a QTAM file.
- To the problem program if a specific phase name is not supplied and no more files remain to be opened.
- To the transient phase specified by the calling phase.

Method: The $$BOFLPT phase provides file protection for DASD files by storing extent limit information in the JIB table. For the IBM 2311 Disk Storage Drive, the IBM 2314 Direct Access Storage Facility, and the IBM 2319 Disk Storage Facility, the lower and upper cylinder limits are stored in a single JIB. For the IBM 2321 Data Cell Drive, subcell and strip information is stored in two chained JIBs, the first containing the lower extent limit, and the second containing the upper extent limit. The extent JIBs are chained to the Logical Unit Block (LUB) entry to which the device is assigned. Further information pertaining to the JIBs and LUBs is found in DOS Supervisor and Related Transients listed in the front of this manual.

The $$BOFLPT phase begins by determining a number of factors:

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

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

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

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

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

The address of the new JIB (or the first new JIB, in the case of a 2321) is calculated by using the pointer to the first unused JIB in the JIBs available chain, found in location FAVP in the supervisor. As in the case of JIBs chained to the LUB, this new JIB contains a pointer to the next available JIB that will be used if needed.

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

### $$BODSPV: VTOC Display, Phase 1 Chart AE

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

Entry: From phases $$BOSD07 or $$BOMSG2 when the operator's response is DSPLYV.

Exit:

- To the second phase of VTOC display, $$BODSPW.
- To job control via an SVC 11 if the operator's response to message 4V95A is EOB or CANCEL and the open was for job control.
- To phase $$BCNCL via an SVC 6 to cancel the job if the operator's response to message 4V96A is EOB or CANCEL and the open was not for job control.

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

### $$BODSPW: VTOC Display, Phase 2 Charts AF-AH

Objective: To display, on either SYSLST or SYSLOG, the Volume Table of Contents for the volume (pack or cell) currently being opened.

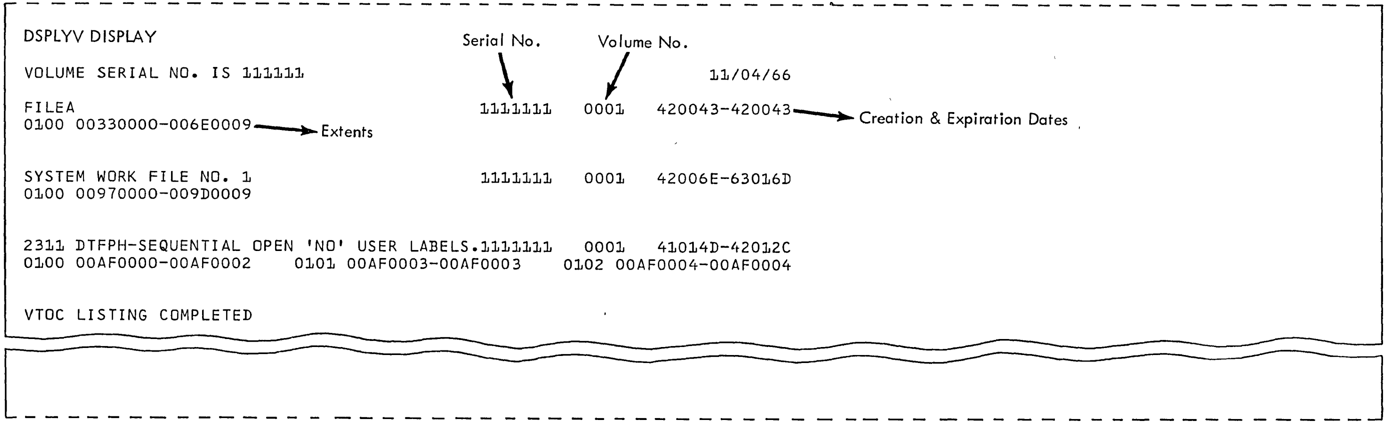
Entry: From the first phase of VTOC display, $$BODSPV.

Exit: To $$BOSD07, $$BOMSG1, or $$BODSMW.

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

The file label for each file on the volume (pack or cell) is displayed by printing the contents of the label. The first line printed for each Format 1 label contains the first 59 bytes of the label and includes:

- Filename
- Format identifier
- File serial number
- Volume sequence number
- Creation date
- Expiration date


```
DSPLYV DISPLAY

VOLUME SERIAL NO. IS 111111                                          11/04/66

FILEA                                        1111111   0001   420043-420043
0100 00330000-006E0009

SYSTEM WORK FILE NO. 1                       1111111   0001   42006E-63016D
0100 00970000-009D0009

2311 DTFPH-SEQUENTIAL OPEN 'NO' USER LABELS.1111111   0001   41014D-42012C
0100 00AF0000-00AF0002    0101 00AF0003-00AF0003    0102 00AF0004-00AF0004

VTOC LISTING COMPLETED
```
Serial No.
Volume No.
Extents
Creation & Expiration Dates

Figure 3. VTOC Display (DSPLYV Response)

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

When all Format 1 labels in the VTOC have been processed, message 4V09I is issued and the job is cancelled. Figure 3 is a sample of the VTOC display printed by this phase.

## $$BOVDMP: VTOC Dump Charts AJ-AK

Objective: To provide a list of all the labels in the Volume Table of Contents (VTOC), for the volume (pack or cell) being opened.

Entry: From phase 2 of the Disk Open Message Writer, $$BOMSG2, or $$BOSDO7, when the operator's response is CANCELV, or from the problem program.

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

Method: Phase $$BOVDMP reads the VOL 1 label to retrieve the volume serial number and the CCHHR address of the VTOC for the volume (pack or cell) being opened. A header line is then printed on SYSLST to indicate the date and identify the volume with the volume serial number. If SYSLST is not assigned to a printer, the VTOC Dump is ignored.

## $$BOWDMP: List VTOC Charts AL-AM

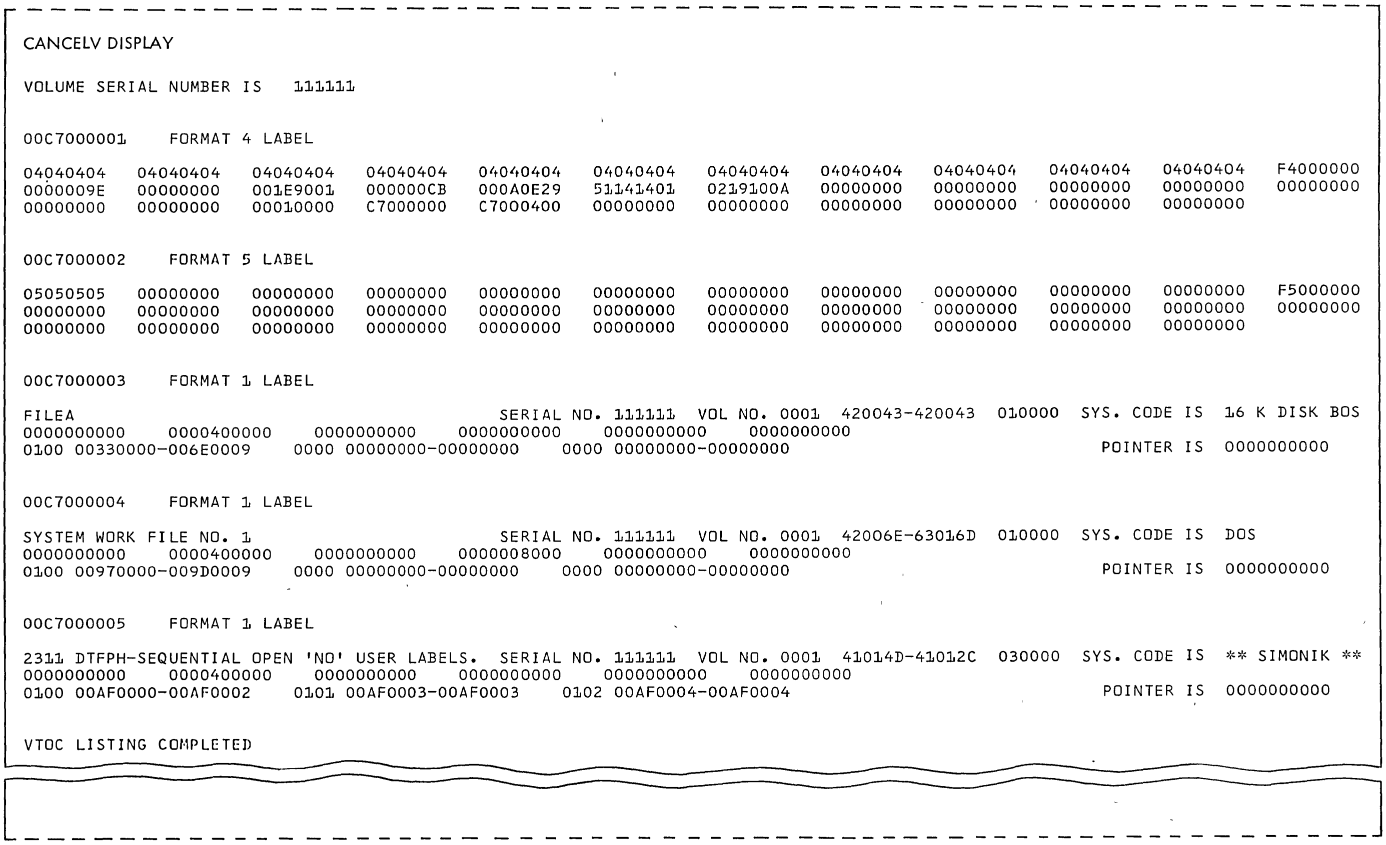
Objective: To provide a listing of all the labels in the VTOC.

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

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

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

Figure 4 is a sample of the VTOC Dump printed by this phase.



```
CANCELV DISPLAY

VOLUME SERIAL NUMBER IS   111111

00C7000001    FORMAT 4 LABEL

04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  F4000000
0000009E  00000000  001E9001  000000CB  000A0E29  51141401  0219100A  00000000  00000000  00000000  00000000  00000000
00000000  00000000  00010000  C7000000  C7000400  00000000  00000000  00000000  00000000  00000000  00000000

00C7000002    FORMAT 5 LABEL

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

00C7000003    FORMAT 1 LABEL

FILEA                                          SERIAL NO. 111111  VOL NO. 0001  420043-420043  010000  SYS. CODE IS  16 K DISK BOS
0000000000    0000400000    0000000000    0000000000    0000000000    0000000000
0100 00330000-006E0009    0000 00000000-00000000    0000 00000000-00000000                 POINTER IS  0000000000

00C7000004    FORMAT 1 LABEL

SYSTEM WORK FILE NO. 1                         SERIAL NO. 111111  VOL NO. 0001  42006E-63016D  010000  SYS. CODE IS  DOS
0000000000    0000400000    0000000000    0000008000    0000000000    0000000000
0100 00970000-009D0009    0000 00000000-00000000    0000 00000000-00000000                 POINTER IS  0000000000

00C7000005    FORMAT 1 LABEL

2311 DTFPH-SEQUENTIAL OPEN 'NO' USER LABELS.  SERIAL NO. 111111  VOL NO. 0001  41014D-41012C  030000  SYS. CODE IS  ** SIMONIK **
0000000000    0000400000    0000000000    0000000000    0000000000    0000000000
0100 00AF0000-00AF0002    0101 00AF0003-00AF0003    0102 00AF0004-00AF0004                 POINTER IS  0000000000

VTOC LISTING COMPLETED
```

Figure 4. VTOC Dump (CANCELV Response)

## $$BOMSG1 Disk Open Error Message Writer, Phase 1 Charts AN-AP

Objective: To initialize the message output area, SYSLOG CCB and CCWs, and to fetch phase 2 of the message writer, $$BOMSG2.

Entry:

- From the disk VTOC display phase, $$BODSPW.
- From a DASD open or close phase.
- From the DTFCP open phases, $$BOCP01, $$BOCP02, $$BOCP11 or $$BOCP12.

Exit: Phase 2 of the open error message writer, $$BOMSG2.

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

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

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

This phase first saves the last four characters in the name of the phase requesting the message. It initializes the SYSLOG message output area with the organization type numeric code, DTF filename and symbolic unit and constant. It builds the SYSLOG CCWs for writing the message and reading the response and determines if the required message is in this phase of the message writer. If it is not in this phase, the routine determines in which overlay phase the message is located (either $$BOMSG3, $$BOMSG4, $$BOMSG5, $$BOMSG6, or $$BOMSG7) and fetches $$BOMSG2 to load the required overlay phase.

## $$BOMSG2 Disk Open Error Message Writer, Phase 2 Charts AQ-AS

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

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

Exit:

- To the VTOC dump phase, $$BOVDMP,
- To phase 1 of the VTOC display routine, $$BODSPV,
- To the DASD open/close organization phase requesting the message (if a cancel was not encountered).

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

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

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

For a file overlap type, the last character of the message is initialized to a 'D', the CCW is initialized to write the 44-byte file key, and the file overlap switch is set to NO-OP. The file overlap switch set to NO-OP allows a test for deleting an unexpired file later in the routine.

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

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

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

If the message is a D-type, the message return indicator is set, the address of the next phase name is retrieved, and an SVC 2 is issued to fetch the return phase. If the message is not a D-type, the routine tests the file-overlap switch and the wrong-pack switch as previously mentioned. During this portion of the routine, a 'B' for bypass, or a 'D' for delete is stored in the transient region +1186 for use by the calling phase.

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

1. Operator reply of IGNORE for a D-type message.
2. Message for ISFMS.
3. Equal file ID message.
4. No EXTENT to be bypassed.
5. Next pack not mounted.

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

Figure 5 shows the message code (passed via register 3) together with the last two digits and action indicator of the associated message number. For reference purposes, the text of the message is also included.

### $$BODSMW Data Security Message Writer Charts AT-AU

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

Entry: From $$BODSPW, $$BOSDI2, $$BODAI1, $$BODA02, $$BOIS06, $$BORTV1, and return from $$BODSPV.

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

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

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

| Message Code | Message Number | Message |
|---|---|---|
| 0 | 44A | OVERLAP ON UNEXPRD FILE |
| 1 | 55A | WRONG PACK, MOUNT nnnnnn |
| 2 | 40A | EXTENT OVERLAP ON ANOTHER |
| 3 | 41A | EXTENT OVERLAPS ON VTOC |
| 4 | 42A | NO MATCHING EXTENT |
| 5 | 33A | EQUAL FILE ID IN VTOC |
| 6 | 66A | 1 TRACK USER LBL EXTENT |
| 7 | 59A | INVALID EXTENT |
| 15 | 84D | NEED FILE PROTECT RING |
| 16 | 31D | VOLUME SEQUENCE ERROR |
| 17 | 38D | USER HDR LBL IS NOT STD |
| 18 | 39D | USER TRL LBL IS NOT STD |
| 19 | 08D | NO UTL0 FILE MARK FOUND |
| 20 | 47A | EXTENTS NOT ON SAME UNIT |
| 21 | D | Reserved for D type msg. |
| 22 | 00I | NO RECORD FOUND |
| 23 | 01I | NO RECORD FOUND |
| 24 | 02I | NO RECORD FOUND |
| 25 | 03I | NO RECORD FOUND |
| 26 | 04I | NO RECORD FOUND |
| 27 | 05I | NO RECORD FOUND |
| 28 | 06I | NO RECORD FOUND |
| 29 | 07I | NO RECORD FOUND |
| 30 | 08D | NO RECORD FOUND |
| 31 | 09I | NO RECORD FOUND |
| 32 | 00I | NO LABEL SPACE IN VTOC |
| 33 | 01I | NO FORMAT 1 LABEL FOUND |
| 34 | 02I | NO FORMAT 2 LABEL FOUND |
| 35 | 03I | NO FORMAT 3 LABEL FOUND |

Figure 5. Message Codes for Disk Open Error Message Writer (Part 1 of 3)

| Message Code | Message Number | Message |
|---|---|---|
| 36 | 04I | NO FORMAT 4 LBL IN VTOC |
| 37 | 06I | NO STANDARD VOL1 LABEL |
| 38 | 41I | EXTENT OVERLAPS ON VTOC |
| 39 | 46I | DISCONT INDEX EXTENTS |
| 40 | 51I | SYSUNITS NOT IN SEQUENCE |
| 41 | 52I | DISCONT TYPE 1 EXTENTS |
| 42 | 54I | DSKXTN ENTRY TABLE FULL |
| 43 | 62I | NO PRIME DATA EXTENT |
| 44 | 45I | TOO MANY EXTENTS |
| 45 | 49I | DATA TRACK LIMIT INVALID |
| 46 | 59I | INVALID EXTENT |
| 47 | 60I | NO EXTENTS, ALL BYPASSED |
| 48 | 61I | INVALID DLBL FUNCTION |
| 49 | 63I | LOAD FILE NOT CLOSED |
| 50 | 80I | INVALID FILE TYPE |
| 51 | 81I | NO LABEL INFORMATION |
| 52 | 83I | INVALID LOGICAL UNIT |
| 53 | 90I | NO AVAILABLE JIB |
| 54 | 87I | SYS FILE EXTENT EXCEEDED |
| 55 | 35I | DELETED WORKFILE LABEL |
| 56 | 34I | CURRENT FILE LBL DELETED |
| 57 | 40I | EXTENT OVERLAP ON ANOTHER |
| 58 | 36I | NO MORE AVAIL/MATCH XTNT |
| 59 | 48I | SYSIN/SYSOUT UNSUPPORTED |
| 60 | 70I | 1ST XTNT CD NOT INDX VOL |
| 61 | 71I | EXTENT INFO NEEDED |
| 62 | 72I | MOD AND DTF INCOMPATIBLE |
| 63 | 58I | NO EXTENT FOR OUTPUT FILE |
| 64 | 88I | EOF ON SYSTEM INPUT FILE |

Figure 5. Message Codes for Disk Open Error Message Writer (Part 2 of 3)

| Message Code | Message Number | Message |
|---|---|---|
| 68 | 98I | OVLAP UNEXPRD SECRD FILE |
| 69 | 69I | FILE IS OPEN FOR ADD |
| 70 | 97I | OVLAP EXPIRED SECRD FILE |
| NONE | 99D | DATA SECURED FILE ACCESSED |

Figure 5. Message Codes for Disk Open Error Message Writer (Part 3 of 3)

## SEQUENTIAL ACCESS DASD FILES

Sequentially-organized DASD (SD) files are contained on 2311, 2314, 2319, or 2321 DASD devices, and are processed by the Sequential Disk 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 under Device Independent Files in Volume 1.

### RECORD FORMATS

Logical records in a sequential access DASD file may be blocked or unblocked records and may be in one of four formats: fixed-length, (Format F) variable-length or spanned (Format V), or undefined (Format U). The format of the record and whether the file is blocked, is specified by the user in the DTFSD macro instruction used to define the file.

### FIXED-LENGTH RECORDS (FORMAT F)

Fixed-length records may be blocked or unblocked. The number of logical records within a block (blocking factor) is normally constant for every block in the file unless the block is truncated (short block). In unblocked format, the logical record constitutes the block (refer to Figure 6).

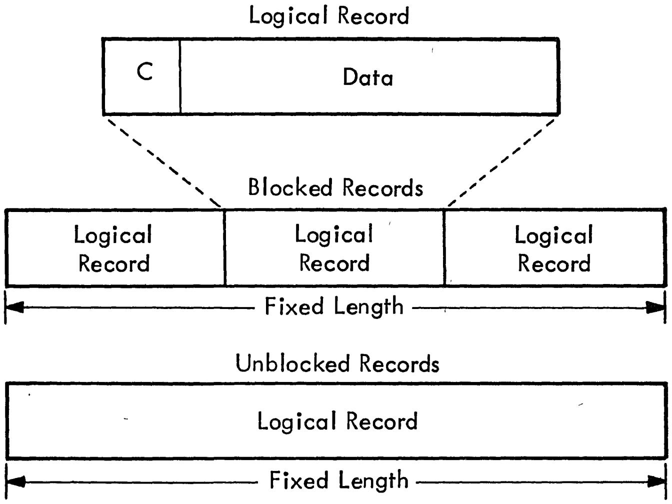


Figure 6. Fixed-Length Record Format (Format F)

VARIABLE-LENGTH RECORDS (FORMAT V)

Variable-length records may be in blocked or unblocked format. Either format may be spanned or unspanned. Since the length of the record is not constant, each record describes its own length. For blocked records, each block describes its own block length.

Figure 7 illustrates the format of variable-length records. The first four characters (bytes) of each logical record contain control information; '$L_2$' represents the length of the logical record and 'bb' represents two bytes reserved for system use. The user must provide these characters when he is creating the record. An optional control character, represented by C in Figure 7 may be specified as the fifth character of each logical record.

For blocked records, '$L_1$' represents the block length and 'bb' represents the two bytes reserved for system use. Although these four bytes do not appear in the logical record furnished to the user, input and output areas must be large enough to accommodate them.

For unblocked records, the logical record and the block control information constitute the block, unless the logical record is larger than the physical block and spanned processing has been specified. In such a case, the logical record is divided into segments, which are written into each block until all the bytes of the logical record have been written. The last block, therefore, contains only enough space to hold the remaining bytes of the logical record plus the block control information.

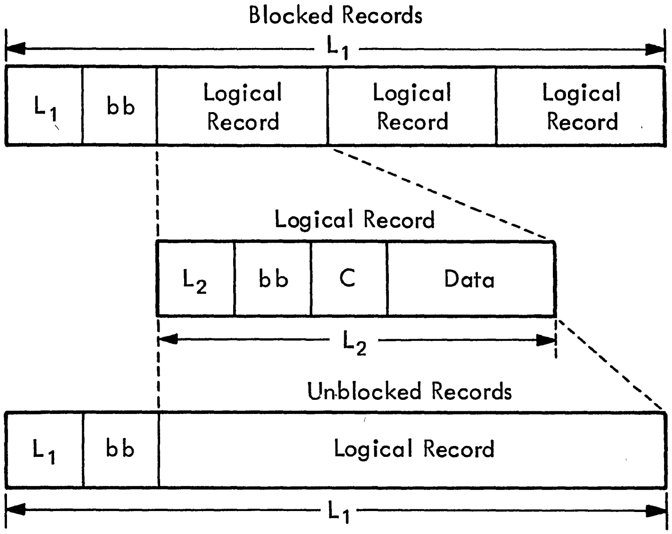


Figure 7. Variable-Length Record Format (Format V)

<u>SPANNED RECORDS</u>: Spanned records are format V records, each of which specifies its own length. Spanned record processing is an extension of variable-length record processing. In this technique the user need not be concerned with the restrictions the system imposes on the length of physical records. Thus, he can maximize his secondary storage efficiency, while organizing his data files with logical record lengths most suited to his needs. The Sequential DASD access method allows a logical record, either blocked or unblocked, to span multiple physical records. This implies that:

1. The user only concerns himself with logical records. The IOCS segments and blocks his logical records for him, while it makes a most efficient use of the track capacities on his DASD devices.

2. The user is allowed greater flexibility in transferring logical records from one type of DASD device to another, when he uses the Sequential DASD access method.

Figure 8 shows spanned records. The first four bytes of every spanned record, whether blocked or unblocked, constitute the <u>Block Descriptor Word</u>, which describes the information portion of the block that immediately follows it. The first two bytes contain the block length (LL) supplied by data management when the data set is written. The last two bytes (RR) are reserved and set to binary zeros. The user is required to reserve, for use by IOCS, the four bytes occupied by the block descriptor word, at the beginning of his input and output areas.

The length of each logical record ($\ell\ell$), including two bytes for the length field and two bytes for system use (rr), must be supplied by the problem programmer when the record is written.

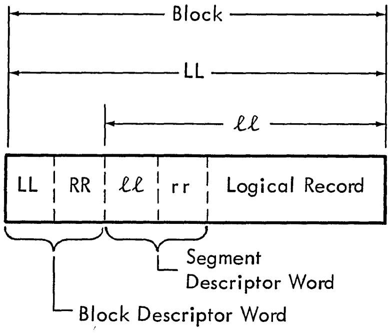


Figure 8. Spanned Records (Unblocked)

Because the length of a logical record may exceed the size of a single physical record on the associated device, IOCS may write a spanned record in sections called <u>segments</u>. Figure 9 shows segmented spanned records.

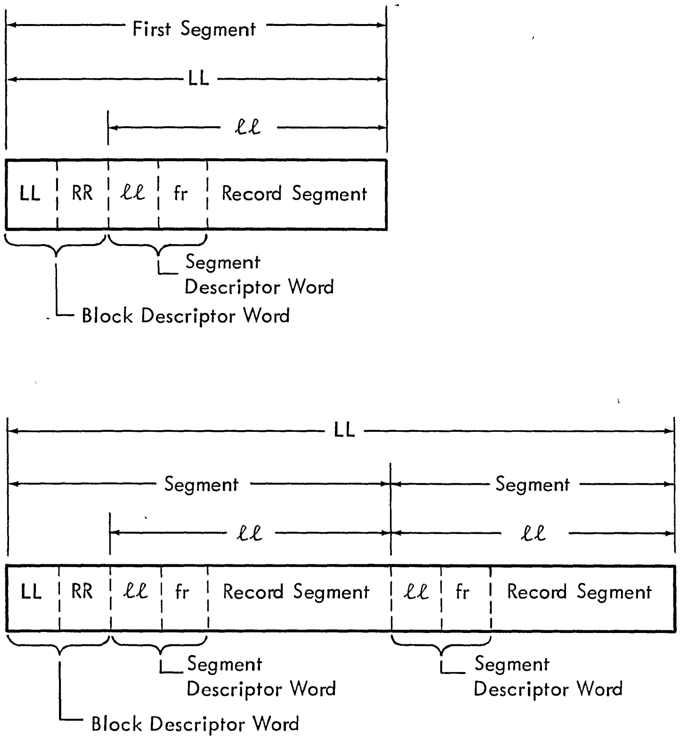


Figure 9. Segmented Spanned Records (Blocked)

When the logical record is written in segments, each segment includes a <u>segment descriptor word</u>. The segment descriptor word is an additional four-byte field that describes the data portion of the segment which immediately follows it.

The segment length, including the four bytes occupied by the segment descriptor word itself, is contained in bits 1-15 of the first two bytes ($\ell\ell$). The value must lie in the range $4 \le \ell\ell \le 32,763$.

Bit 0 describes the segment type. If the bit is off, it indicates that the segment is a normal one. If the bit is on, it indicates a null segment containing the eight descriptor bytes only.

The last two bytes of the segment descriptor word are reserved and set to binary zeros, with the exception of bits 6 and 7, which contain a value (f). This value specifies the relative position of the segment with respect to other segments, if any; that is, whether it is a single segment or whether it is the first, the last, or an intermediate segment of a multisegment logical record. When a spanned record is read, the segment lengths specified in each segment descriptor word are added together to provide the problem program with the length ($\ell\ell$) of the logical record.

The first segment of a spanned record may begin at any point in the physical record on the associated device.

## UNDEFINED RECORDS (FORMAT U)

Undefined records are treated as unblocked records, and any deblocking must be performed by the problem program. The optional control character may be used in each logical record (refer to Figure 10).

## 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.

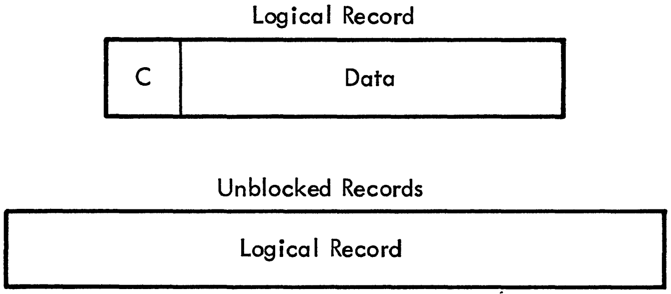


Figure 10. Undefined Record Format

## 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 11 through 20 illustrate the format of the save area for each logic module.

SDMODFI - Fixed-Length Input

| Byte | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| Displ.<br>DEC | <br>HEX | | | | | | | | |
| 0 | 0 | User Register 9 | | | | User Register 10 | | | |
| 8 | 8 | User Register 11 | | | | User Register 12 | | | |
| 16 | 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 | | | | | Maximum Block Size (SDMODFI With Truncation) | | | |
| 64 | 40 | Not used | | | | | | | |

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

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

Figure 11. SDMODFI Save Area

SDMODFO - Fixed-Length Output

| Byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Displ.<br>DEC HEX<br>0 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 there are no more DTFs to be opened.

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

Figure 12. SDMODFO Save Area

SDMODFU - Fixed-Length Input with Update

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

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

**If ERREXT=YES, bytes 60-67 contain the parameter list that includes the address of DTF and core address of the block in error.

Figure 13. SDMODFU Save Area

SDMODVI - Variable-Length Input

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

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

**If ERREXT=YES, bytes 36-43 contain the parameter list that includes the address of DTF and core address of the block in error.

Figure 14. SDMODVI Save Area

SDMODVO - Variable-Length Output

| Byte | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| Displ.<br>DEC | <br>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 there are no more DTFs to be opened.

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

Figure 15. SDMODVO Save Area

SDMODVU - Variable-Length Input with Update

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

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

**Last Record Switch

***If ERREXT=YES, bytes 44-51 contain the parameter list that includes the address of DTF and core address of the block in error.

Figure 16. SDMODVU Save Area

SDMODUI - Undefined Input

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

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

**If ERREXT=YES, bytes 40-47 contain the parameter list that includes the address of DTF and core address of the block size in error.

Figure 17. SDMODUI Save Area

SDMODUO - Undefined Output

| Byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Displ.<br>DEC HEX<br>0 0 | User Register 7 | | | | User Register 8 | | | |
| 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 OPEN that there are no more DTFs to be opened.

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

Figure 18. SDMODUO Save Area

SDMODUU - Undefined Input with Update

| Byte | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| Displ.<br>DEC<br>0 | HEX<br>0 | User Register 8 | | | | User Register 9 | | | |
| 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 OPEN that there are no more DTFs to be opened.

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

Figure 19. SDMODUU Save Area

SDMODW - Work File

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

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

**If ERREXT=YES, bytes 28-35 contain the parameter list that includes the address of DTF and core address of the block in error.

Figure 20. SDMODW Save Area

## DTFSD MACRO

### DATA FILES

To process a sequentially-organized DASD file of data records by the Sequential Access Method, the file must first be defined by the declarative macro DTFSD (Define The File for Sequential DASD). This 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 main storage areas and routines used for the file.

A DTF table is then generated according to the parameters specified in the operand of the DTFSD macro instruction. Figure 21 illustrates the DTF table generated for sequential DASD files.

### WORK FILES

If a TYPEFLE=WORK parameter is specified in the operand of a DTFSD macro instruction, the file being defined is a work file, and work file macro instructions READ, WRITE, CHECK, etc., are provided. This type of file is used in applications where records are to be alternately read and written from and to a DASD device, which is used as a temporary extension of main storage.

Figure 22 illustrates the DTF table that is generated when the TYPEFLE=WORK parameter is specified.

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| &Filename | 0-15<br>(0-F) | | Command Control Block (CCB). |
| | 16<br>(10) | 0 | 1 = Dequeue old volume extents. |
| | | 1 | i = Dummy OPEN to obtain extents from label track. |
| | | 2 | 1 = File assigned 'IGN' (COBOL). |
| | | 3 | 1 = Track hold option specified. |
| | | 4 | 1 = DTF relocated by OPENR. |
| | | 5 | 1 = Input trailer labels to be processed at close time (COBOL only). |
| | | 6 | 1 = Spanned processing. |
| | | 7 | 1 = COBOL end-of-extent option specified. |
| | 17-19<br>(11-13) | | Address of logic module. |
| | 20<br>(14) | | DTF type for OPEN/CLOSE (X'20' = sequential access DASD files). |
| | 21<br>(15) | 0 | 1 = 2321, 0 = 2311, 2314, 2319. |
| | | 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<br>(16-1C) | | Filename (DTF Name). |
| | 29<br>(1D) | | Device Type Code (Version 3)<br>X'00' = 2311<br>X'01' = 2314, 2319<br>X'02' = 2321 |
| | | | Note: In previous versions, last byte of filename contains device type code. |
| | 30-35<br>(1E-23) | | Address of Format 1 label in VTOC (BCCHHR). |
| | 36-37<br>(24-25) | | Volume sequence number. |
| | 38<br>(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 21. DTFSD Table -- Data Files (Part 1 of 12)

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| &Filename.S | | | Output File |
| | | 0 | 1 = No more extents. |
| | | 1 | 1 = Extents needed at close time. |
| | | 2 | 1 = Process trailer labels. |
| | | 3 | 1 = Process header labels. |
| | | 4 | 1 = Next extent on new volume. |
| | | 5 | 1 = Extents entered via 1052. |
| | | 6 | 1 = Process trailer labels at close. |
| | | 7 | 1 = Check extent for minimum of 2 tracks. |
| | 39<br>(27) | 0 | 1 = Extent bypassed before file is opened (Input only). |
| | | 1 | 1 = FEOVD has been issued (input only) |
| | | 0-7 | Sequence number of current extent opened (Output only). |
| | 40<br>(28) | | Sequence number of last extent opened. |
| | 41-43<br>(29-2B) | | Address of user's label routine. |
| | 44-47<br>(2C-2F) | | Address of IOAREA1. |
| | 48-51<br>(30-33) | | CCHH address of user's label track (X'80000000'). |
| | 52-53<br>(34-35) | | Lower head limit (HH). |
| | 54-57<br>(36-39) | | Extent upper limit (CCHH). |
| | 58-59<br>(3A-3B) | | Seek address (BB):<br>X'0000' if 2311, 2314, or 2319.<br>X'00nn' if 2321, where nn = bin number. |
| | 60-63<br>(3C-3F) | | Search argument (CCHH). |
| | 64<br>(40) | | Record number. |
| | 65-67<br>(41-43) | | EOF address if input file. Key length and data length if output file. |
| | 68-71<br>(44-47) | | CCHH control field<br>CCHH=X'00C80009' if 2311 type 1<br>CCHH=X'00C80013' if type 1 2314, or 2319.<br>CCHH=X'13090413' if 2321 type 1<br>CCHH=X'00C700nn' if type 128 2311, 2314, 2319<br>CCHH=X'130904nn' if 2321 type 128 where nn = current upper head number. |
| | 72<br>(48) | | Number of records per track (computed at generation time). |
| | 73<br>(49) | | Switch byte used by the logic modules for various switching purposes. Functions indicated are for the ON condition (1) of the respective bit. |

Figure 21. DTFSD Table -- Data Files (Part 2 of 12)

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

Figure 21. DTFSD Table -- Data Files (Part 3 of 12)

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| | 84-87<br>(54-57) | | Instruction to load user's register IOREG. (NOTE: This field is a NOP unless blocked records are processed in one I/O area, or two I/O areas are specified and records are processed in the I/O areas.) |
| | 88-91<br>(58-5B) | | Address of current available input/output area. |
| | 92-95<br>(5C-5F) | | Logical record size. |
| | 96-99<br>(60-63) | | Address of end of input/output area. |
| | 100<br>(64) | | Logical indicators |
| | | 0 | 1 = ERROPT = address. |
| | | 1 | 1 = ERROPT = IGNORE. |
| | | 2 | 1 = ERROPT = SKIP. |
| | | 3 | 1 = VERIFY = YES. |
| | | 4 | 1 = 2 I/O areas. |
| | | 5 | 1 = WLRERR = address (Fixed-length and variable records).<br>1 = Output file (Undefined length records). |
| | | 6 | 1 = Fixed-length records.<br>0 = Variable or undefined length records. |
| | | 7 | Control parameter specified. |
| | 101-103<br>(65-67) | | Address of user's read error routine. |
| | 104-111<br>(68-6F) | | Seek CCW. |
| | 112-119<br>(70-77) | | Search ID Equal CCW. |
| | 120-127<br>(78-7F) | | TIC CCW. |
| | 128-135<br>(80-87) | | Read/Write Data CCW. |

This is the end of the common portion of the DTFSD table. The following sections are added depending on the parameters specified in the operand of the DTFSD macro instruction. Refer to Figure 24 for the sequential access CCW chains which are generated by the DTFSD macro according to parameters specified in the operand of the DTFSD macro instruction.

Figure 21. DTFSD Table -- Data Files (Part 4 of 12)

The following section is added to the DTFSD table for fixed-length record input files.

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

Figure 21. DTFSD Table -- Data Files (Part 5 of 12)

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| If RECFORM=FIXBLK, TRUNCS=YES, and UPDATE=YES: | | | |
| | 136-143<br>(88-8F) | | Read Count CCW. |
| | 144-151<br>(90-97) | | Search ID Equal CCW. |
| | 152-159<br>(98-9F) | | TIC CCW. (Bytes 158-159 contain saved block length if two files are using same logic module.) |
| | 160-167<br>(A0-A7) | | Verify CCW. |
| | 168-175<br>(A8-AF) | | Count field input area. |
| | If CONTROL=YES, the following section is added: | | |
| | 176-191<br>(B0-BF) | | Control CCB. |
| | 192-199<br>(C0-C7) | | Control CCW. |
| If TRUNCS or UPDATE are not specified, no additions are made to the DTFSD table except when CONTROL=YES is specified. Then, the following section is added: | | | |
| | 136-151<br>(88-97) | | Control CCB. |
| | 152-159<br>(98-9F) | | Control CCW. |

Figure 21. DTFSD Table -- Data Files (Part 6 of 12)

The following section is added to the DTFSD table for fixed-length record output files.

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| | 136-143<br>(88-8F) | | Search ID Equal CCW. |
| | 144-151<br>(90-97) | | TIC CCW. |
| | 152-159<br>(98-9F) | | Verify CCW. |
| | If CONTROL is not specified: | | |
| | 160-163<br>(A0-A3) | | End-of-extent routine address (primarily used by COBOL compiler). |
| | If CONTROL=YES: | | |
| | 160-175<br>(A0-AF) | | Control CCB. |
| | 176-183<br>(B0-B7) | | Control CCW. |
| | 184-187<br>(B8-BB) | | End-of-extent routine address (primarily used by COBOL compiler). |

Figure 21. DTFSD Table -- Data Files (Part 7 of 12)

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

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| | 136-143<br>(88-8F) | | Read Count CCW. |
| If UPDATE is not specified: | | | |
| | 144-151<br>(90-97) | | Count field input area. |
| | If CONTROL=YES:* | | |
| | 152-167<br>(98-A7) | | Control CCB. |
| | 168-175<br>(A8-AF) | | Control CCW. |
| | 176-179<br>(B0-B3) | | Logical record length. |
| | 180-183<br>(B4-B7) | | RX type instruction. |
| | 184<br>(B8) | 0 | Not used. |
| | | 1 | 1 = Skip segment. |
| | | 2 | 1 = Spanned first time. |
| | | 3 | Not used. |
| | | 4 | Not used. |
| | | 5 | Not used. |
| | | 6 | Not used. |
| | | 7 | Not used. |
| | 185-187<br>(B9-BB) | | Pointer in logical record. |

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

Figure 21. DTFSD Table -- Data files (Part 8 of 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<br>(90-97) | | Search ID Equal CCW. |
| | 152-159<br>(98-9F) | | TIC CCW. |
| | 160-167<br>(A0-A7) | | Verify CCW. |
| | 168-175<br>(A8-AF) | | Count field input area. |
| | 176-183<br>(B0-B7) | | Count field save area if one I/O area. |
| | 184-191<br>(B8-BF) | | Count field save area if two I/O areas. |
| | If CONTROL=YES:* | | |
| | 192-207<br>(C0-CF) | | Control CCB. |
| | 208-215<br>(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<br>(D8-DB) | | Logical record length. |
| | 220-223<br>(DC-DF) | | RX type instruction. |
| | 224<br>(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<br>(E1-E3) | | Pointer in logical record. |
| | 228-235<br>(E4-EB) | | Count save area. |
| | 236-239<br>(EC-EF) | | Extent status save area. |

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

Figure 21. DTFSD Table -- Data Files (Part 9 of 12)

The following section is added to the DTFSD table for variable-length record output files.

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| | 136-143<br>(88-8F) | | Search ID Equal CCW. |
| | 144-151<br>(90-97) | | TIC CCW. |
| | 152-159<br>(98-9F) | | Verify CCW. |
| | 160-163<br>(A0-A3) | | Space remaining in output area. |
| | 164-165<br>(A4-A5) | | Track capacity. |
| | 166-169<br>(A6-A9) | | Instruction to load user's register VARBLD. (If VARBLD is not specified, instruction is NO-OP.) |
| | If CONTROL=YES:* | | |
| | 170-172<br>(AA-AC) | | Not used. |
| | 173-175<br>(AD-AF) | | End-of-extent routine address (primarily used by COBOL compiler). |
| | 176-191<br>(B0-BF) | | Control CCB. |
| | 192-199<br>(C0-C7) | | Control CCW. |

Figure 21. DTFSD Table -- Data Files (Part 10 of 12)

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

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| | 200-203 (C8-CB) | | Logical record length. |
| | 204-207 (CC-CF) | | RX type instruction. |
| | 208 (D0) | 0 | Not used. |
| | | 1 | Not used. |
| | | 2 | 1 = Leading segment. |
| | | 3 | 1 = Output block truncated. |
| | | 4 | 1 = End of track. |
| | | 5 | 1 = Track truncated. |
| | | 6 | 1 = Save count. |
| | | 7 | 1 = Volumes spanned. |
| | 209-211 (D1-D3) | | Pointer in logical record. |
| | 212-219 (D4-DB) | | Count save area. |
| | 220-223 (DC-DF) | | Extent status save area. |

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

Figure 21. DTFSD Table -- Data Files (Part 11 of 12)

The following section is added to the DTFSD table for undefined length record output files.

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| | 136-143<br>(88-8F) | | Search ID Equal CCW. |
| | 144-151<br>(90-97) | | TIC CCW. |
| | 152-159<br>(98-9F) | | Verify CCW. |
| | 160-161<br>(A0-A1) | | Track Capacity. |
| | If CONTROL=YES: | | |
| | 162-164<br>(A2-A4) | | Not used. |
| | 164-167<br>(A4-A7) | | End-of-extent routine address (primarily used by COBOL compiler). |
| | 168-183<br>(A8-B7) | | Control CCB. |
| | 184-191<br>(B8-BF) | | Control CCW. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 21. DTFSD Table -- Data Files (Part 12 of 12)

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| &Filename | 0-15<br>(0-F) | | Command Control Block (CCB). |
| | 16<br>(10) | 0-1 | Not used |
| | | 2 | 1 = File assigned 'IGN' (COBOL). |
| | | 3 | 1 = Track hold option specified. |
| | | 4 | 1 = DTF relocated by OPENR. |
| | | 5-7 | Not used. |
| | 17-19<br>(11-13) | | Address of logic module. |
| | 20<br>(14) | | DTF type for OPEN/CLOSE (X'20' = sequential access DASD files). |
| | 21<br>(15) | 0 | 0 = 2311, 2314, or 2319. |
| | | 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<br>(16-1C) | | Filename (DTF Name). |
| | 29<br>(1D) | | Device Type Code (Version 3):<br>X'00' = 2311<br>X'01' = 2314, 2319.<br>NOTE: In previous versions, last byte of filename contains device type code. |
| | 30-31<br>(1E-1F) | | Track capacity counter. |
| | 32-35<br>(20-23) | | Address of Format 1 label in VTOC (CHHR). |
| | 36<br>(24) | | Extent sequence number. |
| | 37<br>(25) | | Open Communications Byte. |
| | | 0-2 | Not used. |
| | | 3 | 1 = Symbolic unit in DTF. |
| | | 4 | 1 = Next extent on new volume. |
| | | 5 | 1 = Extent opened. |
| | | 6-7 | Not used. |
| | 38<br>(26) | | Lower head limit. |
| | 39<br>(27) | | Upper head limit. |

Figure 22. DTFSD Table -- Work Files (Part 1 of 3)

| DTF Assembly Label | Bytes | Bits | Function |
|---|---|---|---|
| &Filename.L | 40-41<br>(28-29) | | Record Length. |
| | 42-45<br>(2A-2D) | | Initial extent lower limit. |
| | 46-49<br>(2E-31) | | Current extent lower limit. |
| | 50-53<br>(32-35) | | Extent upper limit. |
| &Filename.S | 54-55<br>(36-37) | | Seek address (BB=X'0000'). |
| | 56-59<br>(38-3B) | | Search address (CCHH). |
| | 60<br>(3C) | | Record number. |

Figure 22. DTFSD Table -- Work Files (Part 2 of 3)

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

Numbers in parentheses are displacements in hexadecimal notation.

Figure 22. DTFSD Table -- Work Files (Part 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=2311/2314/2321 and MOUNTED=SINGLE. Figure 23 illustrates the DTF table generated to define the file for physical IOCS.

| Bytes | Bits | Function |
|---|---|---|
| 0-15 | | CCB. |
| (0-F) | | |
| 16 | 0 | 1 = Dequeue old volume extents. |
| (10) | 1 | Not used. |
| | 2 | 1 = File assigned 'IGN' (COBOL). |
| | 3 | Not used. |
| | 4 | 1 = DTF relocated by OPENR. |
| | 5-7 | Not used. |
| 17-19 | | 3X'00' |
| (11-13) | | |
| 20 | | DTF type (X'21'). |
| (14) | | |
| 21 | | Open/Close indicators. |
| (15) | 0 | 1 = 2321; 0 = 2311, 2314, 2319. |
| | 1 | 1 = Blocked files. |
| | 2 | 1 = Work file. |
| | 3 | 1 = Work area. |
| | 4 | 1 = Not Version 1 table type. |
| | 5 | 1 = Open; 0 = Closed. |
| | 6 | 1 = Input; 0 = Output. |
| | 7 | 1 = User labels specified. |
| 22-28 | | Filename (see byte 29). |
| (16-1C) | | |
| 29 | | Version 3 device type code: |
| (1D) | | X'00' = 2311 |
| | | X'01' = 2314, 2319 |
| | | X'02' = 2321 |
| | | Last byte of filename in previous versions. |
| 30 | | C'F' = EOF indicator for DTFPH. |
| (1E) | | |
| 30-35 | | (BCCHHR) Address of F1 label in VTOC (output). |
| (1E-23) | | (BCCHHR) Address of next DLBL-EXTENT record (input). |
| 36-37 | | Volume sequence number. |
| (24-25) | | |
| 38 | | Open communications byte. |
| (26) | | Output |
| | 0 | 1 = No more EXTENTS. |
| | 1 | 1 = EXTENTS for LIOCS at close. |
| | 2 | 1 = Process trailer labels. |
| | 3 | 1 = Process header labels. |
| | 4 | 1 = New volume on next EXTENT. |
| | 5 | 1 = EXTENTS entered via 1052. |
| | 6 | 1 = Process trailer labels at close. |
| | 7 | 1 = Check EXTENT for minimum of 2 tracks. |
| | | Input |
| | 0 | 1 = No more EXTENTS. |
| | 1 | Not used. |
| | 2 | 1 = No F1 label, process EXTENTS only. |
| | 3 | Not used. |
| | 4 | 1 = New volume on next EXTENT. |
| | 5 | Not used. |
| | 6 | 1 = Process header labels. |
| | 7 | Not used. |

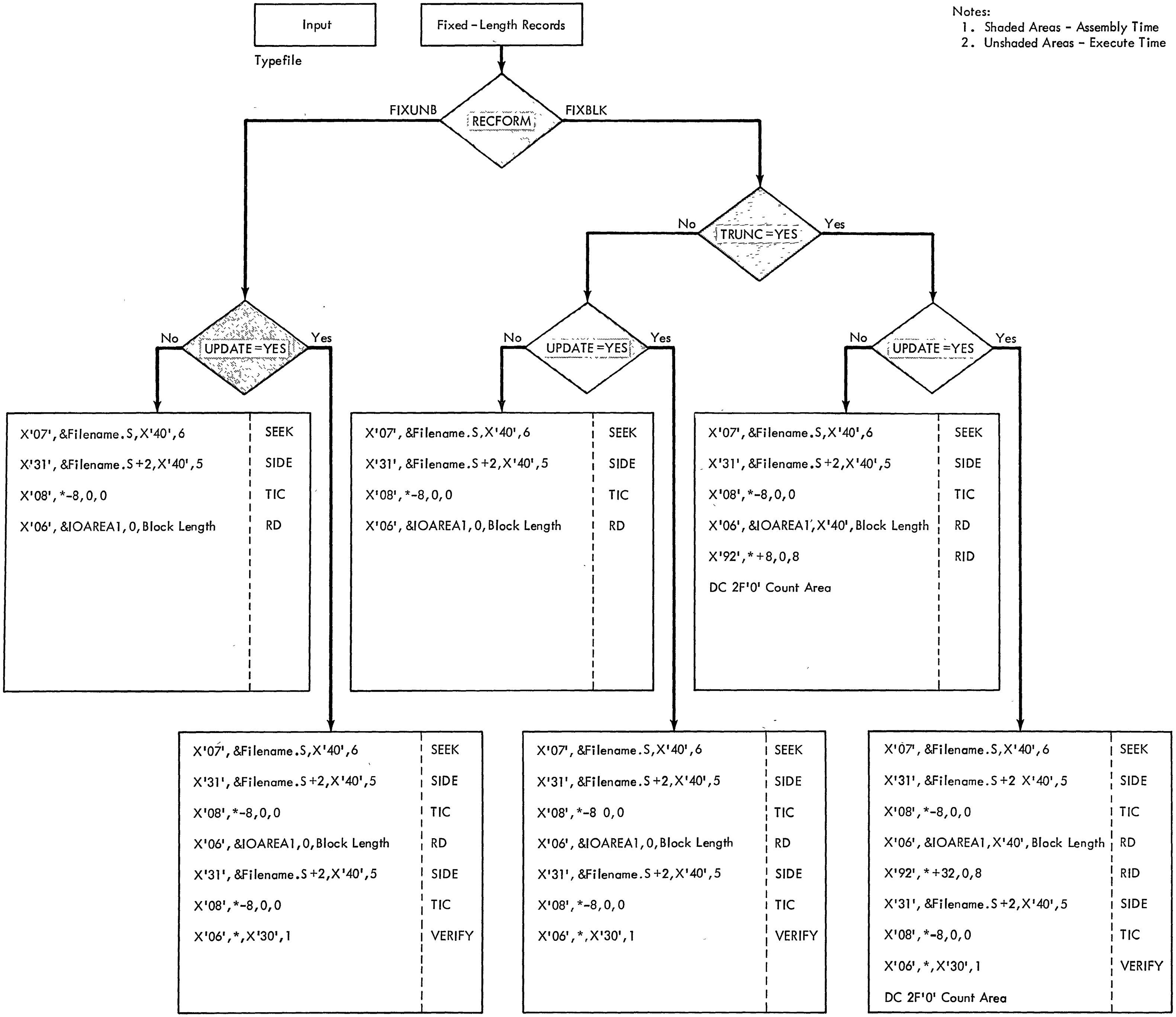
Figure 23. DTFPH Table for Sequential Disk (Part 1 of 2)

| Bytes | Bits | Function |
|---|---|---|
| 39<br>(27) | | Sequence number of current EXTENT being opened. |
| 40<br>(28) | | Sequence number of last EXTENT opened (not a 1050 EXTENT entry). |
| 41-43<br>(29-2B) | | Address of user's label routine. |
| 44-47<br>(2C-2F) | | Address of IOAREA1. |
| 48-51<br>(30-33) | | CCHH address of user's label track. Initially X'80000000'. |
| 52-53<br>(34-35) | | Lower head limit (HH) X'0000' if type 1; X'00nn' if type 128 (n = head limit). |
| 54-57<br>(36-39) | | EXTENT upper limit (CCHH). |
| 58-59<br>(3A-3B) | | BB seek address:<br>=X'0000' if 2311, 2314, or 2319.<br>=X'00nn' if 2321 where 'nn' = bin number. |
| 60-63<br>(3C-3F) | | EXTENT lower limit (CCHH). |
| 64<br>(40) | | Record number. 1 = Input, 0 = Output. |
| 65-67<br>(41-43) | | Not used. |
| 68-71<br>(44-47) | | CCHH control bucket.<br>CCHH = X'13090413' if 2321 - type 1.<br>CCHH = X'00C80009' if 2311, or X'00C80013' if 2314, 2319 - type 1.<br>CCHH = X'130904nn' if 2321 - type 128.<br>CCHH = X'00C800nn' if 2311, 2314, 2319 - type 128 where n = current upper head number. |
| 72<br>(48) | | Record number. |
| 73<br>(49) | | Not used. |
| 74-75<br>(4A-4B) | | Not used. |
| 76-80<br>(4C-50) | | CCHHR bucket = extent lower limit and record number. |
| 81-83<br>(51-53) | | Not used. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 23. DTFPH Table for Sequential Disk (Part 2 of 2)

Notes:
1. Shaded Areas - Assembly Time
2. Unshaded Areas - Execute Time



```
X'07', &Filename.S,X'40',6          SEEK
X'31', &Filename.S+2,X'40',5        SIDE
X'08',*-8,0,0                       TIC
X'06', &IOAREA1,0,Block Length      RD
```

```
X'07', &Filename.S,X'40',6          SEEK
X'31', &Filename.S+2,X'40',5        SIDE
X'08',*-8,0,0                       TIC
X'06', &IOAREA1,0,Block Length      RD
```

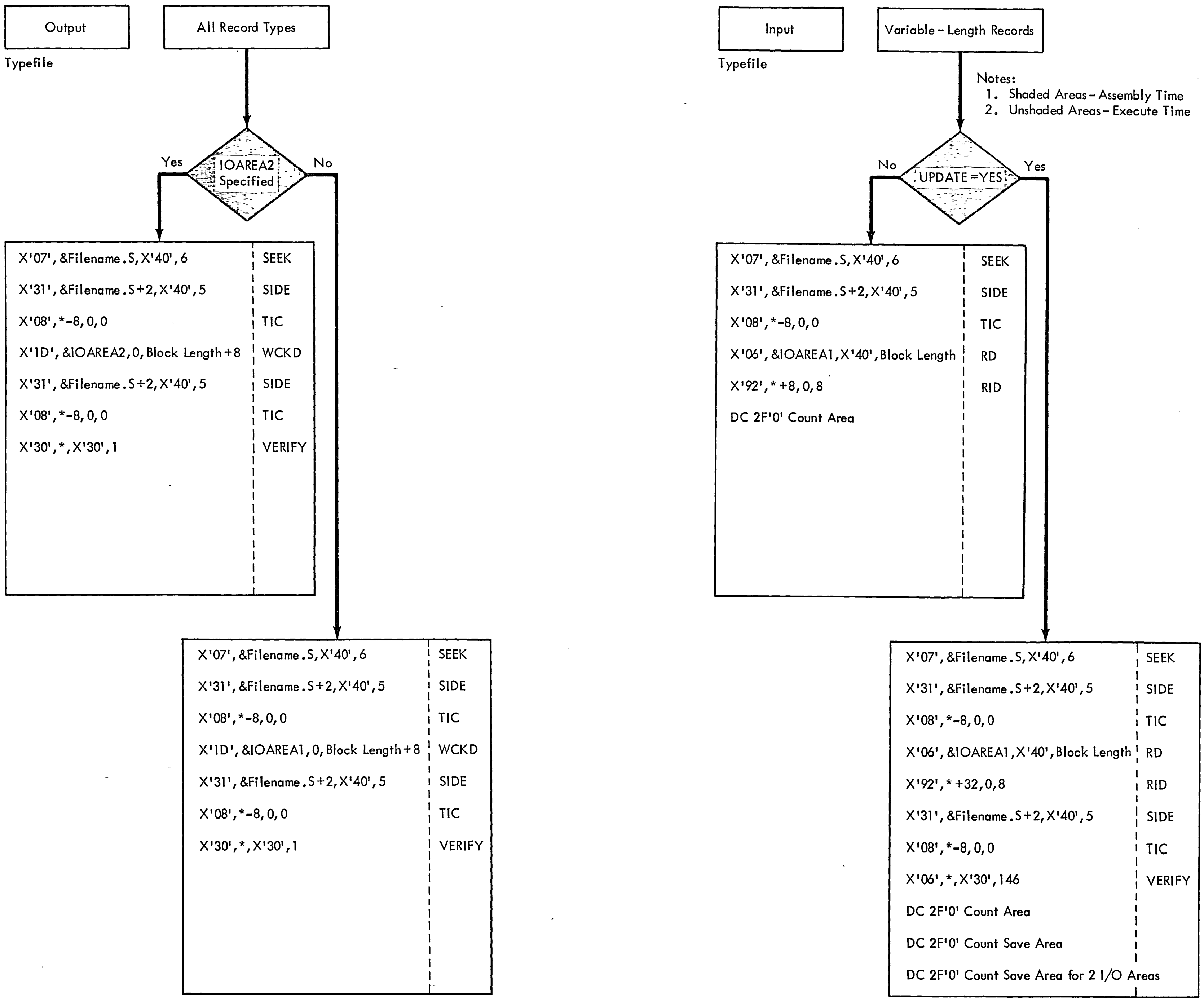
```
X'07', &Filename.S,X'40',6              SEEK
X'31', &Filename.S+2,X'40',5            SIDE
X'08',*-8,0,0                           TIC
X'06', &IOAREA1,X'40',Block Length      RD
X'92',*+8,0,8                           RID
DC 2F'0' Count Area
```

```
X'07', &Filename.S,X'40',6          SEEK
X'31', &Filename.S+2,X'40',5        SIDE
X'08',*-8,0,0                       TIC
X'06', &IOAREA1,0,Block Length      RD
X'31', &Filename.S+2,X'40',5        SIDE
X'08',*-8,0,0                       TIC
X'06',*,X'30',1                     VERIFY
```

```
X'07', &Filename.S,X'40',6          SEEK
X'31', &Filename.S+2,X'40',5        SIDE
X'08',*-8 0,0                       TIC
X'06', &IOAREA1,0,Block Length      RD
X'31', &Filename.S+2,X'40',5        SIDE
X'08',*-8,0,0                       TIC
X'06',*,X'30',1                     VERIFY
```

```
X'07', &Filename.S,X'40',6              SEEK
X'31', &Filename.S+2 X'40',5            SIDE
X'08',*-8,0,0                           TIC
X'06', &IOAREA1,X'40',Block Length      RD
X'92',*+32,0,8                          RID
X'31', &Filename.S+2,X'40',5            SIDE
X'08',*-8,0,0                           TIC
X'06',*,X'30',1                         VERIFY
DC 2F'0' Count Area
```

Figure 24. DTFSD Channel Programs (Part 1 of 3)



Notes:

1. Shaded Areas - Assembly Time
2. Unshaded Areas - Execute Time

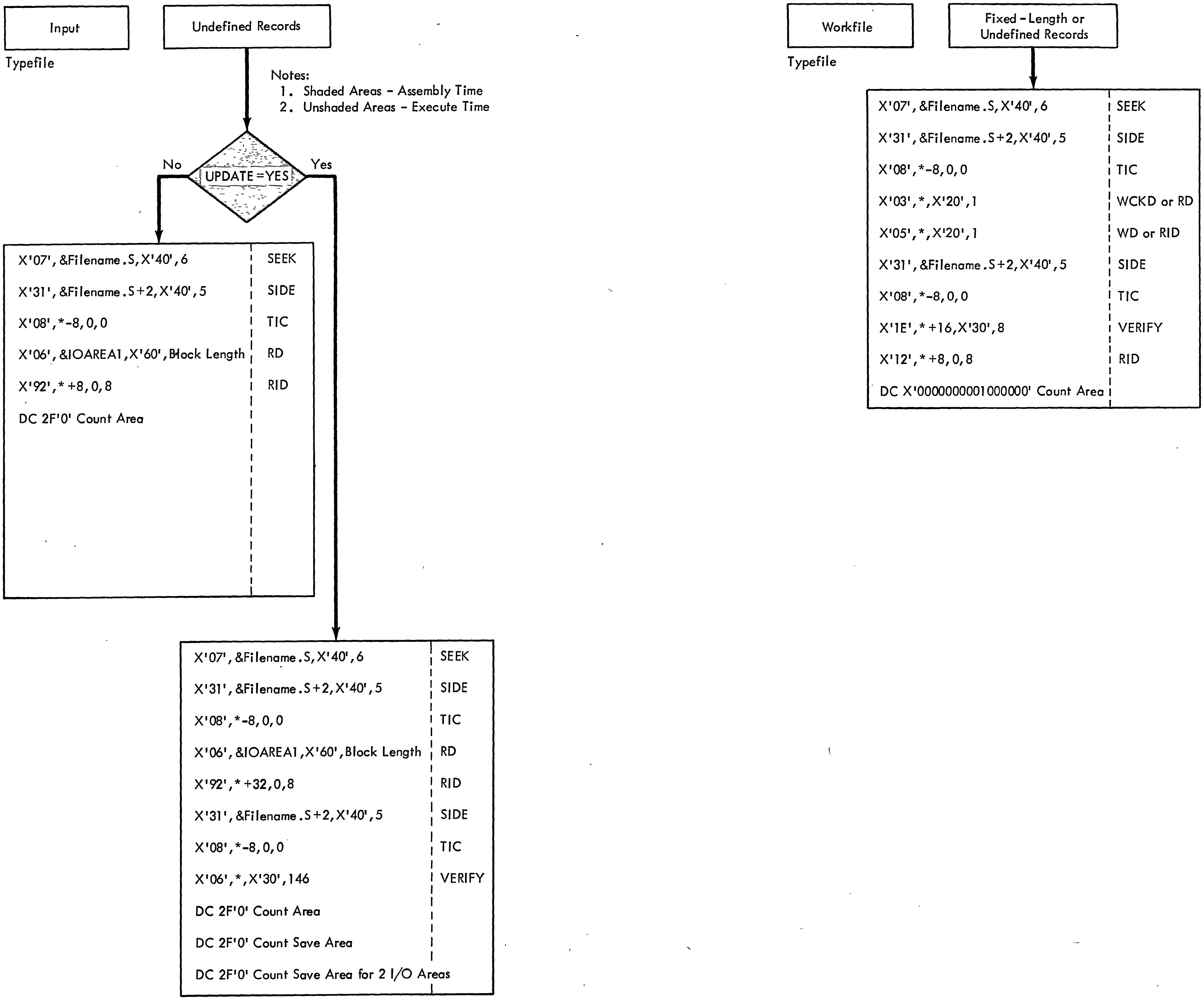
```
X'07', &Filename.S,X'40',6            SEEK
X'31', &Filename.S+2,X'40',5          SIDE
X'08',*-8,0,0                         TIC
X'1D', &IOAREA2,0,Block Length+8      WCKD
X'31', &Filename.S+2,X'40',5          SIDE
X'08',*-8,0,0                         TIC
X'30',*,X'30',1                       VERIFY
```

```
X'07', &Filename.S,X'40',6            SEEK
X'31', &Filename.S+2,X'40',5          SIDE
X'08',*-8,0,0                         TIC
X'1D', &IOAREA1,0,Block Length+8      WCKD
X'31', &Filename.S+2,X'40',5          SIDE
X'08',*-8,0,0                         TIC
X'30',*,X'30',1                       VERIFY
```

```
X'07', &Filename.S,X'40',6            SEEK
X'31', &Filename.S+2,X'40',5          SIDE
X'08',*-8,0,0                         TIC
X'06', &IOAREA1,X'40',Block Length    RD
X'92',*+8,0,8                         RID
DC 2F'0' Count Area
```

```
X'07', &Filename.S,X'40',6            SEEK
X'31', &Filename.S+2,X'40',5          SIDE
X'08',*-8,0,0                         TIC
X'06', &IOAREA1,X'40',Block Length    RD
X'92',*+32,0,8                        RID
X'31', &Filename.S+2,X'40',5          SIDE
X'08',*-8,0,0                         TIC
X'06',*,X'30',146                     VERIFY
DC 2F'0' Count Area
DC 2F'0' Count Save Area
DC 2F'0' Count Save Area for 2 I/O Areas
```

Figure 24. DTFSD Channel Programs (Part 2 of 3)



| | |
|---|---|
| X'07', &Filename.S,X'40',6 | SEEK |
| X'31', &Filename.S+2,X'40',5 | SIDE |
| X'08',*-8,0,0 | TIC |
| X'06', &IOAREA1,X'60',Block Length | RD |
| X'92',*+8,0,8 | RID |
| DC 2F'0' Count Area | |

| | |
|---|---|
| X'07', &Filename.S,X'40',6 | SEEK |
| X'31', &Filename.S+2,X'40',5 | SIDE |
| X'08',*-8,0,0 | TIC |
| X'06', &IOAREA1,X'60',Block Length | RD |
| X'92',*+32,0,8 | RID |
| X'31', &Filename.S+2,X'40',5 | SIDE |
| X'08',*-8,0,0 | TIC |
| X'06',*,X'30',146 | VERIFY |
| DC 2F'0' Count Area | |
| DC 2F'0' Count Save Area | |
| DC 2F'0' Count Save Area for 2 I/O Areas | |

| | |
|---|---|
| X'07', &Filename.S,X'40',6 | SEEK |
| X'31', &Filename.S+2,X'40',5 | SIDE |
| X'08',*-8,0,0 | TIC |
| X'03',*,X'20',1 | WCKD or RD |
| X'05',*,X'20',1 | WD or RID |
| X'31', &Filename.S+2,X'40',5 | SIDE |
| X'08',*-8,0,0 | TIC |
| X'1E',*+16,X'30',8 | VERIFY |
| X'12',*+8,0,8 | RID |
| DC X'0000000001000000' Count Area | |

Figure 24. DTFSD Channel Programs (Part 3 of 3)

## MODULE GENERATION MACROS

Logical IOCS provides a number of logical-file accessing routines called logic modules. These modules provide an interface between the user's processing function and physical IOCS. They are generated by module generation macros (SDMODxx), and are executed in response to imperative macro instructions issued by the problem program.

The sequential access DASD logic modules and module generation macros differ from other IOCS logic modules and module generation macros in that the characteristics of the sequential access DASD modules are separated into ten categories. Each category has a unique macro instruction associated with it. The categories vary according to record form (fixed, variable, or undefined length) and function (input, output, input with update function, and work files).

The user must define his file by the DTFSD macro and issue the appropriate module generation macro to process that file.

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 BA-BE

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.

### SDMODFI Without Truncation, GET Macro Charts BF-BJ

Objective: To read fixed-length blocked or unblocked records from a sequential DASD file without provisions for truncated records.

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 problem program in a work area, if one is specified.

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. In the case of a wrong-length record, this routine tests the residual count. 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 negative or zero. If negative, a wrong-length record exists. If zero, the record is a valid short record and is processed as a normal record.

If there are no errors, or after all errors are processed, the search argument is updated to the address of the next consecutive record in the file.

### SDMODFO With Truncation: PUT Macro Charts BK-BP

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 truncated blocks of records as well as fixed-length blocked, or unblocked records. A block can only be shortened by some multiple of the logical record size. Since various size records can be written, a count of the number of bytes remaining on the track must be kept.

The first PUT macro issued to the file initializes the count field in the I/O area with the search argument, and initializes the track capacity counter. If the user has specified a work area and the TRUNC switch is not on, this routine moves the logical record to the output area. It then tests the truncation switch to see if the TRUNC macro has been issued. If it was issued, no more logical records are built in the block. Otherwise, the current block size and I/O area address are updated by the logical record size, and a test determines if the end of the block has been reached. If not, control returns to the problem program.

If the block is full, or the truncation switch has been set, the module determines whether the block will fit on the track. If the entire block cannot be written in the space remaining on the track, the module updates the search argument and writes the block as the first record on the next track. If the extent upper limit is exceeded, end of file has not been reached, and there is no partial block to be written, this phase calls $$BOPEN to open the next extent, and to write the record as the first record on the new extent. Control then returns to the problem program.

This module also processes errors if the problem program has specified error routines. If ERROPT is specified, errors are processed as each record is written. If ERREXT is specified, additional errors are returned to the problem program for processing.

### SDMODFO With Truncation: Close Routine Chart BQ

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, $$BOSDC1.

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

Method: First, a test determines if there are any logical records to be written. If so, the routine sets a partial block switch and 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 set, and another branch-and-link is taken to the PUT routine to set up the end-of-file record. When this routine regains control, the data length is set to zero in the count field and the end-of-file record is written. This phase then exits to $$BOSDC1 via an SVC 9.

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 With Truncation: TRUNC Macro Chart BQ

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 Charts BR-BU

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, $$BOPEN 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.

### SDMODFO Without Truncation: Close Routine Chart BV

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, $$BOSDC1.

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 $$BOSDC1 via an SVC 9.

SDMODFU With Truncation:

GET Macro Charts BW-CF

PUT Macro Chart CG

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 problem program, and rewrites the blocked or unblocked records in sequence if the problem program has issued a PUT macro instruction. Because records are not removed from or added to the file, a GET macro instruction must precede a PUT macro instruction. Only one PUT macro instruction can be issued for a record. If a work area is specified, the module moves the logical record to and from the work area.

If blocked records are being processed, and the 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 record to be read or written. 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 (CCHHR), and the byte count field of the Read/Write Data CCW is initialized with the data length.

The second read operation reads the data area of the record, and then the count field of the next consecutive record in the file. Subsequent GET macros read the data area of a record and the count field of the next consecutive record.

If unblocked records are being processed, or the 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.

When a PUT macro is issued to the file, SDMODFU sets a switch in the DTF table to indicate that the PUT command has been issued. If a work area is specified, it moves the logical record to the output area. Control then returns to the problem program. The routines that write the record are incorporated within the GET macro logic.

The GET routine reads each record, and stores the address and length of the record for use by the output routine. If track hold has been specified, every read operation (except the first read count) reads a record and holds a track.

After the initial entry to the module, a test determines whether another record is needed. If not, the next logical record is made available to the problem program in a work area, if specified, and control returns to the problem program.

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 not on, the module reads the next record without performing a write operation. 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 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.

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

If ERROPT is specified and the problem program has specified error routines, errors are processed.

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

## SDMODFU Without Truncation:

### GET Macro Charts CH-CR

### PUT Macro Chart CS

Objective: To read fixed-length blocked or unblocked records to be updated (optionally) by the problem program, and rewrite the 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 and rewrites the records in sequence if the problem program has issued a PUT macro instruction. Because records are not removed from or added to the file, a GET macro instruction must precede a PUT macro instruction. Only one PUT macro instruction can be issued for a record. If a work area is specified, the module moves the logical record to and from the work area.

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 BW-CF

### Without Truncation Charts CH-CR

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.

## CNTRL Macro, Fixed-Length Records Chart CT

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 2311, 2314, 2319 and 2321 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.

## RELSE Macro, Fixed-Length Records Chart CT

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.

## VARIABLE-LENGTH RECORD MODULES

The basic modules for accessing variable length records are:

- SDMODVI - Variable-length input records.
- SDMODVO - Variable-length output records.
- SDMODVU - Variable-length input records for update.

Each module provides logical data-handling routines that work with one or more unique DTF tables to perform its specified functions. Each module can function using:

- A work area (required for spanned processing).
- 1 or 2 I/O areas.
- Error options (if specified at module generation time).
- ERET macro (if ERREXT is specified at module generation time).
- Blocked or unblocked (spanned or unspanned) variable-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 (only for input files with update function).

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

## SDMODVI: GET Macro Charts DA-DE

Objective: To read variable-length blocked or unblocked (spanned or unspanned) records from a sequential DASD file.

Entry: From a GET macro expansion.

Exit: To the problem program.

Method: For unspanned records, the first GET macro issued to the file results in two separate input operations. The module performs a read count operation to obtain the length of the first block of data. It initializes the Read Data CCW with the block length, and performs a second input operation to read the data area of the first record and the count field of the next sequential record in the file. Subsequent GET macros issued to the file read the data area and then the count field of the next sequential record.

After each physical record is read, the module determines if the problem program has specified two I/O areas, or a work area with unblocked records being processed. If either of these conditions exists, the module performs another input operation to read the data area of the next sequential record before returning control to the problem program.

As each GET macro is issued (other than the first), the module updates the input area address by the logical record size. If the end of the block has not been reached, the logical record data is moved to a work area if one is specified. If the end of the block is reached, the module performs another read operation, unless two I/O areas or a work area with unblocked records have been specified. In this case, the next record for processing has already been read.

Spanned processing is similar to unspanned processing except that spanned record segments are assembled into logical records in the user's work area. The user need not fill in the length field because either the module passes the length of the logical record through the register specified under RECSIZE in the DTF table, or (if no specification has been given) the module issues a current MNOTE and register 2 is assumed. Null segments are recognized but not assembled into logical records.

If end of file is reached, the module fetches $$BOPEN to exit to the problem program's end-of-file routine. This module also processes errors as each record is read, if the problem program has specified ERROPT as a parameter. If ERREXT is specified, additional errors are returned to the problem program.

If spanned processing is specified and ERROPT=SKIP, the entire spanned record or block of spanned records is skipped. Conversely, if ERROPT=IGNORE, the entire spanned record in which the error occurred is not skipped.

## SDMODVI: RELSE Macro Chart DE

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.

## SDMODVO: PUT Macro Charts DG-DP

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.

### SDMODVO: Close Routine Chart DT

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.

Method: If the problem program (with or without a work area) has specified two I/O areas, the close routine waits for the completion of the last write operation. It determines whether a record is to be written. If not, it sets the data length to zero and writes the end-of-file record. If there is a record to be written and blocking has been specified, this routine branches to the PUT macro logic to write the truncated record. Control returns to this routine which then writes the end-of-file record. The close routine exits to the close transient, $$BOSDC1.

### SDMODVO: TRUNC Macro Chart DU

Objective: To cause a truncated record to be written.

Entry: From the TRUNC macro expansion.

Exit: To the SDMODVO PUT logic.

Method: This routine sets the truncation switch to indicate that a TRUNC macro instruction has been issued. It stores the problem program registers, and branches to the SDMODVO PUT logic to write the truncated record.

### SDMODVU: GET Macro Charts DV-EF

Objective: To read variable-length blocked or unblocked records (spanned or unspanned) from a sequential DASD file that are to be updated optionally by the problem program.

Entry: From the GET macro expansion.

Exit: To the problem program.

Method: This logic module reads variable-length blocked or unblocked records (spanned or unspanned), and makes the logical record available to the problem program in a work area, if specified.

The first GET macro issued to the file results in two separate read operations. The first operation reads the count field to determine the length of the block to be read. The module initializes the Read Data CCW with the length, and initializes the search argument with the address of the block. The second operation then reads the data area of the first record and the count field of the next sequential record on the file. Subsequent GET macros then read the data portion of a record and the count field of the next sequential record.

If specified, the module stores the count field of each physical record in the DTF table to be used by a subsequent PUT macro if the record has been updated by the problem program.

As each GET macro is issued, the input area address is updated by the logical record size. If the end of the block has not been reached, the logical record is moved to a work area, if one is specified. A test determines whether there are two I/O areas. If so, the module tests to determine whether it is necessary to read another record. If it is necessary, the module performs another read operation. Control then returns to the problem program. If track hold is specified, each read data operation reads a record and holds the track via an SVC 35.

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.

When the end of the block is reached, the routine determines if the track hold option has been specified, and if it is necessary to free a track. If so, an SVC 36 is issued to free the held track. The routine then tests for two I/O areas. If there is only one I/O area, no overlap is possible, and the routine reads another record. If there are two I/O areas, the routine determines whether the next record has already been read. If not, it reads the record and performs a wait operation. Processing then continues on this record.

Spanned processing is similar to unspanned processing except that spanned record segments are assembled into logical records in the user's work area. Null segments are recognized, but not assembled. On each GET macro instruction, the pointer to the physical record in which the logical record begins is stored in the DTF table to be used for repositioning the device by a subsequent PUT instruction if the record is updated. The extent sequence number is also stored in the DTF table in case the logical record spans extents.

If the extent upper limit is reached, the routine processes all records or segments pertaining to that extent before fetching $$BOPEN to open the next extent. When end of file is reached, any records that have already been read are processed, and $$BOPEN is fetched to exit to the problem program's end-of-file routine.

### SDMODVU: PUT Macro Charts EG-EM

Objective: To write variable-length blocked or unblocked records (spanned or unspanned) that were updated optionally by 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, $$BOSDC1.

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, $$BOSDC1. 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 CCB.

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 EN

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: 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 EN

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.

### CNTRL Macro, Variable Length Records Chart EP

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 2311, 2314, 2319, and 2321, or a restore operation on a 2321. The routine waits for the completion of any previous I/O operation. It 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 exits to the problem program.

## UNDEFINED LENGTH RECORD MODULES

The basic modules for accessing undefined records are:

- SDMODUI - Undefined length input records.
- SDMODUO - Undefined length output records.
- SDMODUU - Undefined length input records for update.

Each module provides logical data-handling routines that work with one or more unique DTF tables to perform its specified 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 specified at module generation time).
- Unblocked records only.
- 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, (only for input files with update function).

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

### SDMODUI: GET Macro Charts EQ-EU

Objective: To read undefined length records from a sequential DASD file.

Entry: From the GET macro expansion.

Exit: To the problem program.

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

The first time the module is entered, a first-time switch is turned on and the DTF is initialized. On each subsequent entry, the module determines if two I/O areas or a work area is specified. If either is specified, the module issues an SVC 7 (WAIT) to ensure that the previous GET operation is complete. If neither, it performs a read operation.

The read operation determines the address and length of the data area of the record. It compares the data length of the record to the defined block size in the DTF table. If the data length is less than the block size and the first record is being processed, a Read Count CCW is executed, and a test determines if the record length is correct. If the length is incorrect, the module initializes the Read Data CCW with the block size and sets the incorrect length indicator in the CCB. The second and subsequent GET macros read the data area of the record and then the count field of the next consecutive record.

If the extent upper limit is exceeded, this module issues an SVC 2 to fetch $$BOPEN, which opens the next extent for the file. If end of file is found, $$BOPEN is fetched to give control to the user's end-of-file routine. Errors are processed if ERROPT is specified as a parameter and the user specifies an address of an error routine. If ERREXT is specified, additional errors are returned to the problem program for processing.

Finally, the module determines if two I/O areas, or a work area is specified. If either is specified, the module executes the Read Data CCW and returns control to the problem program. If neither is specified, control returns to the problem program.

## SDMODUO: PUT Macro Charts EV-EZ

Objective: To write records on a DASD file in sequential order.

Entry: From the PUT macro expansion.

Exit: To the problem program.

Method: The routine determines if this is the first entry. If it is , the first-time switch is turned on. If it is not, a test determines if two I/O areas or a work area are specified in the DTF statement. If either is specified, the module issues an SVC 7 (wait for completion of I/O). If 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 FA

Objective: To write an EOF record.

Entry: From the close transient, $$BOSDC1.

Exit: To the close transient, $$BOSDC1 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 Charts FB-FC

PUT Macro Chart FD

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 macro expansion.
- From the 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.

CNTRL Macro, Undefined-Length Records Chart GA

Objective: To perform a nondata operation.

Entry: From a CNTRL macro expansion.

Exit: To the problem program.

Method: This routine is used for nondata operations on the file. For a 2311, 2314, 2319, and 2321, the control operation seeks to the address specified in the DTF table. For the 2321 data cell, the operation also returns a strip to the subcell.

The routine puts the symbolic device address in the control CCB, moves the seek address to the CCW, performs the control operation, and branches back to the problem program.

## WORK FILE MODULE

Logical IOCS macros READ, WRITE, CHECK, NOTE, POINTS, POINTR, and POINTW access the work file module. A work file can be used for input, output, or both.

The CHECK macro must be issued after a READ or WRITE macro to ensure completion of the operation before issuing another instruction. The NOTE macro is used with the POINTR or POINTW macros to position the file at a predetermined record. The POINTS macro positions the file to the beginning of the file.

GET and PUT macros are not used with the work file module. The work file module does not support blocking and deblocking, or automatic I/O area switching.

The work file module provides routines that work with one or more unique DTF tables to perform the specified functions. The module can function using:

- CNTRL macro--control function (if specified at module generation time).
- Error options (if specified at module generation time).
- ERET macro (if specified at module generation time).
- Fixed-length unblocked records or undefined length records.
- Update function.
- Verify option.

- RDONLY option (if specified at module generation time).

- Track hold function (if specified at module generation time).

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

SDMODW: READ Macro Chart GB

Objective: To read all or part of a physical record from a sequential DASD work file.

Entry: From a READ macro expansion.

Exit: To the WRITE macro.

Method: The READ macro provides the user with the ability to access data on a work file. This macro requires the user to specify the area that the record is to be read into, and also allows the user to read only a portion of the record. Record deblocking is not handled by the READ macro, but is the responsibility of the problem program.

For undefined records, the first READ macro issued to the file results in two separate input operations. The module performs a READ COUNT to obtain the length of the first block of data. It initializes the READ DATA CCW with the block length and performs a second input operation to read the data of the first record and the count field of the next sequential record. Subsequent READ macros issued to the file will first read the data area and then read the count field of the next sequential record.

For FIXUNB records this routine initializes the Read CCW chain to perform a read data operation followed by a read count operation, in order to obtain the count field ID of the next sequential record on the file. If a POINTS macro has been issued before the READ macro, this routine reinitializes to read the first record on the file. If the problem program has specified both the track hold option and update option, the routine issues an SVC 35 to read the record and hold the track. Otherwise, this routine issues an SVC 0 to read the record.

If the letter 'S' is specified in the operand of the READ macro, the entire record is read. Otherwise, the actual length, as stated in the operand, is read. This parameter is only present in the case of records of undefined format.

This routine updates track and cylinder addresses, when all records of a track or cylinder have been read. If the extent upper limit is reached, $$BOPEN is fetched to open the next extent.

SDMODW: WRITE Macro Chart GC

Objective: To write a record on a DASD work file.

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 $$BOPEN 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 GE

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

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

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 Charts GF-GK

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 GL

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 NOTE macro instruction obtains the identification of the last physical record that was read or written in the specified file. The problem program must issue a CHECK macro before the NOTE macro to ensure that the last operation was completed satisfactorily.

The NOTE macro routine saves the current search address, which is loaded into register 1 just before returning to the problem program. If the NOTE macro was issued after a WRITE macro or the initial OPEN macro, this routine returns the remaining track capacity in register 0 to the problem program. After loading register 1, control returns to the problem program.

The identification in register 1 is returned in the form 0chr, where 0 = eight binary zeros, c = cylinder number, h = track number, and r = record number within the track; c,h, and r are binary numbers. The remaining track capacity returned in register 0 is in the binary form, 00nn.

## SDMODW: POINTR Macro Chart GL

Objective: To reposition a file in order to read a record previously identified by a NOTE macro instruction.

Entry: From the POINTR macro expansion.

Exit: To the problem program.

Method: This routine expands the record identification, previously supplied by the NOTE macro, to a search address. If the record identification does not fall within the current extent limits, the routine fetches $$BOPEN to open a new extent. It initializes the count field ID with the search address, and returns control to the problem program.

SDMODW: POINTW Macro Chart GL

Objective: To reposition a file in order to write a record following one previously identified by a NOTE macro instruction.

Entry: From the POINTW macro expansion.

Exit: To the problem program.

Method: This routine uses the record identification supplied by a previous NOTE macro to position the file in order to write a record immediately following the one identified by the NOTE macro.

The routine expands the 0chr identification to a DASD search address. If the identification does not fall within the current extent limits, $$BOPEN is fetched to open the next extent. The routine updates the count field ID and initializes the remaining track capacity field in the DTF table with the value supplied by the NOTE macro. (This value is available only if the NOTE macro followed a WRITE macro.) Control then returns to the problem program.

The POINTW macro may be followed by a WRITE macro only if the space remaining on the track is available.

SDMODW: POINTS Macro Chart GM

Objective: To reposition a file to the beginning of the file for the next read or write operation.

Entry: From the POINTS macro expansion.

Exit: To the problem program.

Method: This routine reinitializes the search address and the count field ID to the lower limit of the first extent supplied for the file. Control then returns to the problem program.

SDMODW: FREE Macro Chart GM

Objective: To free a track if the track hold option has been specified.

Entry: From the FREE macro expansion.

Exit: To the problem program.

Method: This routine determines whether the track hold option has been specified. If so, it obtains the address and length of the record on the held track and initializes the CCW chain with that information. The routine then issues an SVC 36 to free the track. Control returns to the problem program.

SDMODW: CNTRL Macro Chart GM

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 2311, 2314, and 2319. 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 GN

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

Entry: From $$BOSDEV.

Exit: To $$BOSDEV 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, as well as information on the simplified DLBL and EXTENT statements provided by Version 3. Job control stores this DASD label information on the SYSRES DASD label information cylinder. The open monitor logical transient, $$BOPEN prepares to read the label information from the SYSRES label information cylinder into the logical transient area, and then fetches $$BOSD00.

The sequential DASD open logical transients read the DASD label information from SYSRES into main storage. Figure 2 illustrates the format of the SYSRES DASD label information. 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.

Chart 01. Sequential Access DASD Open, General Flow

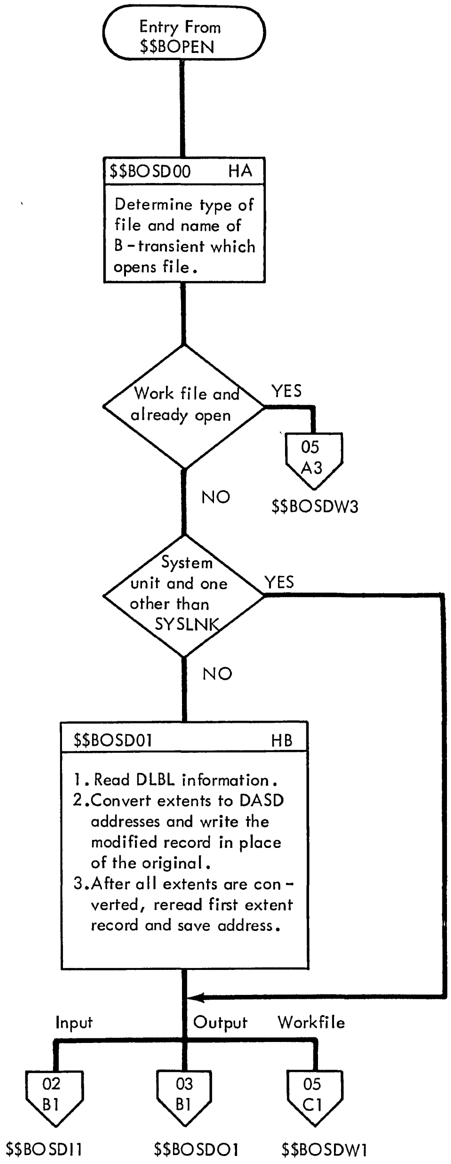

Entry From $$BOPEN
$$BOSD00 HA
Determine type of file and name of B-transient which opens file.
Work file and already open
YES
05 A3
$$BOSDW3
NO
System unit and one other than SYSLNK
YES
NO
$$BOSD01 HB
1. Read DLBL information.
2. Convert extents to DASD addresses and write the modified record in place of the original.
3. After all extents are converted, reread first extent record and save address.
Input
Output
Workfile
02 B1
03 B1
05 C1
$$BOSDI1
$$BOSDO1
$$BOSDW1

Chart 02. Sequential Access DASD Open, Input Files

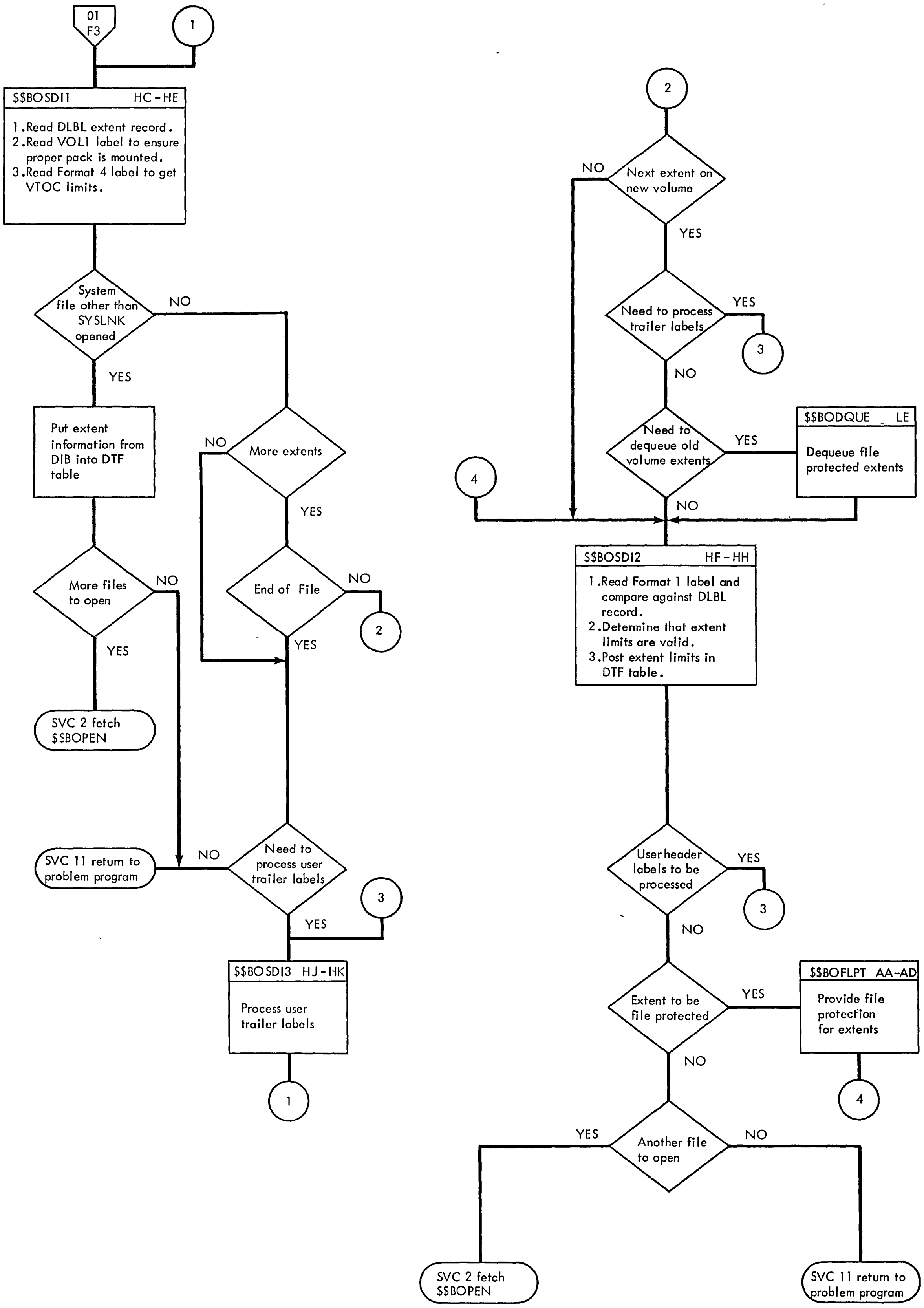

01
F3
1
$$BOSD11 HC-HE
1.Read DLBL extent record.
2.Read VOL1 label to ensure proper pack is mounted.
3.Read Format 4 label to get VTOC limits.
System file other than SYSLNK opened
NO
YES
Put extent information from DIB into DTF table
NO
More extents
YES
More files to open
NO
YES
End of File
NO
2
YES
SVC 2 fetch $$BOPEN
SVC 11 return to problem program
NO
Need to process user trailer labels
3
YES
$$BOSD13 HJ-HK
Process user trailer labels
1
2
NO
Next extent on new volume
YES
Need to process trailer labels
YES
3
NO
$$BODQUE LE
Need to dequeue old volume extents
YES
Dequeue file protected extents
4
NO
$$BOSD12 HF-HH
1.Read Format 1 label and compare against DLBL record.
2.Determine that extent limits are valid.
3.Post extent limits in DTF table.
User header labels to be processed
YES
3
NO
$$BOFLPT AA-AD
Extent to be file protected
YES
Provide file protection for extents
NO
4
YES
Another file to open
NO
SVC 2 fetch $$BOPEN
SVC 11 return to problem program

Chart 03. Sequential Access DASD Open, Output Files (Section 1 of 2)

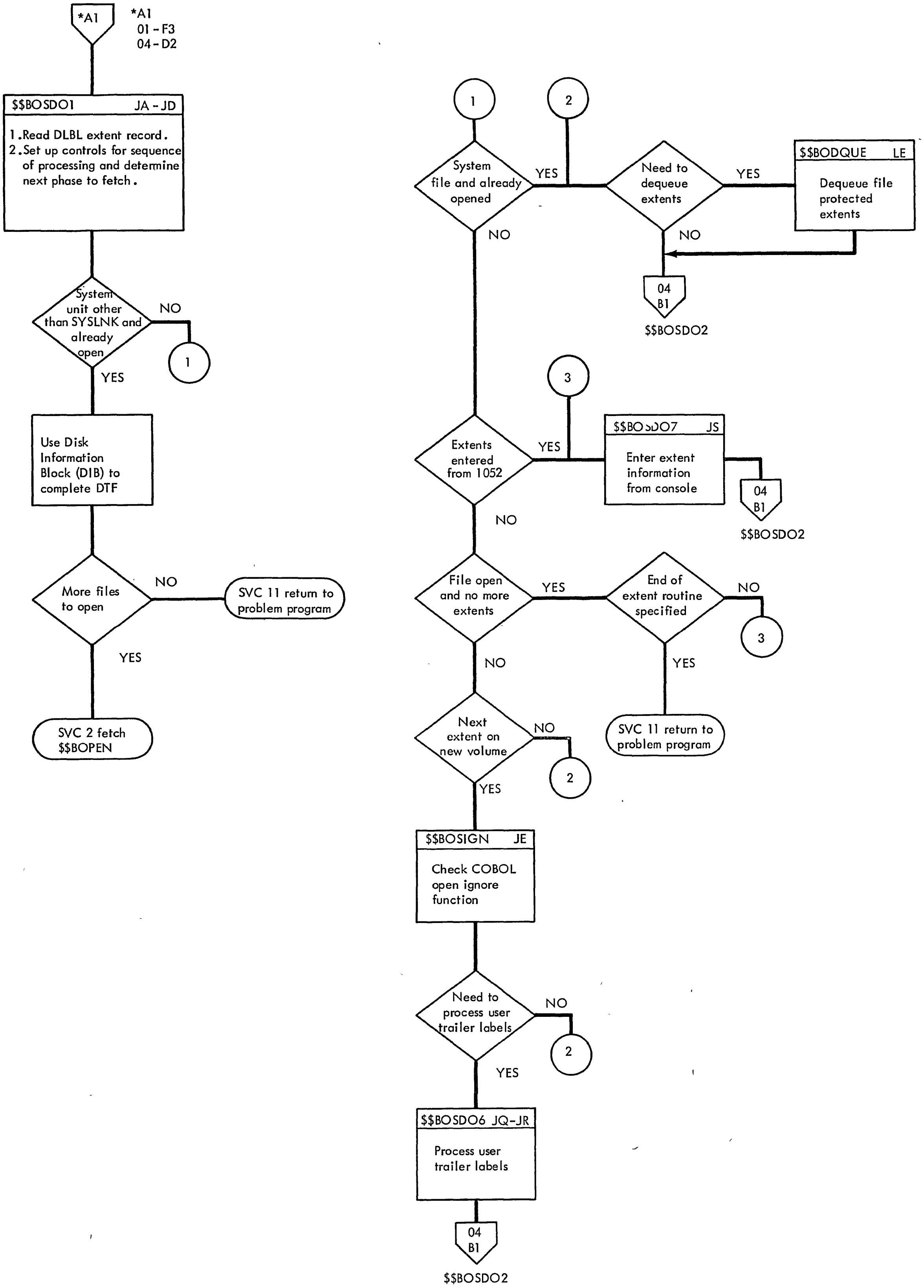

*A1
*A1
01 - F3
04 - D2
$$BOSDO1 JA - JD
1.Read DLBL extent record.
2.Set up controls for sequence of processing and determine next phase to fetch.
System unit other than SYSLNK and already open
NO
1
YES
Use Disk Information Block (DIB) to complete DTF
More files to open
NO
SVC 11 return to problem program
YES
SVC 2 fetch $$BOPEN
1
2
System file and already opened
YES
NO
Need to dequeue extents
YES
NO
$$BODQUE LE
Dequeue file protected extents
04
B1
$$BOSDO2
3
Extents entered from 1052
YES
NO
$$BOSDO7 JS
Enter extent information from console
04
B1
$$BOSDO2
File open and no more extents
YES
NO
End of extent routine specified
NO
3
YES
SVC 11 return to problem program
Next extent on new volume
NO
2
YES
$$BOSIGN JE
Check COBOL open ignore function
Need to process user trailer labels
NO
2
YES
$$BOSDO6 JQ-JR
Process user trailer labels
04
B1
$$BOSDO2

Chart 04. Sequential Access DASD Open, Output Files (Section 2 of 2)

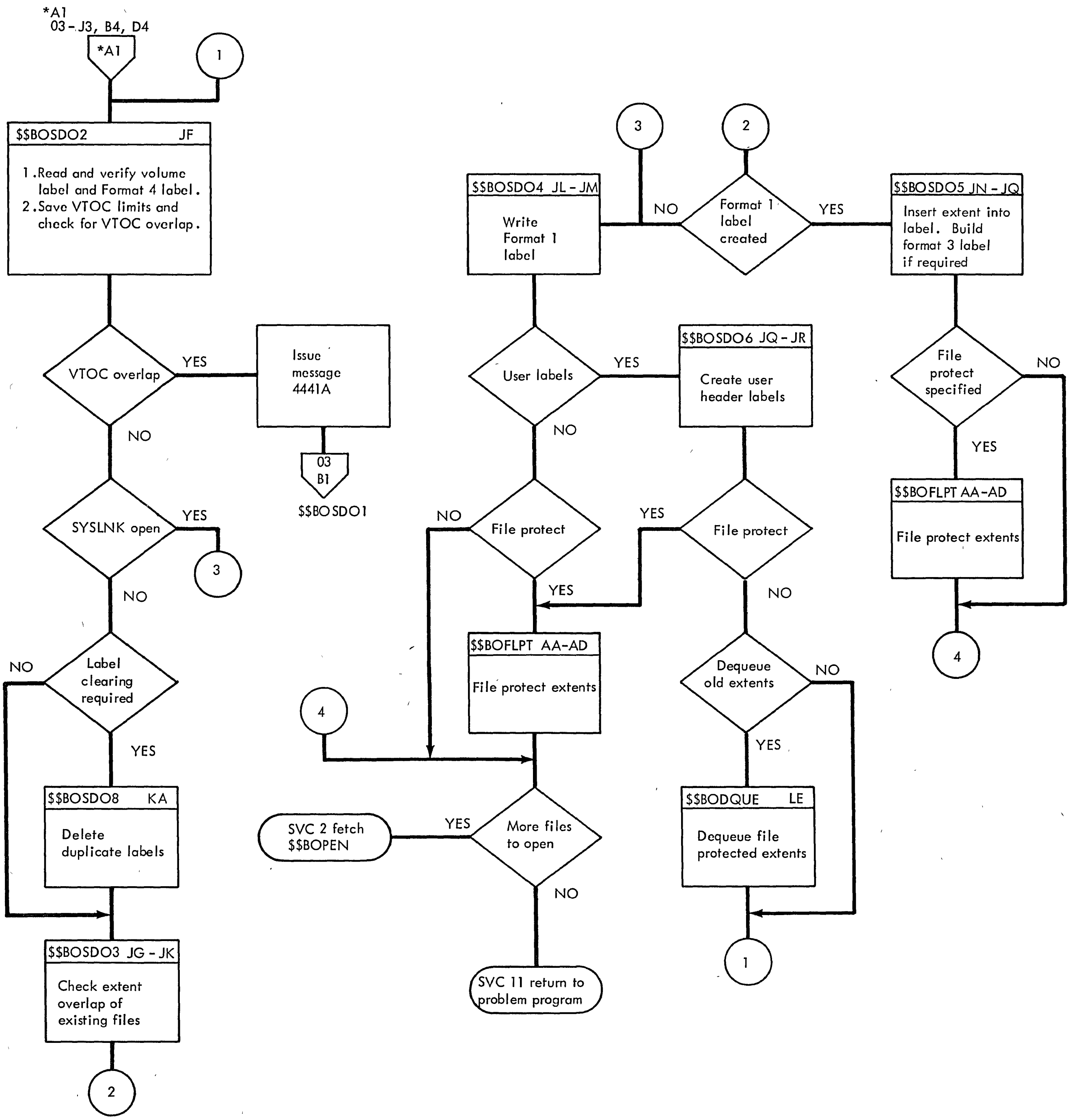


Chart 05. Sequential Access DASD Open, Work Files

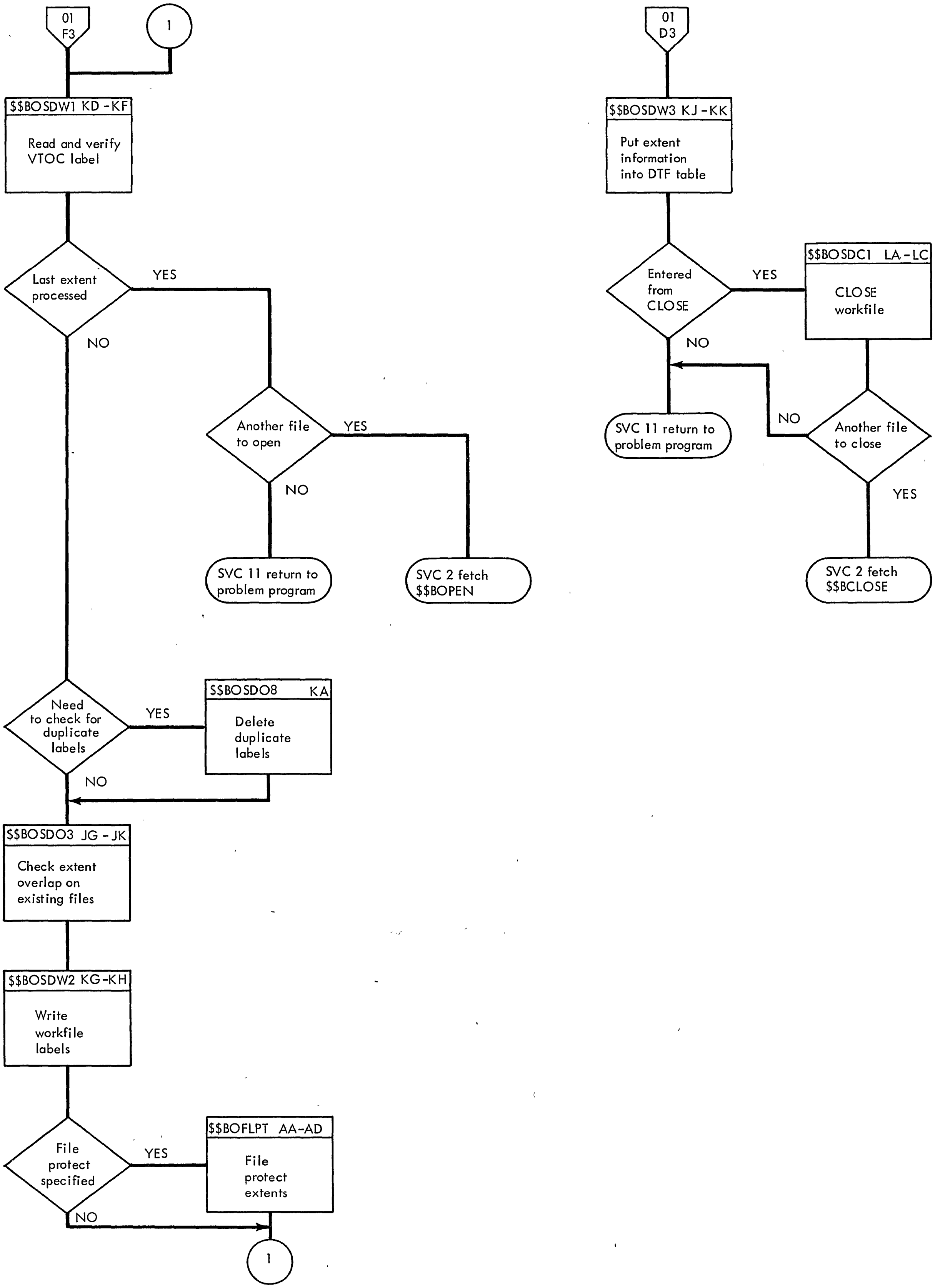
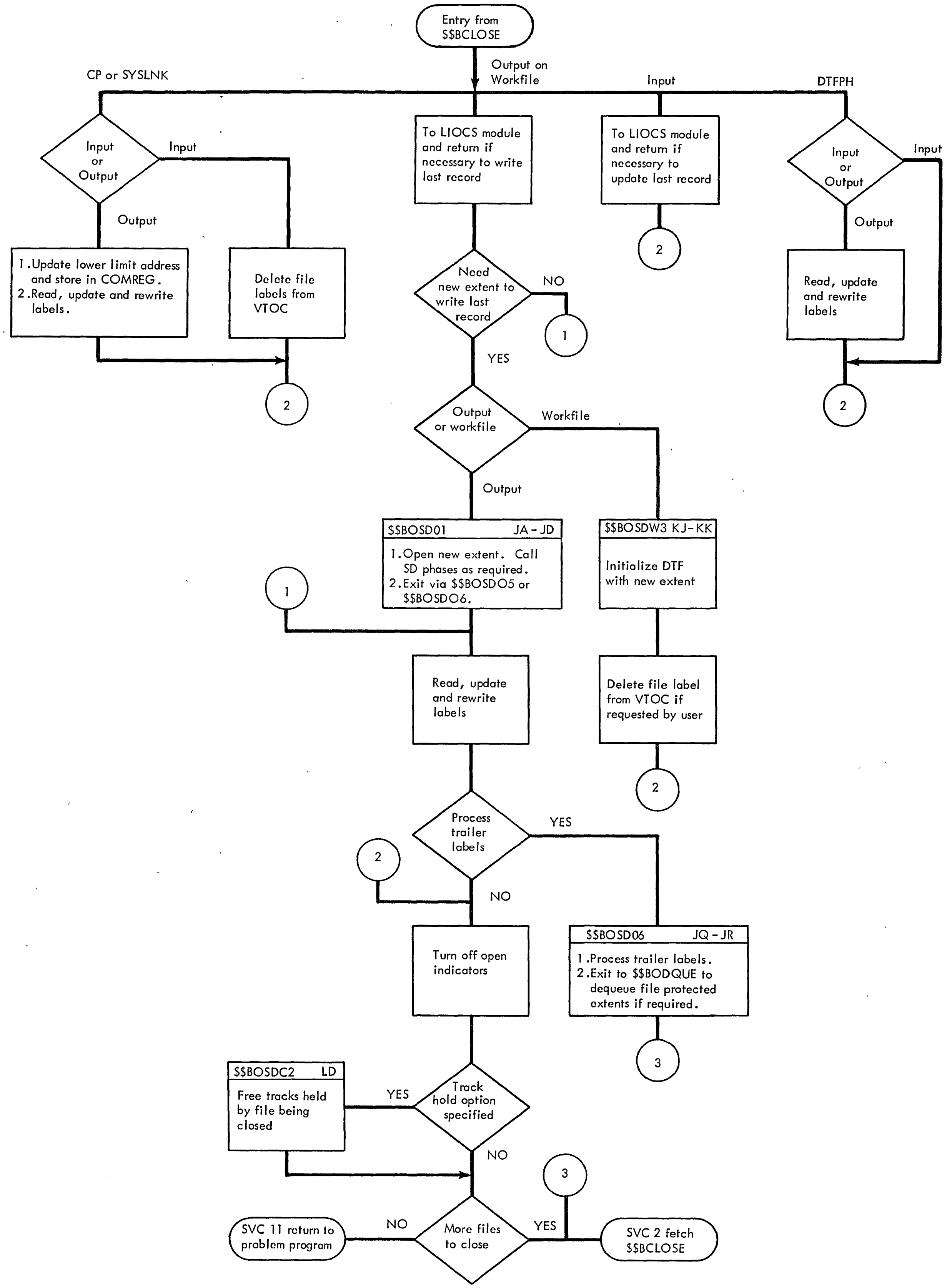


Chart 06. Sequential Access DASD Close, All Files


Entry from $$BCLOSE
CP or SYSLNK
Output on Workfile
Input
DTFPH
Input or Output
Input
Output
1.Update lower limit address and store in COMREG.
2.Read, update and rewrite labels.
Delete file labels from VTOC
2
To LIOCS module and return if necessary to write last record
Need new extent to write last record
NO
1
YES
Output or workfile
Workfile
Output
$$BOSD01 JA-JD
1.Open new extent. Call SD phases as required.
2.Exit via $$BOSDO5 or $$BOSDO6.
1
Read, update and rewrite labels
$$BOSDW3 KJ-KK
Initialize DTF with new extent
Delete file label from VTOC if requested by user
2
To LIOCS module and return if necessary to update last record
2
Input or Output
Input
Output
Read, update and rewrite labels
2
Process trailer labels
YES
2
NO
Turn off open indicators
$$BOSD06 JQ-JR
1.Process trailer labels.
2.Exit to $$BODQUE to dequeue file protected extents if required.
3
$$BOSDC2 LD
Free tracks held by file being closed
YES
Track hold option specified
NO
3
SVC 11 return to problem program
NO
More files to close
YES
SVC 2 fetch $$BCLOSE

## SEQUENTIAL DASD OPEN/CLOSE LOGIC

### Open/Close Sequential DASD Files  Chart 01

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 02

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.

### OPEN (Output Sequential DASD) Charts 03-04

If the file has not been previously opened, the extents and labels are checked against the DLBL cards. The labels for the next extent to be opened are read and checked for overlap. Labels are created and written as directed by the DLBL information, pertinent information is posted in the DTF table, user labels are processed at the option of the user, and control returns to the user.

When the file has been previously opened, the labels and extents are checked, the file labels are created, and the current extents are inserted in the VTOC. The pertinent data is posted in the DTF table, user labels are checked, (if specified) and control branches to the user.

### OPEN (Work File Sequential DASD)  Chart 05

When WORKFILE is specified, the volume and EXTENT labels are checked against the DLBL information, the VTOC limits are saved, and the extents are checked for overlap on the VTOC. File labels that overlap are deleted if the expiration date is passed. Format 1 and 3 labels are created as required. Phase 1 is called in to process more extents, if available. If not, control returns to user.

### CLOSE Sequential DASD  Chart 06.

The close routine for sequential DASD work files and output files is logically the same. Any additional records are processed, the file labels are updated or deleted as required, and the file is indicated as being closed. A compiler file is closed in the same manner as a work file or an output file, except that processing of additional records is bypassed. If the file is an input file, it is simply indicated as being closed.

### $$BOSD00: SD Open, Initialization Chart HA

Objectives:

- To determine which B-transient phase to fetch following $$BOSD01.
- To bypass $$BOSD01 (if possible), by fetching the determined phase.
- To perform other initialization functions.

Entry: From $$BOPEN.

Exit: To $$BOSD01, $$BOSDI1, $$BOSDO1, $$BOSDW1, $$BOSDW3.

Method: If the specified file is a work file and is already open, this routine goes directly to $$BOSDW3. For system units other than SYSLNK, it goes to $$BOSDI1, $$BOSDO1, or $$BOSDW1 for input, output, or work file, respectively. For all other files, it goes to $$BOSD01 after first determining the maximum allowable seek address.

### $$BOSD01: SD Open, DLBL Extents Chart HB

Objective: To convert DLBL extents to DASD addresses and to indicate those extents which already exist as label information.

Entry: From $$BOSD00, and reentry from $$BOMSG1.

Exits:

- To $$BOSDI1, $$BOSD01, or $$BOSDW1 as predetermined by phase $$BOSD00.
- To $$BOMSG1 for operator communication.

Method: This routine gets the address of the DLBL information as supplied by the open monitor, $$BOPEN. It converts each extent to DASD addresses and writes the converted record in place of the original. When the last extent has been converted, this routine rereads the first extent record, saves its address, and exits to the next phase.

### $$BOSDI1: SD Open Input, DLBL Extents Charts HC-HE

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 and reentry from $$BOSDI2, $$BOSDI3, and $$BOMSG1.

Exits:

- To $$BOSDI2 to continue OPEN processing.
- To $$BODQUE to dequeue old extents.
- To $$BOSDI3 to process trailer labels.
- To $$BOPEN if the last DLBL extent has been processed and another file is to be opened.
- To $$BOMSG1 for operator communication.

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). 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 Charts HF-HH

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 $$BODSMW to print message if data secured file is uncountered.
- To $$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 Charts HJ-HK

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.

Exits:

- To $$BOSDI1 to continue OPEN processing.
- To the user's label routine.
- To $$BOFLPT if file protect has been specified.
- To $$BCLOSE if the phase was entered from the close routine and track hold has not been specified.
- To $$BOSDC2 if the phase was entered from the close routine and track hold has been specified.
- To $$BOMSG1 for operator communication.
- To the problem program following an end-of-file condition.

Method: This phase reads the user's labels and passes them to the problem program for processing. This process continues until the maximum number of labels have been read (and rewritten, if any labels have been updated by the user) or the user signals that he does not want to process any more labels.

If this phase is entered from the close routine, and if track hold is specified, control passes to $$BOSDC2. If track hold is not specified, control passes to $$BCLOSE. If the routine is entered from $$BOSDI1 or $$BOSDI2, control passes to $$BOSDI1.

When the maximum number of labels have been processed, end-of-file is reached, or the user signals that he does not want to process any more labels, control returns to the problem program.

## $$BOSDI4: SD Open Input, Initialization of DTF Table Charts HL-HM

Objective: To complete initialization of the DTF table.

Entry: From $$BOSDI2.

Exits:

- To $$BOSDI3 to process user header labels.
- To $$BOPEN to open the next parameter or to return to the user.
- To $$BOFLPT to queue the extent for file protection.
- To $$BOQ001 to open a QTAM file.

Method: Upper and lower extent limits are posted in the DTF, and the DTF open switch is set on. Switches are also set to indicate whether the last extent has been processed, if the next extent is on a new volume, or if it is on a new pack and the extents need dequeueing.

If user header labels are to be processed, exit is made to $$BOSDI3. If file protect is specified for a system file open, $$BOFLPT is fetched. For a QTAM open, $$BOQ001 is fetched. If any parameters remain to be processed, $$BOPEN is fetched; otherwise control returns to the user.

## $$BOSDO1: SD Open Output, Control Charts JA-JD

Objective: To control the sequence of operations for opening each file extent, providing for an entry of extents from the console, and for an entry of user trailer labels.

Entry: From $$BOSD00, $$BOSD01, and reentry from $$BOSDO2, $$BOMSG1, $$BOSIGN or $$BOSDC1. The phase is entered at least once for each DLBL extent.

Exits:

- To $$BOSDO2 to read and verify the volume label (VOL 1) and VTOC label (Format 4) and prevent any extent from overlapping the VTOC.
- To $$BOSDO6 to provide for user trailer labels if specified and if the last extent for the file has been processed.

- To $$BOSDO7 to allow the operator to enter extents from the console following the last DLBL extent for the file.

- 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.

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.

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.

### $$BOSIGN: SD Open Ignore Chart JE

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

Entry: From $$BOSDO1.

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 $$BOSDO1 to continue processing.

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

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 JG-JK

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.

Exits:

- To $$BOSDO4 to build a file label.
- To $$BOSDO5 to insert the extent in a file label.
- To $$BOMSG1 for operator communication.
- To $$BOSDW2 if this phase is used for opening a work file.
- To $$BOIS03 if this phase is being used by the Indexed Sequential File Management System (refer to Volume 4).

Method: This routine checks the incoming extent against all of the existing files in the VTOC for any overlap. If overlap occurs, the file label in the VTOC is deleted if the expiration date has been reached. If the expiration date has not been reached, a message is issued and action is taken depending on the operator's response.

This phase is also used by the open work file routines and the indexed sequential file management system to check extents for overlap and data security.

### $$BOSDO4: SD Open Output, File Label Charts JL-JM

Objective: To build a Format 1 label, insert the first extent into the label, write it out, and update the DTF table.

Entry: From $$BOSDO2 (SYSLNK open) or $$BOSDO3.

Exits:

- To $$BOSDO6 if user labels are specified.
- To $$BOFLPT if file-protect is specified.
- To $$BOPEN if the open processing for the file is complete and another file remains to be processed.
- To $$BOMSG1 for operator communication.

Method: This routine builds a Format 1 label for the file, inserts the first extent into the label, writes it out, and posts the pertinent information into the DTF table. Then, it tests to determine if user header labels are to be processed. If yes, it fetches $$BOSDO6. If no, a test determines whether the extent is to be file-protected. If yes, this routine fetches $$BOFLPT. If no, it exits to $$BOPEN to open the next file, or returns to the problem program if no more files are to be opened.

### $$BOSDO5: SD Open Output, Format 3 Label Charts JN-JP

Objective: To insert each extent into its file label, building a Format 3 label if required, and to update the DTF table.

Entry: From $$BOSDO3.

Exits:

- To $$BOFLPT if file-protect is specified.
- To $$BOPEN if the open processing for the file is complete and another file remains to be opened.
- To $$BOSDC1 if the open processing was entered from $$BOSDC1.
- To $$BOMSG1 for operator communication.

Method: This routine inserts each successive extent into its appropriate Format 1 label. If more than three extents are specified, it is necessary to build one or more Format 3 labels. This routine posts appropriate extent information in the file DTF table.

### $$BOSDO6: SD Open Output, User Labels Charts JQ-JR

Objective: To create user header and trailer labels.

Entry: From $$BOSDO1 or $$BOSDC1 to allow user trailer labels, from $$BOSDO4 to allow user header labels, reentry from $$BOMSG1, reentry from user label routine via SVC 9.

Exits:

- To user's label routine via SVC 8.
- To $$BOSDO2 to continue processing extents.
- To $$BOFLPT if file-protect is specified.
- To $$BODQUE if old extents are to be dequeued.

- To $$BOPEN if another file is to be opened.

- To $$BCLOSE if entry was from $$BOSDC1.

- To $$BOMSG1 for operator communication.

Method: This routine determines whether header labels or trailer labels are required. If trailer labels, it sets an asterisk in the DTF table for testing by the user. This routine finds the file mark that precedes the user label location. It then issues an SVC 8 to exit to the user-supplied address. It writes the user-supplied label when control returns from the user. $$BOSD06 allows up to eight user labels unless the user indicates otherwise. It writes a file mark following the last label.

## $$BOSD07: SD Open Output, Extents from Console Chart JS

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

Entry: From $$BOSD01, or $$BODSPW.

Exits:

- To $$BOSD02 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 1052 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 $$BOSD02 to process it.

## $$BOSD08: SD Open Output, Delete Label Chart KA

Objective: To prevent creation of identical file labels.

Entry: From $$BOSD02, $$BOSDW1, $$BODA01, or $$BOIS03, and reentry from $$BOMSG1.

Exits:

- To $$BODA01 for Direct Access Method.

- To $$BOSD03 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.

## $$BOSDW1: SD Open Work File, Volume Label Charts KD-KF

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, $$BOSDW2, and return from $$BOMSG1.

Exits:

- To $$BOSD03 to continue processing a work file extent.

- To $$BOSD08 to prevent duplicate file labels.

- To $$BOPEN if the last extent has been processed and another file remains to be opened.

- To $$BOMSG1 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 2311, 2314 or 2319, 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, $$BOSD08 is fetched to eliminate possible duplication. Subsequent exits are to $$BOSD03. (See $$BOSD03 SD Open Output, Extent Overlap.)

## $$BOSDW2: SD Open Work File, File Label Charts KG-KH

Objective: To create a file label (Format 1 and Format 3 labels as required).

Entry: From $$BOSD03, and return from $$BOMSG1.

Exits:

- To $$BOFLPT if file-protect is specified.

- To $$BOSDW1 as the normal exit.

- To $$BOMSG1 for operator communication.

Method: This routine builds a Format 1 label, inserting the first extent. This routine puts the extent information into the DTF and writes the label.

For each succeeding extent, this phase inserts the new extent (three maximum) and rewrites the label. For additional extents, it creates and writes a Format 3 label.

### $$BOSDW3: SD Open Work File, Extent to DTF Charts KJ-KK

Objective: To get extent information for the DTF table as requested by a POINT macro instruction or as required by a need to write a record beyond the limits of the current extent.

Entry: From $$BOSD00 or $$BOSDC1, with reentry from $$BOMSG1.

Exits:

- To $$BOSDC1 if entered from that phase.

- To the problem program.

Method: This phase reads the file label (Format 1) and determines whether the label has been deleted. If this open was not initiated by a POINTR or a POINTS macro, this routine uses the extent sequence number to find the next extent. If the open was initiated by a POINTR or a POINTS macro, a new extent low limit was put in the DTF. This phase uses the new low limit to find the next extent. It then enters the extent information in the DTF.

### $$BOSDC1: SD Close Charts LA-LC

Objective: To allow writing the last block of data, provide a linkage to a user's trailer label routine, and indicate in the DTF table that the file is closed.

Entry: From $$BCLOSE and reentry from $$BOSDW3, $$BOSD05, or $$BOSD06.

Exits:

- To the problem program upon completion of the close.

- To $$BCLOSE if another file remains to be closed.

- To $$BOSD01 if an additional file extent is required for writing the last block of data in an output file.

- To $$BOSDW3 if an additional file extent is required for writing the last block of data in a work file.

- To $$BOSD06 for processing user labels.

- To the LIOCS logic module if a last data block must be written or updated.

- To $$BOMSG1 for operator communication.

Method: On the basis of file type and usage, this routine determines whether a last block of data must be processed. If so, it enters the SD logic module. Upon return, it determines whether another extent must be opened to provide for the actual write operation. For this, it goes to $$BOSDW3 for a work file or to $$BOSD01 for an output file.

After the last data block is written on an output file, it deletes the labels, if so specified. Otherwise, it updates and rewrites work file or output file labels.

It also indicates in the DTF table that the file is closed and restores the unit exception indicator. If the track hold option has been specified, it issues an SVC 2 to fetch $$BOSDC2 to free any tracks held by the file being closed.

### $$BOSDC2: SD Close: Free Track Function Chart LD

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

Entry: From $$BOSDC1.

Exits:

- To the close monitor, $$BCLOSE.

- To $$BCISOA for ISAM files.

- To the problem program.

Method: This routine searches the track hold table to determine whether a track is being held by the file being closed. If so, an SVC 36 is issued to free the track.

If another 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.

### $$BODQUE: Dequeue Extent JIBs Chart LE

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 $$BOSDO1, $$BOSDO6, or $$BOSDI1 to the label DEQUERTN.

Exit: To the problem program if no files remain to be opened, or to the open monitor, $$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 $$BODQUE 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 LUB entry to determine if any JIBs are chained to the LUB.

If JIBs are chained to the LUB; that is, if the second byte of the LUB is not hex 'FF', the address of the first JIB in the chain is calculated by adding the pointer (byte 2 of the LUB) multiplied by 4 (the length of a JIB entry) to the starting address of the JIB table.

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

When all the chained JIBs have been checked, or if no JIBs are chained to the LUB, 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 open monitor, $$BOPEN, if the name of the calling phase was not supplied and there is another file to be opened. If the name of the calling phase was not supplied and there are no other files to be opened, phase $$BODQUE returns control to the problem program via an SVC 11.

### $$BOSDEV: SD Close Charts LF-LG

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

Entry:

- From the FEOVD macro
- From $$BOSDO5 (phase 5 of open sequential output)
- From LIOCS via SVC 9

Exits:

- To the open monitor $$BOPEN
- To the close phase $$BOSDC2
- 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.

Direct Access (DA) files refer to files contained on IBM 2311, 2314, 2319, or 2321 DASD devices and processed by the Direct Access Method. Note that the term Direct Access applies to a method of processing DASD records and not to a type of file organization.

## DIRECT ACCESS METHOD

The Direct Access Method provides a flexible set of macro instructions for creating and maintaining a data file on a DASD device. This technique applies specifically to records organized in a random order, but it can also be used to process records sequentially. The macro language offered by this data management method permits the user to load, read, write, update, add, or replace records on a DASD file.

The Direct Access Method is an IOCS processing method specifically designed to utilize the capabilities of direct access storage devices. This method provides the following facilities:

- Processing of records organized in a random order.
- Processing, in physical sequence, of a file of records stored by record key.
- Utilizing of track capacities.
- Two referencing methods:
  1. Record ID (physical track and record address),
  2. Record KEY (control field of the logical record).
- Multiple track searching beyond the specified track for resolving the key argument.
- Providing a means of supplying the user with the Record Identifier (ID) of either the current record or the next record after a READ or a WRITE operation has been executed.

The Direct Access Method is subject to the following restrictions:

- Only unblocked records are processed.
- No work area and only one I/O area can be specified for the file.
- The user must supply either a track reference or a record identifier for every record read or written by logical IOCS.

DASD files processed by the Direct Access Method must be defined for logical IOCS by a DTFDA macro. If a DASD file is processed by physical IOCS in a manner similar to the Direct Access Method, the file must be defined by a DTFPH macro.

## DTFDA MACRO

Whenever a file of DASD records is processed by the Direct Access Method, the logical file must be defined by a DTFDA macro. This macro generates a partial DTF table to describe the characteristics of the file for logical IOCS as shown in Figure 25. The DTF table is completed by the channel program builder subroutine in the DA logic module. This subroutine builds, and inserts into the DTF table, the channel program CCWs needed to process the file. The number and specific nature of the CCWs varies with the imperative macros used with the file. Figure 36 summarizes the CCW chains needed to accomplish the function of a particular imperative macro.

| DTF Assembly Label | Module DSECT Label | Bytes | Bits | Function |
|---|---|---|---|---|
| &Filename | IJICCB | 0-15<br>(0-F) | | Command Control Block (CCB). |
| | IJIMOD | 16 | 0 | 1 = Trailer labels |
| | | (10) | 1 | Used by FREE macro |
| | | | 2 | 1 = COBOL Open/Ignore option |
| | | | 3 | 1 = Track hold option specified |
| | | | 4 | 1 = DTF relocated by OPENR |
| | | | 5 | Not used |
| | | | 6 | 1 = SPNUNB |
| | | | 7 | Used by CNTRL macro |
| | | 17-19<br>(11-13) | | Address of logic module. |
| | | 20<br>(14) | | DTF type for OPEN/CLOSE (X'22' = direct access files). |
| | IJISWI | 21 | 0 | 1 = Output; 0 = Input. |
| | | (15) | 1 | 1 = Verify option specified. |
| | | | 2 | 1 = Search multiple track (SRCHM) specified. |
| | | | 3 | 1 = WRITE AFTER or WRITE RZERO macro used. |
| | | | 4 | 1 = IDLOC specified. |
| | | | 5 | 1 = Undefined; 0 = FIXUNB, VARUNB, or SPNUNB |
| | | | 6 | 1 = RELTYPE = DEC |
| | | | 7 | 1 = End of file. |
| | IJIFNM | 22-28<br>(16-1C) | | Filename (DTF Name). |
| | IJIDVTP | 29<br>(1D) | | Device Type.<br>X'00' = 2311, X'01' = 2314, 2319, X'02' = 2321. |
| | IJIUNT | 30-31<br>(1E-1F) | | Starting logical unit address of the first volume containing the data file. This value is supplied by the OPEN from EXTENT cards (can be initially zero). |
| | IJIULB | 32-35<br>(20-23) | | Address of user's label routine. |
| | IJIUXT | 36-39<br>(24-27) | | Address of user's routine for processing EXTENT information. |
| | IJIRELPT | 40<br>(28) | | Pointer to relative address area:<br>$\frac{\text{\&Filename.P - \&Filename}}{2}$ |
| | IJIERC | 41-43<br>(29-2B) | | Address of a 2-byte field in which IOCS can store the error condition or status codes. |
| | IJITST | 44-45<br>(2C-2D) | | Macro code switch for internal use:<br>X'0000' = READ ID<br>X'0001' = READ KEY<br>X'0002' = WRITE ID<br>X'0003' = WRITE KEY<br>X'0004' = WRITE RZERO<br>X'0005' = WRITE AFTER |
| | IJIBPT | 46-47<br>(2E-2F) | | Pointer to channel program build area (&Filename.B) minus 32. |
| | IJICB2 | 48-63<br>(30-3F) | | Control seek CCB |

Figure 25. DTFDA Table (Part 1 of 6)

| DTF Assembly Label | Module DSECT Label | Bytes | Bits | Function |
|---|---|---|---|---|
| &Filename.Z | IJICCW | 64-71<br>(40-47) | | Control Seek CCW for overlap seek routine. |
| | IJIXMD | 72-75<br>(48-4B) | | Channel program builder instruction:<br>XI 36(2),C'0' |
| | IJIMSZ | 76-77<br>(4C-4D) | | Maximum data length for FIXUNB or UNDEF records; BLKSIZE for VARUNB or SPNUNB records. |
| | IJISPT | 78<br>(4E) | | Pointer to READ ID string (Filename.0); X'00' if no READ ID issued. |
| | | 79<br>(4F) | | Pointer to READ KEY string (Filename.1); X'00' if no READ KEY issued. |
| | | 80<br>(50) | | Pointer to WRITE ID string (Filename.2); X'00' if no WRITE ID issued. |
| | | 81<br>(51) | | Pointer to WRITE KEY string (Filename.3); X'00' if no WRITE KEY issued. |
| | | 82<br>(52) | | Pointer to WRITE RZERO string (Filename.4); X'00' if no WRITE RZERO issued. |
| | | 83<br>(53) | | Pointer to WRITE AFTER string (Filename.5); X'00' if no WRITE AFTER issued. |
| | IJITRK | 84-85<br>(54-55) | | Track constant:<br>2311: X'0000' if key length = 0;<br>X'0020' if key length ≠ 0.<br>2314: X'0000' if key length = 0;<br>X'0045' if key length ≠ 0.<br>2321: X'0000' if key length = 0;<br>X'0016' if key length ≠ 0. |
| | IJIRIC | 86-87<br>(56-57) | | 2311: X'0061'<br>2314: X'0101'<br>2321: X'0084' |
| | IJILAT | 88<br>(58) | 0 | Not used |
| | | | 1 | 1 = Wrong-length record |
| | | | 2 | 1 = Non data transfer error. |
| | | | 3 | Not used. |
| | | | 4 | 1 = No room found |
| | | | 5-6 | Not used |
| | | | 7 | 1 = Record out of extent area. |
| | | 89<br>(59) | 0 | 1 = Data check in count area. |
| | | | 1 | 1 = Track overrun. |
| | | | 2 | 1 = End of cylinder. |
| | | | 3 | 1 = Data check when reading key or data. |
| | | | 4 | 1 = No record found. |
| | | | 5 | 1 = End of file. |
| | | | 6 | 1 = End of volume. |
| | | | 7 | Not used. |
| | IJILBTK | 90-95<br>(5A-5F) | | Label track address, XBCCHH, where X is the volume sequence number of the device on which the label track is located. |

This is the end of the common DTFDA table.

Figure 25. DTFDA Table (Part 2 of 6)

The following section is included if UNDEF, AFTER, or RZERO is specified.

| DTF Assembly Label | Module DSECT Label | Bytes | Bits | Function |
|---|---|---|---|---|
| &Filename.L | IJILST | 96-143<br>(60-8F) | | Basic CCWs to build channel program (see Figure 33) |
| | | 144-183<br>(90-B7) | | Basic CCWs for undefined length or formatting macros (see Figure 33). |
| | IJIVIT | 184-185<br>(B8-B9) | | Instruction to give record length to user if record length is undefined. (NOPR 0 if no RECSIZE specified.) |
| | IJIFRU | 186-187<br>(BA-BB) | | Instruction to get record length from user if record length is undefined. (NOPR 0 if no RECSIZE specified.) |
| &Filename.F | IJIFLD | 188-192<br>(BC-C0) | | Work area (used for R0 address - CCHH0). |
| &Filename.K | IJICNT | 193-200<br>(C1-C8) | | Work area (used for R0 data field). |
| &Filename.C | IJICTS | 201-208<br>(C9-D0) | | Work area (included only for spanned or variable records for record count field). |

The channel program builder strings are generated following the DTFDA table, and preceding the channel program building area. (See Figure 32 for the channel program builder string to be used for each macro.)

| | | | | |
|---|---|---|---|---|
| &Filename.0 | | Variable | | Channel program builder string for READ ID macro. If READ ID is not specified, the string is not generated. |
| &Filename.1 | | Variable | | Channel program builder string for READ KEY macro. If READ KEY is not specified, the string is not generated. |
| &Filename.2 | | Variable | | Channel program builder string for WRITE ID macro. If WRITE ID is not specified, the string is not generated. |
| &Filename.3 | | Variable | | Channel program builder string for WRITE KEY macro. If WRITE KEY is not specified, the string is not generated. |
| &Filename.4 | | Variable | | Channel program builder string for WRITE RZERO macro. If WRITE RZERO or WRITE AFTER is not specified, the string is not generated. |
| &Filename.5 | | Variable | | Channel program builder strings for WRITE AFTER macro. If WRITE RZERO or WRITE AFTER is not specified, the string is not generated. |

Figure 25. DTFDA Table (Part 3 of 6)

The following section contains the channel program build areas and varies in size.

| DTF Assembly Label | Module DSECT Label | Bytes | Bits | Function |
|---|---|---|---|---|
| &Filename.B | | 0-7 | | Seek CCW that is generated at program assembly time and used by all channel programs. |
| | | Variable | | Area to build seven CCWs; or a maximum of nine CCWs if spanned or variable records and AFTER=YES is specified. |
| | | Variable | | Area to build verify CCWs; five CCWs for fixed and undefined records, eight CCWs if spanned or variable records and AFTER=YES is specified. (This area is generated only if VERIFY=YES.) |

The following section is added for spanned records only.

| DTF Assembly Label | Module DSECT Label | Bytes | Bits | Function |
|---|---|---|---|---|
| | | 8 bytes | | Count save area. |
| | | 8 bytes | | SEEKADR save area. |
| | | 1 byte | 0 | 1 = Relative addressing. |
| | | | 1 | 1 = IJIGET switch on. |
| | | | 2 | 1 = Ignore hold switch on. |
| | | | 3 | Reserved for use by DAMODV. |
| | | | 4 | 1 = New volume SEEKADR. |
| | | | 5-7 | Not used. |
| | | 1 byte | | Reserved. |
| | | 2 bytes | | Record size. |
| | | 12 bytes | | Work area. |
| | | 8 bytes | | Control word save area. |

Figure 25. DTFDA Table (Part 4 of 6)

The following section is added to the DTFDA table if DSKXTNT (relative addressing) is specified.

| DTF Assembly Label | Module DSECT Label | Bytes | Bits | Function |
|---|---|---|---|---|
| &Filename.P | | 3 bytes | | 3X'00' for padding. |
| &Filename.I | | 5 bytes | | IDLOC record area (bucket used by module). |
| &Filename.S | | 8 bytes | | SEEKADR in form:<br>M,B1,B2,C1,C2,H1,H2,R |
| | | 4 bytes | | DC A(&SEEKADR) |
| | | 4 bytes | | DC A(&IDLOC) |
| | | 8 bytes | | Work area for RELTYPE=DEC. |
| &Filename.X | | 4 bytes | | Save area for CCHH portion of actual DASD address. |
| | | 4 bytes | | Alteration factor for C1 in SEEKADR (see bytes 112-119):<br>2311: X'00000001'<br>2314, 2319: X'00000001'<br>2321: X'000003E8' |
| | | 4 bytes | | Alteration factor for C2 in SEEKADR (see bytes 112-119):<br>2311: X'0000000A'<br>2314, 2319: X'00000014'<br>2321: X'00000064' |

Figure 25. DTFDA Table (Part 5 of 6)

| DTF Assembly Label | Module DSECT Label | Bytes | Bits | Function |
|---|---|---|---|---|
| | | 4 bytes | | Alteration factor for H1 in SEEKADR (see bytes 112-119):<br>2311: X'00000001'<br>2314, 2319: X'00000001'<br>2321: X'00000014' |
| | | Variable to end of DTF table | | DSKXTNT table composed of a variable number of 8-byte entries containing extent information in the following format:<br>Bytes 0-2 TTT2 -cumulative number of tracks in the DSKXTNT table entries up to and including the current entry.<br>3 M -volume sequence number.<br>4 B -bin number (0 for 2311, 2314, or 2319)<br>Bytes 5-7 TTT1 -relative track number of lower limit of this entry.<br>A 2-byte end-of-table indicator containing X'FFFF' follows the last entry in the DSKXTNT table. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 25. DTFDA Table (Part 6 of 6)

| Bytes | Bits | Function |
|---|---|---|
| 0-15<br>(0-F) | | CCB. |
| 16<br>(10) | | X'08' indicates DTF relocated by OPENR. |
| 17-19<br>(11-13) | | 3X'00' |
| 20<br>(14) | | DTF type (X'23'). |
| 21<br>(15) | | Option codes. |
| | 0 | 1 = Output, 0 = Input. |
| | 1 | Not used. |
| | 2 | Not used. |
| | 3 | Not used. |
| | 4 | Not used. |
| | 5 | Not used. |
| | 6 | 1 = 2321, 0 = 2311, 2314, or 2319. |
| | 7 | Not used. |
| 22-28<br>(16-1C) | | Filename. |
| 29<br>(1D) | | Version 3 device type code: |
| | | X'00' = 2311<br>X'01' = 2314, 2319<br>X'02' = 2321<br>Last byte of filename in previous versions. |
| 30-31<br>(1E-1F) | | Logical unit address of first volume containing the file. |
| 32-35<br>(20-23) | | Address of user label routine. |
| 36-39<br>(24-27) | | Address of user routine to process EXTENT information. |

Numbers in parentheses are displacements in hexadecimal notation.

Figure 26. DTFPH Table For Direct Access

## DTFPH MACRO

Figure 26 illustrates the DTF table generated by the DTFPH macro when the parameters DEVICE=2311/2314/2321 and MOUNTED=ALL are specified in the macro operand. The table contains the information to define a DASD file for processing by physical IOCS, in a manner similar to the Direct Access Method.

## REFERENCE METHODS AND ADDRESSING SYSTEMS

Each record read or written must be identified by providing the logical IOCS routines of the Direct Access Method with two references:

1. Track reference - location of the track within the pack or cell.
2. Record number (ID), or Record Key (control information) - position of the record on the track.

The user can specify the track reference or record ID as either an actual physical DASD address or as an address relative to the start of the file. If relative addressing is used, the address provided by the user has been converted to either a 4-byte hexadecimal or a 10-byte decimal address. Actual physical addresses are supplied as 8-byte DASD addresses. Further details of the addressing systems are presented in the following discussion of reference methods.

## TRACK REFERENCE

Before issuing a read or write instruction, the user must supply the proper track identification in the track reference field in main storage. (This field is identified by the SEEKADR= parameter specified in the DTFDA macro.) The track identification can be expressed in one of three formats depending on the addressing system used.

1. <u>Actual physical addressing</u> - the track identification is contained in the first seven bytes of the 8-byte track reference field (MBBCCHHR).

2. <u>Relative addressing (RELTYPE=HEX)</u> - the track identification is contained in the first three bytes of the 4-byte track reference field (TTTR).

3. <u>Relative addressing (RELTYPE=DEC)</u> - the track identification is contained in the first eight zoned decimal bytes of the 10-byte track reference field (TTTTTTTTRR).

The track reference selects the channel and unit on which the referenced track is found.

## RECORD ID

Reference to a particular record can be made by supplying a specific number in the track reference field. This number (ID) refers to the consecutive position of the record on the given track; that is, the first data record on a track is number 1, the second is number 2, etc.

The form in which the record ID is supplied in the track reference field also depends on the addressing system used.

1. <u>Actual physical addressing</u> - the record ID is the last byte (R-byte) in the 8-byte track reference field (MBBCCHHR).

2. <u>Relative addressing (RELTYPE=HEX)</u> - the record ID is the last byte (R-byte) in the 4-byte track reference field (TTTR).

3. <u>Relative addressing (RELTYPE=DEC)</u> - the record ID is the last two zoned decimal bytes (RR) in the 10-byte track reference field (TTTTTTTTRR).

When a READ or WRITE macro that searches for record ID is executed, logical IOCS refers to the track reference field to determine which record is requested by the program. The number in this field is compared with the corresponding field in the count areas of the DASD records.

When a READ ID macro is executed, IOCS searches the specified track for the particular record. If the record is found, the key area (if present and defined by the KEYLEN= parameter in the DTFDA macro) and the data area of the record are transferred into the main storage I/O area. If the corresponding record ID (R portion of the count area on the track) is not found, a no record found indicator is placed in the user's error status indicator. The WRITE ID operation is the same as the READ ID except a record is written instead of read.

## RECORD KEY

If the DASD records include <u>key</u> areas, the records can be identified by the control information contained in the key. Whenever this method of referencing is used, the problem program must supply the key of the desired record to logical IOCS before a READ or WRITE macro is issued. When a READ or WRITE macro is executed, IOCS searches the track identified by the track reference field for the desired key. The search is confined to one track unless multiple track search is specified by the user. (See <u>Multiple Track Search</u>.)

If the desired key is not found on the track, IOCS posts a no record found indication in the user's error status indicator. When the desired key is found, IOCS reads the data area of the DASD record into main storage if a READ KEY macro was issued.

When a WRITE KEY macro is executed and the desired key is found, IOCS transfers the data in main storage to the data area of the DASD record. This replaces the information previously recorded in the data area.

## CONVERSION OF RELATIVE ADDRESSES

When the record address supplied by the user in the track reference field (SEEKADR) is in relative address form, it must be converted to an actual DASD address (CCHHR) before it can be handled by the routines of the DA logic modules. The Seek Overlap subroutine in the logic module performs the conversion.

If the user wants to express the relative address as a 10-byte zoned decimal number (RELTYPE=DEC), the address is packed and converted to binary so that it takes the hexadecimal TTTR form before conversion to an actual address.

Conversion to an actual DASD address starts by comparing the TTT value given in the user-supplied relative address with the TTT2 value of each entry in the DSKXTNT table. (Refer to Figure 35 and to Relative Addressing under Initialization and Termination in this section of the manual.) The proper DSKXTNT entry is reached when the TTT2 value of the entry exceeds the TTT value in the address. The M and B2 values from the table entry are inserted into the seek address, MBBCCHHR (B1 is always 0). The reconversion factor is calculated by subtracting the TTT1 value of the current extent entry from the TTT2 value of the previous entry. The reconversion factor is saved for reconversion of an actual address to a relative address if IDLOC is specified.

The user's TTT value is then divided, in turn, by the three device-dependent alteration factors; C1, C2, and H1 (refer to Figure 38). The quotient after each divide operation is placed in the respective position in the seek address. For example; the quotient (after the TTT value is divided by the C1 alteration factor), is inserted in the first C-byte of the seek address, MBBCCHHR. The remainder after each divide operation becomes the dividend for the next divide operation. The remainder after the final divide operation is the H2 value in the seek address, MBBCCHHR. The R-byte of the actual seek address is identical to the R-byte (or equivalent to the RR bytes if decimal relative addressing is used) in the TTTR relative address.

If a record ID is returned to the user in relative address form after a READ or WRITE macro instruction is executed (IDLOC specified), reconversion is accomplished by reversing the conversion process. Thus, the corresponding CCHH portions of the actual address are multiplied by the respective alteration factors and the reconversion factor is added to the result. Again, the R-byte remains unmodified throughout the reconversion process. If the decimal form of relative addressing is specified, the TTTR hexadecimal form is further converted to the 10-byte zoned decimal form TTTTTTTTRR.

## MULTIPLE TRACK SEARCH

The Direct Access READ KEY and WRITE KEY macro routines for processing DASD files normally search one track for the desired logical record. The user can specify a search of multiple tracks by including the DTFDA entry SRCHM (SeaRCH Multiple tracks) in the DTF. When SRCHM is specified, IOCS begins the search for a specified record key on the track specified in the track reference field. The search continues until one of two conditions occur:

1. An equal compare occurs between the key argument (record key) in main storage and the key of the required record.

2. The end of the specified cylinder is reached.

The search for multiple tracks continues through the cylinder, even though part of the cylinder may be assigned to a different logical file. This occurs with or without relative addressing. IOCS provides the user with an end of cylinder indicator when the search reaches the end of a cylinder. This indicator is placed into the error/status byte by IOCS.

## IDLOC

The parameter IDLOC= is provided (in both the DTFDA and DAMOD or DAMODV macros) if the user wants to identify records after each READ or WRITE operation is complete. If specified, IDLOC identifies a main storage location where IOCS supplies the address (either actual or relative) of a DASD record. If spanned records are being processed, the ID returned will be that of the first segment of the record. The address returned in location IDLOC after a particular macro depends on a variety of conditions. These conditions and the addresses returned are summarized in Figure 27.

When the problem program references a record by ID or KEY and does not specify the search multiple tracks (SRCHM) option, IOCS returns the ID of the next record under normal conditions. If the user is

| Read/Write Function | SRCHM = YES | | | SRCHM ≠ YES | | | EOF record read | Seek address not in extent area |
|---|---|---|---|---|---|---|---|---|
| | Normal I/O complete | No record found | *End of cylinder | Normal I/O complete | No record found | *End of cylinder | | |
| READ Filename,KEY | Same record | Blank | Next record | Next record | Dummy record | Next record | Dummy record | Dummy record |
| WRITE Filename,KEY | Same record | Blank | Next record | Next record | Dummy record | Next record | Dummy record | Dummy record |
| READ Filename,ID | Next record | Dummy record | Next record | Next record | Dummy record | Next record | Dummy record | Dummy record |
| WRITE Filename,ID | Next record | Dummy record | Next record | Next record | Dummy record | Next record | Dummy record | Dummy record |
| WRITE Filename,AFTER | None | Dummy record | Dummy record | None | Dummy record | Dummy record | Dummy record | Dummy record |
| WRITE Filename,RZERO | None | Dummy record | Dummy record | None | Dummy record | Dummy record | Dummy record | Dummy record |

*If an end-of-cylinder condition coincides with either a physical or a logical end of volume, the ID supplied is that of the first record on the next volume. If this condition occurs on the last volume, the ID supplied in IDLOC is equal to the maximum number of tracks for the file. A dummy record is supplied when a physical end of volume is reached if actual DASD addressing is used.

Dummy record:
Actual addressing ---------- 5 bytes (CCHHR) containing X'FFs
Relative addressing (HEX) -- 4 bytes (TTTR) containing X'FFs
Relative addressing (DEC) -- 10 bytes containing decimal 9s

Figure 27. Record ID Returned to IDLOC

processing records sequentially on the basis of the next ID, he can check the ID supplied by IOCS against his file limits to determine when he has reached the end of his logical file.

If the next record ID is returned to IDLOC, LIOCS searches for the ID of the next record on the specified cylinder. If an end of cylinder occurs befcre the next record is found, logical IOCS:

1. Posts the end-of-cylinder bit in the error/status indicator, and

2. Updates the address to head 0, record 1 of the next cylinder, and posts this updated address in IDLOC.

It is possible that there will be no record at this new address. In this case, logical IOCS posts a no-record-found in the error/status indicator. Two ways to avoid this possibility:

1. Initialize the volume by writing a dummy record at the beginning of each cylinder.

2. Add 1 to the record address and read or write again, and continue this process until logical IOCS finds the desired record.

## CONTROL FIELD - SPANNED RECORDS

Figure 28 illustrates the format of the 8-byte control field associated with each spanned record. The first four bytes are called the block descriptor word and contain information supplied by LIOCS wher the record is written. The second four bytes are called the segment descriptor word and contain segment type information, the user supplied record length, and the segment control flag.

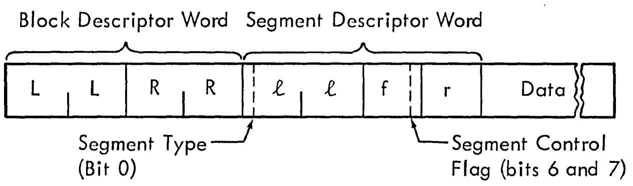
Normal Segment: The term normal segment refers to any segment of the kind described by the segment control flag.

Null Segment: The term null segment refers to a special 8-byte segment (control field only) that may be written by a WRITE AFTER macro when the file is being created. A null segment is written as the last record on a volume and indicates that the next logical record is written on a new volume. Spanned records do not span volumes; that is, the first portion of a logical record cannot exist on one volume and the remainder on another.

## ERROR/STATUS INDICATOR

When processing records in a DASD environment, certain exceptional conditions must be handled within the program. Because the method used for handling these exceptional conditions depends on the application and operating environment, the logical IOCS routines of the Direct Access Method provide the user with exception indicators.

The user must specify a symbolic name for the address of a 2-byte field where IOCS places the exceptional condition codes. The symbolic name is written by the user in the DTFDA entry ERRBYTE. When needed, IOCS sets one or more of the bits in this error/status indicator for the conditions illustrated in Figure 29.



Block Descriptor Word

LL = Record length including the 8-byte control field (ℓℓ+4).

RR = Used by the system.

Segment Descriptor Word

ℓℓ = Record length including the 4-byte segment descriptor (data length+4).

Segment Type:

0 = Normal segment
1 = Null segment

f = Contains binary zeros except bits 6 and 7.

Segment Control Flag:

00 = This segment is not followed or preceded by another segment; that is, a single contiguous segment contains the entire logical record.

01 = This segment is the first segment of a multi-segment logical record.

10 = This segment is the last segment of a multi-segment logical record.

11 = This segment is neither the first nor the last segment of a multisegment logical record.

r = Contains binary zeros.

Figure 28. Spanned Record Control Field

| Byte | Bit | Error/Status Indicator | Explanation |
|---|---|---|---|
| 0 | 0 | ----- | Not used. |
| 0 | 1 | Wrong-length record | FIXUNB records: This bit is set on whenever the data length or key length of a record differs from the original record. If an updated record is shorter than the original record, the updated record is padded with binary zeros to the length of the original record. If the updated record is longer than the original record, the original record positions are filled and the rest of the updated record is truncated and lost.<br>UNDEF records: This bit is set on under the following conditions:<br>• When a READ is issued and the record is greater than the maximum data size (BLKSIZE minus KEYLEN; or BLKSIZE minus the value of KEYLEN plus eight, if AFTER is used), a wrong-length error condition is given and the value returned in the RECSIZE register is that of the actual record length.<br>• When a WRITE ID or KEY is issued and the record to be written is greater than the maximum data size, a wrong-length error condition is given and the record written is equal to that of the maximum data length. If the DASD record is larger than the maximum data size, the remainder of the record is padded with binary zeros. The value in the RECSIZE register is set equal to that of the maximum data length.<br>• When a WRITE AFTER is issued and the record to be written is greater than the maximum data size, a wrong-length error condition is given and the record written is truncated to the maximum data length. The value in the RECSIZE register is set equal to that of the maximum data length.<br>VARUNB records: This bit is set on under the following conditions:<br>• When a READ is issued and the LL (data length + 8) count of the record read is greater than the maximum value specified by the BLKSIZE= parameter in the DTFDA macro.<br>• When a WRITE ID or KEY macro is issued and the LL count is greater than the value specified by the BLKSIZE parameter in the DTFDA macro. The record is written with the low-order bytes truncated. |

Figure 29. Error/Status Indicator (Part 1 of 4)

| Byte | Bit | Error/Status Indicator | Explanation |
|---|---|---|---|
| | | | • When a WRITE AFTER macro is issued and the LL count is greater than the value specified by the BLKSIZE parameter in the DTFDA macro. The record is written with the low-order bytes truncated.<br>SPNUNB records: This bit is set on under the following conditions:<br>• When a READ macro is issued, the wrong-length record error indicator is set if the LL (data length + 8) count is larger than the value specified by the BLKSIZE parameter in the DTFDA macro. The number of data bytes read into the I/O area is equal to the value of BLKSIZE minus 8 bytes for the control words.<br>• When a WRITE ID or KEY macro is issued, the wrong-length record indicator is set if:<br>1. The LL count for the record to be written exceeds the value specified by the BLKSIZE parameter in the DTFDA macro.<br>2. The data length of the record to be written exceeds the data length of the original record.<br>In either of the above cases the record is written with the low-order bytes truncated.<br>The wrong-length record indicator is also set when the first segment encountered for the original record is not type 00 or 01. In this case the no-record-found (bit 4 in byte 1) indicator is also set on and no portion of the new record is written.<br>The wrong-length record indicator is set for multisegment records if any segment of the original record encountered after the first segment is not type 11 or 10. In this case the remainder of the new record is not written.<br>• When a formatting WRITE AFTER macro is issued and the LL count for the record being written exceeds the value specified by the BLKSIZE parameter in the DTFDA macro. The record is written with the low-order bytes truncated. |
| 0 | 2 | Nondata Transfer Error | The record in error was neither read nor written. If ERREXT is specified and this bit is off (0), transfer took place and the problem programmer should check for other errors in the ERRBYTE field. |

Figure 29. Error/Status Indicator (Part 2 of 4)

| Byte | Bit | Error/Status Indicator | Explanation |
|---|---|---|---|
| 0 | 3 | ----- | Not used. |
| 0 | 4 | No-room-found | The no-room-found indication is applicable only when the formatting WRITE AFTER macro is used for a file. If the bit is set on, IOCS has determined that there is not enough room left to write the record; consequently, the record is not written. |
| 0 | 5 | ----- | Not used. |
| 0 | 6 | ----- | Not used. |
| 0 | 7 | Record out of extent area | The relative address given is outside the extent area of the file. No I/O activity has been started and no other error indicators are set. |
| 1 | 0 | Data check in count area | This is an unrecoverable error. |
| 1 | 1 | Track overrun | The number of bytes on the track exceeds the theoretical capacity. (Will not occur when DOS/360 macro instructions are used.) |
| 1 | 2 | End-of-cylinder | The end-of-cylinder indicator bit is set on when SRCHM is specified for a READ or WRITE KEY and the end-of-cylinder is reached before the record is found (bit 4 of byte 1 is also turned on). If IDLOC is also specified, certain conditions also turn this bit on, possibly in conjunction with the no-record-found indicator (bit 4 in byte 1) - for further information see IDLOC. |
| 1 | 3 | Data check when reading key or data | This is an unrecoverable error. |
| 1 | 4 | No-record-found | The no-record-found indication is given when a SEARCH ID or KEY is issued and the record is not found.<br>If SRCHM=YES is specified, the end-of-cylinder indicator (bit 2 in byte 1) is also set on.<br>For SPNUNB records: the no-record-found indicator is also set on if, when the identified record is found, the control flag in the first segment encountered is <u>not</u> 00 or 01. In this case, the no-record-found indicator is set on in conjunction with the wrong-length-record indicator (bit 1 of byte 0). |

Figure 29. Error/Status Indicator (Part 3 of 4)

| Byte | Bit | Error/Status Indicator | Explanation |
|---|---|---|---|
| 1 | 5 | End-of-file | The end-of-file indication is applicable only when the record to be read has a data length of zero. The ID returned in IDLOC, if specified, is hexadecimal FFFFF. The bit is set only after all the data records have been processed. For example, in a file having n data records (record n+1 is the end-of-file record), the end-of-file indicator is set on when the user reads the n+1 record. This bit is also posted when an end of volume marker is detected. It is the user's responsibility to determine if this bit means true EOF or end of volume on a SAM file. |
| 1 | 6 | End-of-volume | The end-of-volume indication is given in conjunction with the end-of-cylinder indicator (bit 2 of byte 1). This bit is set on if the next record ID (CCHHR where CC = n+1, HH = 0, and R = 1) that is returned on an end-of-cylinder is higher than the volume address limit. The volume address limit is: cylinder 199, head 9, for a 2311; cylinder 199, head 19, for a 2314, or 2319; and subcell 19, strip 5, cylinder 4, head 19, for a 2321. These limits allow for the reserved alternate track area.<br>If both end-of-cylinder and end-of-volume indicators are set on, the ID returned in IDLOC is FFFFF. |
| 1 | 7 | ----- | Not used. |

Figure 29. Error/Status Indicator (Part 4 of 4)

## CAPACITY RECORD (RZERO OR R0)

The Direct Access Method utilizes the first record on each track, R0, to monitor the amount of available space on the track. This record is unique in that it does not contain a key area even though keys may be specified for the data records of the file.

The Direct Access Method reads the data portion of the R0 record into the Filename.K location in the DTF table. The data portion has the following format:

- 5-bytes - The identifier (CCHHR) of the last record written on the track.
- 2-bytes - The number of unused bytes remaining on the track.
- 1-byte - Flag for the Direct Access Method for Operating System/360.

## WRITE RZERO MACRO

The WRITE Filename,RZERO macro is used to erase a specified track. To do this, the programmer must supply the track address in the track-reference field identified by the SEEKADR= parameter of the DTFDA macro. The system locates the track, restores the number-of-bytes-remaining information in the data field of the R0 record to the maximum capacity of the track, and erases the remainder of the track after the R0 record.

## FORMATTING MACRO

The formatting WRITE Filename,AFTER macro is used to write a record after the last current record on a specified track. To perform this function, the problem programmer must supply, in the location specified by the SEEKADR= parameter of the DTFDA macro, the address of the track on which the new record is to be written. This form of the WRITE macro cannot return the ID of the new record in the IDLOC field.

When the formatting WRITE AFTER macro is used to write FIXUNB or UNDEF records on a file, the first eight bytes of the user's I/O area must be reserved for LIOCS. Therefore, the blocksize (BLKSIZE) must be equal to:

8 + (KEYLEN, if specified) + $D_L$

The ID of the new record can be found in the first five bytes of the I/O area after the write operation is complete because LIOCS uses the eight bytes that are reserved for the record count field with the following format:

- 5-byte track ID (CCHHR)
- 1-byte key length ($K_L$ )
- 2-byte data length ($D_L$ )

When the formatting WRITE AFTER macro is used to write VARUNB or SPNUNB records on a file, the first eight bytes of the user's I/O area contain the record control information. (Refer to Figure 28 for the format of the 8-byte control field.) Therefore, the blocksize (BLKSIZE) must be equal to:

Maximum $D_L$ + 8

The ID of the new record can be found in the DTF table at location Filename.C after the write operation is complete. This area of the DTF table is generated specifically for VARUNB and SPNUNB records and is used for the count field of the new record. It has the following format:

- 5-byte track ID (CCHHR)
- 1-byte key length ($K_L$ )
- 2-byte data length ($D_L$ ) *

* For VARUNB and SPNUNB records, $D_L$ includes the 8-byte control field.

## DA LOGIC MODULE MACROS

IBM supplies two macros to generate independent logic modules needed to process records under the Direct Access Method. These macros are:

- DAMOD - for fixed-length unblocked and undefined records.
- DAMODV - for variable-length unblocked and spanned unblocked records.

The modules generated by these macros include routines for the basic imperative macros READ and WRITE, which allow the user to read, write, update, add, or replace records in the file.

## DIRECT ACCESS MODULES

Under the Direct Access Method an individual module, either DAMCD or DAMODV (depending on the record format specified), provides the logic to support all the imperative macros used to process the file. In each case, the CNTRL, FREE, and WAITF macros have individual entries into the required module; that is, the logic for executing these macro functions is tailored to the specific macro. On the other hand, the input/output macros (READ and WRITE, with their variations) have a common entry to the respective module and a common logic in the module for performing their functions.

When the user issues a READ or WRITE macro instruction for a file, program control transfers to a logical IOCS routine, which builds the proper channel program to accomplish the command. The IOCS routine issues an execute channel program that causes the I/O request to start. IOCS then returns control to the problem program. A WAITF macro instruction must be issued by the user before the next READ or WRITE for the file. The WAITF macro routines test the status of the channel to ensure that the operation is complete. If the channel is busy, the routine waits for I/O completion. The WAITF macro routines supply indications of exceptional conditions to the problem program in the error/status indicator. At the completion of the I/O operation, control returns to the problem program.

### DAMOD: Input/Output Macros Charts NA-NC

Objective: To read or write a fixed-length unblocked or undefined record on a direct access file.

Entry: From any Input/Output macro used with the Direct Access Method.

Exit: To the problem program via linkage register 14.

Method: Each of the six Input/Output macros has a unique expansion that results in a branch to a different entry point in the module. The entry point is at one of a series of exclusive OR instructions. The exclusive OR instructions cause a unique bit structure to be set up in a one-byte macro switch in the DTFDA table. From this macro switch, the module determines which macro has been issued.

After the macro switch has been set, a test is made for undefined records or an end-of-file condition. If neither, the data length is set to the maximum length. If end of file, the data length is set to zero. If undefined and a read operation, the data length is set to the maximum. For a WRITE AFTER, WRITE KEY, or WRITE ID instruction, this routine gets the data length from the user, and determines whether it is greater than the maximum length. If so, it is set to maximum and the wrong-length record bit is set on in the DTF table. The CCW data areas are then updated, and a branch and link is made to the seek overlap routine to position the device for subsequent processing. If track hold has been specified, and entry to the seek overlap routine was not from the CNTRL or FREE macros, an SVC 35 is performed to hold the track.

Next, this routine branches to the channel program builder to build the CCW chain for the macro that is being processed (refer to Figure 36). A test is then made for a WRITE AFTER or WRITE RZERO macro being processed. If neither of these, this routine issues the SVC 0 to perform a read or write operation. Control then returns to the problem program.

If the macro is a formatting macro (WRITE AFTER or WRITE RZERO), additional processing is necessary. If the macro is WRITE AFTER, R0 is read and the capacity of the track is checked. If the space remaining on the track is not large enough for the record, the no-room-found bit is set on in the DTF and control returns to the problem program.

If the track capacity is large enough, the routine calculates the space remaining on the track after the record is written and stores it in the R0 write area. The channel program builder then builds a CCW chain to WRITE AFTER, updates the previous record ID by 1 in the R0 write area, and tests for end of file. If end of file, the key and data length fields in the count field are set to zero. If not end of file, the key and data lengths of the record are inserted in the count field. An SVC 0 is then issued to write out the record. If track hold has been specified, an SVC 36 is issued to free the held track, and control returns to the problem program.

If the macro issued is a WRITE RZERO, CCWs are modified, and a new R0 record is written. If track hold has been specified, the module issues an SVC 36 to free the track. Control then returns to the problem program.

DAMOD: WAITF Macro Charts ND-NG

Objective: To ensure that the transfer of a record has been completed, to supply the ID of a record to the user, if IDLOC is specified, and to post error conditions in the error/status indicator, if necessary.

Entry: From the WAITF macro.

Exit: To the problem program.

Method: After saving the user's registers, this routine first issues an SVC 7 WAIT macro to ensure that the previous I/O operation is complete. The second error byte from the CCB is placed in the error/status indicator in the DTF table.

If IDLOC is specified, IOCS supplies the user with the ID of a record after each READ or WRITE is completed (see Figure 25). If IDLOC is specified, a test is made for the type of macro issued. If a READ KEY or WRITE KEY macro, the routine determines if the search multiple track option (SRCHM) has been specified. If so, the ID returned to the user is the ID of current record transferred.

If a READ or WRITE KEY macro has been issued without a search multiple track option, or a READ or WRITE ID macro has been issued, the ID returned to the user is the ID of the next record location, unless an end-of-cylinder condition is encountered. In this case, the ID returned is that of the first record of the next cylinder. If an end-of-volume condition is detected while updating the cylinder address, the end-of-volume bit is set in the error status indicator in the DTF table, and a dummy record is returned in IDLOC.

After the module determines the contents of IDLOC, the error/status bytes are set in accordance with the conditions posted to the CCB by physical IOCS, and returned to the user. Then, if record length is undefined and a READ macro has been issued, the record length is calculated and returned to the user. This routine then restores the user's registers, resets the macro switch in the DTF table, and returns control to the problem program via linkage register 14.

DAMOD: CNTRL Macro Chart NH

Objective: To perform nondata operations on a file. For a 2311, 2314, or 2319, a seek operation is executed. For a 2321, either a seek operation or a restore operation is executed.

Entry: From the CNTRL macro.

Exit: To the problem program.

Method: This routine saves the user's registers, and then branches and links to the seek-overlap routine, which performs the nondata operation (seek for a 2311, 2314, or 2319; seek or restore for a 2321). When the operation has been completed, the user's registers are restored, and control returns to the problem program via linkage register 14.

DAMOD: FREE Macro Chart NH

Objective: To release a protected (held) track on a direct access storage device.

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

Exit: To the problem program.

Method: After storing the user's registers, the FREE routine branches to the seek-overlap subroutine. The subroutine determines the seek address of the held track from the seek CCW in the channel program build area. The module (M) number from the seek address calculates the symbolic unit address which is then inserted into the control-seek CCB. An SVC 36 is issued to free the held track. After completing the subroutine, the FREE routine restores the user's registers and returns control to the problem program.

DAMODV: Input/Output Macros Charts NM-NX

Objective: To read or write a variable-length unblocked or a spanned unblocked record on a Direct Access file.

Entry: From any Input/Output macro used with the Direct Access Method.

Exit: To the problem program via linkage register 14.

Method: Each of the six Input/Output macros has a unique expansion that results in a branch to a different entry point in the module. The entry point is to one of a series of exclusive OR instructions, which cause a unique bit pattern to be set up in a 1-byte macro switch in the DTFDA table. The module determines which macro has been issued by testing this switch.

READ Macro - VARUNB Records: The procedure followed for both the READ ID and the READ KEY macros is exactly the same. The only difference between the two macros is in the CCW chain built by the channel program builder subroutine, IJISBLD. Refer to Figure 36, Chart I for READ ID; Chart J for READ KEY.

The byte count in the basic read data CCW (Figure 33) is set equal to the length specified by the user in the BLKSIZE= parameter for the DTFDA macro. The IJISOVP subroutine is then entered to execute a controlled seek to the proper track. Next, the channel program builder subroutine is used to build the required channel program. The channel program is executed to read the record into the I/O area and control is returned to the problem program.

READ Macro - SPNUNB Records: The procedure followed for both the READ ID and the READ KEY macros is exactly the same. The only difference between the two macros is in the CCW chain built by the channel program builder subroutine, IJISBLD. Refer to Figure 36, Chart I for READ ID; Chart J for READ KEY.

The byte count in the basic read data CCW (Figure 33) is set equal to the length specified by the user in the BLKSIZE= parameter for the DTFDA macro. The IJISOVP is then entered to execute a controlled seek to the proper track. Next, the channel program builder subroutine is used to build the required channel program, which is then executed to start the read operation. If the segment descriptor for the record read indicates that it is a null segment or that the segment contains the entire logical record (segment type 00), control is returned to the problem program.

If the record read is segment type 01 (the first segment of a multisegment record - at this point, segment types 10 or 11 would be in error) which indicates that the rest of the logical record continues on another track, the CCW chain is modified and the seek address is updated to the next track. One of the modifications made to the READ ID CCW chain is the substituting of a TIC*+8 CCW for the RDKD CCW when

KEYLEN is specified. This is done because the record key is associated only with the first segment of a multisegment record.

The last eight bytes of the last portion of the record read into the I/O area are temporarily stored in the DTF table to allow the control words (block descriptor and segment descriptor) of the next segment to be read in along with the data (see Figure 30); these bytes are later restored after the control word information for the next segment is processed. The modified channel program is reexecuted to read the next segment into the I/O area and its length is added to the combined length of the previous segments. The combined total length is then compared to the BLKSIZE specified by the user. Should the combined length exceed the BLKSIZE, a wrong-length-record (WLR) indicator is set. If the segment just read is type 11 neither the first nor the last segment of a multisegment record) the procedure described in this paragraph is repeated.

When the last segment (type 10) is read, the combined length of all the record segments is posted to the segment descriptor word in the I/O area and control is returned to the problem program.

WRITE Macro - VARUNB Records: The procedure followed for both the WRITE ID and the WRITE KEY macros is exactly the same. The only difference between the two macros is in the CCW chain built by the channel program builder subroutine, IJISBLD. Refer to Figure 36, Chart K for WRITE ID and VERIFY, Chart L for WRITE ID and No VERIFY; Chart M for WRITE KEY and VERIFY, and Chart N for WRITE KEY and No VERIFY.

The logical record length ( *ℓℓ* ) is obtained from the user's segment descriptor word in the I/O area. The length specified for the record plus four bytes for the block descriptor word is then tested to see if it is greater than the maximum block length specified in the BLKSIZE= parameter of the DTFDA macro. If it is not greater than the BLKSIZE value, the byte count in the basic read data CCW (RDD CCW - Figure 31) is set equal to the specified *ℓℓ* + 4 (that is, LL) value. If, on the other hand, the LL value is greater than the BLKSIZE value, the record capacity

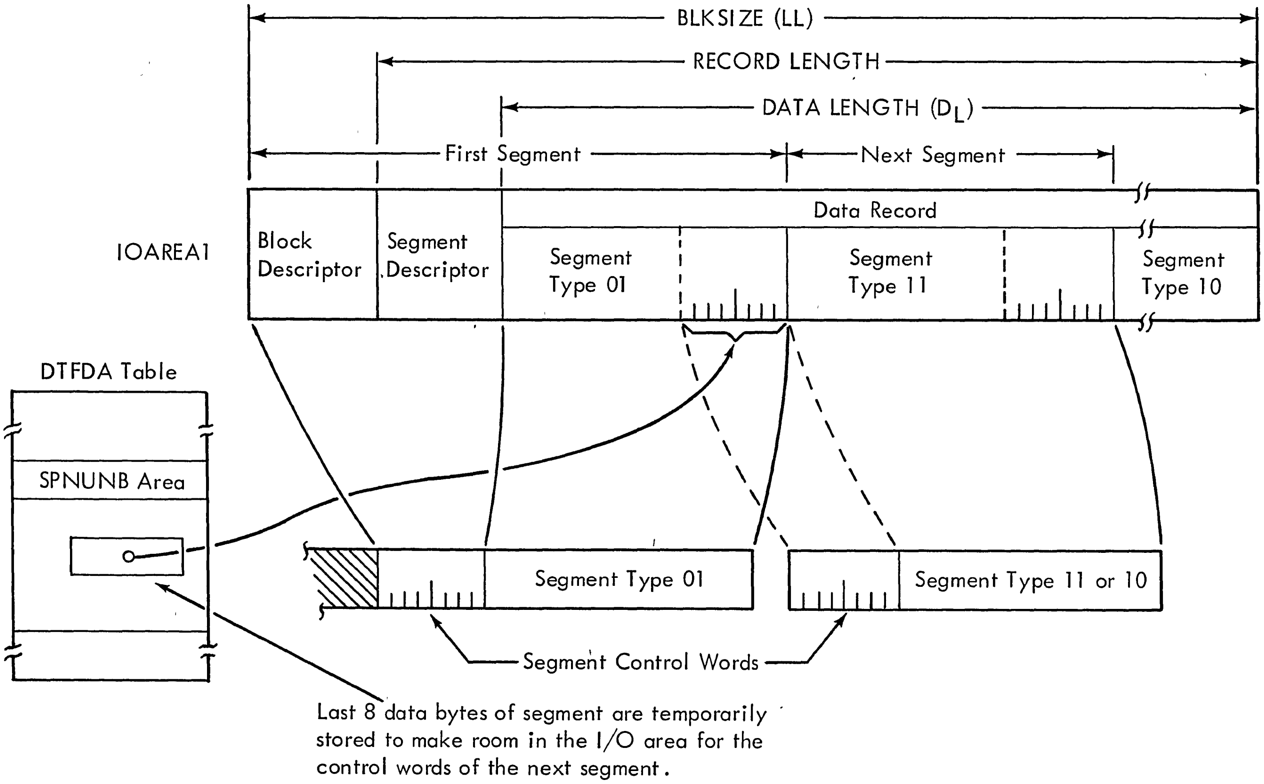


Figure 30. Multisegment Spanned Record

register(IJICPR) and the RDD CCW byte count are set equal to the BLKSIZE value and the wrong-length-record (WLR) indicator is set. This causes truncation of the record when it is written.

The IJISOVP subroutine is entered to execute a controlled seek to the proper track. Next, the channel program builder subroutine is used to build the required channel program, which is then executed to write (and verify, if so specified) the record. Control is then returned to the problem program. If the track hold option has been specified, the track on which the record written resides is freed before control is returned.

WRITE Macro - SPNUNB Records: The procedure followed for both the WRITE ID and the WRITE KEY macros is exactly the same. The only difference between the two macros is in the CCW chain built by the channel program builder subroutine, IJISBLD. Refer to Figure 36, Chart K for WRITE ID and VERIFY, Chart L for WRITE ID and No VERIFY; Chart M for WRITE KEY and VERIFY, and to Chart N for WRITE KEY and No VERIFY.

The IJISOVP subroutine is entered to execute a controlled seek to the proper track. Next, the channel program builder subroutine is used to build the first portion of the WRITE macro channel program. It is at this point that spanned record handling differs markedly from the handling of records of other formats.

The first portion of the WRITE macro channel program (refer to Figure 36, Charts K or L for WRITE ID; Charts M or N for WRITE KEY) is actually a CCW chain to read the eight bytes of control information contained in the existing DASD record. This read operation is necessary because, before a spanned record can be written, the arrangement of the record being replaced must be determined. That is, it must be known if the existing record is contained in a single DASD segment (type 00) or in multiple DASD segments and, if in multiple segments, the lengths of the individual segments. Thus, for each segment of a multisegment spanned record, it is necessary to execute a read and a write operation.

If the segment control flag in the segment descriptor of the existing record is type 00, the record to be written is handled in a manner similar to a normal variable-length record. That is, the channel program builder subroutine is entered to build the write CCW chain, the channel program is executed to write the new record, and control is returned to the problem program.

If the segment control flag in the DASD segment read is type 01 (the first segment of a multisegment record), the channel program builder subroutine is entered to build the write CCW chain. The CCW chain is then modified according to the various options specified for the type of macro being used. Next, the length of the current DASD segment is determined from the control words obtained by the read operation and compared to the user-specified length of the record to be written (LL). If the record length is less than the length of the current segment, the byte count in the write data (WRD) CCW is changed to the length of the record (if VERIFY is specified, the byte count in the verify read data CCW is likewise changed). Otherwise, the CCW byte count remains equal to the length of the segment that can be accommodated on the track; that is, the length of the current segment. The channel program is then executed to write the segment.

After the first segment of the record is written, the seek address is updated to the next track and a similar procedure is followed for the next segment(s) of the record. During the procedure for writing segments after the first segment, the last eight data bytes of the preceding segment in the I/O are temporarily stored in the DTF table to allow the control words of the subsequent segment to be read into the I/O area (see Figure 30). The segment length obtained from the control words is used to set the byte count in the WRD CCW for all type 11 segments. Each time a segment is written, its length is added to the combined lengths of the previously written segments, and the total is subtracted from the user-specified record length. The result of this calculation is the number of bytes in the record that remain to be written. When the last segment (type 10) is written, this remainder is used to determine if the new record is larger or smaller than the original record. If it is larger, a WLR indicator is set and the truncated remainder of the record is written; if smaller, the byte count in the WRD CCW is reduced to the value necessary to write the remainder of the record.

Because each segment of a multisegment spanned record is handled as an individual physical DASD record, if the VERIFY option is specified, each segment is verified after it is written and before the next segment is read. Therefore, if VERIFY is used, three I/O operations are required for each segment: read, write, and read.

WRITE AFTER Macro - VARUNB Records: The byte count of the basic read data CCW (Figure 33) is set equal to the block length (LL) of the record to be written, and the IJISOVP subroutine is entered to execute a controlled seek to the track specified by the user. The channel program builder subroutine, IJISBLD, is then used to build the first portion (read RZERO) of the channel program for the WRITE AFTER macro (refer to Figure 36, Chart O).

Next, the ID (CCHHR) of the R0 record on the specified track is set up in the DTF table, at location Filename.F, and the channel program is executed to read the 8-byte data field of R0 into the DTF table at location Filename.K. The data field of the R0 record contains the following information:

Bytes 0-4: The CCHHR of the last record currently written on the track.

Bytes 5-6: The number of unused bytes currently remaining on the track.

Byte 7: Not used by DOS.

Using the information contained in bytes 5 and 6 of the R0 data field, a test is made to determine if sufficient room exists on the track to write the new record. If enough room is not available, the no-room-found indicator is set in the DTF table and control is returned to the problem program.

If there is enough room on the track for the new record, the DASD space that remains after the new record is written is calculated to update the R0 record. Next, the channel program builder subroutine is used to build the rest of the WRITE AFTER channel program, which includes the CCWs needed to write the updated R0 record and the new record (and verify both, if so specified).

The channel program is then executed and control is returned to the problem program. If the track hold option has been specified, the track is freed before control is returned.

WRITE AFTER Macro - SPNUNB Records: The procedure followed for the WRITE AFTER macro for spanned records is the same as that followed for variable-length records up to the point of testing to determine if there is sufficient room on the specified track for the new record. For spanned records, the test is first made to determine if a minimum length (KEYLEN + 9) segment can be written in the space remaining on the track. If not, the no-room-found indicator is set in the DTF table and control is returned to the problem program. If the minimum length segment can fit, a second test determines if the entire record can be written on the track. If it can, the record is written in the manner described for variable-length records.

If the entire record will not fit in the space remaining on the specified track, the length of the portion that can fit is calculated and subtracted from the user-specified length of the record. The seek address is then updated to the next track.

The R0 record for the next track is read and checked for full availability; that is, if the track is not empty, a no-room-found indicator is set and control is returned to the problem program. The data field of the R0 record is tested to determine if all the remaining bytes of the record (plus eight bytes for control words) can be contained on the new track. If not, the full track capacity is subtracted from the number of record bytes remaining to be written, and the seek address is once again updated. This process is repeated until the point is reached where the entire logical record can be accommodated. If the track hold option has been specified, a hold is placed on all the tracks checked.

The channel program builder subroutine is then used to build the second portion of the WRITE AFTER channel program, and the first segment of the record is written on the specified track. If KEYLEN is specified, the key is written with the first segment. The rest of the record is then written in as many segments as necessary, along with the R0 records for each of the tracks involved. If the track hold option has been specified, the tracks are individually freed after the respective segment is written.

If, during the checking of the series of tracks needed to write the record, the updated seek address indicates a change to a new volume, the R0 records of all the tracks between the user-specified track and the first track on the new volume are rewritten with their respective data fields indicating no space available. Checking is reinitiated on the new volume and, when it is established that sufficient room is available on the new volume, the first segment (and, if specified, the record key) is written on the first track. The rest of the record is written on subsequent tracks in the normal manner.

WRITE RZERO Macro - VARUNB or SPNUNB Records: The IJISOVP and IJISBLD subroutines are entered in sequence to execute a controlled seek to the specified track and build the channel program. The ID for the R0 record (CCHH0) on the specified track is set up in locations

Filename.F and Filename.K in the DTF table. The number of bytes remaining on the track is set equal to the full track capacity and inserted into bytes 5 and 6 of the R0 data field (Filename.K). The channel program is then executed to erase the track and write the updated R0 record, after which control is returned to the problem program.

### DAMODV: CNTRL Macro Chart NY

Objective: To perform nondata operations on a file. For a 2311, 2314, or 2319, a seek operation is executed. For a 2321, either a seek operation or a restore operation is executed.

Entry: From the CNTRL macro.

Exit: To the problem program.

Method: This routine saves the user's registers, and then branches and links to the seek-overlap routine, which performs the nondata operation (seek for a 2311, 2314, or 2319; seek or restore for a 2321). When the operation has been completed, the user's registers are restored, and control returns to the problem program via linkage register 14.

### DAMODV: FREE Macro Chart NY

Objective: To release a protected (held) track on a direct access storage device.

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

Exit: To the problem program.

Method: After storing the user's registers, the FREE routine branches to the seek-overlap subroutine. The subroutine determines the seek address of the held track from the seek CCW in the channel program build area. The module (M) number from the seek address calculates the symbolic unit address which is then inserted into the control-seek CCB. An SVC 36 is issued to free the held track. After completing the subroutine, the FREE routine restores the user's registers and returns control to the problem program.

### DAMODV: WAITF Macro Charts NZ-PB

Objective: To ensure that the transfer of a record has been completed, to supply the ID of a record to the user, if IDLOC is specified, and to post error conditions in the error/status indicator, if necessary.

Entry: From the WAITF macro.

Exit: To the problem program.

Method: After saving the user's registers, this routine first issues an SVC 7 WAIT macro to ensure that the previous I/O operation is complete. The second error byte from the CCB is placed in the error/status indicator in the DTF table.

If IDLOC is specified, IOCS supplies the user with the ID of a record after each READ or WRITE is completed (see Figure 25). If IDLOC is specified, a test is made for the type of macro issued. If a READ KEY or WRITE KEY macro, the routine determines if the search multiple track option (SRCHM) has been specified. If so, the ID returned to the user is the ID of the current record transferred.

If a READ or WRITE KEY macro has been issued without a search multiple track option, or a READ or WRITE ID macro has been issued, the ID returned to the user is the ID of the next record location, unless an end-of-cylinder condition is encountered. In this case, the ID returned is that of the first record of the next cylinder. If an end-of-volume condition is detected while updating the cylinder address, the end-of-volume bit is set in the error status indicator in the DTF table, and a dummy record is returned in IDLOC.

After the module determines the contents of IDLOC, the error/status bytes are set in accordance with the conditions posted to the CCB by physical IOCS, and returned to the user. Then, if record length is undefined and a READ macro has been issued, the record length is calculated and returned to the user. This routine then restores the user's registers, resets the macro switch in the DTF table, and returns control to the problem program via linkage register 14.

### Channel Program Builder Subroutine Chart NL

Objective: To construct a channel program in accordance with the processing macro issued in the problem program.

Note: Figure 36 provides a summary of the channel programs built to process DASD records by the Direct Access Method.

Entry: From a direct access logic module (either DAMOD or DAMODV) via a branch and link instruction.

Exit: To the calling routine.

Method: To perform direct access processing, many different channel programs, varying in length from 5 to 17 CCWs, are needed in DOS (refer to Figure 34). The many CCWs required can be built from 11 basic CCWs by modifying command codes and/or flag bytes. Of these 11 CCWs, 5 are required for initial file loading. The other 6 are needed for normal file maintenance processing. TIC CCWs are built directly from storage addresses.

For each channel program that is built, a string of descriptor bytes are generated in the DTF table at program assembly time. The content of the string depends on the imperative macro issued by the problem program to access the file. There is one descriptor byte for each CCW in the channel program. This descriptor byte is divided into three subfields, which perform the functions illustrated by Figure 31.

| | | | |
|---|---|---|---|
| Bit | 0 | 1 2 3 4 | 5 6 7 |
| Field | A | B | C |

Field A: References the command code.

Field B: A relative pointer to select one of the 11 basic CCWs (see Figure 33).

Field C: Further defines the command code, and modifies the flag byte as required.

Figure 31. Direct Access Descriptor Byte

Because the first CCW in any Disk Operating System channel program must be a seek command, the seek CCW is generated at program assembly time as the first CCW in the CCW build area, and is never modified. As each channel program is requested, the channel program builder subroutine is called to build the remainder of the CCW chain.

Before entering this subroutine, the logic module uses the macro switch to determine the address of the string of descriptor bytes for the macro issued (refer to Figure 32). After pointers are set to the current descriptor byte and the CCW build area, the subroutine isolates the relative pointer to the basic CCW needed (see Figure 33) and tests to determine if the CCW is to be a Transfer In Channel (TIC). Figure 33 shows the Basic CCWs used to build channel programs.

If fields A and C of the descriptor byte are zero, the CCW is to be a TIC. Field B determines the address of the CCW to which control is to be transferred. This address and the TIC command code are stored in the TIC CCW (see Figure 34). If the end of the descriptor string has not been reached, the subroutine returns to build the next CCW; otherwise, control returns to the calling routine.

If the CCW is not a TIC, Field B determines which of the basic CCWs is moved to the build area. Fields A and C of the descriptor byte are tested to see which fields in the CCW, if any, are to be modified (see Figure 34). A test is then made for the end of the descriptor string. If the end has not been reached, the routine returns to build the next CCW in the chain; otherwise, control returns to the calling routine.

| Macro | Option | FIXUNB | UNDEF | VARUNB | SPNUNB |
|---|---|---|---|---|---|
| READ ID | No options | DC X'871814' | DC X'C718BF14' | DC X'871816' | DC X'871816' |
| | KEYLEN | DC X'87182C' | DC X'C718BF2C' | DC X'87182A16' | DC X'87182A16' |
| | IDLOC | DC X'8718979E' | DC X'C719BF129C' | DC X'8718129E' | DC X'8718129E' |
| | KEYLEN, IDLOC | DC X'8718AF9E' | DC X'C718BF2A9C' | DC X'87182A129E' | DC X'87182A129E' |
| READ KEY | No options | DC X'8F1814' | DC X'BF8F1014' | DC X'A7188F1816' | DC X'A7180A1816' |
| | SRCHM | DC X'A718881814' | DC X'A718B8881014' | DC X'A718881816' | DC X'A7188A1816' |
| | IDLOC | DC X'8F18979E' | DC X'BF8F10129C' | DC X'A7188F18129E' | DC X'A7180A18129E' |
| | SRCHM, IDLOC | DC X'A7189A881014' | DC X'A718B8881014' | DC X'A7189A881016' | DC X'A7189A8A1016' |
| WRITE ID (No VERIFY)* | No options | DC X'871895' | DC X'871895' | DC X'871895' | DC X'871814871895' |
| | KEYLEN | DC X'8718AD' | DC X'8718AD' | DC X'8718AB95' | DC X'8718148718AB95' |
| | IDLOC | DC X'8718919E' | DC X'8718939C' | DC X'8718919E' | DC X'87181487189I9E' |
| | KEYLEN, IDLOC | DC X'8718A99E' | DC X'8718AB9C' | DC X'8718AB919E' | DC X'8718148718AB919E' |
| WRITE ID (VERIFY) | No options | DC X'871891871815' | DC X'871893871815' | DC X'871891871815' | DC X'871814871891871815' |
| | KEYLEN | DC X'8718A987182D' | DC X'8718AB87182D' | DC X'8718AB9187182B15' | DC X'8718148718AB9187182B15' |
| | IDLOC | DC X'87189187181I9E' | DC X'8718938718139C' | DC X'871891871811I9E' | DC X'87181487189187181I9E' |
| | KEYLEN, IDLOC | DC X'8718A98718299E' | DC X'8718AB87182B9C' | DC X'8718AB9187182B119E' | DC X'8718148718AB9187182B119E' |
| WRITE KEY (No VERIFY) | No options | DC X'8F1895' | DC X'8F1895' | DC X'A7188F1895' | DC X'A7180A18140A1895' |
| | SRCHM | DC X'A718881895' | DC X'A718881895' | DC X'A718881895' | DC X'A7189A8A10148A1895' |
| | IDLOC | DC X'8F18919E' | DC X'8F18939C' | DC X'A7188F18919E' | DC X'A7180A18140A18919E' |
| | SRCHM, IDLOC | DC X'A7189A881095' | DC X'A718B8881095' | DC X'A7189A881095' | DC X'A7189A8A1014BA1895' |
| WRITE KEY (VERIFY) | No options | DC X'8F18918F1815' | DC X'8F18938F1815' | DC X'A7188F18910A1815' | DC X'A7180A18140A18910A1815' |
| | SRCHM | DC X'A7188818918F1815' | DC X'A7188818938F1815' | DC X'A71888189I0A1815' | DC X'A7189A8A10148A18910A1815' |
| | IDLOC | DC X'8F18918F18119E' | DC X'8F18938F18139C' | DC X'A7188F18910A18119E' | DC'X'A7180A18140A18910A18119E' |
| | SRCHM, IDLOC | DC X'A7189A8810918F1815' | DC X'A718B88810938F1815' | DC X'A7189A8810910A1815' | DC X'A7189A8A10148A18910A1815' |

Macros WRITE AFTER and WRITE RZERO use the same strings. If AFTER is not specified in the DEFDA macro, the strings are not generated.

If AFTER is specified, these strings are generated for all record formats:

| | |
|---|---|
| DC X'C718D752C718B5' | WRITE RZERO |
| DC X'C71834' | READ RZERO |

| | | |
|---|---|---|
| And one of the following strings: | | |
| No VERIFY | DC X'C718B18718CD' | No options DC X'C718B18718CB95'<br>KEYLEN DC X'C718B18718CBAB95' |
| VERIFY | DC X'C718B18718C9C71831871840' | No options DC X'C718B18718CB91C7183187184B15'<br>KEYLEN DC X'C718B18718CBAB91C7183187184B2B15' |

One string for each macro to be used is generated, dependent upon the options specified in the DTFDA macro.

* Indicates the operation used in the example given of the Channel Program Builder.

Figure 32. Direct Access Channel Program Builder Strings

| Field B | Basic CCW | Function |
|---|---|---|
| 0000 | X'31',&SEEKADR+3,X'40',5 | Search identifier equal using the address specified in the user's track-reference field. |
| | X'31',&Filename.S+3,X'40',5 | If relative addressing is used. |
| 0001 | X'29',KEYARG,X'40',Key length | Search key equal for key specified by user in KEYARG field. |
| 0010 | X'06',&IOAREA+16,X'40',Data length | Read data into I/O area (FIXUNB and UNDEF records). |
| | X'06',&IOAREA,X'40',BLKSIZE | Read data into I/O area (VARUNB and SPNUNB records). |
| 0011 | X'12',&IDLOC,X'40',5 | Read count (CCHHR) into IDLOC. |
| | X'12',&Filename.I,X'40',5 | Read count (CCHHR) into work area in DTF table if relative addressing is used. |
| 0100 | X'39',&SEEKADR+3,X'40',4 | Search home address equal using the address specified in the user's track-reference field. |
| | X'39',&Filenamc.S,X'40',4 | If relative addressing is used. |
| 0101 | X'0E',&IOAREA+8,X'40',Key and Data length | Read key and data into I/O area (FIXUNB and UNDEF records). |
| | X'0E,&KEYARG,X'C0',Key length | Read key into user's KEYARG field (VARUNB and SPNUNB records). |
| 0110 | X'06',&Filename.K,X'40',8 | Read R0 data into work area in DTF table. |
| 0111 | X'12',&Filename.K,X'40',8 | Read R0 count into work area in DTF table. |
| 1000 | X'31',&Filename.F,X'40',5 | Search identifier equal using the address specified in the 5-byte work area in the DTF table. |
| 1001 | X'1E',&IOAREA,X'40',Count, Key, and Data length | Read count, key, and data into I/O area (FIXUNB and UNDEF records). |
| | X'1E',&Filename.C,X'C0',8 | Read count (CCHHRK$_{L}$D$_{L}$D$_{L}$) into work area in DTF table (SPNUNB records). |
| 1010 | X'11',&Filename.B+32,X'40',Track capacity | Control erase of track. |

Figure 33. Basic CCWs for Direct Access Channel Program Builder

| Field A | Field B | | | | Field C | | | Meaning |
|---|---|---|---|---|---|---|---|---|
| 1 | N | N | N | N | 1 | 1 | 1 | Basic CCW not modified. |
| 1 | N | N | N | N | 0 | 0 | 0 | Modify command code to multiple-track operation. |
| 1 | N | N | N | N | 0 | 0 | 1 | Modify command code in write operation. |
| 1 | N | N | N | N | 0 | 1 | 0 | Modify command code to multi-track, set SLI flag on. |
| 1 | N | N | N | N | 0 | 1 | 1 | Modify command code to write, set SLI flag on. |
| 1 | N | N | N | N | 1 | 0 | 0 | Modify command code to multi-track, set CC flag off. |
| 1 | N | N | N | N | 1 | 0 | 1 | Modify command code to write, set CC flag off. |
| 1 | N | N | N | N | 1 | 1 | 0 | Modify command code to multi-track, set CC flag off, SLI flag on. |
| 0 | N | N | N | N | 0 | 0 | 1 | Set SKIP flag on in CCW. |
| 0 | N | N | N | N | 0 | 1 | 0 | Set SLI flag on in CCW. |
| 0 | N | N | N | N | 0 | 1 | 1 | Set SLI and SKIP flag on in CCW. |
| 0 | N | N | N | N | 1 | 0 | 0 | Set CC flag off in CCW. |
| 0 | N | N | N | N | 1 | 0 | 1 | Set CC flag off, SKIP flag on in CCW. |
| 0 | N | N | N | N | 1 | 1 | 0 | Set CC flag off, SLI flag on in CCW. |
| 0 | N | N | N | N | 1 | 1 | 1 | Set CC flag off, SLI and SKIP flag on in CCW. |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | TIC to *- 32 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | TIC to *- 24 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | TIC to *- 16 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | TIC to *- 8 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | TIC to *- 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | TIC to *+ 8 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | TIC to *+ 16 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | TIC to *+ 24 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | TIC to *+ 32 |

Bit 0 1 2 3 4 5 6 7

NOTE: NNNN = bits 1-4 of the descriptor byte and is one of the 11 bit combinations shown in Figure 31 under the column heading Field B. This field contains the relative pointer to the basic CCW (Figure 31).

CC - Command Chaining
SLI - Suppress Length Indicator
SKIP - Suppress Transfer of Information to main storage.

Figure 34. Direct Access Channel Program Descriptor Bytes

| Descriptor Byte | CCW Built | Meaning |
|---|---|---|
| | X'07', &SEEKADR+1, X'00', 6 | Seek to the address specified in the user's track reference field. |
| X'87' | X'31', &SEEKADR+3, X'40', 5 | Search identifier equal the address specified in the user's track reference field. |
| X'18' | X'08', Pointer to *-8 | TIC to *-8. |
| X'93' | X'05', &IOAREA+16, X'60', Data length | Write the data portion of the record from the IOAREA. |
| X'9C' | X'12', &IDLOC, X'00', 5 | Read the count field into IDLOC. |

Figure 35. Example of the Direct Access Channel Program for a WRITE ID Macro

The following discussion describes how the direct access channel program builder constructs a channel program for the given example.

Example: Write an undefined record referenced by ID in the location specified by the user's track-reference field, and return the corresponding track record identifier (CCHHR) in IDLOC (option).

Figure 35 illustrates the CCWs needed for the complete channel program to accomplish this operation. In all, five CCWs are required. The first CCW (seek) is generated at assembly time and the remaining four CCWs are built using the string of descriptor bytes included as part of the DTF table for the WRITE ID macro. Refer to Figure 32. The descriptor string for the WRITE ID macro is:

X'8718939C'

Except for the Seek CCW that is generated for any channel program at assembly time and never modified, each pair of hexadecimal characters (descriptor byte) corresponds to one CCW. Thus, X'87' corresponds to the CCW to Search Identifier Equal as illustrated in part 1 of the explanation that follows.

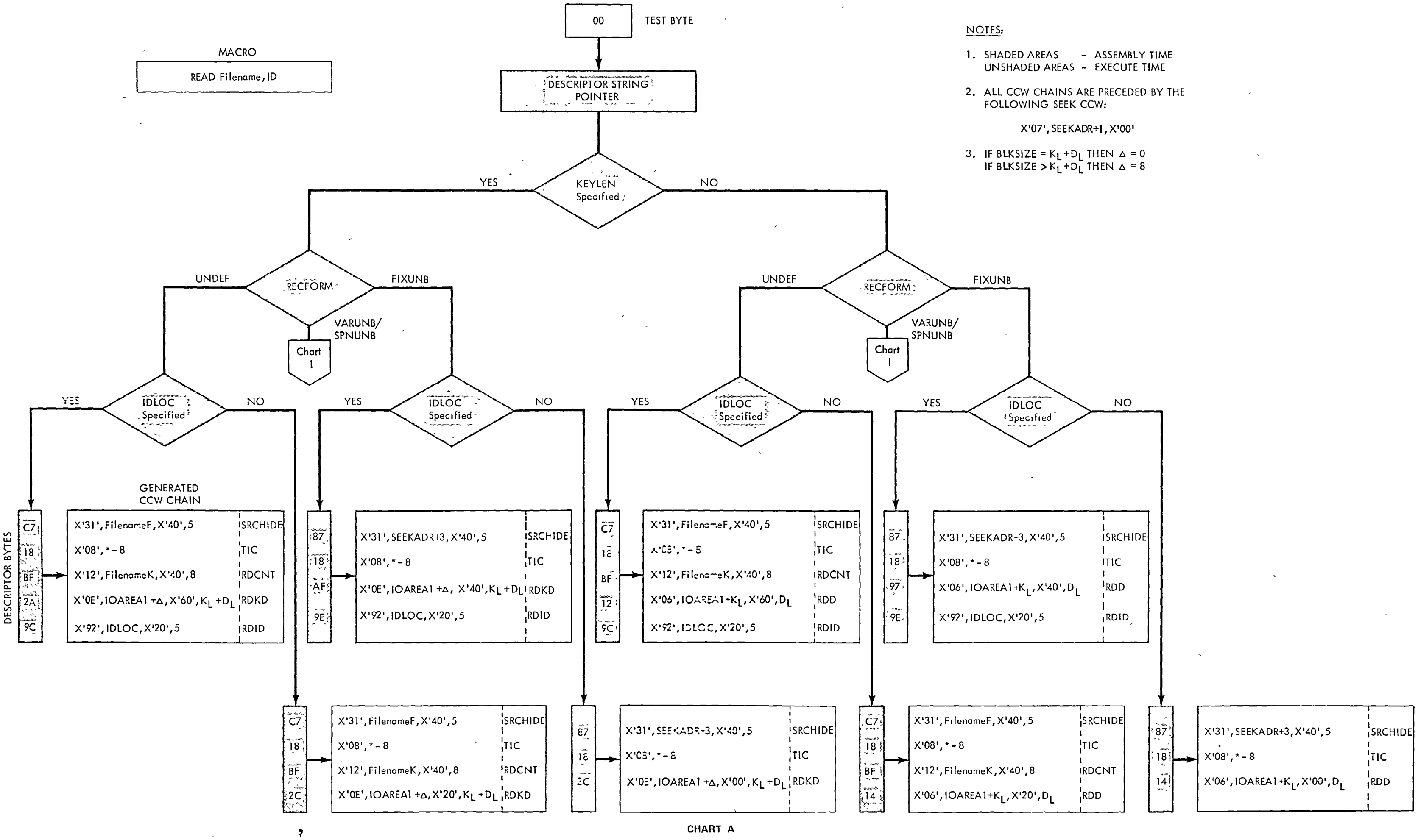
The CCW chain is generated from the descriptor string in this order:

1. X'87' (10000111): Figure 33 illustrates that the CCW for a descriptor byte with a B-field = 0000 is a Search Identifier Equal CCW. Figure 34 further illustrates that a descriptor byte with an A-field = 1 and a C-field = 111 performs no modification of the basic CCW. Therefore, the second CCW (the first being the Seek CCW) in the channel program CCW chain is an unmodified Search Identifier Equal CCW, X'31',&SEEKADR+3,X'40',5 (refer to Figure 35).

2. X'18' (00011000): Because both the A and C fields are all zeros (a characteristic of a descriptor byte used to generate a TIC CCW), the second descriptor byte in the string generates a TIC CCW for the third CCW in the channel program. Figure 34 illustrates that a descriptor byte of this kind with a B-field = 0011 supplies the CCW, TIC to * - 8 (refer to Figure 35 for generated CCW).

3. X'93' (10010011): The B-field = 0010 in this descriptor byte indicates that the next CCW in the channel program chain will be the third basic CCW (refer to Figure 33). Because the A-field = 1 and the C-field = 011, Figure 34 shows that the command code is modified to a WRITE and that the SLI (Suppress Length Indicator) bit is turned on.

4. X'9C' (10011100): The B-field = 0011 in the last descriptor byte indicates that the last CCW in the chain will be the fourth basic CCW in Figure 33, Read Count into IDLOC. A descriptor byte with an A-field = 1 and a C-field = 100 indicates that the command code is modified for a multitrack operation and that the command chaining bit is turned off to signify the end of the channel program (Figure 34).

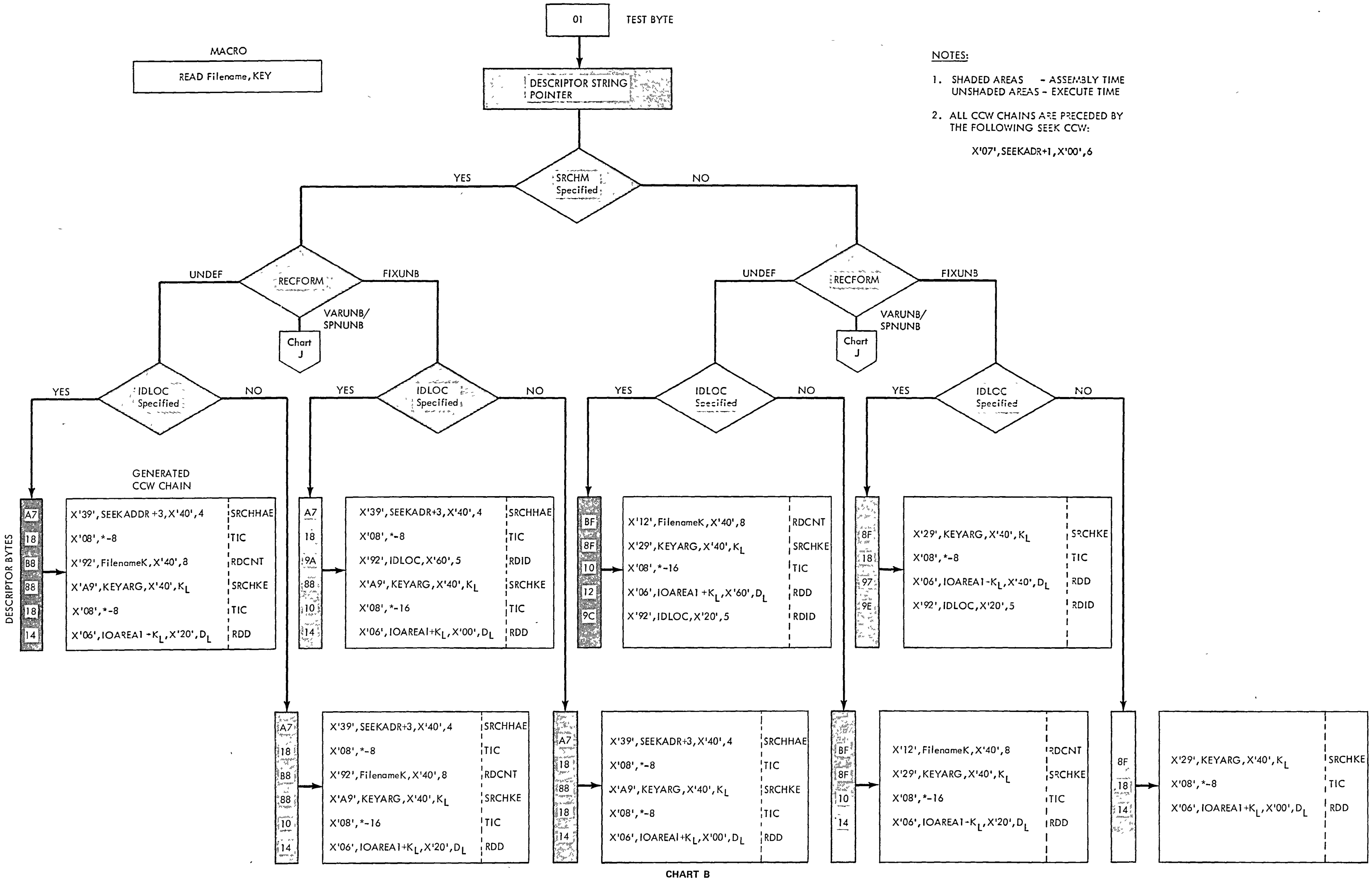

MACRO
READ Filename, ID
00
TEST BYTE
DESCRIPTOR STRING POINTER
YES
KEYLEN Specified
NO
UNDEF
RECFORM
FIXUNB
VARUNB/ SPNUNB
Chart I
YES
IDLOC Specified
NO
GENERATED CCW CHAIN
DESCRIPTOR BYTES
C7
18
BF
2A
9C
X'31', FilenameF, X'40', 5
SRCHIDE
X'08', *-8
TIC
X'12', FilenameK, X'40', 8
RDCNT
X'0E', IOAREA1 +Δ, X'60', K_L +D_L
RDKD
X'92', IDLOC, X'20', 5
RDID
87
18
AF
9E
X'31', SEEKADR+3, X'40', 5
SRCHIDE
X'08', *-8
TIC
X'0E', IOAREA1 +Δ, X'40', K_L +D_L
RDKD
X'92', IDLOC, X'20', 5
RDID
C7
18
BF
12
9C
X'31', FilenameF, X'40', 5
SRCHIDE
X'08', *-8
TIC
X'12', FilenameK, X'40', 8
RDCNT
X'06', IOAREA1 -K_L, X'60', D_L
RDD
X'92', IDLOC, X'20', 5
RDID
87
18
97
9E
X'31', SEEKADR+3, X'40', 5
SRCHIDE
X'08', *-8
TIC
X'06', IOAREA1+K_L, X'40', D_L
RDD
X'92', IDLOC, X'20', 5
RDID
C7
18
BF
2C
X'31', FilenameF, X'40', 5
SRCHIDE
X'08', *-8
TIC
X'12', FilenameK, X'40', 8
RDCNT
X'0E', IOAREA1 +Δ, X'20', K_L +D_L
RDKD
87
18
2C
X'31', SEEKADR+3, X'40', 5
SRCHIDE
X'08', *-8
TIC
X'0E', IOAREA1 +Δ, X'00', K_L +D_L
RDKD
C7
18
BF
14
X'31', FilenameF, X'40', 5
SRCHIDE
X'08', *-8
TIC
X'12', FilenameK, X'40', 8
RDCNT
X'06', IOAREA1+K_L, X'20', D_L
RDD
87
18
14
X'31', SEEKADR+3, X'40', 5
SRCHIDE
X'08', *-8
TIC
X'06', IOAREA1+K_L, X'00', D_L
RDD
CHART A

NOTES:

1. SHADED AREAS - ASSEMBLY TIME
   UNSHADED AREAS - EXECUTE TIME
2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW:

   X'07', SEEKADR+1, X'00'
3. IF BLKSIZE = $K_L + D_L$ THEN $\Delta = 0$
   IF BLKSIZE > $K_L + D_L$ THEN $\Delta = 8$

Figure 36. DA Channel Programs (Part 1 of 14)



MACRO
READ Filename, KEY
01
TEST BYTE
DESCRIPTOR STRING POINTER
SRCHM Specified
YES
NO
RECFORM
UNDEF
FIXUNB
VARUNB/ SPNUNB
Chart J
IDLOC Specified
YES
NO
GENERATED CCW CHAIN
DESCRIPTOR BYTES
A7 18 B8 88 18 14
X'39', SEEKADDR +3, X'40', 4 SRCHHAE
X'08', *-8 TIC
X'92', FilenameK, X'40', 8 RDCNT
X'A9', KEYARG, X'40', K_L SRCHKE
X'08', *-8 TIC
X'06', IOAREA1 -K_L, X'20', D_L RDD
A7 18 9A 88 10 14
X'39', SEEKADR+3, X'40', 4 SRCHHAE
X'08', *-8 TIC
X'92', IDLOC, X'60', 5 RDID
X'A9', KEYARG, X'40', K_L SRCHKE
X'08', *-16 TIC
X'06', IOAREA1+K_L, X'00', D_L RDD
BF 8F 10 12 9C
X'12', FilenameK, X'40', 8 RDCNT
X'29', KEYARG, X'40', K_L SRCHKE
X'08', *-16 TIC
X'06', IOAREA1 +K_L, X'60', D_L RDD
X'92', IDLOC, X'20', 5 RDID
8F 18 97 9E
X'29', KEYARG, X'40', K_L SRCHKE
X'08', *-8 TIC
X'06', IOAREA1-K_L, X'40', D_L RDD
X'92', IDLOC, X'20', 5 RDID
A7 18 B8 88 10 14
X'39', SEEKADR+3, X'40', 4 SRCHHAE
X'08', *-8 TIC
X'92', FilenameK, X'40', 8 RDCNT
X'A9', KEYARG, X'40', K_L SRCHKE
X'08', *-16 TIC
X'06', IOAREA1+K_L, X'20', D_L RDD
A7 18 88 18 14
X'39', SEEKADR+3, X'40', 4 SRCHHAE
X'08', *-8 TIC
X'A9', KEYARG, X'40', K_L SRCHKE
X'08', *-8 TIC
X'06', IOAREA1+K_L, X'00', D_L RDD
BF 8F 10 14
X'12', FilenameK, X'40', 8 RDCNT
X'29', KEYARG, X'40', K_L SRCHKE
X'08', *-16 TIC
X'06', IOAREA1-K_L, X'20', D_L RDD
8F 18 14
X'29', KEYARG, X'40', K_L SRCHKE
X'08', *-8 TIC
X'06', IOAREA1+K_L, X'00', D_L RDD
CHART B

NOTES:

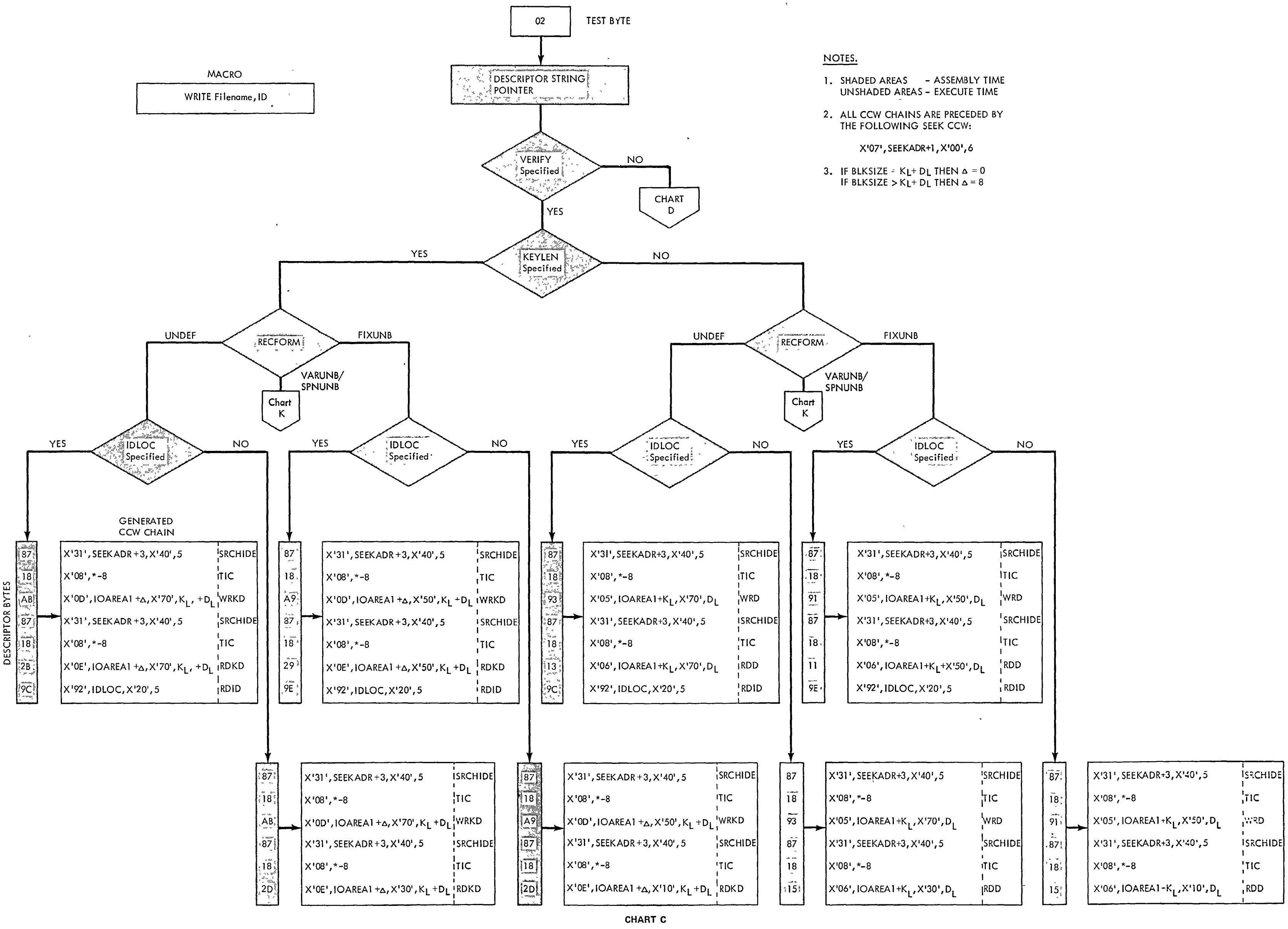
1. SHADED AREAS - ASSEMBLY TIME
   UNSHADED AREAS - EXECUTE TIME

2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW:

   X'07', SEEKADR+1, X'00', 6

Figure 36. DA Channel Programs (Part 2 of 14)

Figure 36. DA Channel Programs (Part 3 of 14)



NOTES.

1. SHADED AREAS - ASSEMBLY TIME
   UNSHADED AREAS - EXECUTE TIME
2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW:

   X'07',SEEKADR+1,X'00',6
3. IF BLKSIZE = $K_L$+ $D_L$ THEN $\Delta$ = 0
   IF BLKSIZE > $K_L$+ $D_L$ THEN $\Delta$ = 8

| DESCRIPTOR BYTES | GENERATED CCW CHAIN | |
|---|---|---|
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| AB | X'0D',IOAREA1 +$\Delta$,X'70',$K_L$, +$D_L$ | WRKD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 2B | X'0E',IOAREA1 +$\Delta$,X'70',$K_L$, +$D_L$ | RDKD |
| 9C | X'92',IDLOC,X'20',5 | RDID |

| | | |
|---|---|---|
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| A9 | X'0D',IOAREA1 +$\Delta$,X'50',$K_L$ +$D_L$ | WRKD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 29 | X'0E',IOAREA1 +$\Delta$,X'50',$K_L$ +$D_L$ | RDKD |
| 9E | X'92',IDLOC,X'20',5 | RDID |

| | | |
|---|---|---|
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 93 | X'05',IOAREA1+$K_L$,X'70',$D_L$ | WRD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 13 | X'06',IOAREA1+$K_L$,X'70',$D_L$ | RDD |
| 9C | X'92',IDLOC,X'20',5 | RDID |

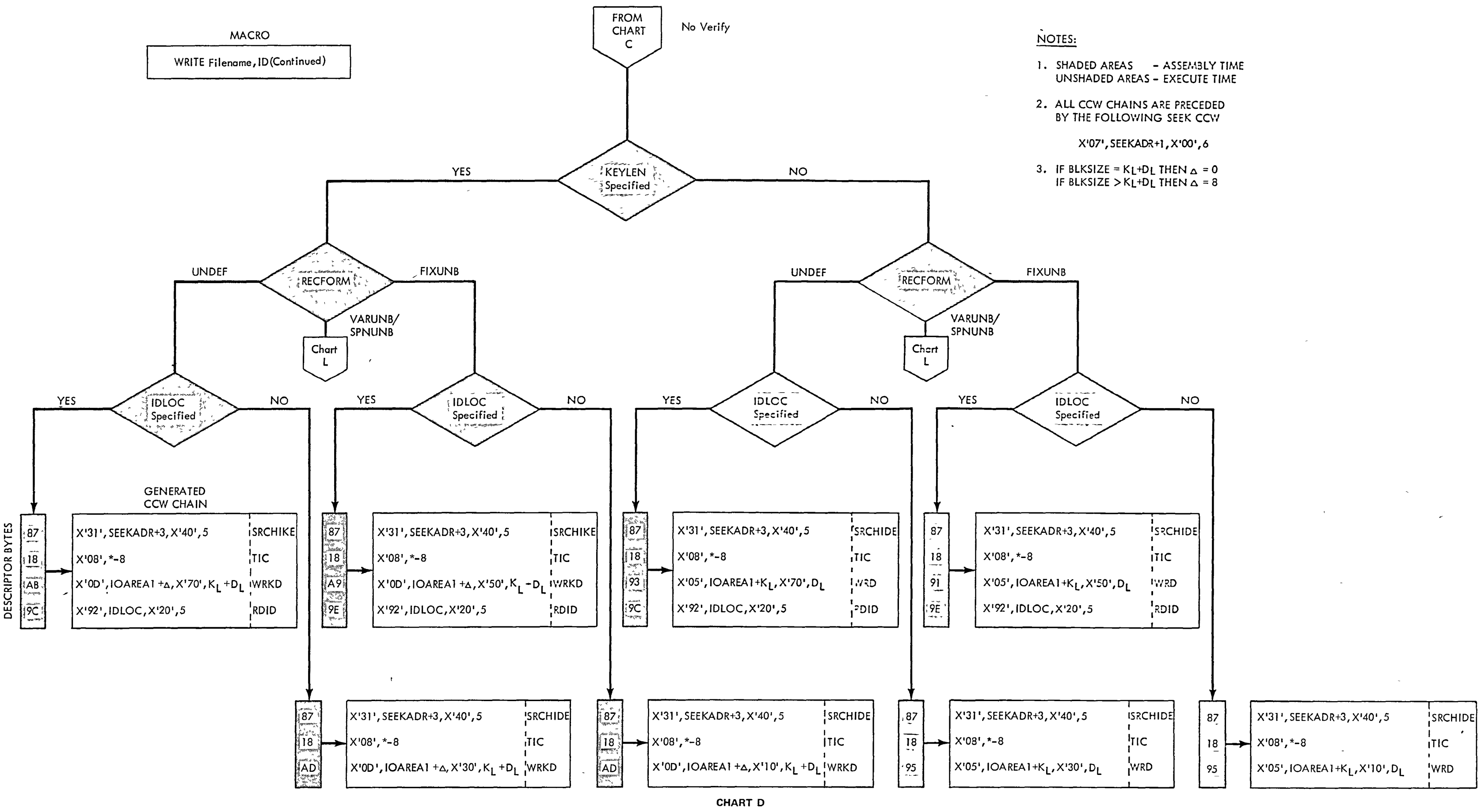
| | | |
|---|---|---|
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 91 | X'05',IOAREA1+$K_L$,X'50',$D_L$ | WRD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 11 | X'06',IOAREA1+$K_L$+X'50',$D_L$ | RDD |
| 9E | X'92',IDLOC,X'20',5 | RDID |

| | | |
|---|---|---|
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| AB | X'0D',IOAREA1 +$\Delta$,X'70',$K_L$ +$D_L$ | WRKD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 2D | X'0E',IOAREA1 +$\Delta$,X'30',$K_L$ +$D_L$ | RDKD |

| | | |
|---|---|---|
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| A9 | X'0D',IOAREA1 +$\Delta$,X'50',$K_L$ +$D_L$ | WRKD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 2D | X'0E',IOAREA1 +$\Delta$,X'10',$K_L$ +$D_L$ | RDKD |

| | | |
|---|---|---|
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 93 | X'05',IOAREA1+$K_L$,X'70',$D_L$ | WRD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 15 | X'06',IOAREA1+$K_L$,X'30',$D_L$ | RDD |

| | | |
|---|---|---|
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 91 | X'05',IOAREA1+$K_L$,X'50',$D_L$ | WRD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 15 | X'06',IOAREA1-$K_L$,X'10',$D_L$ | RDD |



NOTES:

1. SHADED AREAS - ASSEMBLY TIME
   UNSHADED AREAS - EXECUTE TIME
2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW

   X'07', SEEKADR+1, X'00', 6
3. IF BLKSIZE = $K_L+D_L$ THEN Δ = 0
   IF BLKSIZE > $K_L+D_L$ THEN Δ = 8

Figure 36. DA Channel Programs (Part 4 of 14)

Figure 36. DA Channel Programs (Part 5 of 14)

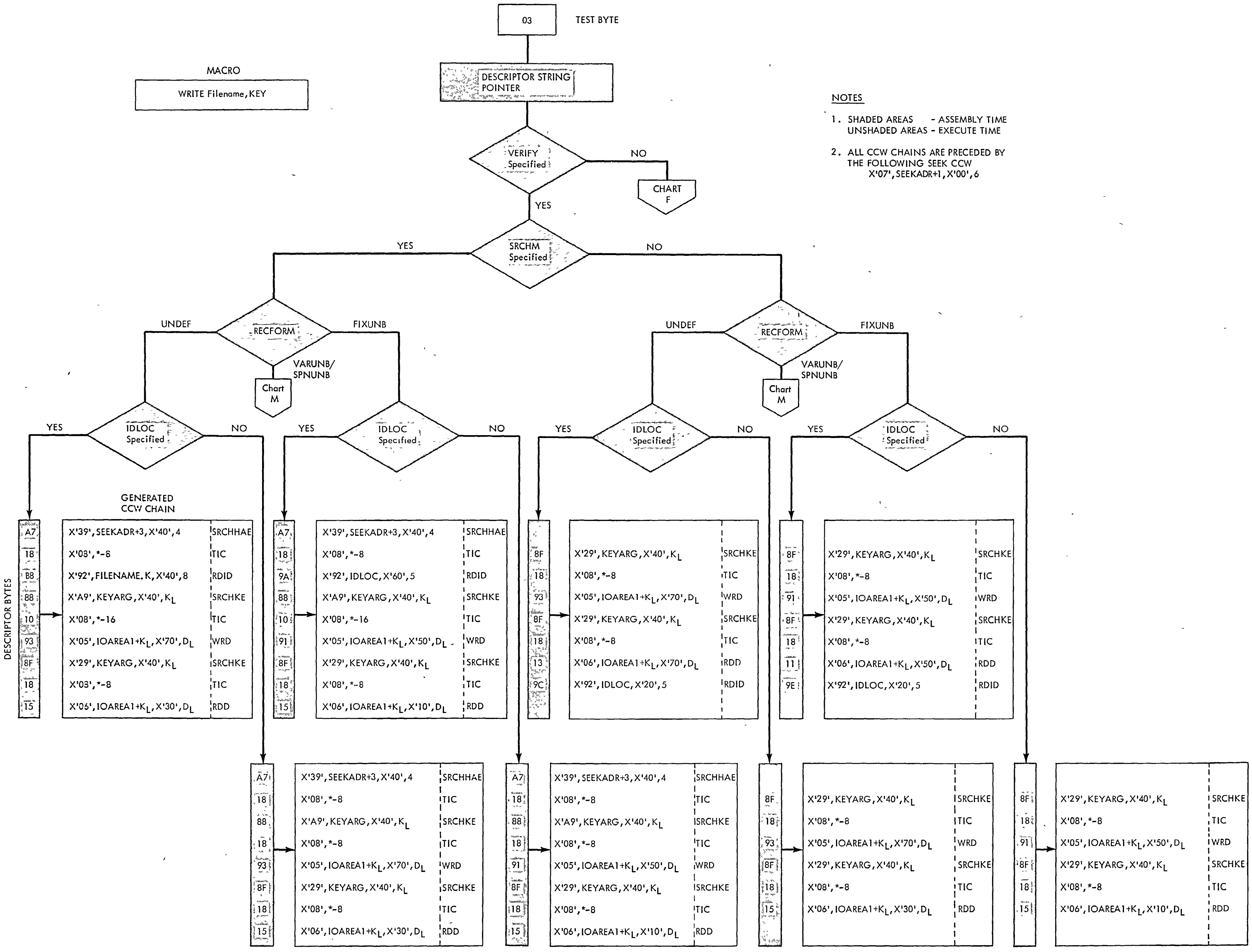

MACRO
WRITE Filename,KEY
03
TEST BYTE
DESCRIPTOR STRING POINTER
VERIFY Specified
NO
CHART F
YES
SRCHM Specified
YES
NO
RECFORM
UNDEF
FIXUNB
VARUNB/ SPNUNB
Chart M
RECFORM
UNDEF
FIXUNB
VARUNB/ SPNUNB
Chart M
IDLOC Specified
YES
NO
IDLOC Specified
YES
NO
IDLOC Specified
YES
NO
IDLOC Specified
YES
NO
GENERATED CCW CHAIN
DESCRIPTOR BYTES
A7 18 B8 88 10 93 8F 18 15
X'39',SEEKADR+3,X'40',4 SRCHHAE
X'08',*-8 TIC
X'92',FILENAME.K,X'40',8 RDID
X'A9',KEYARG,X'40',K_L SRCHKE
X'08',*-16 TIC
X'05',IOAREA1+K_L,X'70',D_L WRD
X'29',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'06',IOAREA1+K_L,X'30',D_L RDD
A7 18 9A 88 10 91 8F 18 15
X'39',SEEKADR+3,X'40',4 SRCHHAE
X'08',*-8 TIC
X'92',IDLOC,X'60',5 RDID
X'A9',KEYARG,X'40',K_L SRCHKE
X'08',*-16 TIC
X'05',IOAREA1+K_L,X'50',D_L WRD
X'29',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'06',IOAREA1+K_L,X'10',D_L RDD
8F 18 93 8F 18 13 9C
X'29',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'05',IOAREA1+K_L,X'70',D_L WRD
X'29',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'06',IOAREA1+K_L,X'70',D_L RDD
X'92',IDLOC,X'20',5 RDID
8F 18 91 8F 18 11 9E
X'29',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'05',IOAREA1+K_L,X'50',D_L WRD
X'29',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'06',IOAREA1+K_L,X'50',D_L RDD
X'92',IDLOC,X'20',5 RDID
A7 18 88 18 93 8F 18 15
X'39',SEEKADR+3,X'40',4 SRCHHAE
X'08',*-8 TIC
X'A9',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'05',IOAREA1+K_L,X'70',D_L WRD
X'29',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'06',IOAREA1+K_L,X'30',D_L RDD
A7 18 88 18 91 8F 18 15
X'39',SEEKADR+3,X'40',4 SRCHHAE
X'08',*-8 TIC
X'A9',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'05',IOAREA1+K_L,X'50',D_L WRD
X'29',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'06',IOAREA1+K_L,X'10',D_L RDD
8F 18 93 8F 18 15
X'29',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'05',IOAREA1+K_L,X'70',D_L WRD
X'29',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'06',IOAREA1+K_L,X'30',D_L RDD
8F 18 91 8F 18 15
X'29',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'05',IOAREA1+K_L,X'50',D_L WRD
X'29',KEYARG,X'40',K_L SRCHKE
X'08',*-8 TIC
X'06',IOAREA1+K_L,X'10',D_L RDD

CHART E

NOTES

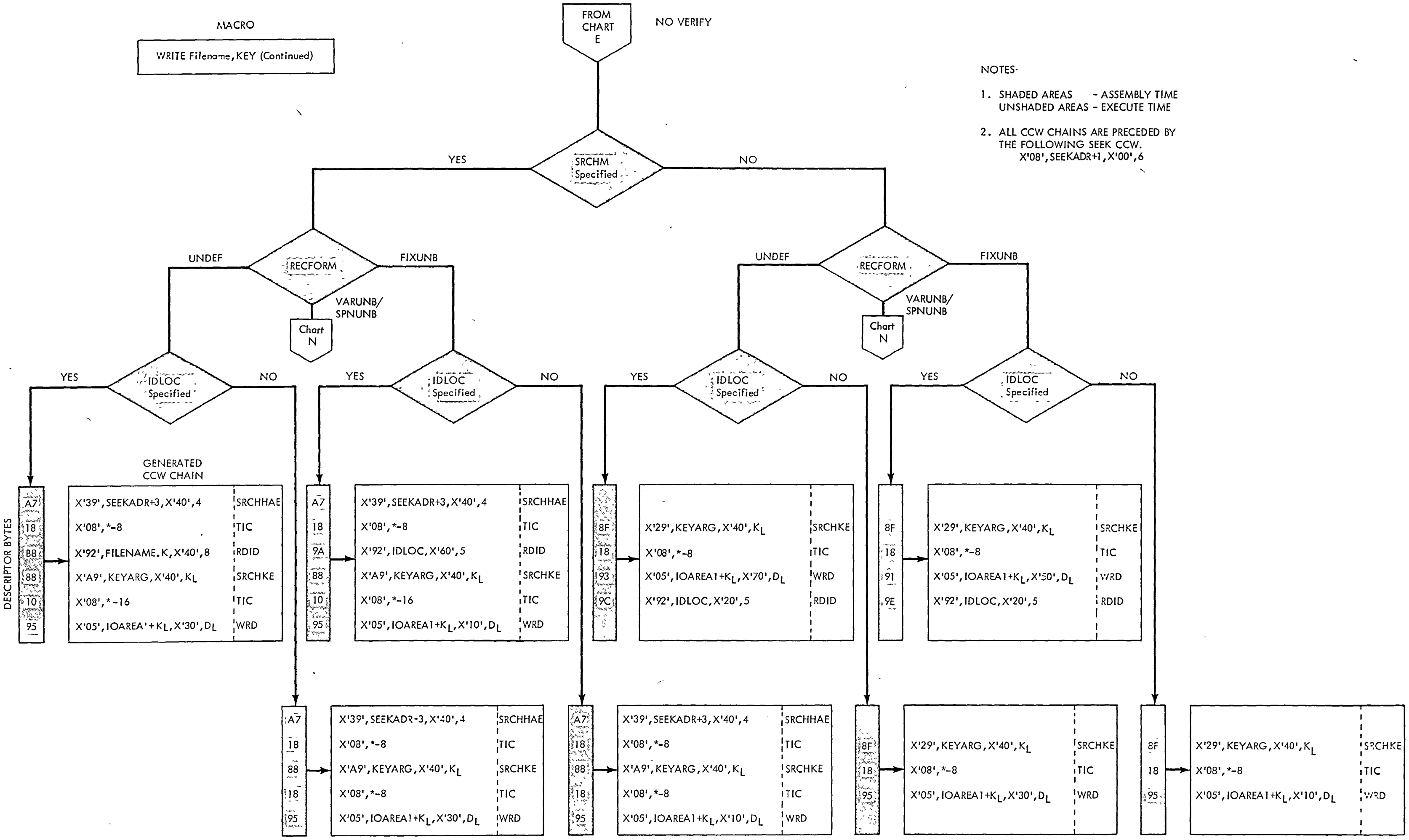
1. SHADED AREAS - ASSEMBLY TIME
   UNSHADED AREAS - EXECUTE TIME
2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW
   X'07',SEEKADR+1,X'00',6

MACRO

WRITE Filename,KEY (Continued)

NOTES:

1. SHADED AREAS - ASSEMBLY TIME
   UNSHADED AREAS - EXECUTE TIME
2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW.
   X'08',SEEKADR+1,X'00',6



| | | |
|---|---|---|
| A7 | X'39',SEEKADR+3,X'40',4 | SRCHHAE |
| 18 | X'08',*-8 | TIC |
| B8 | X'92',FILENAME.K,X'40',8 | RDID |
| 88 | X'A9',KEYARG,X'40',$K_L$ | SRCHKE |
| 10 | X'08',* -16 | TIC |
| 95 | X'05',IOAREA'+$K_L$,X'30',$D_L$ | WRD |

| | | |
|---|---|---|
| A7 | X'39',SEEKADR+3,X'40',4 | SRCHHAE |
| 18 | X'08',*-8 | TIC |
| 9A | X'92',IDLOC,X'60',5 | RDID |
| 88 | X'A9',KEYARG,X'40',$K_L$ | SRCHKE |
| 10 | X'08',*-16 | TIC |
| 95 | X'05',IOAREA1+$K_L$,X'10',$D_L$ | WRD |

| | | |
|---|---|---|
| 8F | X'29',KEYARG,X'40',$K_L$ | SRCHKE |
| 18 | X'08',*-8 | TIC |
| 93 | X'05',IOAREA1+$K_L$,X'70',$D_L$ | WRD |
| 9C | X'92',IDLOC,X'20',5 | RDID |

| | | |
|---|---|---|
| 8F | X'29',KEYARG,X'40',$K_L$ | SRCHKE |
| 18 | X'08',*-8 | TIC |
| 91 | X'05',IOAREA1+$K_L$,X'50',$D_L$ | WRD |
| 9E | X'92',IDLOC,X'20',5 | RDID |

| | | |
|---|---|---|
| A7 | X'39',SEEKADR-3,X'40',4 | SRCHHAE |
| 18 | X'08',*-8 | TIC |
| 88 | X'A9',KEYARG,X'40',$K_L$ | SRCHKE |
| 18 | X'08',*-8 | TIC |
| 95 | X'05',IOAREA1+$K_L$,X'30',$D_L$ | WRD |

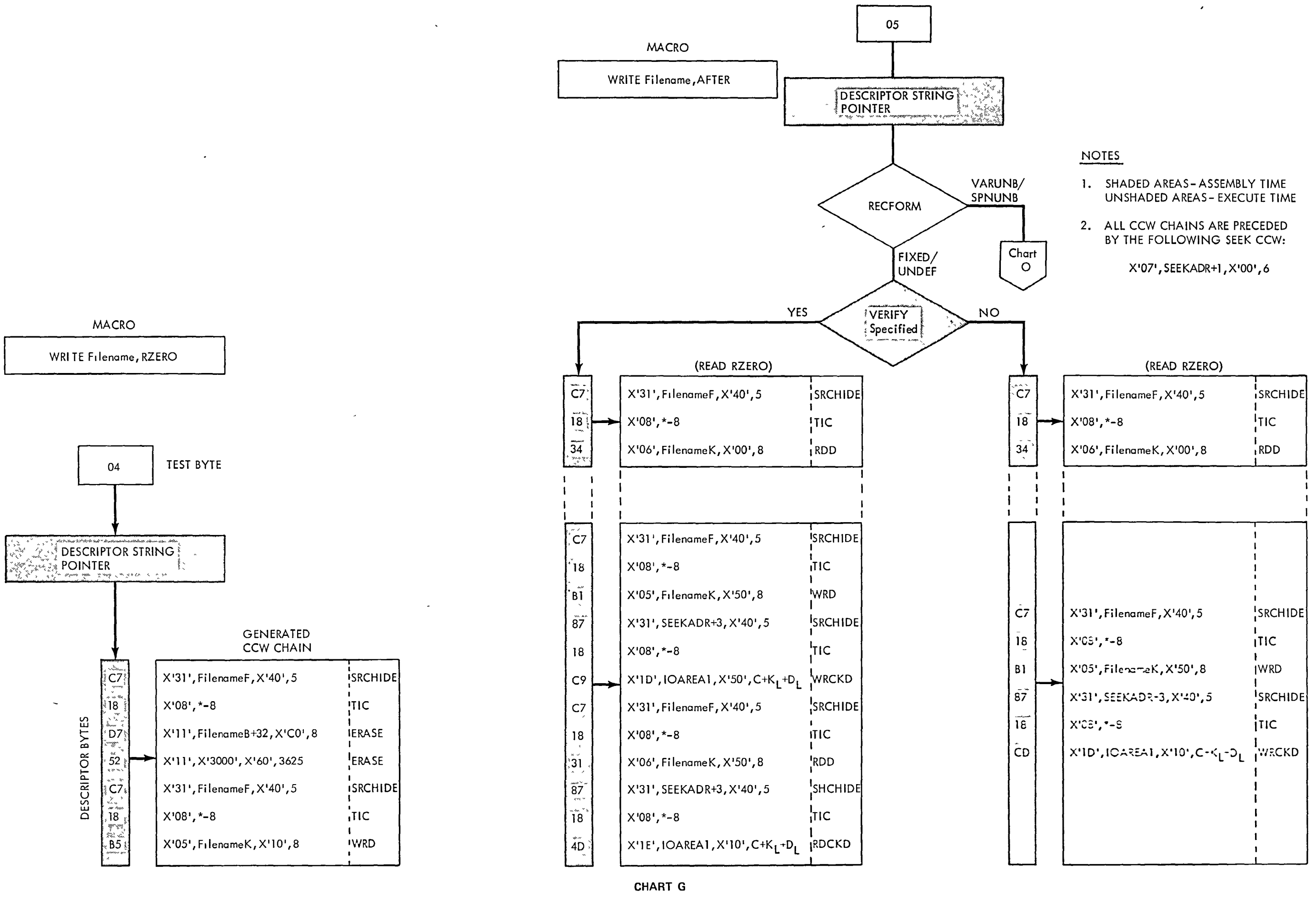
| | | |
|---|---|---|
| A7 | X'39',SEEKADR+3,X'40',4 | SRCHHAE |
| 18 | X'08',*-8 | TIC |
| 88 | X'A9',KEYARG,X'40',$K_L$ | SRCHKE |
| 18 | X'08',*-8 | TIC |
| 95 | X'05',IOAREA1+$K_L$,X'10',$D_L$ | WRD |

| | | |
|---|---|---|
| 8F | X'29',KEYARG,X'40',$K_L$ | SRCHKE |
| 18 | X'08',*-8 | TIC |
| 95 | X'05',IOAREA1+$K_L$,X'30',$D_L$ | WRD |

| | | |
|---|---|---|
| 8F | X'29',KEYARG,X'40',$K_L$ | SRCHKE |
| 18 | X'08',*-8 | TIC |
| 95 | X'05',IOAREA1+$K_L$,X'10',$D_L$ | WRD |

CHART F

Figure 36. DA Channel Programs (Part 6 of 14)


MACRO
WRITE Filename,RZERO
04
TEST BYTE
DESCRIPTOR STRING POINTER
DESCRIPTOR BYTES
GENERATED CCW CHAIN
C7 X'31',FilenameF,X'40',5 SRCHIDE
18 X'08',*-8 TIC
D7 X'11',FilenameB+32,X'C0',8 ERASE
52 X'11',X'3000',X'60',3625 ERASE
C7 X'31',FilenameF,X'40',5 SRCHIDE
18 X'08',*-8 TIC
B5 X'05',FilenameK,X'10',8 WRD
MACRO
WRITE Filename,AFTER
05
DESCRIPTOR STRING POINTER
RECFORM
VARUNB/ SPNUNB
Chart O
FIXED/ UNDEF
VERIFY Specified
YES
NO
(READ RZERO)
C7 X'31',FilenameF,X'40',5 SRCHIDE
18 X'08',*-8 TIC
34 X'06',FilenameK,X'00',8 RDD
C7 X'31',FilenameF,X'40',5 SRCHIDE
18 X'08',*-8 TIC
B1 X'05',FilenameK,X'50',8 WRD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
C9 X'1D',IOAREA1,X'50',C+K_L+D_L WRCKD
C7 X'31',FilenameF,X'40',5 SRCHIDE
18 X'08',*-8 TIC
31 X'06',FilenameK,X'50',8 RDD
87 X'31',SEEKADR+3,X'40',5 SHCHIDE
18 X'08',*-8 TIC
4D X'1E',IOAREA1,X'10',C+K_L+D_L RDCKD
CHART G
(READ RZERO)
C7 X'31',FilenameF,X'40',5 SRCHIDE
18 X'08',*-8 TIC
34 X'06',FilenameK,X'00',8 RDD
C7 X'31',FilenameF,X'40',5 SRCHIDE
18 X'08',*-8 TIC
B1 X'05',FilenameK,X'50',8 WRD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
CD X'1D',IOAREA1,X'10',C+K_L+D_L WRCKD

NOTES

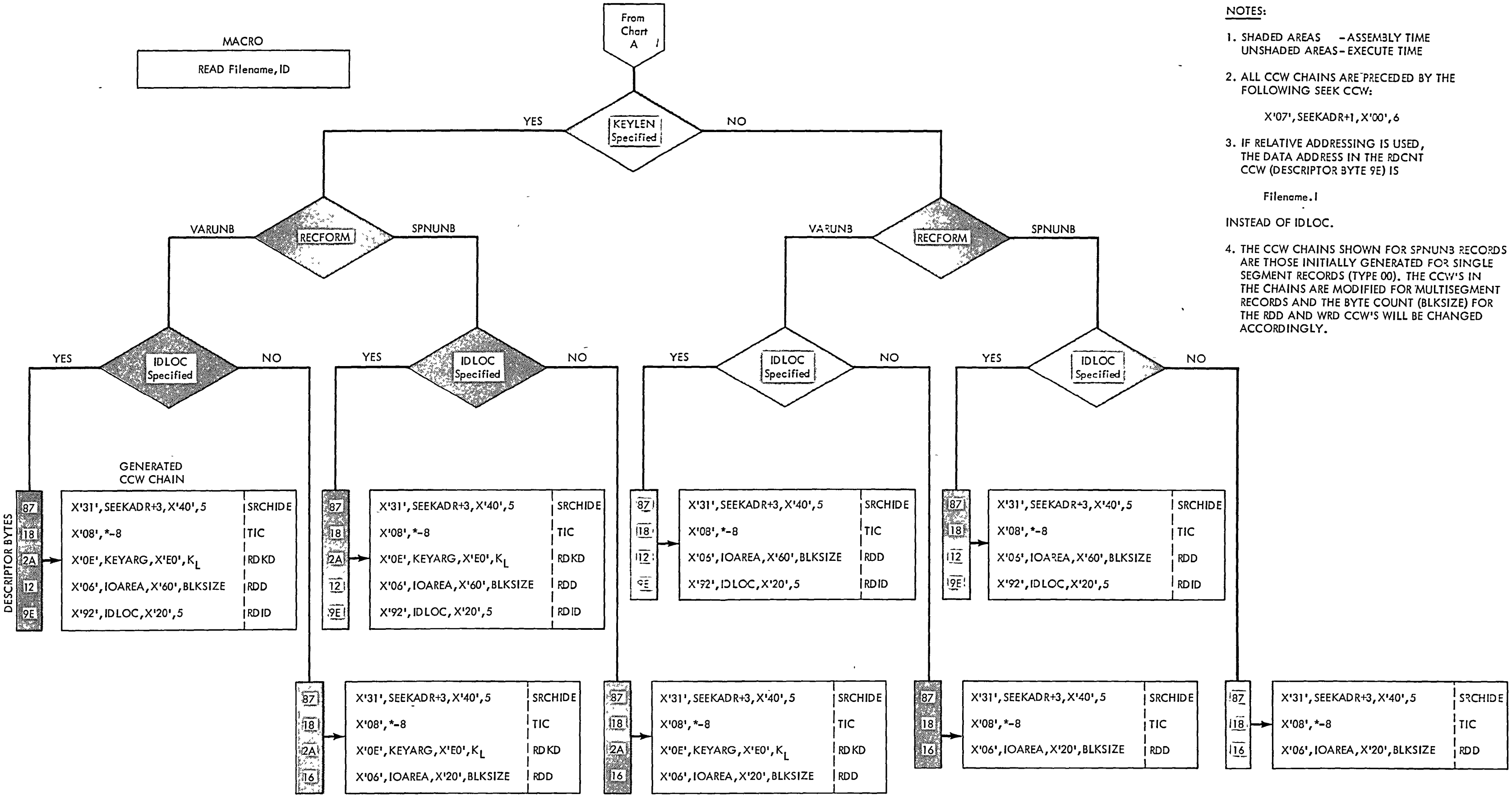
1. SHADED AREAS - ASSEMBLY TIME
   UNSHADED AREAS - EXECUTE TIME

2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW:

   X'07',SEEKADR+1,X'00',6

Figure 36. DA Channel Programs (Part 7 of 14)

Figure 36. DA Channel Programs (Part 8 of 14)



NOTES:

1. SHADED AREAS -ASSEMBLY TIME
   UNSHADED AREAS- EXECUTE TIME

2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW:

   X'07',SEEKADR+1,X'00',6

3. IF RELATIVE ADDRESSING IS USED, THE DATA ADDRESS IN THE RDCNT CCW (DESCRIPTOR BYTE 9E) IS

   Filename.I

INSTEAD OF IDLOC.

4. THE CCW CHAINS SHOWN FOR SPNUNB RECORDS ARE THOSE INITIALLY GENERATED FOR SINGLE SEGMENT RECORDS (TYPE 00). THE CCW'S IN THE CHAINS ARE MODIFIED FOR MULTISEGMENT RECORDS AND THE BYTE COUNT (BLKSIZE) FOR THE RDD AND WRD CCW'S WILL BE CHANGED ACCORDINGLY.

CHART I

Figure 36. DA Channel Programs (Part 9 of 14)

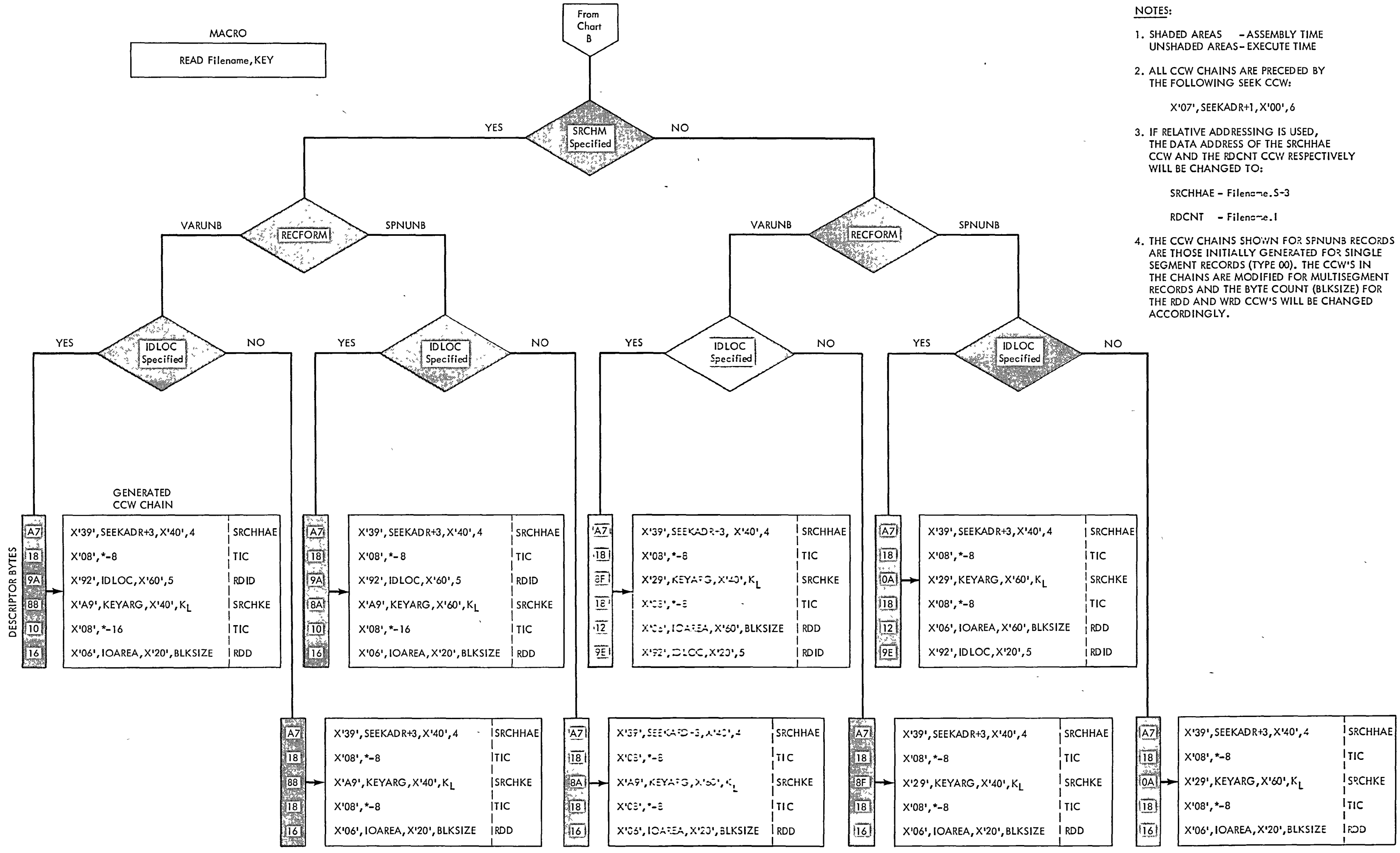

MACRO
READ Filename, KEY
From Chart B
SRCHM Specified
YES
NO
RECFORM
VARUNB
SPNUNB
IDLOC Specified
YES
NO
GENERATED CCW CHAIN
DESCRIPTOR BYTES
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
9A X'92',IDLOC,X'60',5 RDID
88 X'A9',KEYARG,X'40',K_L SRCHKE
10 X'08',*-16 TIC
16 X'06',IOAREA,X'20',BLKSIZE RDD
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
9A X'92',IDLOC,X'60',5 RDID
8A X'A9',KEYARG,X'60',K_L SRCHKE
10 X'08',*-16 TIC
16 X'06',IOAREA,X'20',BLKSIZE RDD
A7 X'39',SEEKADR-3, X'40',4 SRCHHAE
18 X'08',*-8 TIC
8F X'29',KEYARG,X'40',K_L SRCHKE
18 X'08',*-8 TIC
12 X'06',IOAREA,X'60',BLKSIZE RDD
9E X'92',IDLOC,X'20',5 RDID
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
0A X'29',KEYARG,X'60',K_L SRCHKE
18 X'08',*-8 TIC
12 X'06',IOAREA,X'60',BLKSIZE RDD
9E X'92',IDLOC,X'20',5 RDID
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
88 X'A9',KEYARG,X'40',K_L SRCHKE
18 X'08',*-8 TIC
16 X'06',IOAREA,X'20',BLKSIZE RDD
A7 X'39',SEEKADR-3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
8A X'A9',KEYARG,X'60',K_L SRCHKE
18 X'08',*-8 TIC
16 X'06',IOAREA,X'20',BLKSIZE RDD
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
8F X'29',KEYARG,X'40',K_L SRCHKE
18 X'08',*-8 TIC
16 X'06',IOAREA,X'20',BLKSIZE RDD
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
0A X'29',KEYARG,X'60',K_L SRCHKE
18 X'08',*-8 TIC
16 X'06',IOAREA,X'20',BLKSIZE RDD

CHART J

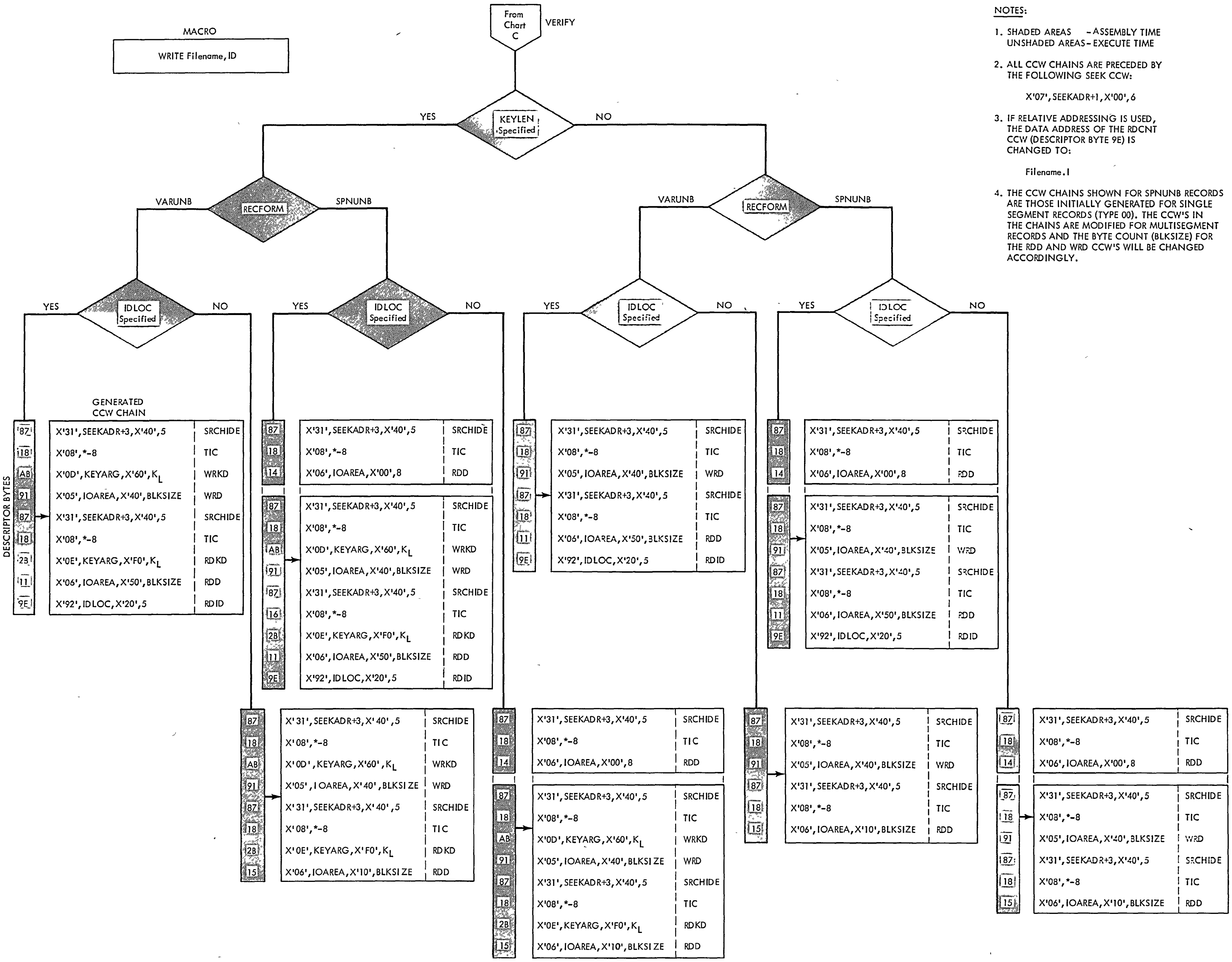
NOTES:

1. SHADED AREAS - ASSEMBLY TIME
   UNSHADED AREAS - EXECUTE TIME
2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW:

   X'07',SEEKADR+1,X'00',6
3. IF RELATIVE ADDRESSING IS USED, THE DATA ADDRESS OF THE SRCHHAE CCW AND THE RDCNT CCW RESPECTIVELY WILL BE CHANGED TO:

   SRCHHAE - Filename.S-3

   RDCNT - Filename.I
4. THE CCW CHAINS SHOWN FOR SPNUNB RECORDS ARE THOSE INITIALLY GENERATED FOR SINGLE SEGMENT RECORDS (TYPE 00). THE CCW'S IN THE CHAINS ARE MODIFIED FOR MULTISEGMENT RECORDS AND THE BYTE COUNT (BLKSIZE) FOR THE RDD AND WRD CCW'S WILL BE CHANGED ACCORDINGLY.


MACRO
WRITE Filename, ID
From Chart C
VERIFY
KEYLEN Specified
YES
NO
RECFORM
VARUNB
SPNUNB
RECFORM
VARUNB
SPNUNB
IDLOC Specified
YES
NO
IDLOC Specified
YES
NO
IDLOC Specified
YES
NO
IDLOC Specified
YES
NO
GENERATED CCW CHAIN
DESCRIPTOR BYTES
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
AB X'0D',KEYARG,X'60',K_L WRKD
91 X'05',IOAREA,X'40',BLKSIZE WRD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
28 X'0E',KEYARG,X'F0',K_L RDKD
11 X'06',IOAREA,X'50',BLKSIZE RDD
9E X'92',IDLOC,X'20',5 RDID
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
AB X'0D',KEYARG,X'60',K_L WRKD
91 X'05',IOAREA,X'40',BLKSIZE WRD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
28 X'0E',KEYARG,X'F0',K_L RDKD
15 X'06',IOAREA,X'10',BLKSIZE RDD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
14 X'06',IOAREA,X'00',8 RDD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
AB X'0D',KEYARG,X'60',K_L WRKD
91 X'05',IOAREA,X'40',BLKSIZE WRD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
16 X'08',*-8 TIC
28 X'0E',KEYARG,X'F0',K_L RDKD
11 X'06',IOAREA,X'50',BLKSIZE RDD
9E X'92',IDLOC,X'20',5 RDID
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
14 X'06',IOAREA,X'00',8 RDD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
AB X'0D',KEYARG,X'60',K_L WRKD
91 X'05',IOAREA,X'40',BLKSIZE WRD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
28 X'0E',KEYARG,X'F0',K_L RDKD
15 X'06',IOAREA,X'10',BLKSIZE RDD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
91 X'05',IOAREA,X'40',BLKSIZE WRD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
11 X'06',IOAREA,X'50',BLKSIZE RDD
9E X'92',IDLOC,X'20',5 RDID
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
91 X'05',IOAREA,X'40',BLKSIZE WRD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
15 X'06',IOAREA,X'10',BLKSIZE RDD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
14 X'06',IOAREA,X'00',8 RDD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
91 X'05',IOAREA,X'40',BLKSIZE WRD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
11 X'06',IOAREA,X'50',BLKSIZE RDD
9E X'92',IDLOC,X'20',5 RDID
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
14 X'06',IOAREA,X'00',8 RDD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
91 X'05',IOAREA,X'40',BLKSIZE WRD
87 X'31',SEEKADR+3,X'40',5 SRCHIDE
18 X'08',*-8 TIC
15 X'06',IOAREA,X'10',BLKSIZE RDD

NOTES:

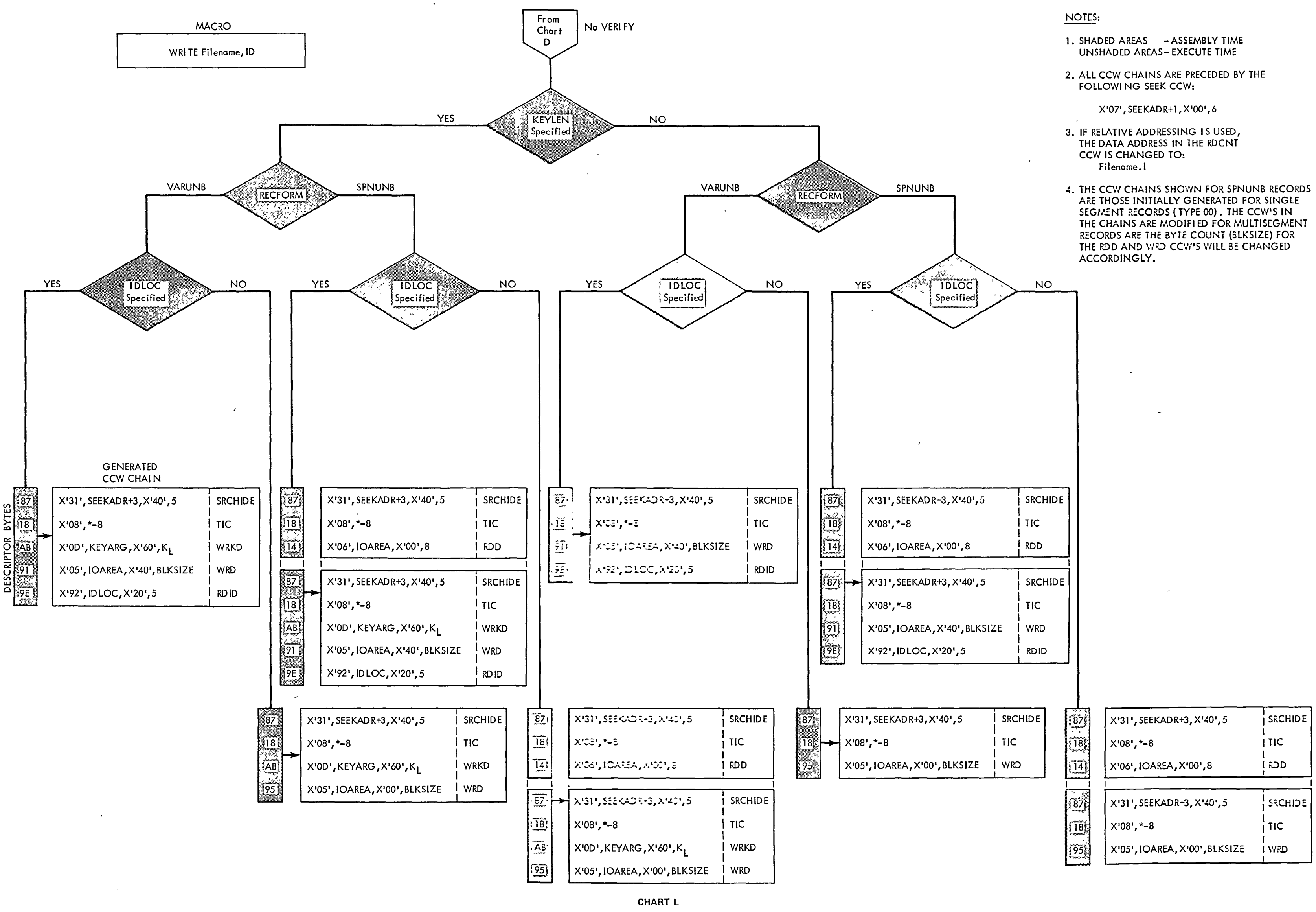
1. SHADED AREAS – ASSEMBLY TIME
   UNSHADED AREAS – EXECUTE TIME
2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW:

   X'07',SEEKADR+1,X'00',6
3. IF RELATIVE ADDRESSING IS USED, THE DATA ADDRESS OF THE RDCNT CCW (DESCRIPTOR BYTE 9E) IS CHANGED TO:

   Filename.I
4. THE CCW CHAINS SHOWN FOR SPNUNB RECORDS ARE THOSE INITIALLY GENERATED FOR SINGLE SEGMENT RECORDS (TYPE 00). THE CCW'S IN THE CHAINS ARE MODIFIED FOR MULTISEGMENT RECORDS AND THE BYTE COUNT (BLKSIZE) FOR THE RDD AND WRD CCW'S WILL BE CHANGED ACCORDINGLY.

CHART K

Figure 36. DA Channel Programs (Part 10 of 14)



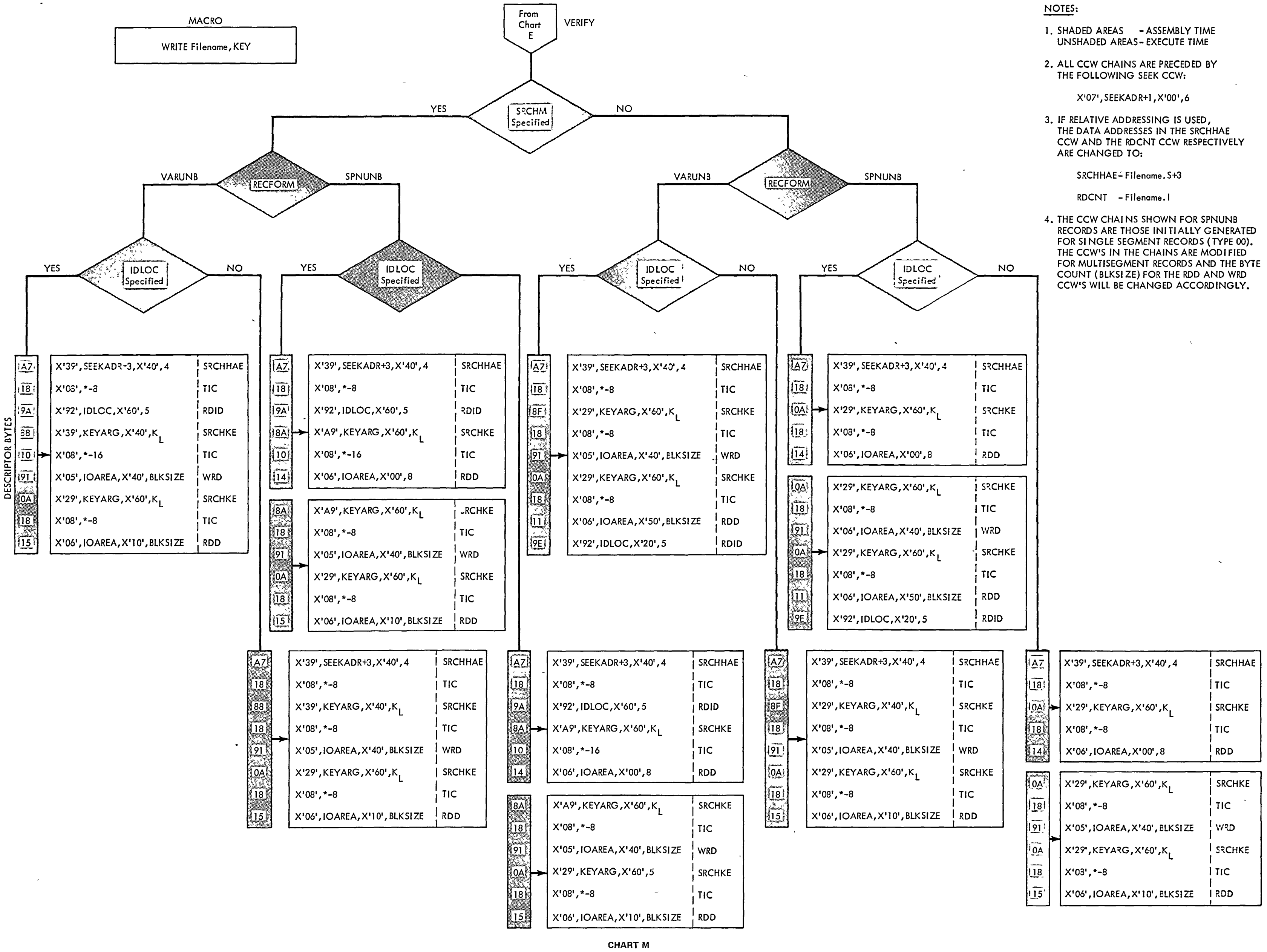
MACRO
WRITE Filename, ID
From Chart D
No VERIFY
KEYLEN Specified
YES
NO
RECFORM
VARUNB
SPNUNB
IDLOC Specified
YES
NO
GENERATED CCW CHAIN
DESCRIPTOR BYTES
87 18 AB 91 9E
X'31',SEEKADR+3,X'40',5 SRCHIDE
X'08',*-8 TIC
X'0D',KEYARG,X'60',K$_L$ WRKD
X'05',IOAREA,X'40',BLKSIZE WRD
X'92',IDLOC,X'20',5 RDID
87 18 14
X'31',SEEKADR+3,X'40',5 SRCHIDE
X'08',*-8 TIC
X'06',IOAREA,X'00',8 RDD
87 18 AB 91 9E
X'31',SEEKADR+3,X'40',5 SRCHIDE
X'08',*-8 TIC
X'0D',KEYARG,X'60',K$_L$ WRKD
X'05',IOAREA,X'40',BLKSIZE WRD
X'92',IDLOC,X'20',5 RDID
87 18 AB 95
X'31',SEEKADR+3,X'40',5 SRCHIDE
X'08',*-8 TIC
X'0D',KEYARG,X'60',K$_L$ WRKD
X'05',IOAREA,X'00',BLKSIZE WRD
87 18 91 9E
X'31',SEEKADR-3,X'40',5 SRCHIDE
X'08',*-8 TIC
X'05',IOAREA,X'40',BLKSIZE WRD
X'92',IDLOC,X'20',5 RDID
87 18 14
X'31',SEEKADR-3,X'40',5 SRCHIDE
X'08',*-8 TIC
X'06',IOAREA,X'00',8 RDD
87 18 AB 95
X'31',SEEKADR-3,X'40',5 SRCHIDE
X'08',*-8 TIC
X'0D',KEYARG,X'60',K$_L$ WRKD
X'05',IOAREA,X'00',BLKSIZE WRD
87 18 14
X'31',SEEKADR+3,X'40',5 SRCHIDE
X'08',*-8 TIC
X'06',IOAREA,X'00',8 RDD
87 18 91 9E
X'31',SEEKADR+3,X'40',5 SRCHIDE
X'08',*-8 TIC
X'05',IOAREA,X'40',BLKSIZE WRD
X'92',IDLOC,X'20',5 RDID
87 18 95
X'31',SEEKADR+3,X'40',5 SRCHIDE
X'08',*-8 TIC
X'05',IOAREA,X'00',BLKSIZE WRD
87 18 14
X'31',SEEKADR+3,X'40',5 SRCHIDE
X'08',*-8 TIC
X'06',IOAREA,X'00',8 RDD
87 18 95
X'31',SEEKADR-3,X'40',5 SRCHIDE
X'08',*-8 TIC
X'05',IOAREA,X'00',BLKSIZE WRD
CHART L

NOTES:

1. SHADED AREAS – ASSEMBLY TIME
   UNSHADED AREAS – EXECUTE TIME
2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW:

   X'07',SEEKADR+1,X'00',6
3. IF RELATIVE ADDRESSING IS USED, THE DATA ADDRESS IN THE RDCNT CCW IS CHANGED TO:
   Filename.I
4. THE CCW CHAINS SHOWN FOR SPNUNB RECORDS ARE THOSE INITIALLY GENERATED FOR SINGLE SEGMENT RECORDS (TYPE 00). THE CCW'S IN THE CHAINS ARE MODIFIED FOR MULTISEGMENT RECORDS ARE THE BYTE COUNT (BLKSIZE) FOR THE RDD AND WRD CCW'S WILL BE CHANGED ACCORDINGLY.

Figure 36. DA Channel Programs (Part 11 of 14)


MACRO
WRITE Filename,KEY
From Chart E
VERIFY
SRCHM Specified
YES
NO
RECFORM
VARUNB
SPNUNB
RECFORM
VARUNB
SPNUNB
IDLOC Specified
YES
NO
DESCRIPTOR BYTES
A7 X'39',SEEKADR-3,X'40',4 SRCHHAE
18 X'03',*-8 TIC
9A X'92',IDLCC,X'60',5 RDID
38 X'39',KEYARG,X'40',$K_L$ SRCHKE
10 X'08',*-16 TIC
91 X'05',IOAREA,X'40',BLKSIZE WRD
0A X'29',KEYARG,X'60',$K_L$ SRCHKE
18 X'08',*-8 TIC
15 X'06',IOAREA,X'10',BLKSIZE RDD
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
88 X'39',KEYARG,X'40',$K_L$ SRCHKE
18 X'08',*-8 TIC
91 X'05',IOAREA,X'40',BLKSIZE WRD
0A X'29',KEYARG,X'60',$K_L$ SRCHKE
18 X'08',*-8 TIC
15 X'06',IOAREA,X'10',BLKSIZE RDD
IDLOC Specified
YES
NO
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
9A X'92',IDLOC,X'60',5 RDID
8A X'A9',KEYARG,X'60',$K_L$ SRCHKE
10 X'08',*-16 TIC
14 X'06',IOAREA,X'00',8 RDD
8A X'A9',KEYARG,X'60',$K_L$ SRCHKE
18 X'08',*-8 TIC
91 X'05',IOAREA,X'40',BLKSIZE WRD
0A X'29',KEYARG,X'60',$K_L$ SRCHKE
18 X'08',*-8 TIC
15 X'06',IOAREA,X'10',BLKSIZE RDD
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
9A X'92',IDLOC,X'60',5 RDID
8A X'A9',KEYARG,X'60',$K_L$ SRCHKE
10 X'08',*-16 TIC
14 X'06',IOAREA,X'00',8 RDD
8A X'A9',KEYARG,X'60',$K_L$ SRCHKE
18 X'08',*-8 TIC
91 X'05',IOAREA,X'40',BLKSIZE WRD
0A X'29',KEYARG,X'60',5 SRCHKE
18 X'08',*-8 TIC
15 X'06',IOAREA,X'10',BLKSIZE RDD
IDLOC Specified
YES
NO
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
8F X'29',KEYARG,X'60',$K_L$ SRCHKE
18 X'08',*-8 TIC
91 X'05',IOAREA,X'40',BLKSIZE WRD
0A X'29',KEYARG,X'60',$K_L$ SRCHKE
18 X'08',*-8 TIC
11 X'06',IOAREA,X'50',BLKSIZE RDD
9E X'92',IDLOC,X'20',5 RDID
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
8F X'29',KEYARG,X'40',$K_L$ SRCHKE
18 X'08',*-8 TIC
91 X'05',IOAREA,X'40',BLKSIZE WRD
0A X'29',KEYARG,X'60',$K_L$ SRCHKE
18 X'08',*-8 TIC
15 X'06',IOAREA,X'10',BLKSIZE RDD
IDLOC Specified
YES
NO
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
0A X'29',KEYARG,X'60',$K_L$ SRCHKE
18 X'03',*-8 TIC
14 X'06',IOAREA,X'00',8 RDD
0A X'29',KEYARG,X'60',$K_L$ SRCHKE
18 X'08',*-8 TIC
91 X'06',IOAREA,X'40',BLKSIZE WRD
0A X'29',KEYARG,X'60',$K_L$ SRCHKE
18 X'08',*-8 TIC
11 X'06',IOAREA,X'50',BLKSIZE RDD
9E X'92',IDLOC,X'20',5 RDID
A7 X'39',SEEKADR+3,X'40',4 SRCHHAE
18 X'08',*-8 TIC
0A X'29',KEYARG,X'60',$K_L$ SRCHKE
18 X'08',*-8 TIC
14 X'06',IOAREA,X'00',8 RDD
0A X'29',KEYARG,X'60',$K_L$ SRCHKE
18 X'08',*-8 TIC
91 X'05',IOAREA,X'40',BLKSIZE WRD
0A X'29',KEYARG,X'60',$K_L$ SRCHKE
18 X'03',*-8 TIC
15 X'06',IOAREA,X'10',BLKSIZE RDD
CHART M

NOTES:

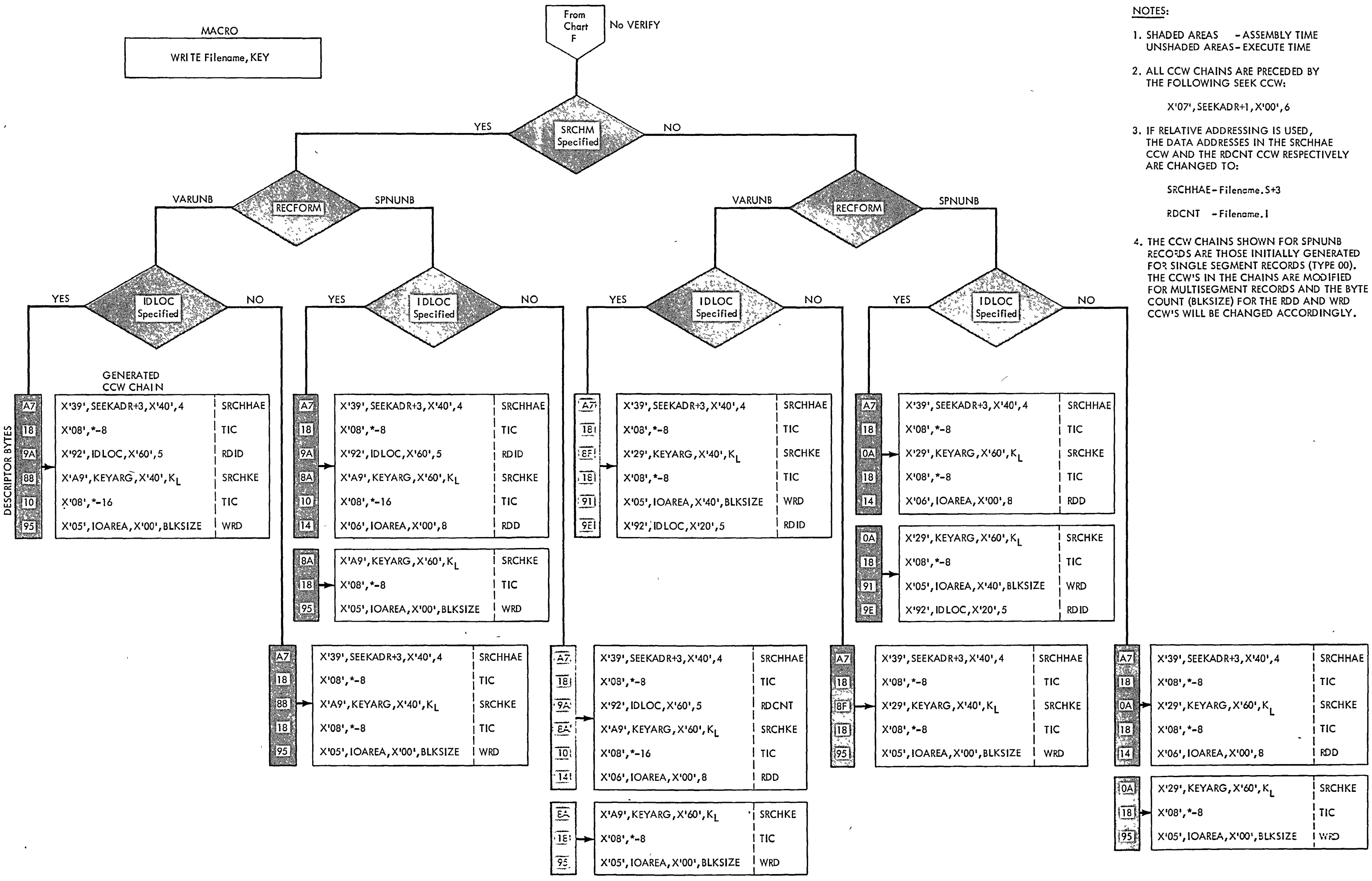
1. SHADED AREAS - ASSEMBLY TIME
   UNSHADED AREAS - EXECUTE TIME
2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW:

   X'07',SEEKADR+1,X'00',6
3. IF RELATIVE ADDRESSING IS USED, THE DATA ADDRESSES IN THE SRCHHAE CCW AND THE RDCNT CCW RESPECTIVELY ARE CHANGED TO:

   SRCHHAE - Filename.S+3

   RDCNT - Filename.I
4. THE CCW CHAINS SHOWN FOR SPNUNB RECORDS ARE THOSE INITIALLY GENERATED FOR SINGLE SEGMENT RECORDS (TYPE 00). THE CCW'S IN THE CHAINS ARE MODIFIED FOR MULTISEGMENT RECORDS AND THE BYTE COUNT (BLKSIZE) FOR THE RDD AND WRD CCW'S WILL BE CHANGED ACCORDINGLY.

Figure 36. DA Channel Programs (Part 12 of 14)



```
A7  X'39',SEEKADR+3,X'40',4      SRCHHAE
18  X'08',*-8                    TIC
9A  X'92',IDLOC,X'60',5          RDID
88  X'A9',KEYARG,X'40',KL        SRCHKE
10  X'08',*-16                   TIC
95  X'05',IOAREA,X'00',BLKSIZE   WRD
```

```
A7  X'39',SEEKADR+3,X'40',4      SRCHHAE
18  X'08',*-8                    TIC
88  X'A9',KEYARG,X'40',KL        SRCHKE
18  X'08',*-8                    TIC
95  X'05',IOAREA,X'00',BLKSIZE   WRD
```

```
A7  X'39',SEEKADR+3,X'40',4      SRCHHAE
18  X'08',*-8                    TIC
9A  X'92',IDLOC,X'60',5          RDID
8A  X'A9',KEYARG,X'60',KL        SRCHKE
10  X'08',*-16                   TIC
14  X'06',IOAREA,X'00',8         RDD

8A  X'A9',KEYARG,X'60',KL        SRCHKE
18  X'08',*-8                    TIC
95  X'05',IOAREA,X'00',BLKSIZE   WRD
```

```
A7  X'39',SEEKADR+3,X'40',4      SRCHHAE
18  X'08',*-8                    TIC
9A  X'92',IDLOC,X'60',5          RDCNT
8A  X'A9',KEYARG,X'60',KL        SRCHKE
10  X'08',*-16                   TIC
14  X'06',IOAREA,X'00',8         RDD

8A  X'A9',KEYARG,X'60',KL        SRCHKE
18  X'08',*-8                    TIC
95  X'05',IOAREA,X'00',BLKSIZE   WRD
```

```
A7  X'39',SEEKADR+3,X'40',4      SRCHHAE
18  X'08',*-8                    TIC
8F  X'29',KEYARG,X'40',KL        SRCHKE
18  X'08',*-8                    TIC
91  X'05',IOAREA,X'40',BLKSIZE   WRD
9E  X'92',IDLOC,X'20',5          RDID
```

```
A7  X'39',SEEKADR+3,X'40',4      SRCHHAE
18  X'08',*-8                    TIC
8F  X'29',KEYARG,X'40',KL        SRCHKE
18  X'08',*-8                    TIC
95  X'05',IOAREA,X'00',BLKSIZE   WRD
```

```
A7  X'39',SEEKADR+3,X'40',4      SRCHHAE
18  X'08',*-8                    TIC
0A  X'29',KEYARG,X'60',KL        SRCHKE
18  X'08',*-8                    TIC
14  X'06',IOAREA,X'00',8         RDD

0A  X'29',KEYARG,X'60',KL        SRCHKE
18  X'08',*-8                    TIC
91  X'05',IOAREA,X'40',BLKSIZE   WRD
9E  X'92',IDLOC,X'20',5          RDID
```

```
A7  X'39',SEEKADR+3,X'40',4      SRCHHAE
18  X'08',*-8                    TIC
0A  X'29',KEYARG,X'60',KL        SRCHKE
18  X'08',*-8                    TIC
14  X'06',IOAREA,X'00',8         RDD

0A  X'29',KEYARG,X'60',KL        SRCHKE
18  X'08',*-8                    TIC
95  X'05',IOAREA,X'00',BLKSIZE   WRD
```

NOTES:

1. SHADED AREAS - ASSEMBLY TIME
   UNSHADED AREAS - EXECUTE TIME
2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW:

   X'07',SEEKADR+1,X'00',6

3. IF RELATIVE ADDRESSING IS USED, THE DATA ADDRESSES IN THE SRCHHAE CCW AND THE RDCNT CCW RESPECTIVELY ARE CHANGED TO:

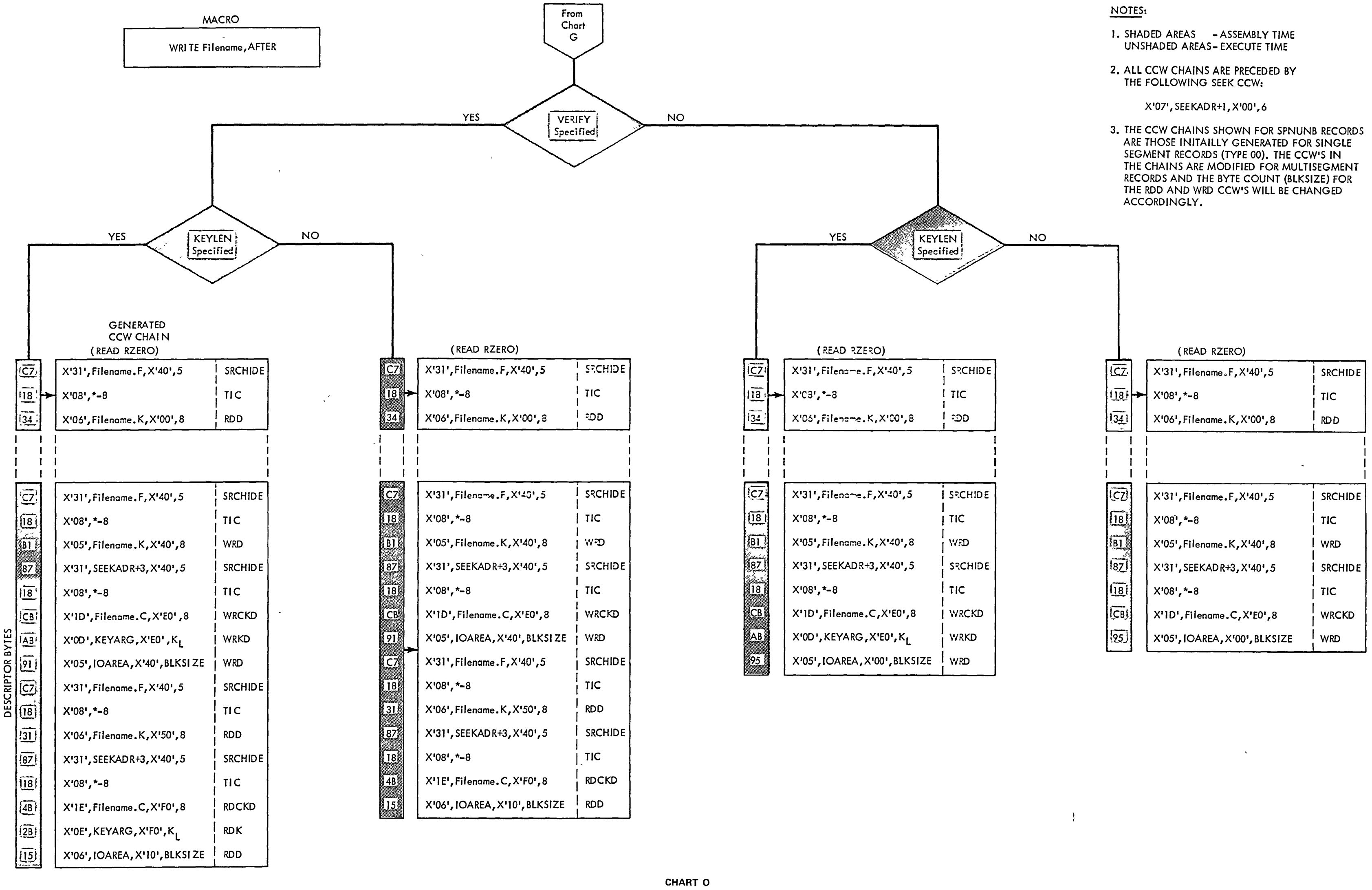
   SRCHHAE - Filename.S+3

   RDCNT - Filename.I

4. THE CCW CHAINS SHOWN FOR SPNUNB RECORDS ARE THOSE INITIALLY GENERATED FOR SINGLE SEGMENT RECORDS (TYPE 00). THE CCW'S IN THE CHAINS ARE MODIFIED FOR MULTISEGMENT RECORDS AND THE BYTE COUNT (BLKSIZE) FOR THE RDD AND WRD CCW'S WILL BE CHANGED ACCORDINGLY.

CHART N

Figure 36. DA Channel Programs (Part 13 of 14)



VERIFY specified, KEYLEN specified — GENERATED CCW CHAIN (READ RZERO)

| | | |
|---|---|---|
| C7 | X'31',Filename.F,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 34 | X'06',Filename.K,X'00',8 | RDD |
| C7 | X'31',Filename.F,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| B1 | X'05',Filename.K,X'40',8 | WRD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| CB | X'1D',Filename.C,X'E0',8 | WRCKD |
| AB | X'0D',KEYARG,X'E0',$K_L$ | WRKD |
| 91 | X'05',IOAREA,X'40',BLKSIZE | WRD |
| C7 | X'31',Filename.F,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 31 | X'06',Filename.K,X'50',8 | RDD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 4B | X'1E',Filename.C,X'F0',8 | RDCKD |
| 2B | X'0E',KEYARG,X'F0',$K_L$ | RDK |
| 15 | X'06',IOAREA,X'10',BLKSIZE | RDD |

VERIFY specified, KEYLEN not specified — (READ RZERO)

| | | |
|---|---|---|
| C7 | X'31',Filename.F,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 34 | X'06',Filename.K,X'00',8 | RDD |
| C7 | X'31',Filename.F,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| B1 | X'05',Filename.K,X'40',8 | WRD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| CB | X'1D',Filename.C,X'E0',8 | WRCKD |
| 91 | X'05',IOAREA,X'40',BLKSIZE | WRD |
| C7 | X'31',Filename.F,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 31 | X'06',Filename.K,X'50',8 | RDD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 4B | X'1E',Filename.C,X'F0',8 | RDCKD |
| 15 | X'06',IOAREA,X'10',BLKSIZE | RDD |

VERIFY not specified, KEYLEN specified — (READ RZERO)

| | | |
|---|---|---|
| C7 | X'31',Filename.F,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 34 | X'06',Filename.K,X'00',8 | RDD |
| C7 | X'31',Filename.F,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| B1 | X'05',Filename.K,X'40',8 | WRD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| CB | X'1D',Filename.C,X'E0',8 | WRCKD |
| AB | X'0D',KEYARG,X'E0',$K_L$ | WRKD |
| 95 | X'05',IOAREA,X'00',BLKSIZE | WRD |

VERIFY not specified, KEYLEN not specified — (READ RZERO)

| | | |
|---|---|---|
| C7 | X'31',Filename.F,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| 34 | X'06',Filename.K,X'00',8 | RDD |
| C7 | X'31',Filename.F,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| B1 | X'05',Filename.K,X'40',8 | WRD |
| 87 | X'31',SEEKADR+3,X'40',5 | SRCHIDE |
| 18 | X'08',*-8 | TIC |
| CB | X'1D',Filename.C,X'E0',8 | WRCKD |
| 95 | X'05',IOAREA,X'00',BLKSIZE | WRD |

NOTES:

1. SHADED AREAS - ASSEMBLY TIME
   UNSHADED AREAS - EXECUTE TIME
2. ALL CCW CHAINS ARE PRECEDED BY THE FOLLOWING SEEK CCW:

   X'07',SEEKADR+1,X'00',6
3. THE CCW CHAINS SHOWN FOR SPNUNB RECORDS ARE THOSE INITAILLY GENERATED FOR SINGLE SEGMENT RECORDS (TYPE 00). THE CCW'S IN THE CHAINS ARE MODIFIED FOR MULTISEGMENT RECORDS AND THE BYTE COUNT (BLKSIZE) FOR THE RDD AND WRD CCW'S WILL BE CHANGED ACCORDINGLY.

CHART O

Figure 36. DA Channel Programs (Part 14 of 14)

## INITIALIZATION AND TERMINATION

When a DASD file is processed by the Direct Access Method, all extents specified by the user must be opened before any data is transferred.

The DA Open logical transients make all the extents for the file available for use by the problem program. To accomplish this, the open routines check and create standard DASD labels or, in the case of nonstandard labels, pass control to the user for label processing.

To open a file, the open routines use label information supplied by the user in job control statements and stored on the SYSRES label information cylinder (refer to Figure 2). This information is used either to check or create the actual file labels in the Volume Table of Contents (VTOC) of the pack or cell containing the file. Refer to DASD Label Processing for details of SYSRES label information and to Appendix G for details and format of the standard DASD file labels processed by DOS logical IOCS.

Close is required for DASD files processed by the Direct Access Method only when user standard trailer labels are specified.

## OPEN DIRECT ACCESS CHART 07

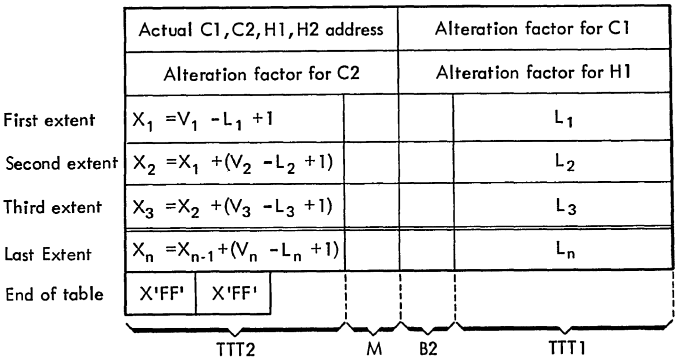
For input files, the volume and Format 1 labels are checked against the SYSRES label information supplied by the user's // DLBL job control card. User labels are then processed, providing LABADDR=address has been specified in the DTFDA macro defining the file. Finally, EXTENT information is passed to the user for checking and/or processing.

For output files, extents are checked to ensure that they do not overlap the VTOC or other extents. Labels are created and written in the VTOC, and user labels are processed, if required.

### Relative Addressing

When relative addressing is specified for a file, the open routines convert extent information supplied as actual physical DASD addresses into a relative addressing format. The converted extent information is stored at the end of the DTF table, in a table (DSKXTNT) at location &Filename.P+48.

The 12 bytes preceding the DSKXTNT table contain device-dependent alteration factors (4 bytes each) used to convert the extent limit addresses. The format of the DSKXTNT table and the location of the alteration factors is illustrated in Figure 37. The alteration factors are summarized in Figure 38.



| | | | | | |
|---|---|---|---|---|---|
| | Actual C1,C2,H1,H2 address | | Alteration factor for C1 | | |
| | Alteration factor for C2 | | Alteration factor for H1 | | |
| First extent | $X_1 = V_1 - L_1 + 1$ | | | | $L_1$ |
| Second extent | $X_2 = X_1 + (V_2 - L_2 + 1)$ | | | | $L_2$ |
| Third extent | $X_3 = X_2 + (V_3 - L_3 + 1)$ | | | | $L_3$ |
| Last Extent | $X_n = X_{n-1} + (V_n - L_n + 1)$ | | | | $L_n$ |
| End of table | X'FF' X'FF' | | | | |

TTT1, L = relative track number of the extent lower limit; that is, the number of tracks from cylinder 0, track 0 to the lower limit of the corresponding extent. (3 bytes)

TTT2 = cumulative total tracks in current extent plus previous extents in the table. (3 bytes)

B2 = second byte of bin number (BB), 0 for 2311, 2314, or 2319. (1 byte)

M = symbolic unit number, incremented by 1 for each new symbolic unit. (1 byte)

V = number of tracks from cylinder 0, track 0 to the upper limit of corresponding extent.

Figure 37. DSKXTNT Table for Relative Addressing

| Factor | 2311 | 2314* | 2321 |
|---|---|---|---|
| C1 | 1 | 1 | 1000 |
| C2 | 10 | 20 | 100 |
| H1 | 1 | 1 | 20 |

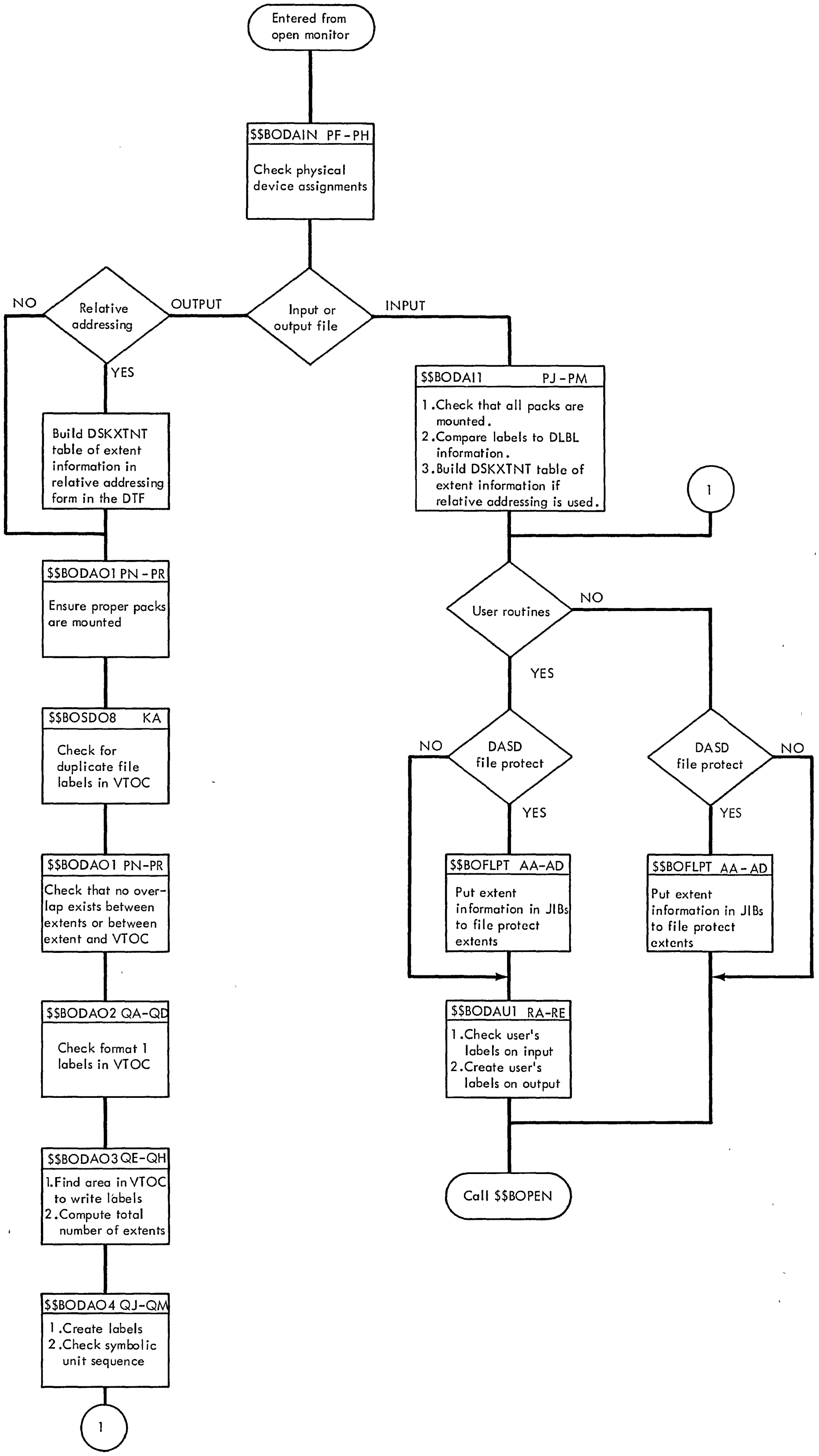
* Also for 2319.

Figure 38. Alteration Factors for Relative Addressing

An actual physical extent address is converted to a relative address in the following manner. Each of the four bytes (CCHH) of the actual address are handled separately and are referred to as C1, C2, H1, and H2. Starting with C1, the first three bytes of the actual address are multiplied, one at a time, by the respective device-dependent alteration factor (refer to Figure 38). The result of each multiply operation is added into an accumulating register. To complete the conversion, H2 is added to the accumulated result. If the conversion is performed on the lower limit address of the extent, the value obtained is the L (or TTT1) value and is stored in the DSKXTNT table (refer to Figure 37).

If the conversion is performed for the upper limit address of the extent, the converted value is increased by 1 and TTT1 is subtracted from the result. The value obtained from this calculation is the total number of tracks included in the extent. The total number of tracks in the extent is then added to the total number of tracks of all previous entries to obtain the TTT2 value for the current extent entry in the DSKXTNT table (refer to Figure 37).

## Chart 07. Open Direct Access, General Logic


Entered from open monitor
$$BODAIN PF-PH
Check physical device assignments
Input or output file
OUTPUT
INPUT
Relative addressing
NO
YES
Build DSKXTNT table of extent information in relative addressing form in the DTF
$$BODAO1 PN-PR
Ensure proper packs are mounted
$$BOSDO8 KA
Check for duplicate file labels in VTOC
$$BODAO1 PN-PR
Check that no overlap exists between extents or between extent and VTOC
$$BODAO2 QA-QD
Check format 1 labels in VTOC
$$BODAO3 QE-QH
1.Find area in VTOC to write labels
2.Compute total number of extents
$$BODAO4 QJ-QM
1.Create labels
2.Check symbolic unit sequence
1
$$BODAI1 PJ-PM
1.Check that all packs are mounted.
2.Compare labels to DLBL information.
3.Build DSKXTNT table of extent information if relative addressing is used.
1
User routines
NO
YES
DASD file protect
NO
YES
DASD file protect
NO
YES
$$BOFLPT AA-AD
Put extent information in JIBs to file protect extents
$$BOFLPT AA-AD
Put extent information in JIBs to file protect extents
$$BODAU1 RA-RE
1.Check user's labels on input
2.Create user's labels on output
Call $$BOPEN

$$BODAIN: DA Open Input/Output Charts PF-PH

Objective: To ensure that the correct device is assigned for each extent, and to build a table (DSKXTNT) of extent values if relative addressing is used.

Entry: From the Open Monitor, $$BOPEN.

Exits:

- To the Open Monitor, $$BOPEN, if the COBOL Open/Ignore option is specified.

- To $$BODAI1 if an input file.

- To $$BODAO1 if an output file.

- To $$BOMSG1 if messages are needed.

Method: This phase determines if the correct device has been assigned to the file. If the correct device has not been assigned, the message writer phase, $$BOMSG1, is fetched to print message 4683I on SYSLOG. If the correct device is assigned, a test is made for relative addressing.

If the file is an input file with relative addressing, the proper alteration factors for the device are inserted in the DTF table and phase $$BODAI1 is fetched. If the file is an input file with actual addressing, phase $$BODAI1 is fetched directly.

For output files with relative addressing, phase $$BODAIN, in addition to storing the proper alteration factors, converts the actual extent limits of each extent to a relative address. The relative address form of each extent is stored in sequence in the DTF table. (This extent information in relative addressing form is referred to as the DSKXTNT table and is located in the DTF table at location &Filename.P+48.) When the extent information is complete, phase $$BODAO1 is fetched.

For output files with actual addressing, phase $$BODAO1 is fetched after the device assignment of each extent is checked.

$$BODAI1: DA Open Input Charts PJ-PM

Objective: To ensure that all packs are mounted, that the symbolic units are specified in sequence, and that the symbolic unit (and bin, if 2321) agrees with the user supplied DLBL information.

Entry: From $$BODAIN, or from a message writer phase.

Exits:

- To $$BODSMW to write data security message.

- To the Open Monitor, $$BOPEN, if no user options are specified.

- To $$BODAU1 to process user's options.

- To $$BOFLPT if DASD file protect is specified.

- To $$BOMSG1 if a message is required.

Method: Phase $$BODAI1 reads the VOL1 label and checks the volume serial number in the label against the volume serial number given in the extent to ensure that the correct pack is mounted. The phase then reads the Format 1 label for the file and checks the symbolic unit specified for each extent for proper sequence.

If the data security indicator in the format 1 label is ON, and the data security message has not been issued for the file, exit is made to $$BODSMW to print the data security message.

If relative addressing is specified, phase $$BODAI1 converts every set of extent limits in the Format 1 label, and the Format 3 label(s) as required, from actual (CCHH) addresses to a relative address form. The extent information in relative form is placed in a table (DSKXTNT) within the DTF table at location &Filename.P+48.

When all extents have been checked, a test is made for file protected extents. If file protect is specified, phase $$BOFLPT is fetched to build JIBs for the protected extents.

Phase $$BODAI1 returns control either to the Open Monitor ($$BOPEN), or to phase $$BODAU1 if processing of user labels or extent information is required.

$$BODAO1: DA Open Output, Phase 1 Charts PN-PR

Objective: To check the volume serial number and determine if the DLBL and EXTENT serial numbers are equal, and to ensure that no extent specified by the user overlaps on the VTOC or on another extent for the same file.

Entry: From $$BODAIN.

Exits:

- To $$BODAO2 if job goes to normal completion.
- To $$BOSDO8 to check for duplicate DLBL in VTOC.
- To $$BOMSG1 if messages are to be put out.

Method: This phase reads the volume label for the symbolic unit (and bin, if the device is a 2321). It checks to see that the correct volume is available by comparing the volume serial number with the serial number specified on the first extent. If they are not equal, message 4755A is initialized and $$BOMSG1 is fetched to write the message on SYSLOG.

If the serial numbers are equal, the Format 4 label is read. Each extent for the current symbolic unit is checked to see that it does not overlap on the VTOC. If an overlap is present, message 4741A is initialized and written out on SYSLOG by $$BOMSG1. Upon reentry to this phase, the current extent is deleted at the user's request.

The next extent is checked to see if it has the same symbolic unit as the previous extent. If it does not, the routine exits to $$BOSDO8 to check for a duplicate DLBL in the VTOC. If the next extent has the same symbolic unit, the phase checks to see if the first extent has two tracks. If it does not, message 4766A is initialized and written on SYSLOG by $$BOMSG1. Upon reentry to this phase, the first extent is deleted at the user's request.

If the first extent has two tracks, a check determines if extents overlap. If an overlap exists, message 4740A is initialized and written on SYSLOG. Upon reentry to this phase, the extent causing the overlap is deleted at the user's request. All remaining extents are moved down the length of one extent so that the upper limit of the extent is decreased by the length of one extent.

When all extents on the file have been checked in this manner, this phase exits to $$BODAO2.

## $$BODAO2: DA Open Output, Phase 2 Charts QA-QD

Objective: To ensure that no extent for the incoming file overlaps on any extent for any file cataloged in the VTOC.

Entry: From $$BODAO1.

Exits:

- To $$BODAO3 to create labels.
- To $$BOMSG1 if messages are to be printed.

Method: This phase reads the VOL1 label to get the starting address (CCHHR) of the VTOC. If the VOL1 label is not found, message 4706I is initialized, and phase $$BOMSG1 is fetched to write the message on SYSLOG.

The Format 4 label, or VTOC definition label, is read to get the upper limit address (CCHH) of the VTOC. If the Format 4 label is not found, message 4704I is initialized, and phase $$BOMSG1 is fetched to write the message.

If the Format 4 label is found, the routine reads DASD labels (count, key, and data) from the VTOC, one at a time, until a Format 1 label is found or the last record in the VTOC is read. If a no-record-found condition results before the end of the VTOC is reached, message 4709I is initialized and written on SYSLOG by phase $$BOMSG1.

If the end of the VTOC is reached without finding a Format 1 label, the routine determines the address of the extent for the next file on a different symbolic unit, and returns to process the extents for that file. If an extent on a different symbolic unit cannot be found, and all extents have been scanned, this phase fetches phase $$BODAO3 to create labels.

When a Format 1 label is found, the extents for the new file are checked for overlap on the extents in the Format 1 label and any other labels chained to it. If overlap is found, the Format 1 label is reread, and the expiration date is checked to see if the file has expired. If the file has not expired, message 4744A is initialized and phase $$BOMSG1 is fetched to write the message on SYSLOG. At this point, the operator sets indicators either to delete the file, or delete the extent. Upon reentry to this phase, the indicators determine the action to be taken, and that action is executed.

If the file has expired, or the operator requests that the file be deleted, it is done by writing zeros over all the labels in the chain. When the file is deleted, the phase returns to read the next Format 1 label from the VTOC and continue processing.

When overlap occurs on an unexpired data secured file, $$BOMSG1 is fetched to issue message 4798I OVLAP UNEXPRD SECRD FILE. If overlap on an expired secured file occurs, message 4797I OVLAP EXPIRED SECRD FILE is issued. In both cases, the job is canceled.

When all the extents for the incoming file have been processed, this phase exits to $$BODAO3 to create labels.

## $$BODAO3: DA Open Output, Phase 3 Charts QE-QH

Objective: To find an open area in the VTOC in which to write labels. To set the DADSM bit on in the Format 4 label and compute the total number of extents.

Entry: From $$BODAO2.

Exits:

- To $$BODAO4 to create Format 1 and Format 3 labels.
- To $$BOMSG1 if messages are to be printed.

Method: This phase first counts the number of extents on the symbolic unit and saves the total number to be inserted in the label later. A test determines if the total number exceeded the limit. If it did, an exit is made to phase $$BOMSG1 to issue a message to that effect.

If the limit has not been exceeded, the VOL1 label is read to get the starting address of the VTOC. The Format 4 label, or VTOC definition label, is then read and the DADSM bit is set on in the Format 4 label. This phase then writes and verifies the Format 4 label.

The VTOC is then searched to find the first open area in which to write a label. This first open area contains the Format 1 label, or in case of more than three extents, the Format 3 label. If a Format 3 label is to be created, the VTOC is searched again to find a second open area for the Format 1 label.

When the necessary number of open areas have been found, this phase exits to phase $$BODAO4 to create the labels.

## $$BODAO4: DA Open Output, Phase 4 Charts QJ-QM

Objective: To create Format 1 and Format 3 labels, and write them in the VTOC. To ensure that symbolic units are in sequence.

Entry: From $$BODAO3.

Exits:

- To $$BODAO3 if additional extents are to be processed.
- To $$BOPEN if there are no user routines.
- To $$BODAU1 if there are user labels.
- To $$BOMSG1 if messages are to be put out.
- To $$BOFLPT if DASD file-protect is present.

Method: This phase tests for user labels. If there are any, a user label extent is created from the first prime data extent for the symbolic unit, and the first track becomes the user label track. If the limits in the prime data extent specify more than one track, the lower limit is updated by one track to form a new lower limit for the prime data extent.

Fields specified in the DLBL are used to create the Format 1 label. The extents are then inserted into the Format 1 label from the extent storage area. If more than three extents on the same symbolic unit are specified, the Format 1 label is written out and the Format 3 label is created. The remaining extents are inserted into the Format 3 label.

When a different symbolic unit or the end of the extent storage area is detected, the label currently being created is written out. If the last symbolic unit has not been processed, the DLBL volume sequence number is updated by 1. The next symbolic unit is processed. If the symbolic unit has changed, the new symbolic unit must be in sequence. If not, phase $$BOMSG1 is fetched to issue a message to that effect.

To process the next symbolic unit, this phase exits to phase $$BODAO3 to find another open area for the next label.

## $$BODAU1: DA Open Input, Output Charts RA-RE

Objective: To check user header labels for an input file if the user elects the user label option. To create user header labels for an output file, to be written by the open routine on the 2311, 2314, 2319, or 2321. To pass extents to the user from the labels if requested to do so by the user.

Entry:

- From $$BODAI1 if an input file.
- From $$BODA04 if an output file.

Exits:

- To $$BOMSG1 if messages are to be printed.
- To $$BOPEN at completion of OPEN routine.

Method: The VOL1 label is read to get the starting address of the VTOC. The VTOC is then searched to find the correct Format 1 label. If the label cannot be found, a message is written by phase $$BOMSG1 to that effect.

When the correct label has been found, a test is made for user labels. If there are none, the routine branches to pass extents if requested by the user.

Then a test determines if the current label is a user label. If it is not, the routine branches to pass extents to the user. If the current label is a user label, a test determines if the device is a 2321. If it is, the maximum number of header labels for input or output files must be modified because only five user labels are possible with a 2321 file. Eight user labels may be used with a 2311, 2314, or 2319 file.

When the maximum constants have been modified (if necessary), this phase determines whether the file is an input or output file. If the file is an input file, the user header labels are read by this phase and checked to ensure that they are correct user header labels. The data portion is then passed to the user if the labels are correct. If the labels are not correct, a message is written by $$BOMSG1 to that effect. The reading of labels continues until the maximum number of labels have been read (and rewritten if the user has updated the label), or the user signals that he does not want to read any more labels.

If the file is an output file, this phase creates the count field, including the ID, key length, and data length and the key field, (UHL1 for the first user label to UHL5 or UHL8 for the maximum number of user labels). It also initializes the first four bytes of the data field which correspond to the key field. This phase then exits to the user in order to create the last 76 bytes of the data field. Upon return from the user, the label is written by this phase.

The creating and writing of labels continues until the maximum number of labels has been written or the user signals that he does not want to create any more labels. When the last label has been written, two file marks are created and written, the first of which is a header label with a data length of zero, and the second of which is a trailer label with a key of UTL0 and a data length of zero. When these file marks have been written, the routine branches to pass extents.

A test determines whether the user wants extents passed. If not, this phase proceeds to process the next symbolic unit. If the user does want extents passed, the extent information is passed via a fourteen byte area illustrated by Figure 39.

| | |
|---|---|
| Byte 0 | Extent Type |
| Byte 1 | Extent Sequence Number |
| Bytes 2 to 5 | Extent Lower Limit |
| Bytes 6 to 9 | Extent Upper Limit |
| Bytes 10, 11 | Symbolic Unit |
| Byte 12 | Original Bin Number<br>(0 for 2311) |
| Byte 13 | Present Bin Number<br>(0 for 2311) |

Figure 39. Format of Extent Information to User

Extents are passed until the end of the extent area in the label is detected. Another test determines whether a pointer to a Format 3 label is present. If so, the Format 3 label is read and the extents in that label are passed to the user. This continues until there are no more extents within a label to be passed to the user.

The phase then finds the next symbolic unit to process those labels and extents. When all the incoming extents in the extent storage area in main storage have been processed, this phase exits to the open monitor, $$BOPEN.

$$BODACL: DA Close, Input/Output Charts RF-RH

Objective: To read or write standard user trailer labels, and to test for track hold.

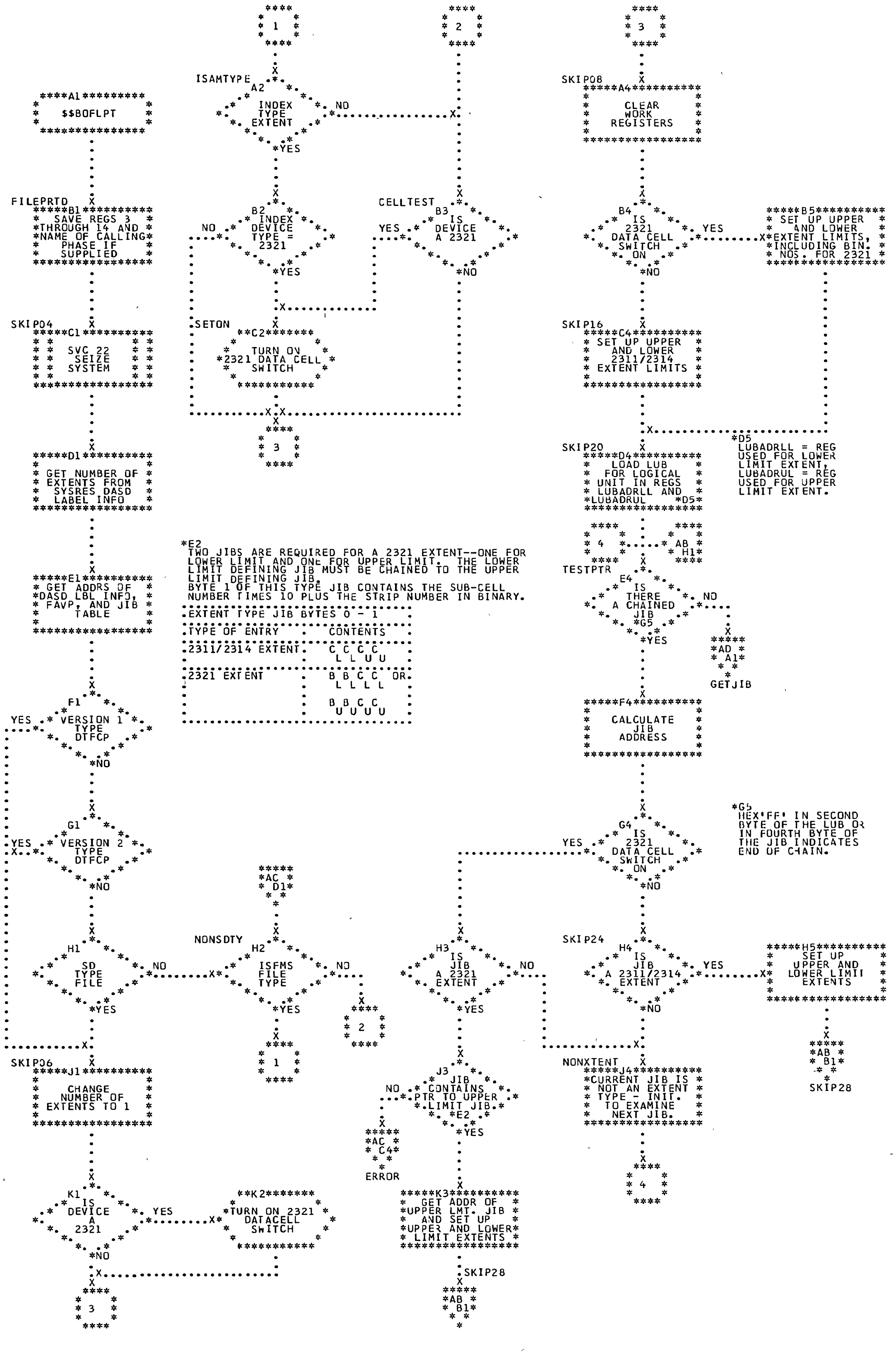
Entry: From the Close Monitor or from a message writer phase.

Exit: To the Close Monitor, $$BCLOSE; to $$BOMSG1 if a message is required; or to $$BOSDC2 to free any tracks.

Method: For input files, phase $$BODACL initializes the search CCW with a key argument of the first standard user trailer label (UTL0). The label is read and control is passed to the user's label routine. Processing of standard user trailer labels continues until either the maximum number of trailer labels are reaḋ (5 for 2321, 8 for 2311, 2314, or 2319), or a file mark (a UTL with a data length of 0) is read. Control then returns to the Close Monitor.

For output files, phase $$BODACL initializes the search CCW with a key argument of UTL0, the end-of-file mark written after the last UHL. When the UTL0 label is found, control passes to the user's label routine. Control returns to $$BODACL to write the standard user trailer label on the user's label track. The first standard user trailer label written is identified by UTL0 and is written over the end-of-file mark previously identified by UTL0. A new end-of-file mark (a standard user trailer label with a data length of 0) is written and the Close Monitor is fetched after all standard user trailer labels are processed. The maximum number of standard user trailer labels permitted (excluding the end-of-file mark), is 5 for a 2321 and 8 for a 2311, 2314, or 2319.

Chart AA. $$BOFLPT: DASD File Protect (Section 1 of 4)


1
2
3
A1
$$BOFLPT
ISAMTYPE
A2
INDEX TYPE EXTENT
NO
YES
SKIP08
A4
CLEAR WORK REGISTERS
FILEPRTD
B1
SAVE REGS 3 THROUGH 14 AND NAME OF CALLING PHASE IF SUPPLIED
B2
INDEX DEVICE TYPE = 2321
NO
YES
CELLTEST
B3
IS DEVICE A 2321
YES
NO
B4
IS 2321 DATA CELL SWITCH ON
YES
NO
B5
SET UP UPPER AND LOWER EXTENT LIMITS, INCLUDING BIN. NOS. FOR 2321
SKIP04
C1
SVC 22 SEIZE SYSTEM
SETON
C2
TURN ON 2321 DATA CELL SWITCH
SKIP16
C4
SET UP UPPER AND LOWER 2311/2314 EXTENT LIMITS
3
D1
GET NUMBER OF EXTENTS FROM SYSRES DASD LABEL INFO
SKIP20
D4
LOAD LUB FOR LOGICAL UNIT IN REGS LUBADRLL AND LUBADRUL
*D5
4
AB H1
TESTPTR
E4
IS THERE A CHAINED JIB
*G5
NO
YES
AD A1
GETJIB
E1
GET ADDRS OF DASD LBL INFO, FAVP, AND JIB TABLE
F1
VERSION 1 TYPE DTFCP
YES
NO
F4
CALCULATE JIB ADDRESS
G1
VERSION 2 TYPE DTFCP
YES
NO
G4
IS 2321 DATA CELL SWITCH ON
YES
NO
AC D1
NONSDTY
H1
SD TYPE FILE
NO
YES
H2
ISFMS FILE TYPE
NO
YES
2
1
H3
IS JIB A 2321 EXTENT
NO
YES
SKIP24
H4
IS JIB A 2311/2314 EXTENT
YES
NO
H5
SET UP UPPER AND LOWER LIMIT EXTENTS
AB B1
SKIP28
SKIP06
J1
CHANGE NUMBER OF EXTENTS TO 1
J3
JIB CONTAINS PTR TO UPPER LIMIT JIB
*E2
NO
YES
AC C4
ERROR
NONXTENT
J4
CURRENT JIB IS NOT AN EXTENT TYPE - INIT. TO EXAMINE NEXT JIB.
4
K1
IS DEVICE A 2321
YES
NO
K2
TURN ON 2321 DATACELL SWITCH
K3
GET ADDR OF UPPER LMT. JIB AND SET UP UPPER AND LOWER LIMIT EXTENTS
SKIP28
AB B1
3

*E2
TWO JIBS ARE REQUIRED FOR A 2321 EXTENT--ONE FOR LOWER LIMIT AND ONE FOR UPPER LIMIT. THE LOWER LIMIT DEFINING JIB MUST BE CHAINED TO THE UPPER LIMIT DEFINING JIB.
BYTE 1 OF THIS TYPE JIB CONTAINS THE SUB-CELL NUMBER TIMES 10 PLUS THE STRIP NUMBER IN BINARY.

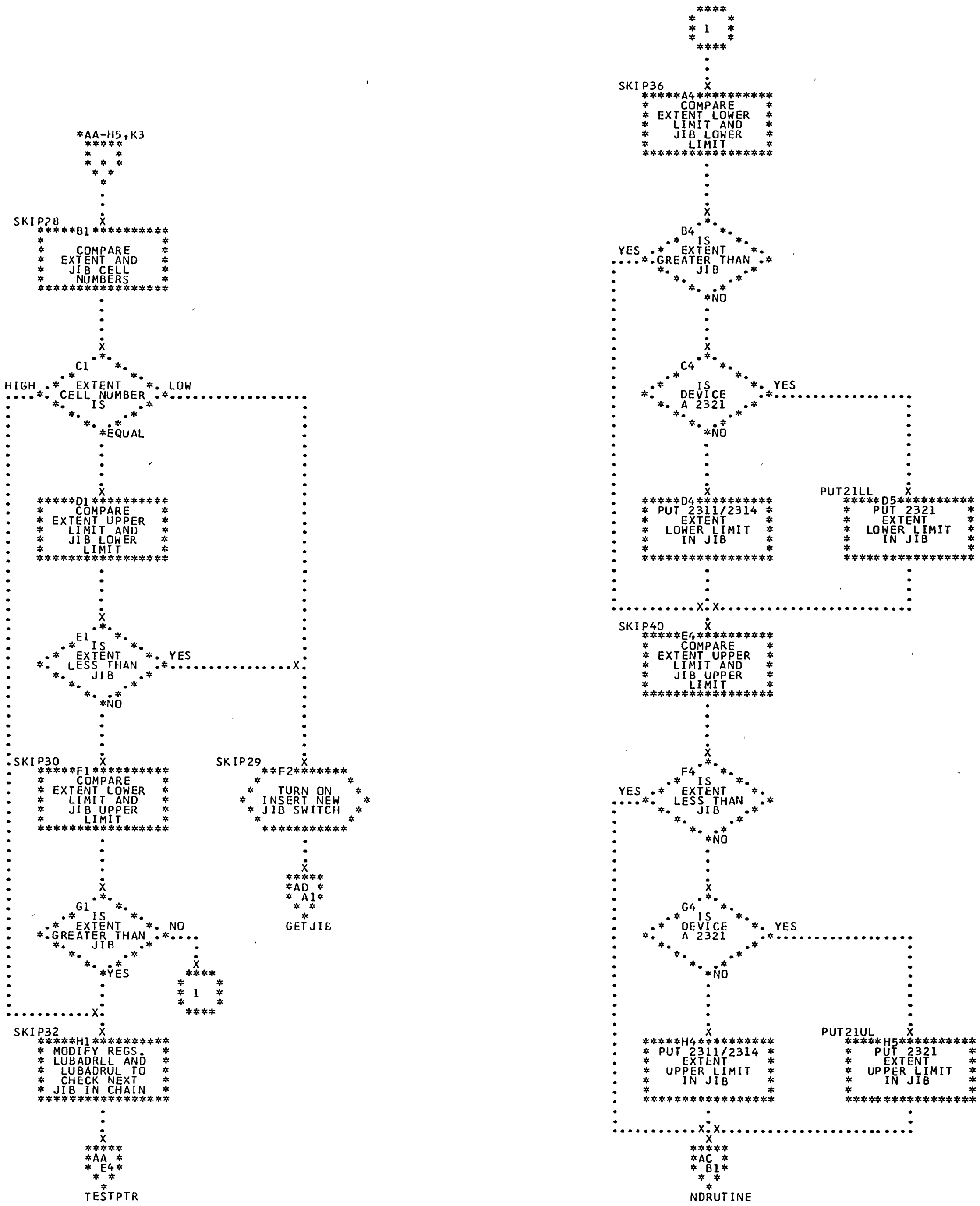
EXTENT TYPE JIB BYTES 0 - 1

| TYPE OF ENTRY | CONTENTS |
|---|---|
| 2311/2314 EXTENT | $C_L\ C_L\ C_U\ C_U$ |
| 2321 EXTENT | $B_L\ B_L\ C_L\ C_L$ OR<br>$B_U\ B_U\ C_U\ C_U$ |

*D5
LUBADRLL = REG USED FOR LOWER LIMIT EXTENT, LUBADRUL = REG USED FOR UPPER LIMIT EXTENT.

*G5
HEX'FF' IN SECOND BYTE OF THE LUB OR IN FOURTH BYTE OF THE JIB INDICATES END OF CHAIN.

Chart AB. $$BOFLPT: DASD File Protect (Section 2 of 4)



*AA-H5,K3
SKIP28
B1
COMPARE EXTENT AND JIB CELL NUMBERS
C1
EXTENT CELL NUMBER IS
HIGH
LOW
EQUAL
D1
COMPARE EXTENT UPPER LIMIT AND JIB LOWER LIMIT
E1
IS EXTENT LESS THAN JIB
YES
NO
SKIP30
F1
COMPARE EXTENT LOWER LIMIT AND JIB UPPER LIMIT
SKIP29
F2
TURN ON INSERT NEW JIB SWITCH
AD A1
GETJIB
G1
IS EXTENT GREATER THAN JIB
NO
YES
1
SKIP32
H1
MODIFY REGS. LUBADRLL AND LUBADRUL TO CHECK NEXT JIB IN CHAIN
AA E4
TESTPTR
1
SKIP36
A4
COMPARE EXTENT LOWER LIMIT AND JIB LOWER LIMIT
B4
IS EXTENT GREATER THAN JIB
YES
NO
C4
IS DEVICE A 2321
YES
NO
D4
PUT 2311/2314 EXTENT LOWER LIMIT IN JIB
PUT21LL
D5
PUT 2321 EXTENT LOWER LIMIT IN JIB
SKIP40
E4
COMPARE EXTENT UPPER LIMIT AND JIB UPPER LIMIT
F4
IS EXTENT LESS THAN JIB
YES
NO
G4
IS DEVICE A 2321
YES
NO
H4
PUT 2311/2314 EXTENT UPPER LIMIT IN JIB
PUT21UL
H5
PUT 2321 EXTENT UPPER LIMIT IN JIB
AC B1
NDRUTINE

## Chart AC. $$BOFLPT: DASD File Protect (Section 3 of 4)

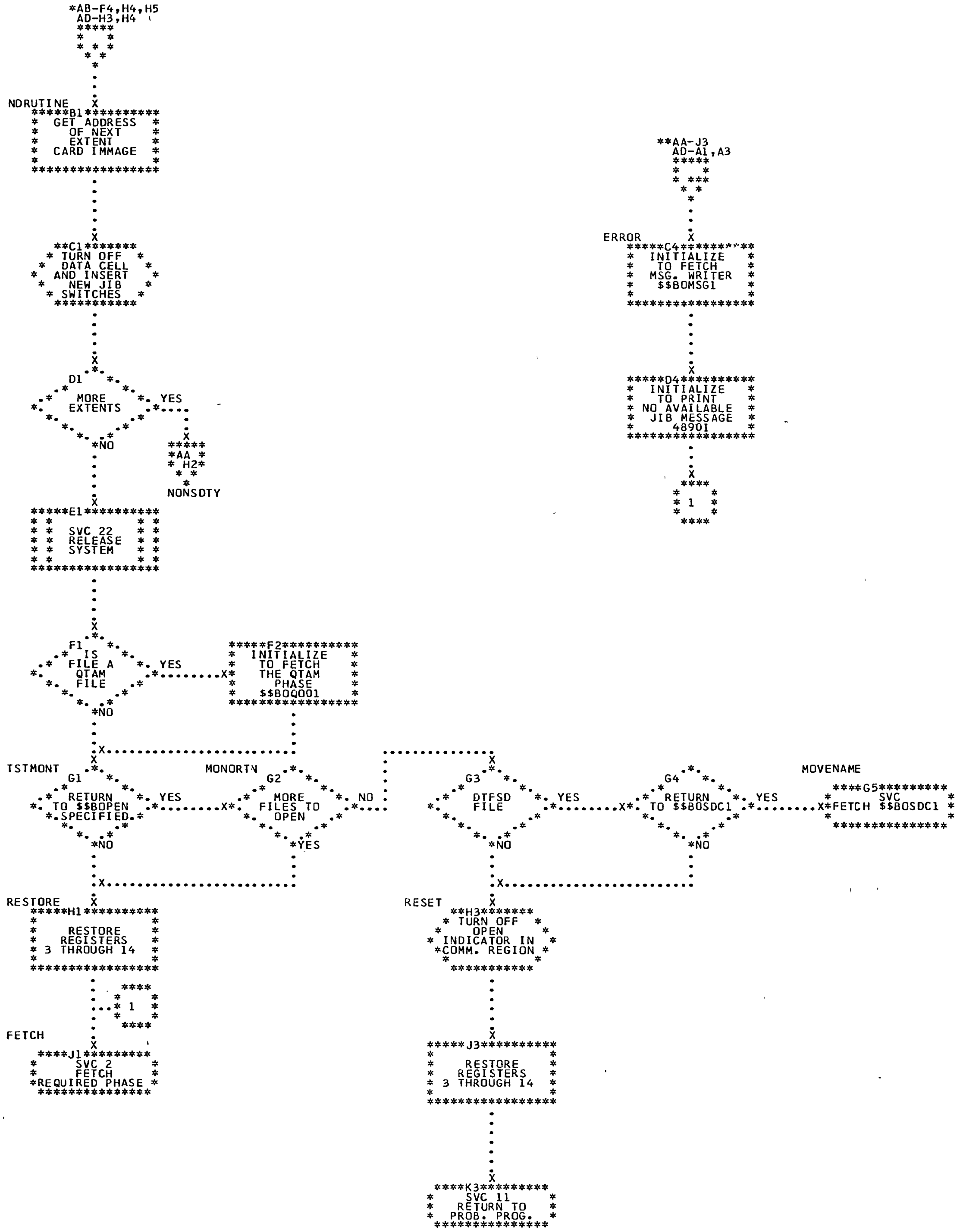
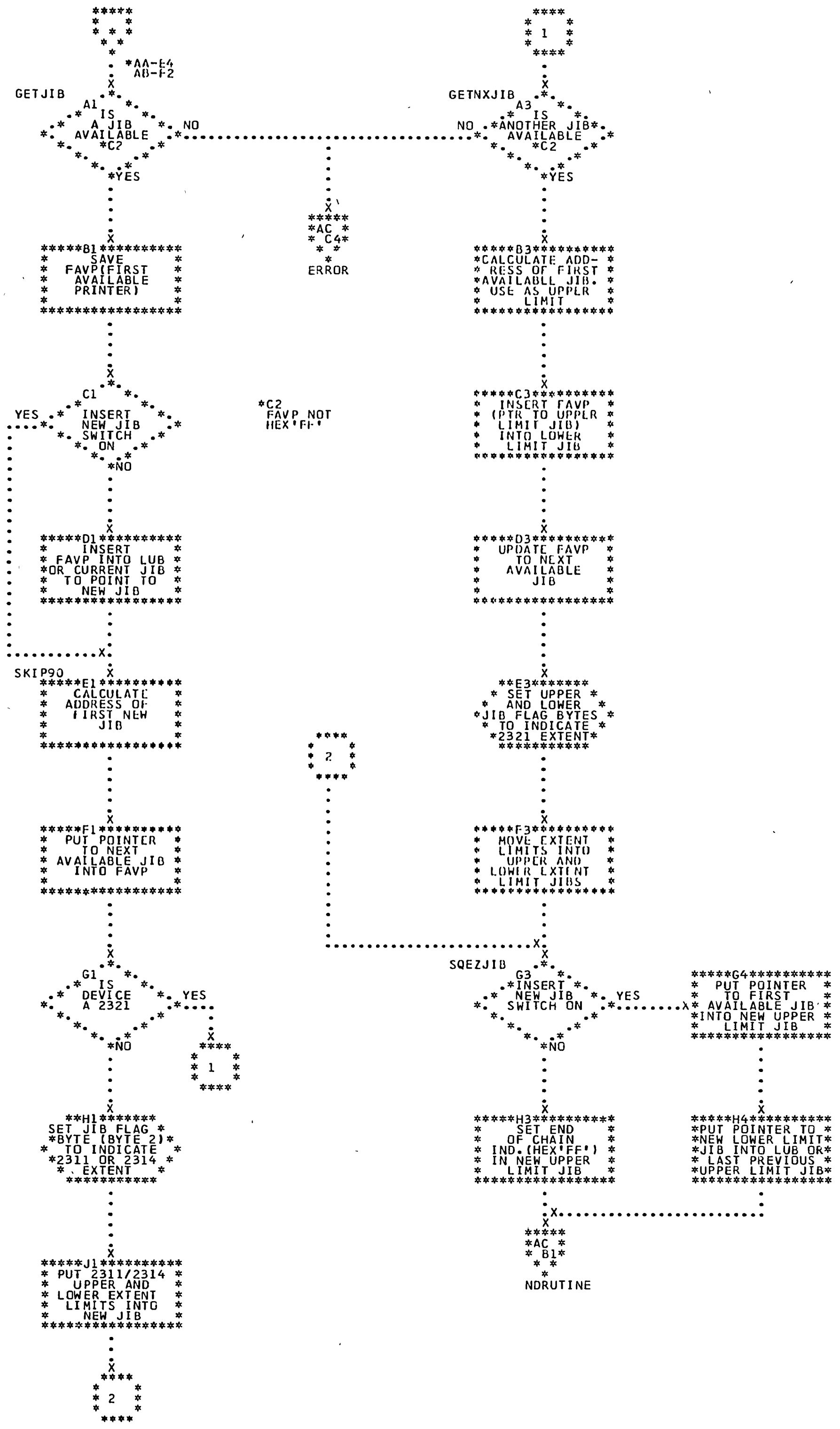

AB-F4,H4,H5
AD-H3,H4
NDRUTINE
B1
GET ADDRESS OF NEXT EXTENT CARD IMMAGE
C1
TURN OFF DATA CELL AND INSERT NEW JIB SWITCHES
D1
MORE EXTENTS
YES
NO
AA H2
NONSDTY
E1
SVC 22 RELEASE SYSTEM
F1
IS FILE A QTAM FILE
YES
NO
F2
INITIALIZE TO FETCH THE QTAM PHASE $$BOQ001
TSTMONT
G1
RETURN TO $$BOPEN SPECIFIED
YES
NO
MONORTN
G2
MORE FILES TO OPEN
NO
YES
RESTORE
H1
RESTORE REGISTERS 3 THROUGH 14
1
FETCH
J1
SVC 2 FETCH REQUIRED PHASE
AA-J3
AD-A1,A3
ERROR
C4
INITIALIZE TO FETCH MSG. WRITER $$BOMSG1
D4
INITIALIZE TO PRINT NO AVAILABLE JIB MESSAGE 4890I
1
G3
DTFSD FILE
YES
NO
G4
RETURN TO $$BOSDC1
YES
NO
MOVENAME
G5
SVC FETCH $$BOSDC1
RESET
H3
TURN OFF OPEN INDICATOR IN COMM. REGION
J3
RESTORE REGISTERS 3 THROUGH 14
K3
SVC 11 RETURN TO PROB. PROG.

Chart AD. $$BOFLPT: DASD File Protect (Section 4 of 4)



AA-E4
AB-F2
GETJIB
A1
IS A JIB AVAILABLE *C2
NO
YES
B1
SAVE FAVP(FIRST AVAILABLE PRINTER)
AC C4
ERROR
C1
INSERT NEW JIB SWITCH ON
YES
NO
*C2 FAVP NOT HEX'FF'
D1
INSERT FAVP INTO LUB OR CURRENT JIB TO POINT TO NEW JIB
SKIP90
E1
CALCULATE ADDRESS OF FIRST NEW JIB
F1
PUT POINTER TO NEXT AVAILABLE JIB INTO FAVP
G1
IS DEVICE A 2321
YES
1
NO
H1
SET JIB FLAG BYTE (BYTE 2) TO INDICATE 2311 OR 2314 EXTENT
J1
PUT 2311/2314 UPPER AND LOWER EXTENT LIMITS INTO NEW JIB
2
1
GETNXJIB
A3
IS ANOTHER JIB AVAILABLE *C2
NO
YES
B3
CALCULATE ADDRESS OF FIRST AVAILABLE JIB. USE AS UPPER LIMIT
C3
INSERT FAVP (PTR TO UPPER LIMIT JIB) INTO LOWER LIMIT JIB
D3
UPDATE FAVP TO NEXT AVAILABLE JIB
E3
SET UPPER AND LOWER JIB FLAG BYTES TO INDICATE 2321 EXTENT
2
F3
MOVE EXTENT LIMITS INTO UPPER AND LOWER EXTENT LIMIT JIBS
SQEZJIB
G3
INSERT NEW JIB SWITCH ON
YES
NO
G4
PUT POINTER TO FIRST AVAILABLE JIB INTO NEW UPPER LIMIT JIB
H3
SET END OF CHAIN IND.(HEX'FF') IN NEW UPPER LIMIT JIB
H4
PUT POINTER TO NEW LOWER LIMIT JIB INTO LUB OR LAST PREVIOUS UPPER LIMIT JIB
AC B1
NDRUTINE

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

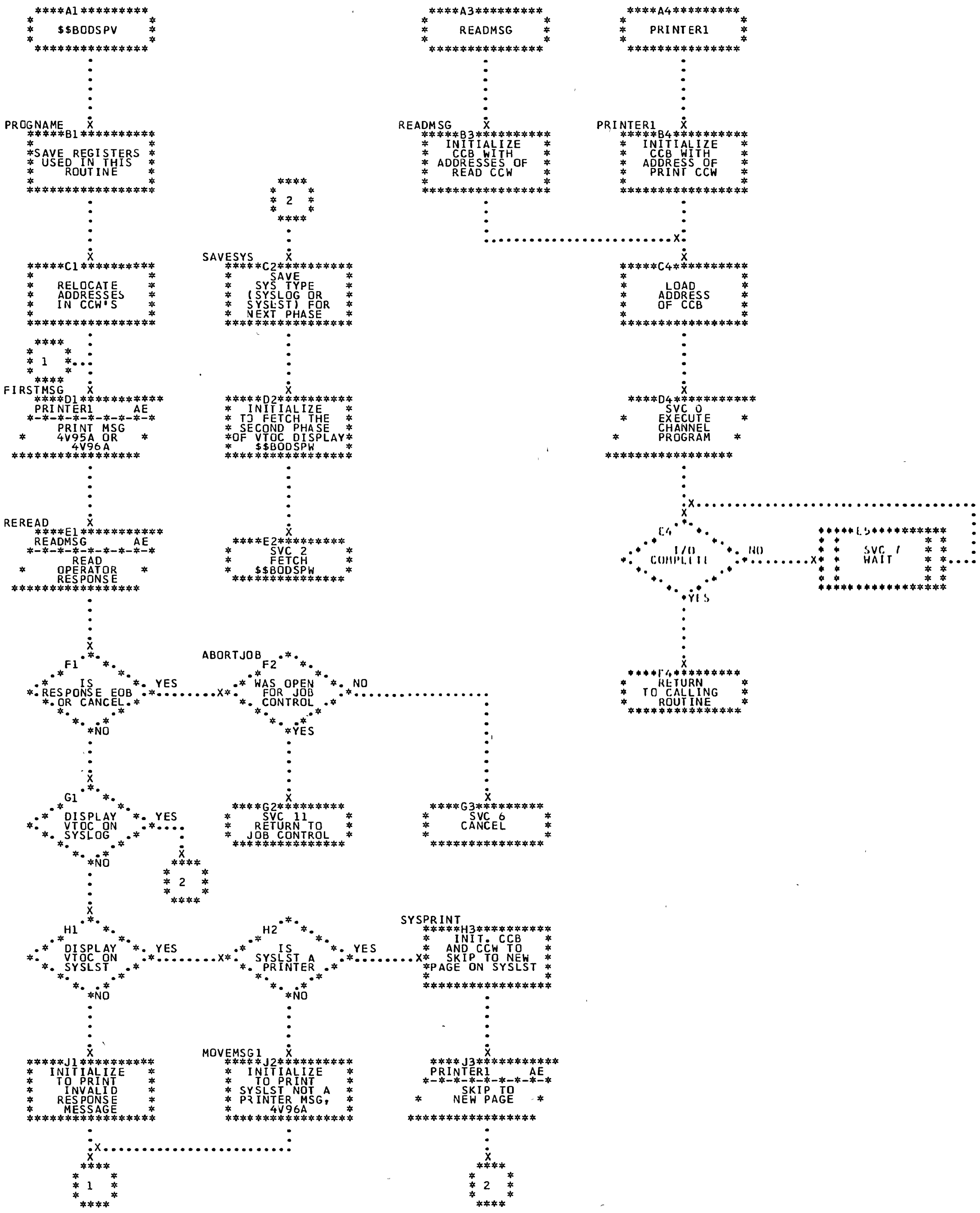
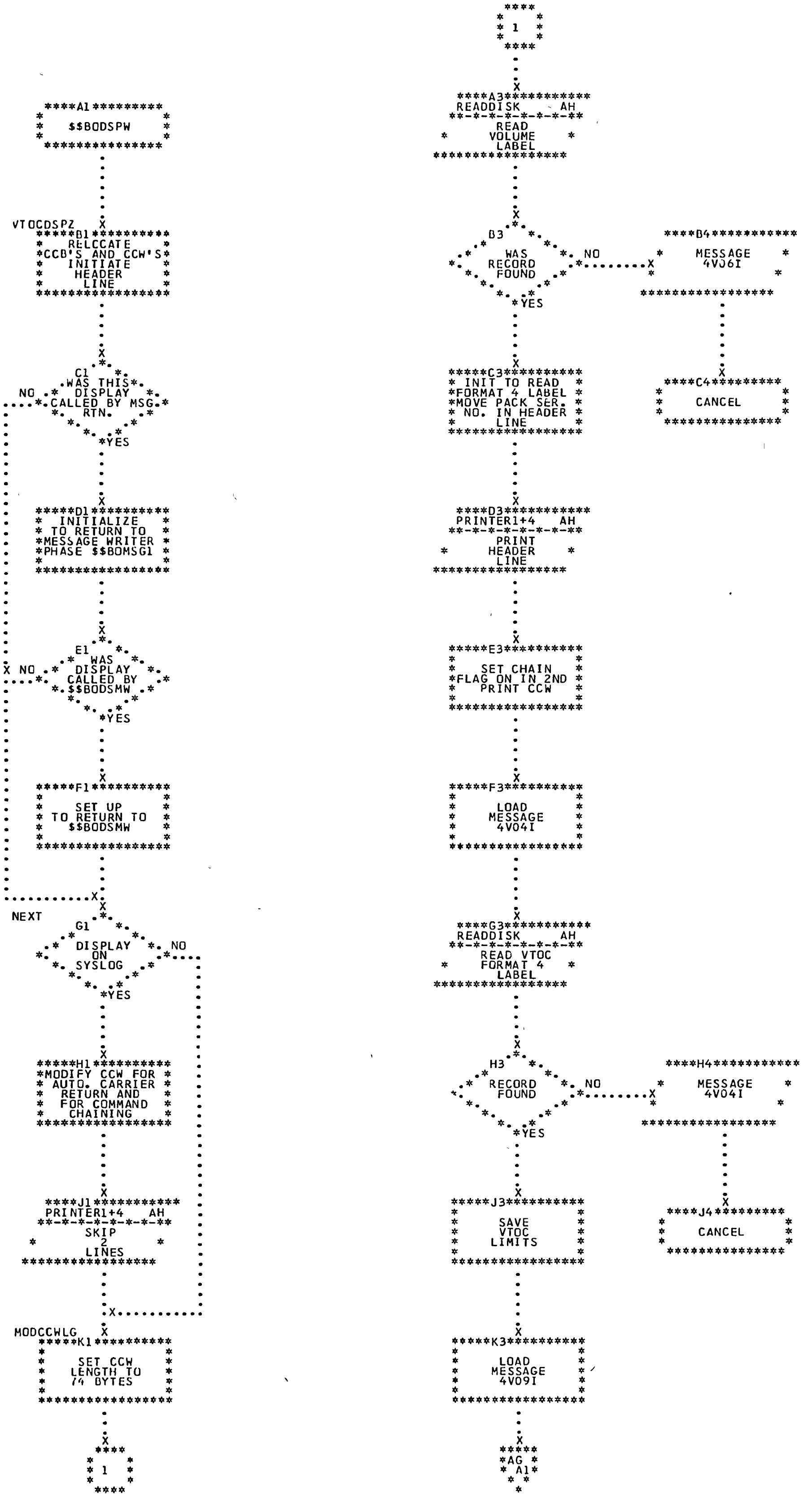

A1
$$BODSPV
A3
READMSG
A4
PRINTER1
PROGNAME
B1
SAVE REGISTERS USED IN THIS ROUTINE
READMSG
B3
INITIALIZE CCB WITH ADDRESSES OF READ CCW
PRINTER1
B4
INITIALIZE CCB WITH ADDRESS OF PRINT CCW
2
C1
RELOCATE ADDRESSES IN CCW'S
SAVESYS
C2
SAVE SYS TYPE (SYSLOG OR SYSLST) FOR NEXT PHASE
C4
LOAD ADDRESS OF CCB
1
FIRSTMSG
D1
PRINTER1 AE
PRINT MSG 4V95A OR 4V96A
D2
INITIALIZE TO FETCH THE SECOND PHASE OF VTOC DISPLAY $$BODSPW
D4
SVC 0 EXECUTE CHANNEL PROGRAM
REREAD
E1
READMSG AE
READ OPERATOR RESPONSE
E2
SVC 2 FETCH $$BODSPW
E4
I/O COMPLETE
NO
YES
L5
SVC 7 WAIT
F1
IS RESPONSE EOB OR CANCEL
YES
NO
ABORTJOB
F2
WAS OPEN FOR JOB CONTROL
NO
YES
F4
RETURN TO CALLING ROUTINE
G1
DISPLAY VTOC ON SYSLOG
YES
NO
2
G2
SVC 11 RETURN TO JOB CONTROL
G3
SVC 6 CANCEL
H1
DISPLAY VTOC ON SYSLST
YES
NO
H2
IS SYSLST A PRINTER
YES
NO
SYSPRINT
H3
INIT. CCB AND CCW TO SKIP TO NEW PAGE ON SYSLST
J1
INITIALIZE TO PRINT INVALID RESPONSE MESSAGE
MOVEMSG1
J2
INITIALIZE TO PRINT SYSLST NOT A PRINTER MSG, 4V96A
J3
PRINTER1 AE
SKIP TO NEW PAGE
1
2

Chart AF. $$BODSPW: VTOC Display, Phase 2 (Section 1 of 3)


A1
$$BODSPW
VTOCDSPZ
B1
RELCCATE CCB'S AND CCW'S INITIATE HEADER LINE
C1
WAS THIS DISPLAY CALLED BY MSG. RTN.
NO
YES
D1
INITIALIZE TO RETURN TO MESSAGE WRITER PHASE $$BOMSG1
E1
WAS DISPLAY CALLED BY $$BODSMW
NO
YES
F1
SET UP TO RETURN TO $$BODSMW
NEXT
G1
DISPLAY ON SYSLOG
NO
YES
H1
MODIFY CCW FOR AUTO. CARRIER RETURN AND FOR COMMAND CHAINING
J1
PRINTER1+4 AH
SKIP 2 LINES
MODCCWLG
K1
SET CCW LENGTH TO 74 BYTES
1
1
A3
READDISK AH
READ VOLUME LABEL
B3
WAS RECORD FOUND
NO
YES
B4
MESSAGE 4V06I
C4
CANCEL
C3
INIT TO READ FORMAT 4 LABEL MOVE PACK SER. NO. IN HEADER LINE
D3
PRINTER1+4 AH
PRINT HEADER LINE
E3
SET CHAIN FLAG ON IN 2ND PRINT CCW
F3
LOAD MESSAGE 4V04I
G3
READDISK AH
READ VTOC FORMAT 4 LABEL
H3
RECORD FOUND
NO
YES
H4
MESSAGE 4V04I
J4
CANCEL
J3
SAVE VTOC LIMITS
K3
LOAD MESSAGE 4V09I
AG A1

## Chart AG. $$BODSPW: VTOC Display, Phase 2 (Section 2 of 3)

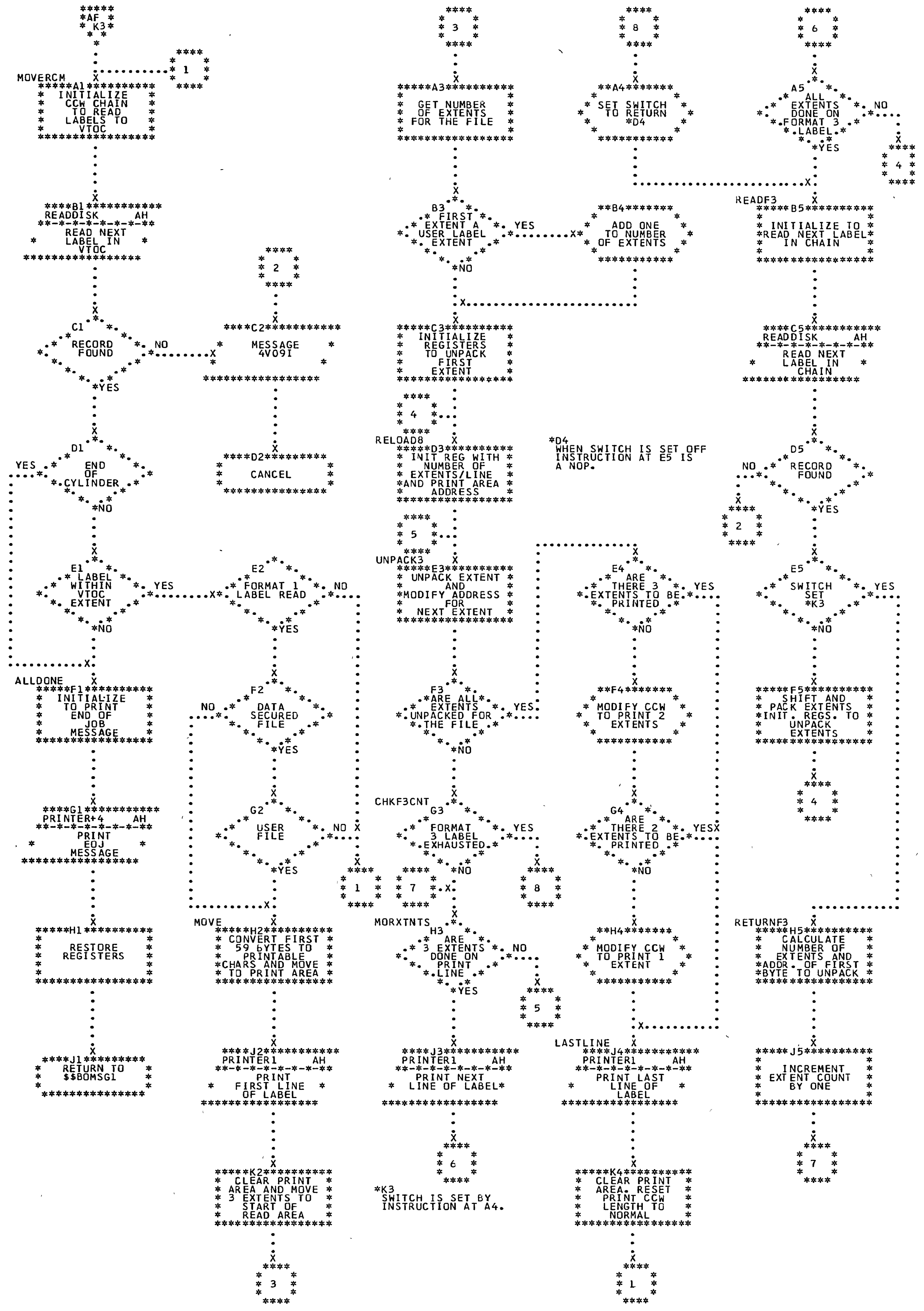

AF K3
1
MOVERCM A1
INITIALIZE CCW CHAIN TO READ LABELS TO VTOC
B1 READDISK AH
READ NEXT LABEL IN VTOC
C1 RECORD FOUND
NO
YES
D1 END OF CYLINDER
YES
NO
E1 LABEL WITHIN VTOC EXTENT
YES
NO
ALLDONE F1
INITIALIZE TO PRINT END OF JOB MESSAGE
G1 PRINTER+4 AH
PRINT EOJ MESSAGE
H1
RESTORE REGISTERS
J1
RETURN TO $$BOMSG1
2
C2
MESSAGE 4V09I
D2
CANCEL
E2 FORMAT 1 LABEL READ
NO
YES
F2 DATA SECURED FILE
NO
YES
G2 USER FILE
NO
YES
1
MOVE H2
CONVERT FIRST 59 BYTES TO PRINTABLE CHARS AND MOVE TO PRINT AREA
J2 PRINTER1 AH
PRINT FIRST LINE OF LABEL
K2
CLEAR PRINT AREA AND MOVE 3 EXTENTS TO START OF READ AREA
3
3
A3
GET NUMBER OF EXTENTS FOR THE FILE
B3 FIRST EXTENT A USER LABEL EXTENT
YES
NO
C3
INITIALIZE REGISTERS TO UNPACK FIRST EXTENT
4
RELOAD8 D3
INIT REG WITH NUMBER OF EXTENTS/LINE AND PRINT AREA ADDRESS
5
UNPACK3 E3
UNPACK EXTENT AND MODIFY ADDRESS FOR NEXT EXTENT
F3 ARE ALL EXTENTS UNPACKED FOR THE FILE
YES
NO
CHKF3CNT G3 FORMAT 3 LABEL EXHAUSTED
YES
NO
7
8
MORXTNTS H3 ARE 3 EXTENTS DONE ON PRINT LINE
NO
YES
5
J3 PRINTER1 AH
PRINT NEXT LINE OF LABEL
6
*K3
SWITCH IS SET BY INSTRUCTION AT A4.
8
A4
SET SWITCH TO RETURN *D4
B4
ADD ONE TO NUMBER OF EXTENTS
*D4
WHEN SWITCH IS SET OFF INSTRUCTION AT E5 IS A NOP.
E4 ARE THERE 3 EXTENTS TO BE PRINTED
YES
NO
F4
MODIFY CCW TO PRINT 2 EXTENTS
G4 ARE THERE 2 EXTENTS TO BE PRINTED
YES
NO
H4
MODIFY CCW TO PRINT 1 EXTENT
LASTLINE J4 PRINTER1 AH
PRINT LAST LINE OF LABEL
K4
CLEAR PRINT AREA. RESET PRINT CCW LENGTH TO NORMAL
L
6
A5 ALL EXTENTS DONE ON FORMAT 3 LABEL
NO
YES
4
READF3 B5
INITIALIZE TO READ NEXT LABEL IN CHAIN
C5 READDISK AH
READ NEXT LABEL IN CHAIN
D5 RECORD FOUND
NO
YES
2
E5 SWITCH SET *K3
YES
NO
F5
SHIFT AND PACK EXTENTS INIT. REGS. TO UNPACK EXTENTS
4
RETURNF3 H5
CALCULATE NUMBER OF EXTENTS AND ADDR. OF FIRST BYTE TO UNPACK
J5
INCREMENT EXTENT COUNT BY ONE
7

Chart AH. $$BODSPW: VTOC Display, Phase 2 (Section 3 of 3)

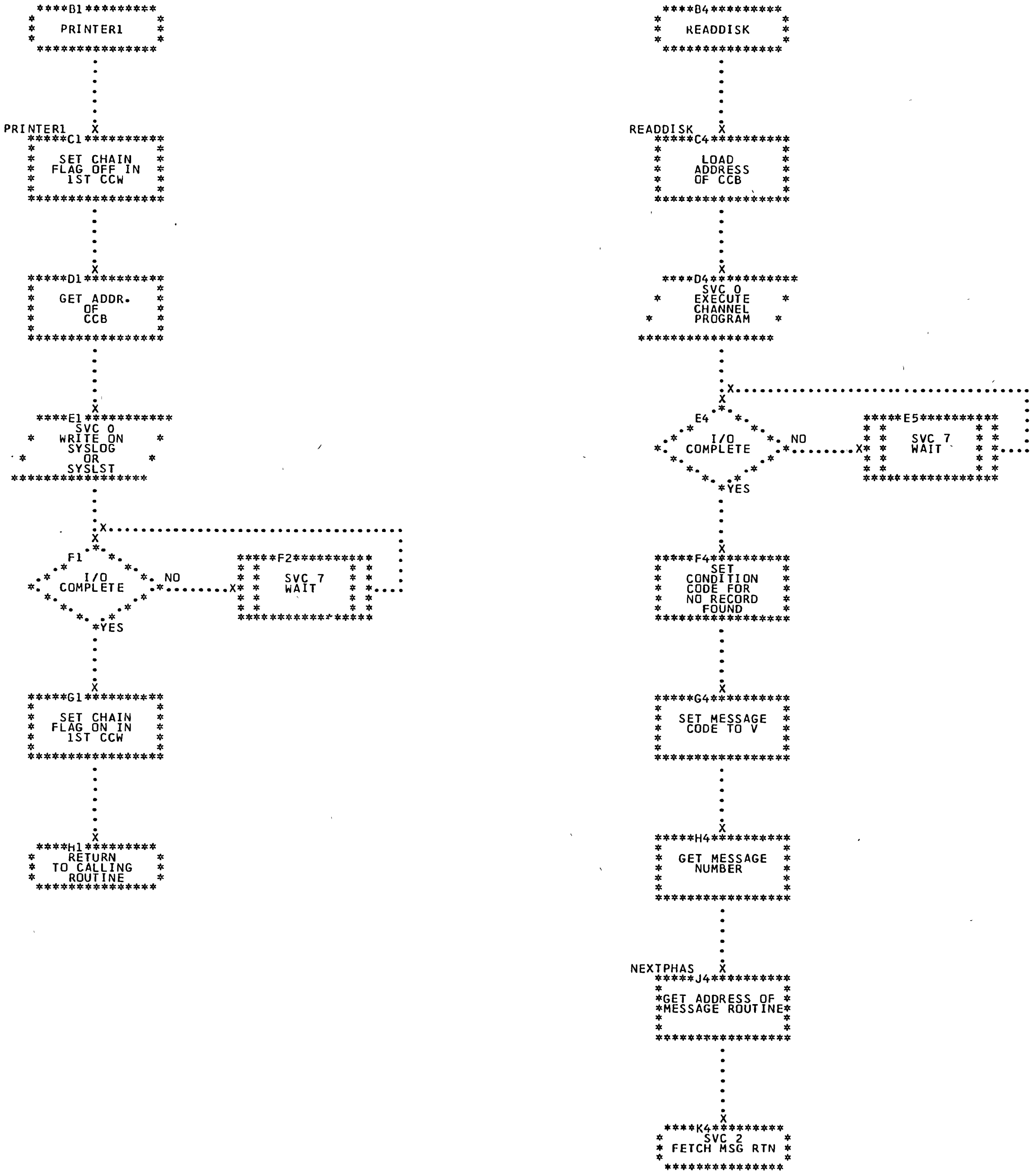


Chart AJ. $$BOVDMP: VTOC Dump (Section 1 of 2)

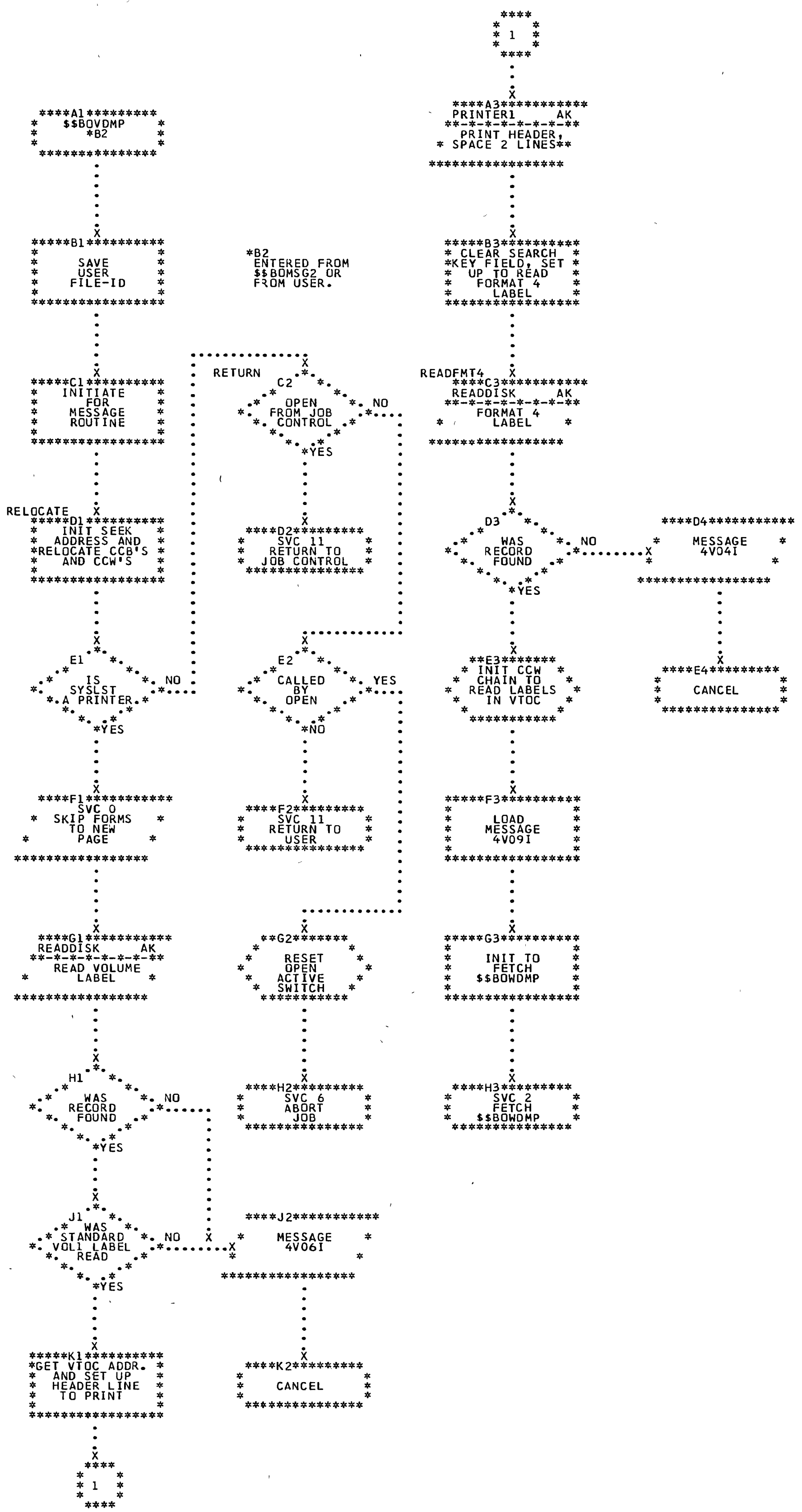
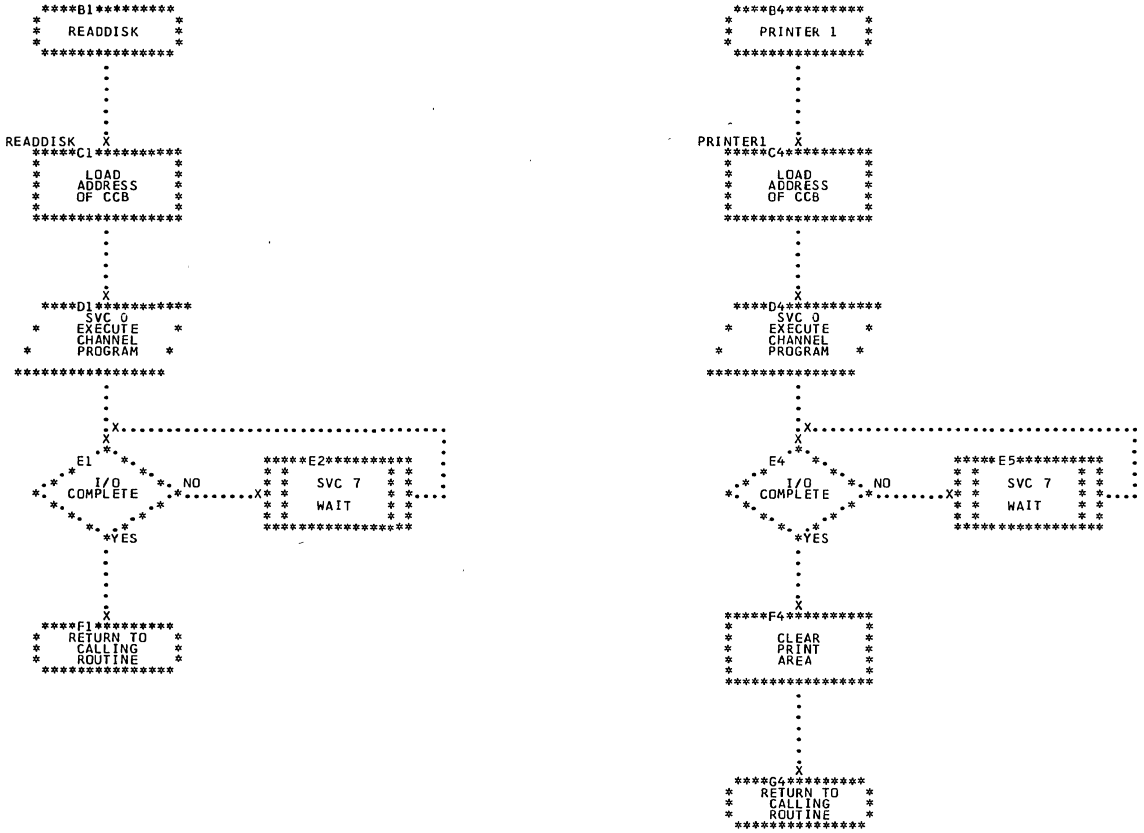


Chart AK. $$BOVDMP: VTOC Dump (Section 2 of 2)



## Chart AL. $$BOWDMP: VTOC Label List (Section 1 of 2)

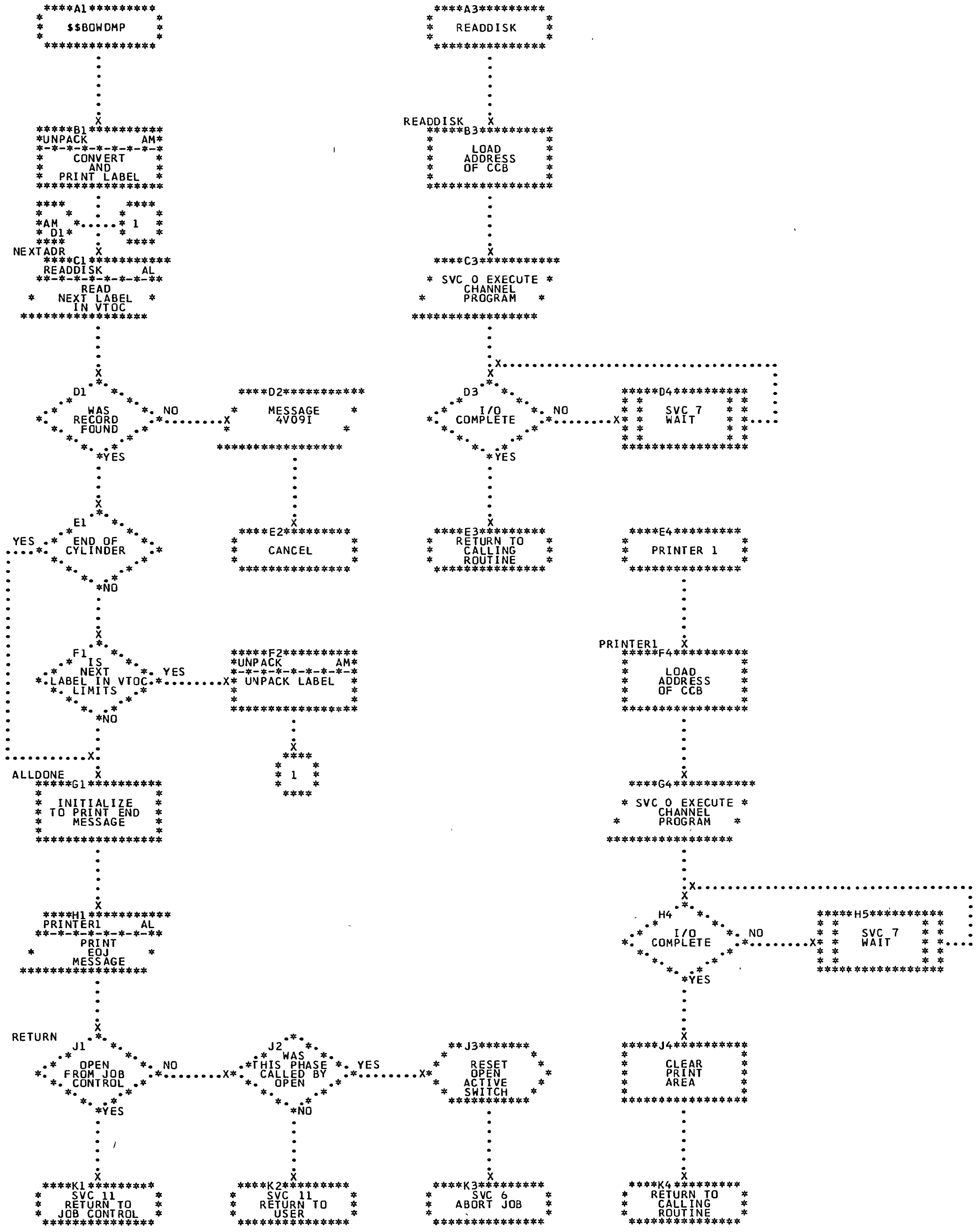
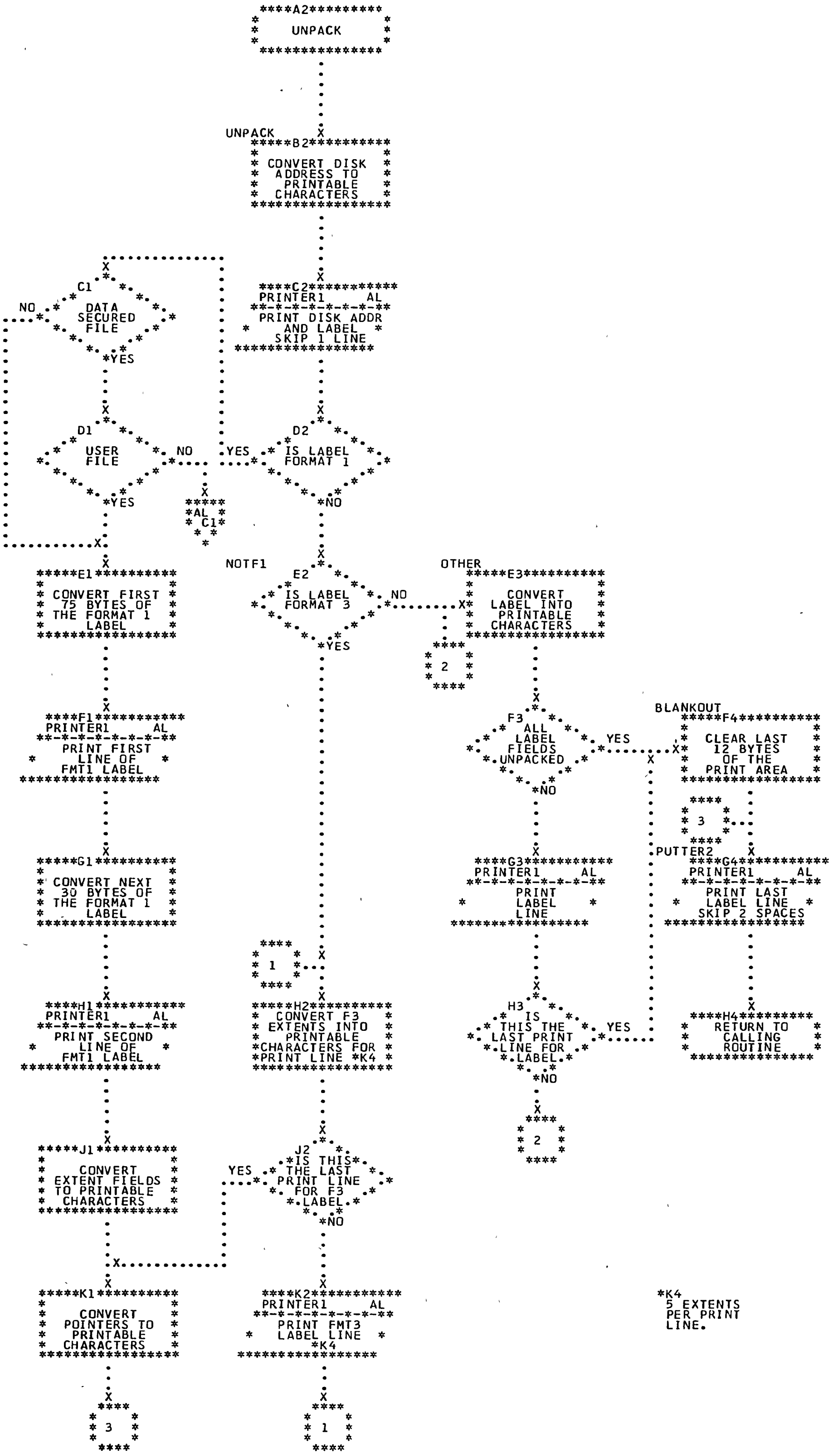


Chart AM. $$BOWDMP: VTOC Label List (Section 2 of 2)


A2
UNPACK
UNPACK
B2
CONVERT DISK ADDRESS TO PRINTABLE CHARACTERS
C1
DATA SECURED FILE
NO
YES
C2
PRINTER1 AL
PRINT DISK ADDR AND LABEL SKIP 1 LINE
D1
USER FILE
NO
YES
AL C1
D2
IS LABEL FORMAT 1
YES
NO
NOTF1
E1
CONVERT FIRST 75 BYTES OF THE FORMAT 1 LABEL
E2
IS LABEL FORMAT 3
NO
YES
OTHER
E3
CONVERT LABEL INTO PRINTABLE CHARACTERS
2
F1
PRINTER1 AL
PRINT FIRST LINE OF FMT1 LABEL
F3
ALL LABEL FIELDS UNPACKED
YES
NO
BLANKOUT
F4
CLEAR LAST 12 BYTES OF THE PRINT AREA
3
G1
CONVERT NEXT 30 BYTES OF THE FORMAT 1 LABEL
G3
PRINTER1 AL
PRINT LABEL LINE
PUTTER2
G4
PRINTER1 AL
PRINT LAST LABEL LINE SKIP 2 SPACES
1
H1
PRINTER1 AL
PRINT SECOND LINE OF FMT1 LABEL
H2
CONVERT F3 EXTENTS INTO PRINTABLE CHARACTERS FOR PRINT LINE *K4
H3
IS THIS THE LAST PRINT LINE FOR LABEL
YES
NO
H4
RETURN TO CALLING ROUTINE
2
J1
CONVERT EXTENT FIELDS TO PRINTABLE CHARACTERS
J2
IS THIS THE LAST PRINT LINE FOR F3 LABEL
YES
NO
K1
CONVERT POINTERS TO PRINTABLE CHARACTERS
K2
PRINTER1 AL
PRINT FMT3 LABEL LINE *K4
*K4 5 EXTENTS PER PRINT LINE.
3
1

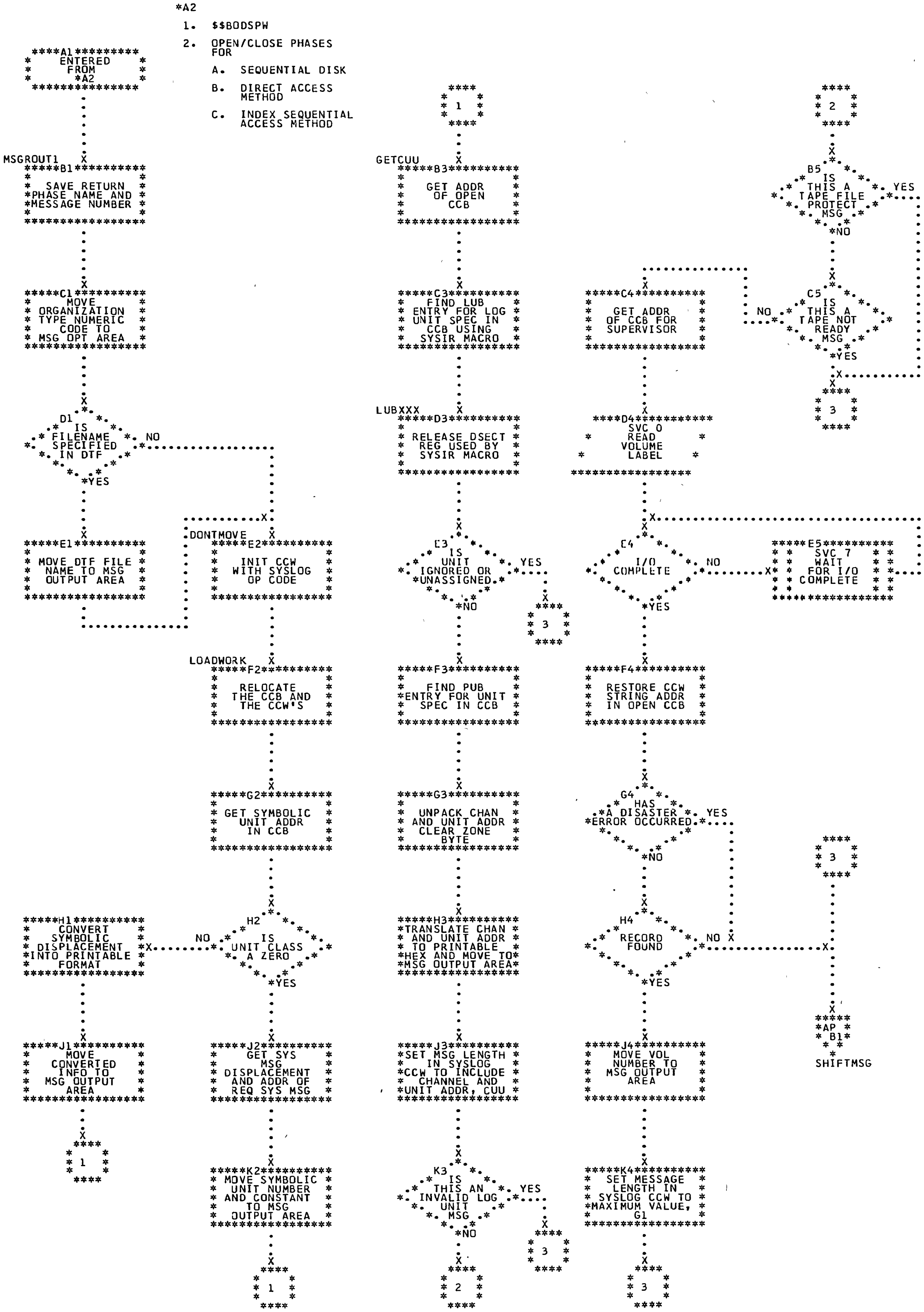
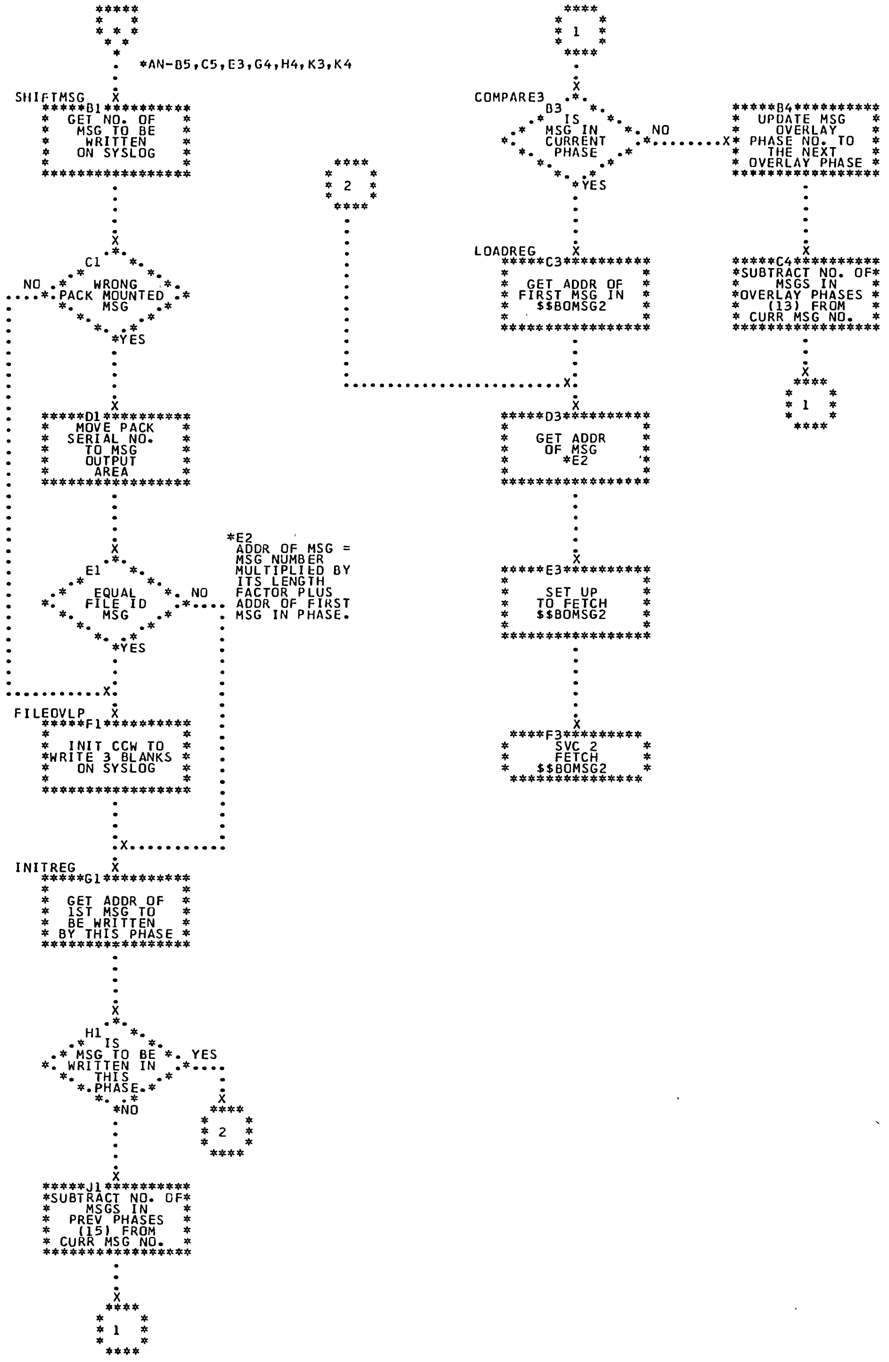
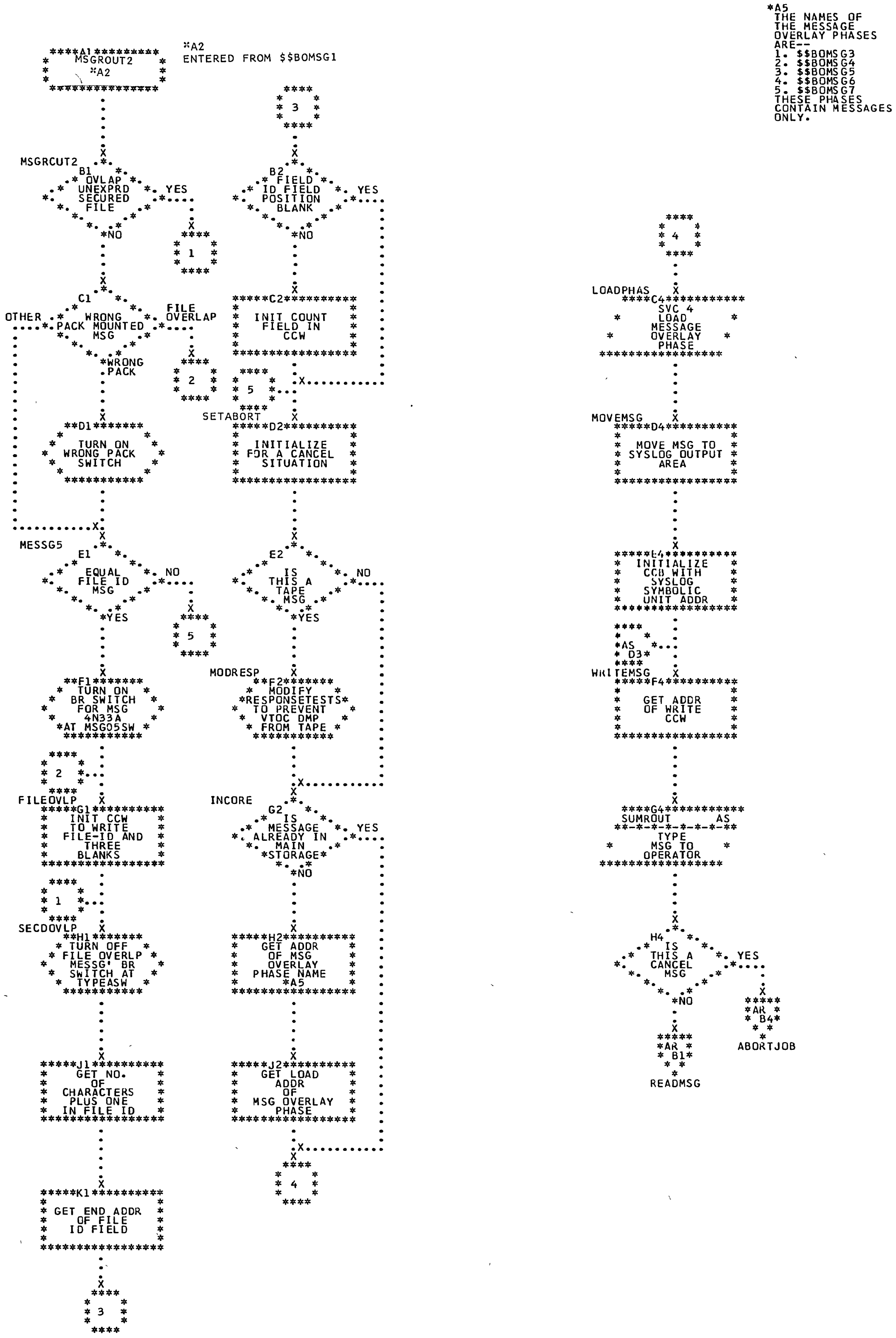

*A2
1. $$BODSPW
2. OPEN/CLOSE PHASES FOR
A. SEQUENTIAL DISK
B. DIRECT ACCESS METHOD
C. INDEX SEQUENTIAL ACCESS METHOD
A1 ENTERED FROM *A2
MSGROUT1
B1 SAVE RETURN PHASE NAME AND MESSAGE NUMBER
C1 MOVE ORGANIZATION TYPE NUMERIC CODE TO MSG OPT AREA
D1 IS FILENAME SPECIFIED IN DTF
NO
YES
E1 MOVE DTF FILE NAME TO MSG OUTPUT AREA
DONTMOVE
E2 INIT CCW WITH SYSLOG OP CODE
LOADWORK
F2 RELOCATE THE CCB AND THE CCW'S
G2 GET SYMBOLIC UNIT ADDR IN CCB
H2 IS UNIT CLASS A ZERO
NO
YES
H1 CONVERT SYMBOLIC DISPLACEMENT INTO PRINTABLE FORMAT
J1 MOVE CONVERTED INFO TO MSG OUTPUT AREA
1
J2 GET SYS MSG DISPLACEMENT AND ADDR OF REQ SYS MSG
K2 MOVE SYMBOLIC UNIT NUMBER AND CONSTANT TO MSG OUTPUT AREA
1
1
GETCUU
B3 GET ADDR OF OPEN CCB
C3 FIND LUB ENTRY FOR LOG UNIT SPEC IN CCB USING SYSIR MACRO
LUBXXX
D3 RELEASE DSECT REG USED BY SYSIR MACRO
E3 IS UNIT IGNORED OR UNASSIGNED
YES
3
NO
F3 FIND PUB ENTRY FOR UNIT SPEC IN CCB
G3 UNPACK CHAN AND UNIT ADDR CLEAR ZONE BYTE
H3 TRANSLATE CHAN AND UNIT ADDR TO PRINTABLE HEX AND MOVE TO MSG OUTPUT AREA
J3 SET MSG LENGTH IN SYSLOG CCW TO INCLUDE CHANNEL AND UNIT ADDR, CUU
K3 IS THIS AN INVALID LOG UNIT MSG
YES
3
NO
2
C4 GET ADDR OF CCB FOR SUPERVISOR
D4 SVC 0 READ VOLUME LABEL
E4 I/O COMPLETE
NO
YES
E5 SVC 7 WAIT FOR I/O COMPLETE
F4 RESTORE CCW STRING ADDR IN OPEN CCB
G4 HAS A DISASTER ERROR OCCURRED
YES
NO
H4 RECORD FOUND
NO
YES
J4 MOVE VOL NUMBER TO MSG OUTPUT AREA
K4 SET MESSAGE LENGTH IN SYSLOG CCW TO MAXIMUM VALUE, G1
3
2
B5 IS THIS A TAPE FILE PROTECT MSG
YES
NO
C5 IS THIS A TAPE NOT READY MSG
NO
YES
3
3
AP B1
SHIFTMSG

Chart AP. $$BOMSG1: Disk Open Error Message Writer, Phase 1 (Section 2 of 2)

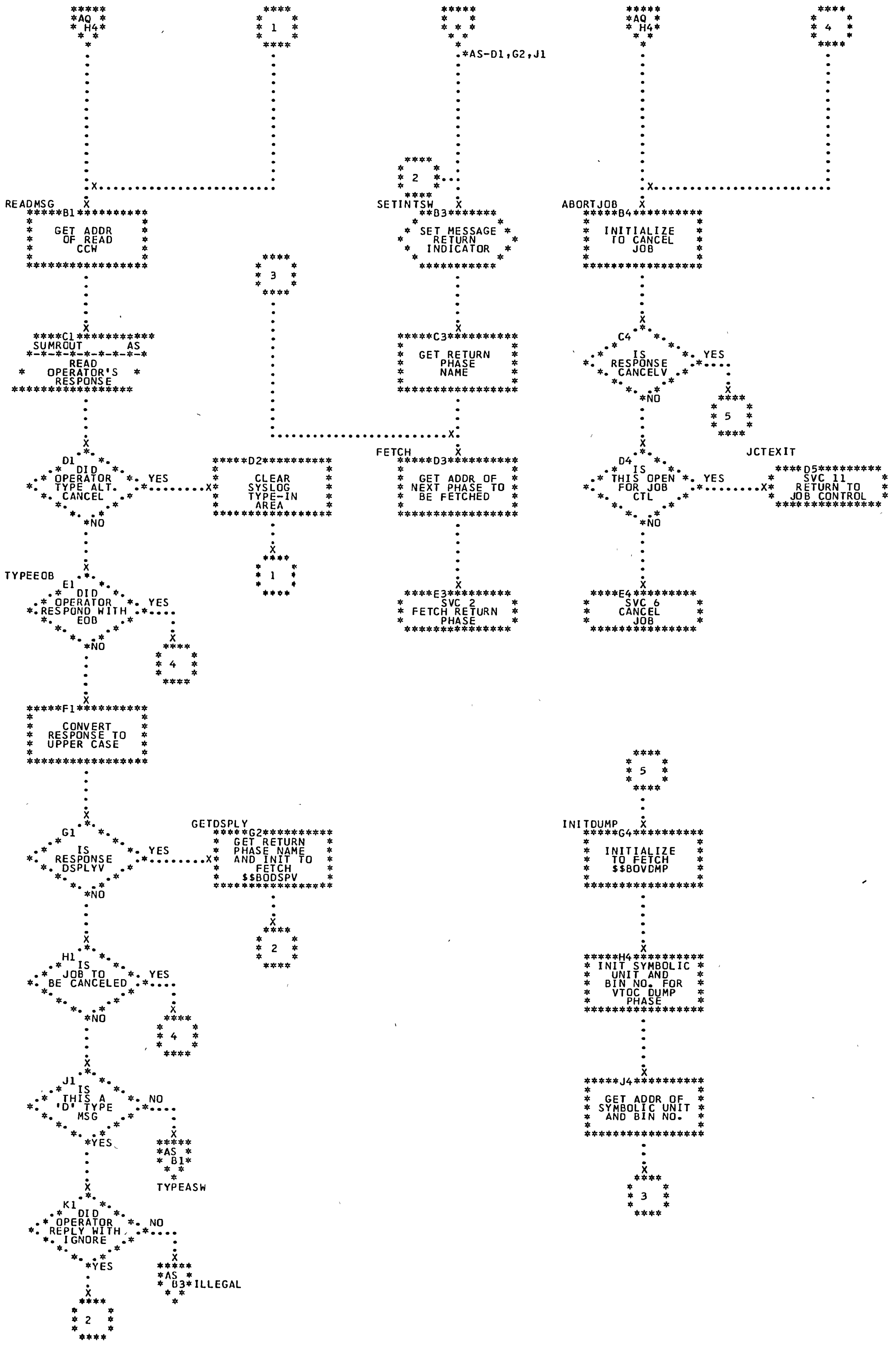

*AN-B5,C5,E3,G4,H4,K3,K4
SHIFTMSG
B1
GET NO. OF MSG TO BE WRITTEN ON SYSLOG
C1
WRONG PACK MOUNTED MSG
NO
YES
D1
MOVE PACK SERIAL NO. TO MSG OUTPUT AREA
E1
EQUAL FILE ID MSG
NO
YES
*E2
ADDR OF MSG = MSG NUMBER MULTIPLIED BY ITS LENGTH FACTOR PLUS ADDR OF FIRST MSG IN PHASE.
FILEOVLP
F1
INIT CCW TO WRITE 3 BLANKS ON SYSLOG
INITREG
G1
GET ADDR OF 1ST MSG TO BE WRITTEN BY THIS PHASE
H1
IS MSG TO BE WRITTEN IN THIS PHASE
YES
NO
2
J1
SUBTRACT NO. OF MSGS IN PREV PHASES (15) FROM CURR MSG NO.
1
2
1
COMPARE3
B3
IS MSG IN CURRENT PHASE
NO
YES
B4
UPDATE MSG OVERLAY PHASE NO. TO THE NEXT OVERLAY PHASE
LOADREG
C3
GET ADDR OF FIRST MSG IN $$BOMSG2
C4
SUBTRACT NO. OF MSGS IN OVERLAY PHASES (13) FROM CURR MSG NO.
1
D3
GET ADDR OF MSG *E2
E3
SET UP TO FETCH $$BOMSG2
F3
SVC 2 FETCH $$BOMSG2

## Chart AQ. $$BOMSG2: Disk Open Error Message Writer, Phase 2 (Section 1 of 3)


A1 MSGROUT2 *A2
*A2 ENTERED FROM $$BOMSG1
3
MSGRCUT2 B1 OVLAP UNEXPRD SECURED FILE
YES
NO
1
B2 FIELD ID FIELD POSITION BLANK
YES
NO
C1 WRONG PACK MOUNTED MSG
OTHER
FILE OVERLAP
WRONG PACK
2
C2 INIT COUNT FIELD IN CCW
5
D1 TURN ON WRONG PACK SWITCH
SETABORT D2 INITIALIZE FOR A CANCEL SITUATION
MESSG5 E1 EQUAL FILE ID MSG
NO
YES
5
E2 IS THIS A TAPE MSG
NO
YES
F1 TURN ON BR SWITCH FOR MSG 4N33A AT MSG05SW
MODRESP F2 MODIFY RESPONSETESTS TO PREVENT VTOC DMP FROM TAPE
2
FILEOVLP G1 INIT CCW TO WRITE FILE-ID AND THREE BLANKS
INCORE G2 IS MESSAGE ALREADY IN MAIN STORAGE
YES
NO
1
SECDOVLP H1 TURN OFF FILE OVERLP MESSG' BR SWITCH AT TYPEASW
H2 GET ADDR OF MSG OVERLAY PHASE NAME *A5
J1 GET NO. OF CHARACTERS PLUS ONE IN FILE ID
J2 GET LOAD ADDR OF MSG OVERLAY PHASE
4
K1 GET END ADDR OF FILE ID FIELD
3
4
LOADPHAS C4 SVC 4 LOAD MESSAGE OVERLAY PHASE
MOVEMSG D4 MOVE MSG TO SYSLOG OUTPUT AREA
E4 INITIALIZE CCB WITH SYSLOG SYMBOLIC UNIT ADDR
AS D3
WRITEMSG F4 GET ADDR OF WRITE CCW
G4 SUMROUT AS TYPE MSG TO OPERATOR
H4 IS THIS A CANCEL MSG
YES
NO
AR B4 ABORTJOB
AR B1 READMSG

*A5
THE NAMES OF
THE MESSAGE
OVERLAY PHASES
ARE--
1. $$BOMSG3
2. $$BOMSG4
3. $$BOMSG5
4. $$BOMSG6
5. $$BOMSG7
THESE PHASES
CONTAIN MESSAGES
ONLY.

Chart AR. $$BOMSG2: Disk Open Error Message Writer, Phase 2 (Section 2 of 3)



## Chart AS. $$BOMSG2: Disk Open Error Message Writer, Phase 2 (Section 3 of 3)

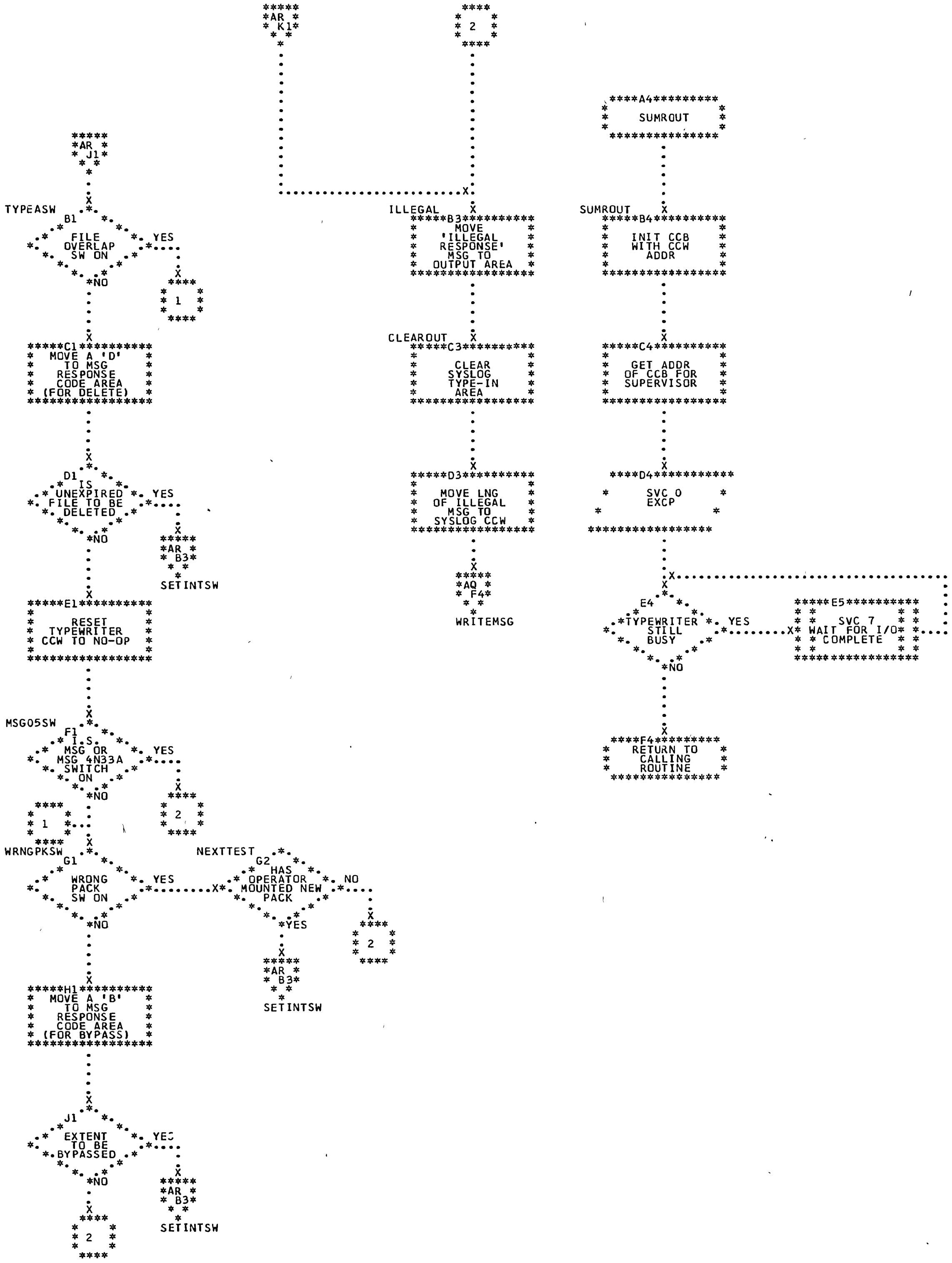

AR K1
2
SUMROUT A4
SUMROUT
AR J1
TYPEASW B1
FILE OVERLAP SW ON
YES
NO
1
ILLEGAL B3
MOVE 'ILLEGAL RESPONSE' MSG TO OUTPUT AREA
SUMROUT B4
INIT CCB WITH CCW ADDR
C1
MOVE A 'D' TO MSG RESPONSE CODE AREA (FOR DELETE)
CLEAROUT C3
CLEAR SYSLOG TYPE-IN AREA
C4
GET ADDR OF CCB FOR SUPERVISOR
D1
IS UNEXPIRED FILE TO BE DELETED
YES
NO
AR B3
SETINTSW
D3
MOVE LNG OF ILLEGAL MSG TO SYSLOG CCW
D4
SVC 0 EXCP
AQ F4
WRITEMSG
E1
RESET TYPEWRITER CCW TO NO-OP
E4
TYPEWRITER STILL BUSY
YES
NO
E5
SVC 7 WAIT FOR I/O COMPLETE
MSG05SW F1
I.S. MSG OR MSG 4N33A SWITCH ON
YES
NO
2
1
F4
RETURN TO CALLING ROUTINE
WRNGPKSW G1
WRONG PACK SW ON
YES
NO
NEXTTEST G2
HAS OPERATOR MOUNTED NEW PACK
NO
YES
2
AR B3
SETINTSW
H1
MOVE A 'B' TO MSG RESPONSE CODE AREA (FOR BYPASS)
J1
EXTENT TO BE BYPASSED
YEC
NO
AR B3
SETINTSW
2

Chart AT. $$BODSMW: Data Security Message Writer (Section 1 of 2)

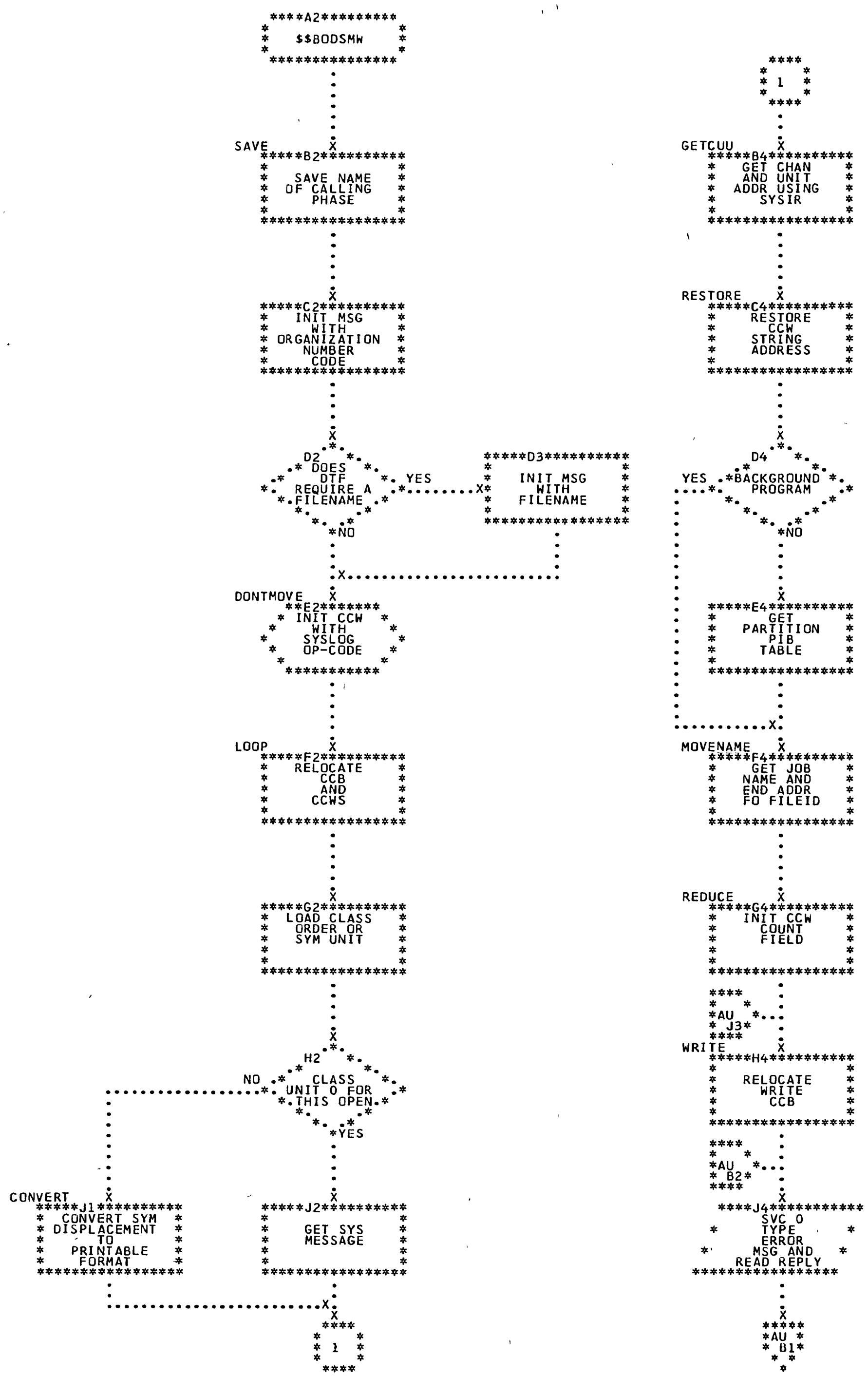
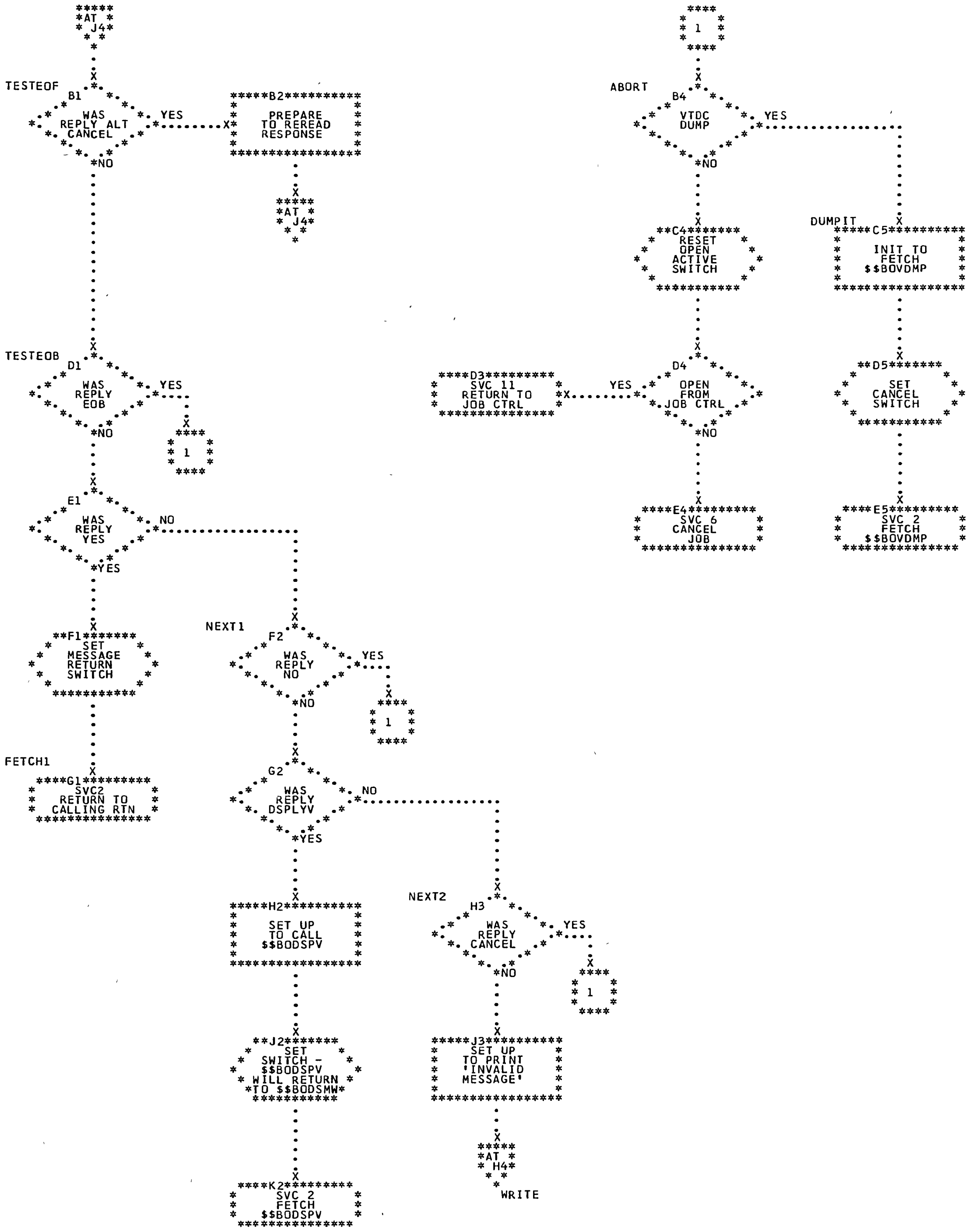

A2
$$BODSMW
SAVE
B2
SAVE NAME OF CALLING PHASE
C2
INIT MSG WITH ORGANIZATION NUMBER CODE
D2
DOES DTF REQUIRE A FILENAME
YES
NO
D3
INIT MSG WITH FILENAME
DONTMOVE
E2
INIT CCW WITH SYSLOG OP-CODE
LOOP
F2
RELOCATE CCB AND CCWS
G2
LOAD CLASS ORDER OR SYM UNIT
H2
CLASS UNIT 0 FOR THIS OPEN
NO
YES
CONVERT
J1
CONVERT SYM DISPLACEMENT TO PRINTABLE FORMAT
J2
GET SYS MESSAGE
1
1
GETCUU
B4
GET CHAN AND UNIT ADDR USING SYSIR
RESTORE
C4
RESTORE CCW STRING ADDRESS
D4
BACKGROUND PROGRAM
YES
NO
E4
GET PARTITION PIB TABLE
MOVENAME
F4
GET JOB NAME AND END ADDR FO FILEID
REDUCE
G4
INIT CCW COUNT FIELD
AU J3
WRITE
H4
RELOCATE WRITE CCB
AU B2
J4
SVC 0 TYPE ERROR MSG AND READ REPLY
AU B1

Chart AU. $$BODSMW: Data Security Message Writer (Section 2 of 2)



## Chart BA. SDMODFI: With Truncation, GET Macro (Section 1 of 5)

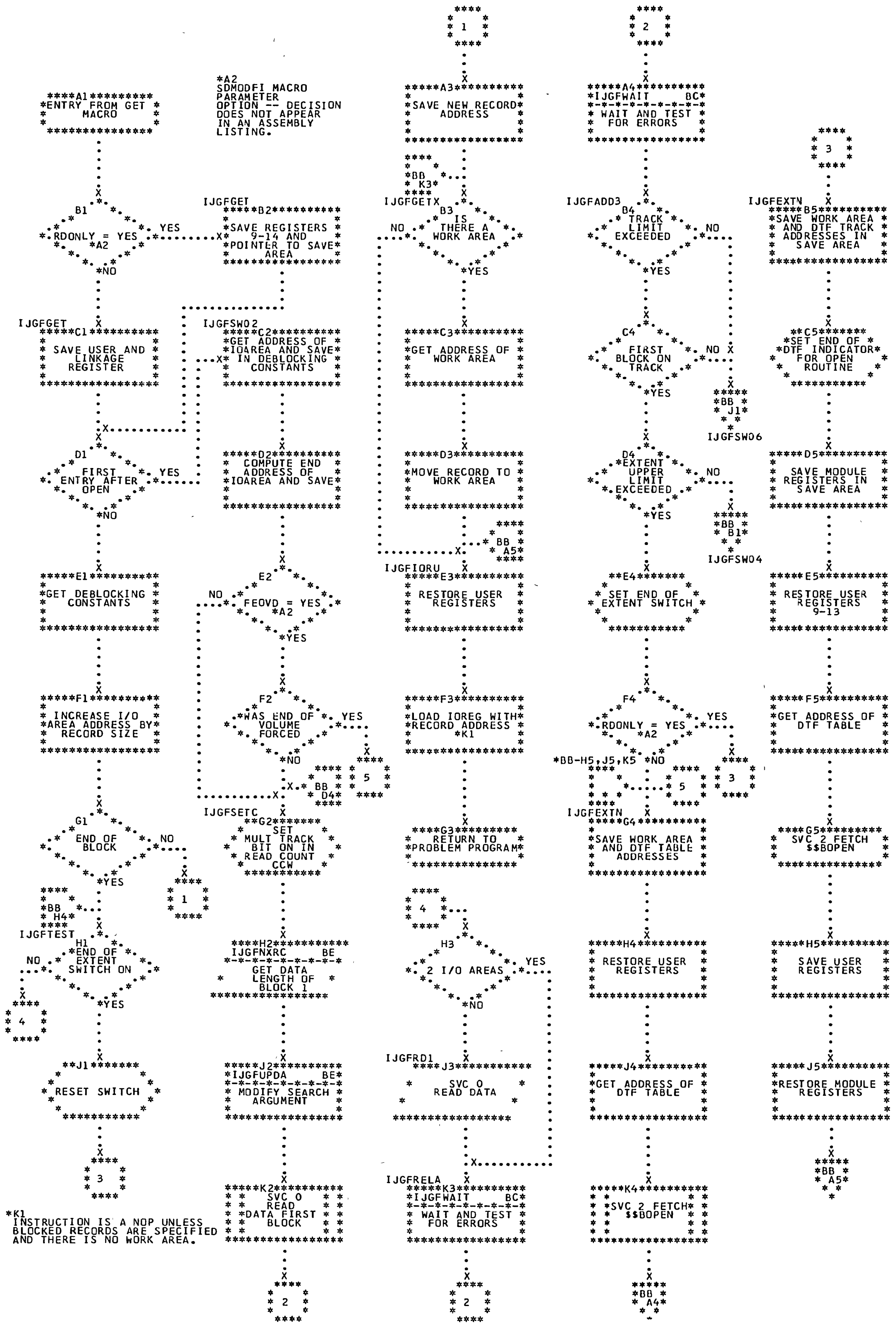

A1 ENTRY FROM GET MACRO
A2 SDMODFI MACRO PARAMETER OPTION -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
1
A3 SAVE NEW RECORD ADDRESS
2
A4 IJGFWAIT BC WAIT AND TEST FOR ERRORS
3
B1 RDONLY = YES *A2
YES
NO
IJGFGET B2 SAVE REGISTERS 9-14 AND POINTER TO SAVE AREA
BB K3
IJGFGETX B3 IS THERE A WORK AREA
NO
YES
IJGFADD3 B4 TRACK LIMIT EXCEEDED
NO
YES
IJGFEXTN B5 SAVE WORK AREA AND DTF TRACK ADDRESSES IN SAVE AREA
IJGFGET C1 SAVE USER AND LINKAGE REGISTER
IJGFSW02 C2 GET ADDRESS OF IOAREA AND SAVE IN DEBLOCKING CONSTANTS
C3 GET ADDRESS OF WORK AREA
C4 FIRST BLOCK ON TRACK
NO
YES
BB J1
IJGFSW06
C5 SET END OF DTF INDICATOR FOR OPEN ROUTINE
D1 FIRST ENTRY AFTER OPEN
YES
NO
D2 COMPUTE END ADDRESS OF IOAREA AND SAVE
D3 MOVE RECORD TO WORK AREA
D4 EXTENT UPPER LIMIT EXCEEDED
NO
YES
BB B1
IJGFSW04
D5 SAVE MODULE REGISTERS IN SAVE AREA
BB A5
E1 GET DEBLOCKING CONSTANTS
E2 FEOVD = YES *A2
NO
YES
IJGFIORU E3 RESTORE USER REGISTERS
E4 SET END OF EXTENT SWITCH
E5 RESTORE USER REGISTERS 9-13
F1 INCREASE I/O AREA ADDRESS BY RECORD SIZE
F2 WAS END OF VOLUME FORCED
YES
NO
5
BB D4
F3 LOAD IOREG WITH RECORD ADDRESS *K1
F4 RDONLY = YES *A2
YES
*BB-H5,J5,K5 NO
5
3
F5 GET ADDRESS OF DTF TABLE
G1 END OF BLOCK
NO
YES
1
IJGFSETC G2 SET MULT TRACK BIT ON IN READ COUNT CCW
G3 RETURN TO PROBLEM PROGRAM
IJGFEXTN G4 SAVE WORK AREA AND DTF TABLE ADDRESSES
G5 SVC 2 FETCH $$BOPEN
BB H4
IJGFTEST H1 END OF EXTENT SWITCH ON
NO
YES
4
H2 IJGFNXRC BE GET DATA LENGTH OF BLOCK 1
4
H3 2 I/O AREAS
YES
NO
H4 RESTORE USER REGISTERS
H5 SAVE USER REGISTERS
J1 RESET SWITCH
J2 IJGFUPDA BE MODIFY SEARCH ARGUMENT
IJGFRD1 J3 SVC 0 READ DATA
J4 GET ADDRESS OF DTF TABLE
J5 RESTORE MODULE REGISTERS
3
BB A5
K2 SVC 0 READ DATA FIRST BLOCK
IJGFRELA K3 IJGFWAIT BC WAIT AND TEST FOR ERRORS
K4 SVC 2 FETCH $$BOPEN
*K1 INSTRUCTION IS A NOP UNLESS BLOCKED RECORDS ARE SPECIFIED AND THERE IS NO WORK AREA.
2
2
BB A4

Chart BB. SDMODFI: With Truncation, GET Macro (Section 2 of 5)

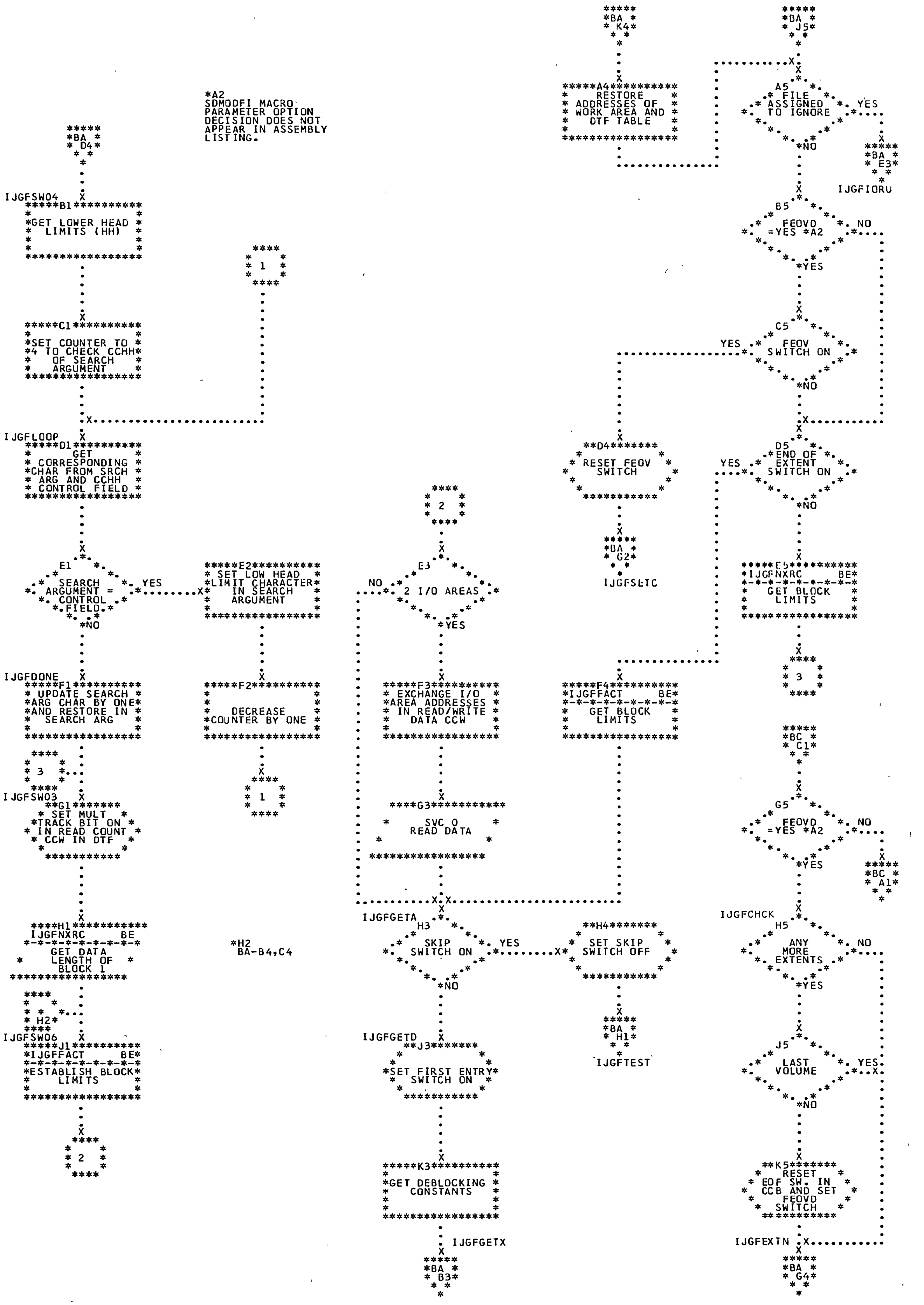
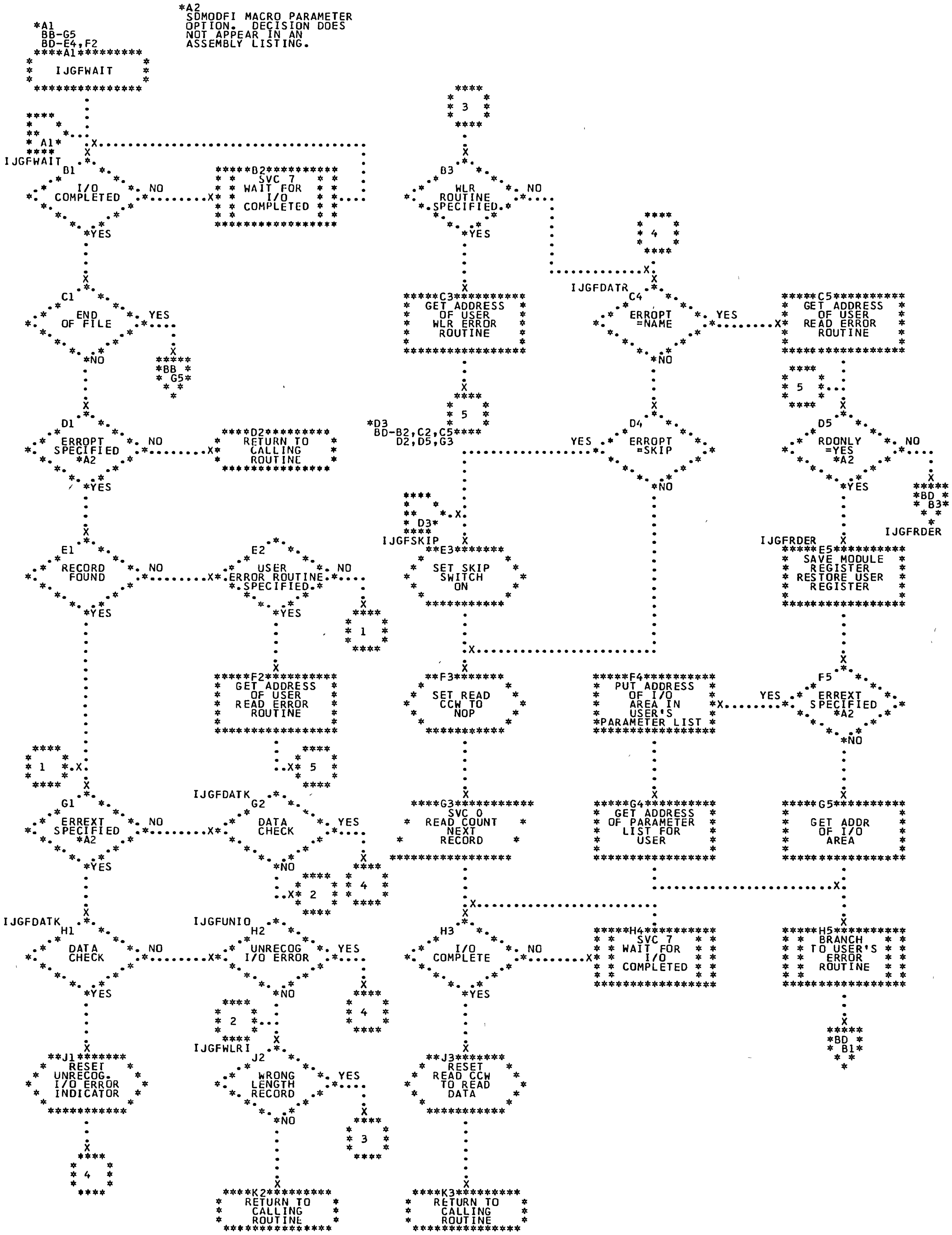

BA D4
IJGFSW04
B1 GET LOWER HEAD LIMITS (HH)
C1 SET COUNTER TO 4 TO CHECK CCHH OF SEARCH ARGUMENT
IJGFLOOP
D1 GET CORRESPONDING CHAR FROM SRCH ARG AND CCHH CONTROL FIELD
E1 SEARCH ARGUMENT = CONTROL FIELD.
YES
NO
IJGFDONE
F1 UPDATE SEARCH ARG CHAR BY ONE AND RESTORE IN SEARCH ARG
3
IJGFSW03
G1 SET MULT TRACK BIT ON IN READ COUNT CCW IN DTF
H1 IJGFNXRC BE GET DATA LENGTH OF BLOCK 1
H2
IJGFSW06
J1 IJGFFACT BE ESTABLISH BLOCK LIMITS
2
*A2 SDMODFI MACRO PARAMETER OPTION DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.
1
E2 SET LOW HEAD LIMIT CHARACTER IN SEARCH ARGUMENT
F2 DECREASE COUNTER BY ONE
1
*H2 BA-B4,C4
2
E3 2 I/O AREAS
NO
YES
F3 EXCHANGE I/O AREA ADDRESSES IN READ/WRITE DATA CCW
G3 SVC 0 READ DATA
IJGFGETA
H3 SKIP SWITCH ON
YES
NO
IJGFGETD
J3 SET FIRST ENTRY SWITCH ON
K3 GET DEBLOCKING CONSTANTS
IJGFGETX
BA B3
BA K4
A4 RESTORE ADDRESSES OF WORK AREA AND DTF TABLE
D4 RESET FEOV SWITCH
BA G2
IJGFSETC
F4 IJGFFACT BE GET BLOCK LIMITS
H4 SET SKIP SWITCH OFF
BA H1
IJGFTEST
BA J5
A5 FILE ASSIGNED TO IGNORE
YES
NO
BA E3
IJGFIORU
B5 FEOVD =YES *A2
NO
YES
C5 FEOV SWITCH ON
YES
NO
D5 END OF EXTENT SWITCH ON
YES
NO
E5 IJGFNXRC BE GET BLOCK LIMITS
3
BC C1
G5 FEOVD =YES *A2
NO
YES
BC A1
IJGFCHCK
H5 ANY MORE EXTENTS
NO
YES
J5 LAST VOLUME
YES
NO
K5 RESET EOF SW. IN CCB AND SET FEOVD SWITCH
IJGFEXTN
BA G4

Chart BC. SDMODFI: With Truncation, GET Macro (Section 3 of 5)



## Chart BD. SDMODFI: With Truncation, GET Macro (Section 4 of 5)

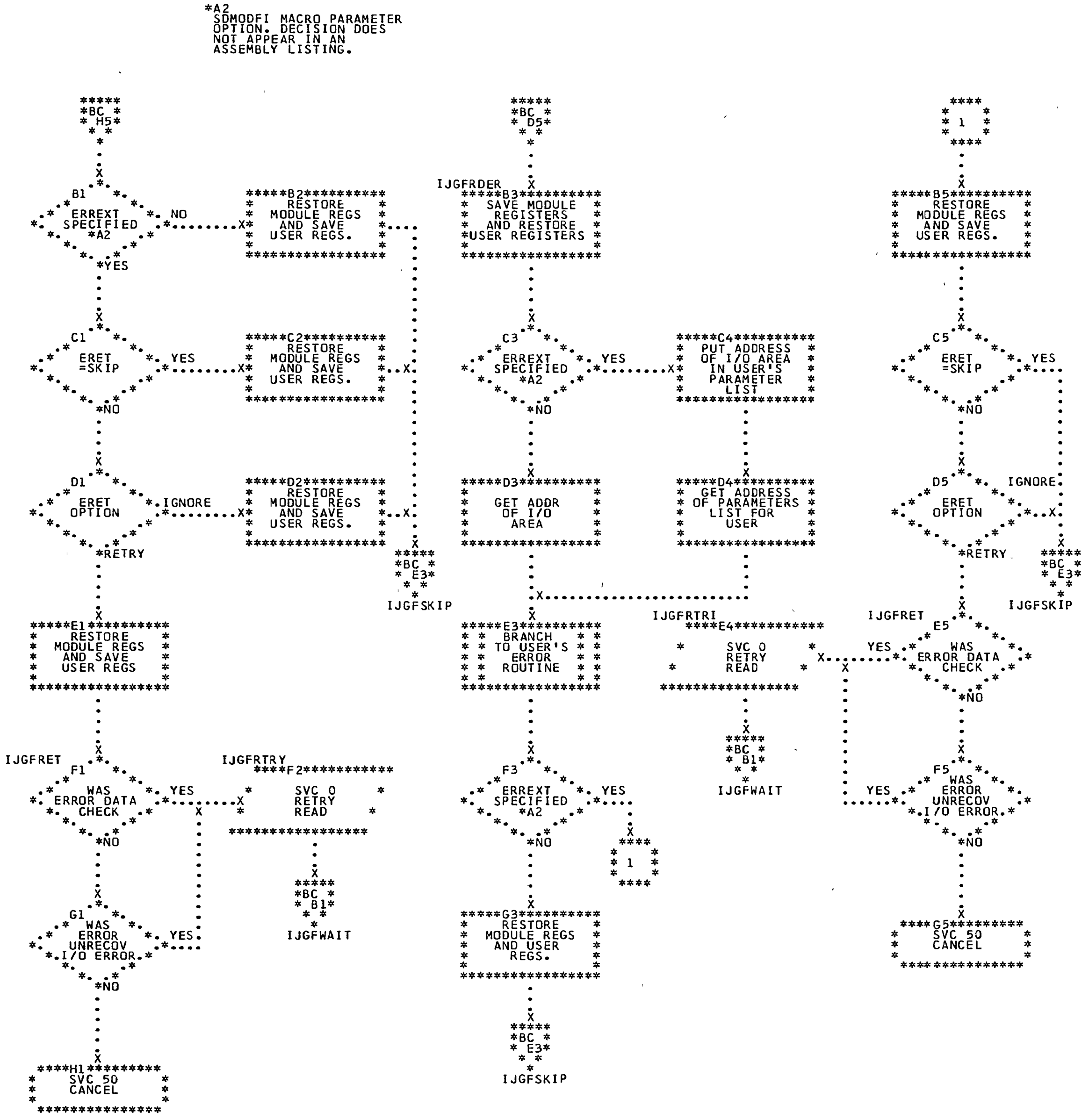
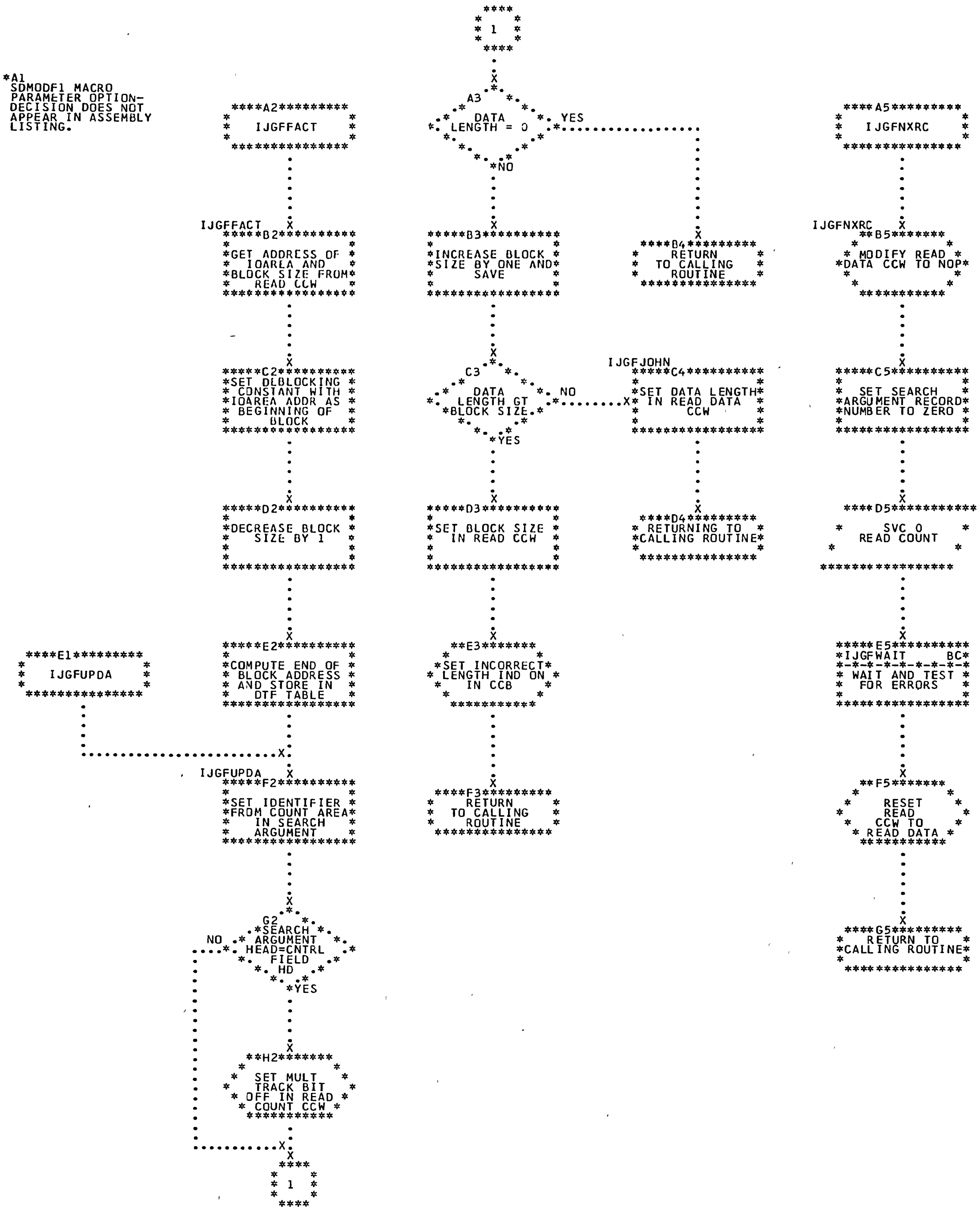

*A2
SDMODFI MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
BC H5
B1 ERREXT SPECIFIED *A2
NO
YES
B2 RESTORE MODULE REGS AND SAVE USER REGS.
BC D5
IJGFRDER
B3 SAVE MODULE REGISTERS AND RESTORE USER REGISTERS
1
B5 RESTORE MODULE REGS AND SAVE USER REGS.
C1 ERET =SKIP
YES
NO
C2 RESTORE MODULE REGS AND SAVE USER REGS.
C3 ERREXT SPECIFIED *A2
YES
NO
C4 PUT ADDRESS OF I/O AREA IN USER'S PARAMETER LIST
C5 ERET =SKIP
YES
NO
D1 ERET OPTION
IGNORE
RETRY
D2 RESTORE MODULE REGS AND SAVE USER REGS.
BC E3
IJGFSKIP
D3 GET ADDR OF I/O AREA
D4 GET ADDRESS OF PARAMETERS LIST FOR USER
D5 ERET OPTION
IGNORE
RETRY
BC E3
IJGFSKIP
E1 RESTORE MODULE REGS AND SAVE USER REGS
E3 BRANCH TO USER'S ERROR ROUTINE
IJGFRTRI
E4 SVC 0 RETRY READ
BC B1
IJGFWAIT
IJGFRET
E5 WAS ERROR DATA CHECK
YES
NO
IJGFRET
F1 WAS ERROR DATA CHECK
YES
NO
IJGFRTRY
F2 SVC 0 RETRY READ
BC B1
IJGFWAIT
F3 ERREXT SPECIFIED *A2
YES
NO
1
F5 WAS ERROR UNRECOV I/O ERROR.
YES
NO
G1 WAS ERROR UNRECOV I/O ERROR.
YES
NO
G3 RESTORE MODULE REGS AND USER REGS.
BC E3
IJGFSKIP
G5 SVC 50 CANCEL
H1 SVC 50 CANCEL

Chart BE. SDMODFI: With Truncation, GET Macro (Section 5 of 5)


*A1 SDMODFI MACRO PARAMETER OPTION-DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.
A2 IJGFFACT
IJGFFACT B2 GET ADDRESS OF IOAREA AND BLOCK SIZE FROM READ CCW
C2 SET DEBLOCKING CONSTANT WITH IOAREA ADDR AS BEGINNING OF BLOCK
D2 DECREASE BLOCK SIZE BY 1
E1 IJGFUPDA
E2 COMPUTE END OF BLOCK ADDRESS AND STORE IN DTF TABLE
IJGFUPDA F2 SET IDENTIFIER FROM COUNT AREA IN SEARCH ARGUMENT
G2 SEARCH ARGUMENT HEAD=CNTRL FIELD HD
NO
YES
H2 SET MULT TRACK BIT OFF IN READ COUNT CCW
1
1
A3 DATA LENGTH = 0
YES
NO
B3 INCREASE BLOCK SIZE BY ONE AND SAVE
C3 DATA LENGTH GT BLOCK SIZE
NO
YES
D3 SET BLOCK SIZE IN READ CCW
E3 SET INCORRECT LENGTH IND ON IN CCB
F3 RETURN TO CALLING ROUTINE
B4 RETURN TO CALLING ROUTINE
IJGFJOHN C4 SET DATA LENGTH IN READ DATA CCW
D4 RETURNING TO CALLING ROUTINE
A5 IJGFNXRC
IJGFNXRC B5 MODIFY READ DATA CCW TO NOP
C5 SET SEARCH ARGUMENT RECORD NUMBER TO ZERO
D5 SVC 0 READ COUNT
E5 IJGFWAIT BC WAIT AND TEST FOR ERRORS
F5 RESET READ CCW TO READ DATA
G5 RETURN TO CALLING ROUTINE

## Chart BF. SDMODFI: Without Truncation, GET Macro (Section 1 of 4)

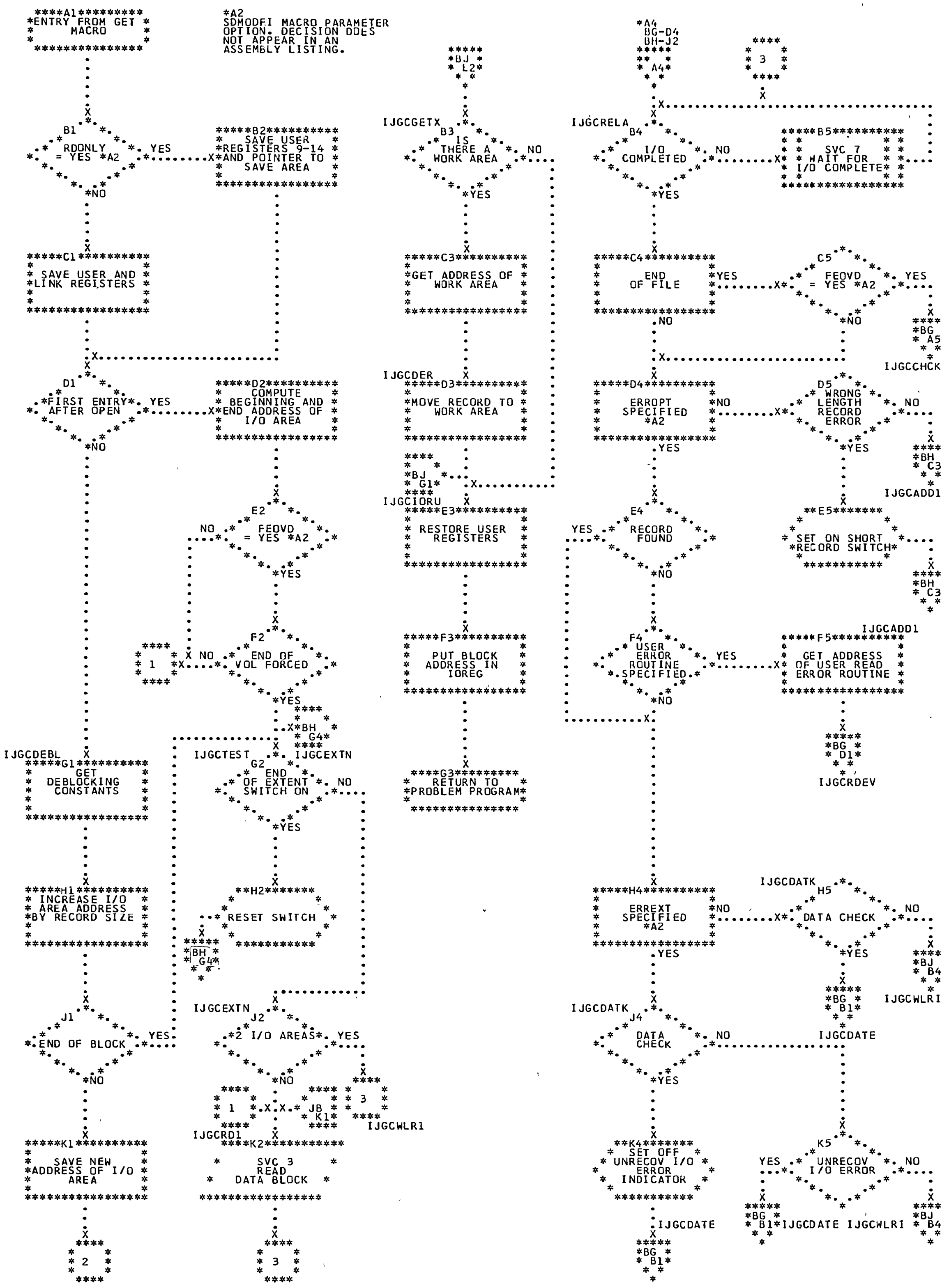
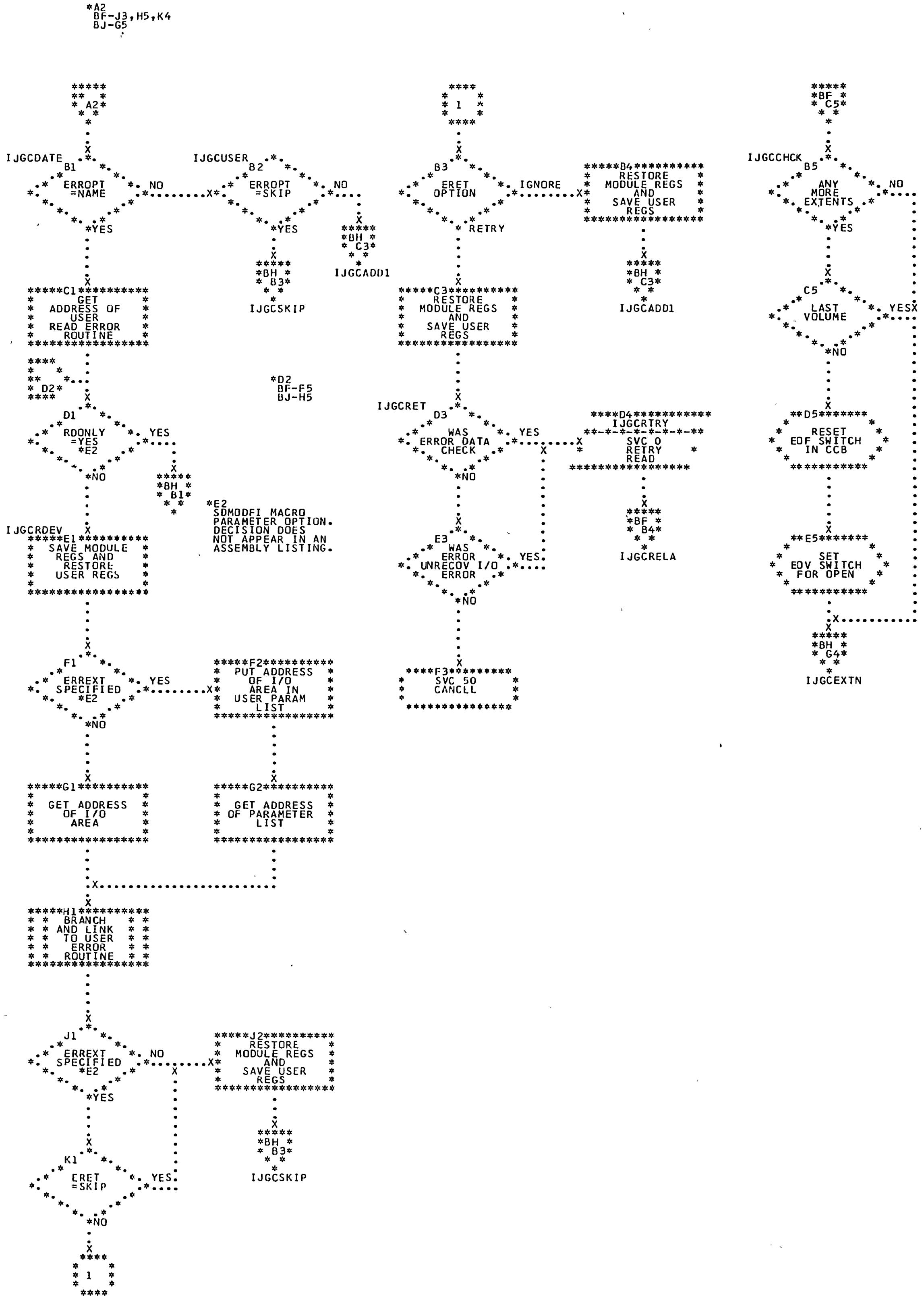


Chart BG. SDMODFI: Without Truncation, GET Macro (Section 2 of 4)



*A2 BF-J3,H5,K4 BJ-G5
A2
IJGCDATE
B1 ERROPT =NAME
NO
YES
IJGCUSER
B2 ERROPT =SKIP
NO
YES
BH B3
IJGCSKIP
BH C3
IJGCADD1
C1 GET ADDRESS OF USER READ ERROR ROUTINE
D2
D1 RDONLY =YES *E2
YES
NO
BH B1
*D2 BF-F5 BJ-H5
*E2 SDMODFI MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
IJGCRDEV
E1 SAVE MODULE REGS AND RESTORE USER REGS
F1 ERREXT SPECIFIED *E2
YES
NO
F2 PUT ADDRESS OF I/O AREA IN USER PARAM LIST
G1 GET ADDRESS OF I/O AREA
G2 GET ADDRESS OF PARAMETER LIST
H1 BRANCH AND LINK TO USER ERROR ROUTINE
J1 ERREXT SPECIFIED *E2
NO
YES
J2 RESTORE MODULE REGS AND SAVE USER REGS
BH B3
IJGCSKIP
K1 ERET =SKIP
YES
NO
1
1
B3 ERET OPTION
IGNORE
RETRY
B4 RESTORE MODULE REGS AND SAVE USER REGS
BH C3
IJGCADD1
C3 RESTORE MODULE REGS AND SAVE USER REGS
IJGCRET
D3 WAS ERROR DATA CHECK
YES
NO
D4 IJGCRTRY SVC 0 RETRY READ
BF B4
IJGCRELA
E3 WAS UNRECOV I/O ERROR
YES
NO
F3 SVC 50 CANCLL
BF C5
IJGCCHCK
B5 ANY MORE EXTENTS
NO
YES
C5 LAST VOLUME
YES
NO
D5 RESET EOF SWITCH IN CCB
E5 SET EOV SWITCH FOR OPEN
BH G4
IJGCEXTN

Chart BH. SDMODFI: Without Truncation, GET Macro (Section 3 of 4)

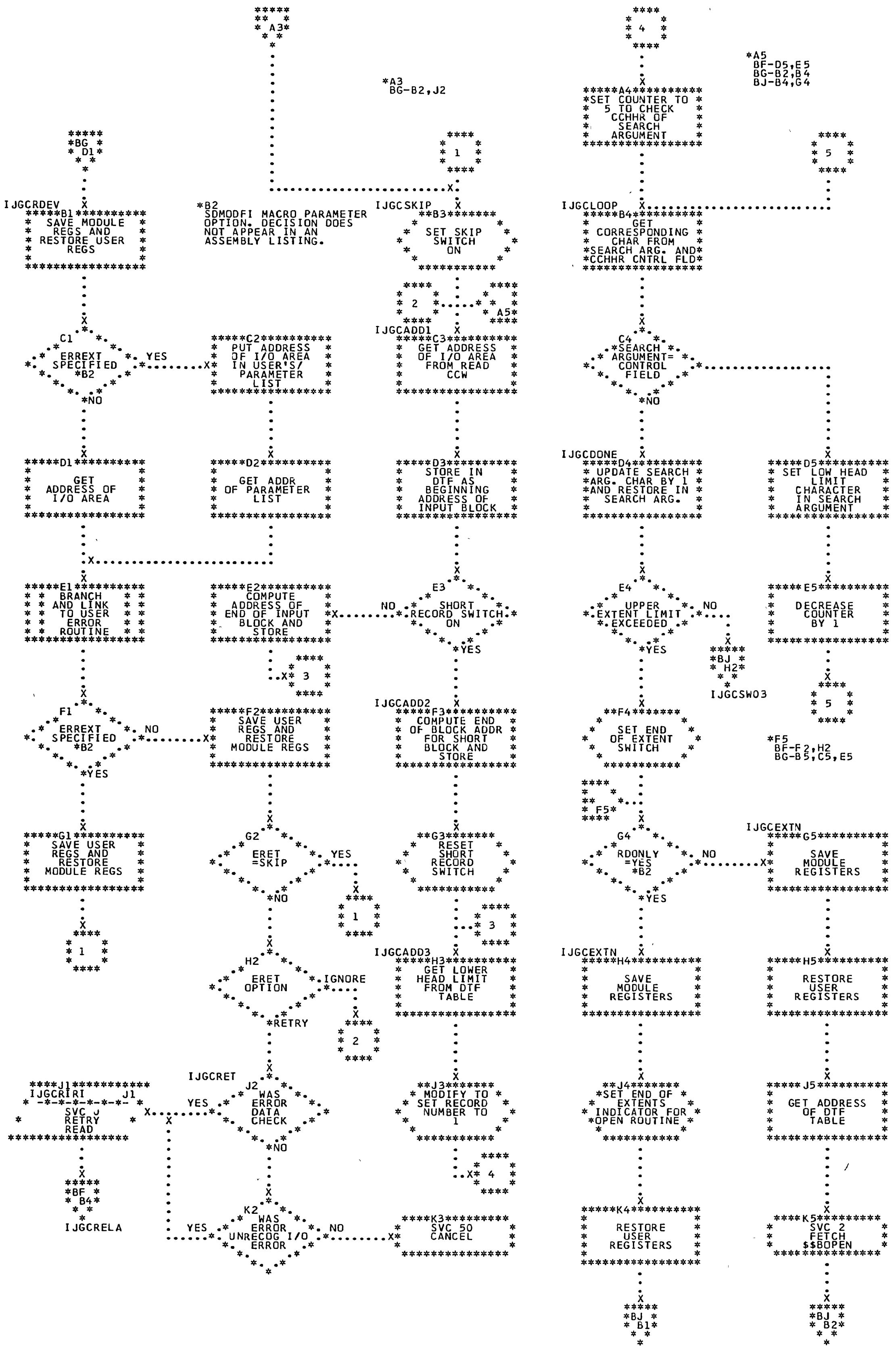
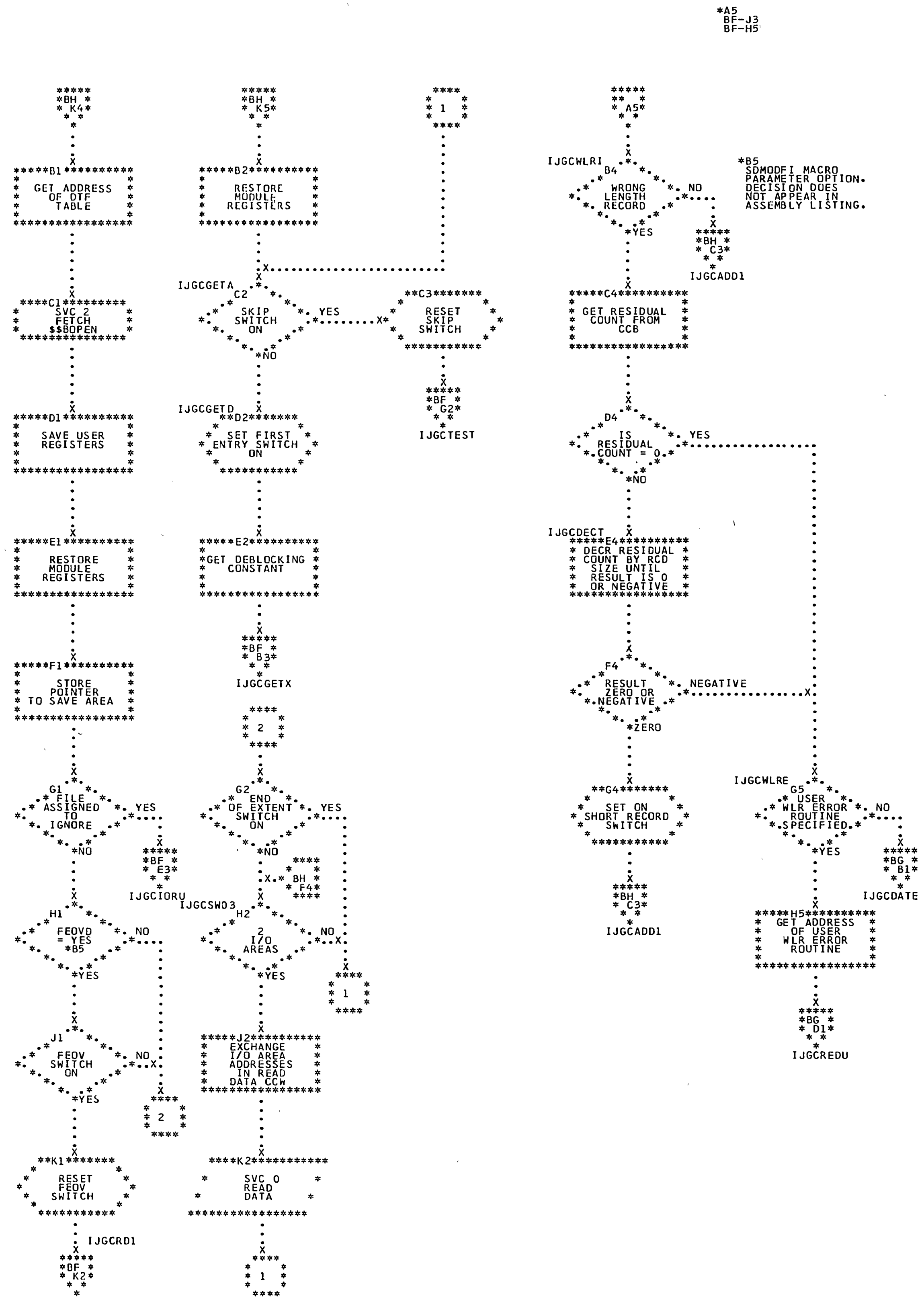

A3
*A3 BG-B2,J2
1
4
A4 SET COUNTER TO 5 TO CHECK CCHHR OF SEARCH ARGUMENT
*A5 BF-D5,E5 BG-B2,B4 BJ-B4,G4
5
BG D1
IJGCRDEV
B1 SAVE MODULE REGS AND RESTORE USER REGS
*B2 SDMODFI MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
IJGCSKIP
B3 SET SKIP SWITCH ON
IJGCLOOP
B4 GET CORRESPONDING CHAR FROM SEARCH ARG. AND CCHHR CNTRL FLD
2
A5
C1 ERREXT SPECIFIED *B2
YES
NO
C2 PUT ADDRESS OF I/O AREA IN USER'S/ PARAMETER LIST
IJGCADD1
C3 GET ADDRESS OF I/O AREA FROM READ CCW
C4 SEARCH ARGUMENT= CONTROL FIELD
NO
D1 GET ADDRESS OF I/O AREA
D2 GET ADDR OF PARAMETER LIST
D3 STORE IN DTF AS BEGINNING ADDRESS OF INPUT BLOCK
IJGCDONE
D4 UPDATE SEARCH ARG. CHAR BY 1 AND RESTORE IN SEARCH ARG.
D5 SET LOW HEAD LIMIT CHARACTER IN SEARCH ARGUMENT
E1 BRANCH AND LINK TO USER ERROR ROUTINE
E2 COMPUTE ADDRESS OF END OF INPUT BLOCK AND STORE
E3 SHORT RECORD SWITCH ON
NO
YES
E4 UPPER EXTENT LIMIT EXCEEDED
NO
YES
BJ H2
IJGCSW03
E5 DECREASE COUNTER BY 1
3
5
F1 ERREXT SPECIFIED *B2
NO
YES
F2 SAVE USER REGS AND RESTORE MODULE REGS
IJGCADD2
F3 COMPUTE END OF BLOCK ADDR FOR SHORT BLOCK AND STORE
F4 SET END OF EXTENT SWITCH
*F5 BF-F2,H2 BG-B5,C5,E5
F5
G1 SAVE USER REGS AND RESTORE MODULE REGS
G2 ERET =SKIP
YES
NO
G3 RESET SHORT RECORD SWITCH
G4 RDONLY =YES *B2
NO
YES
IJGCEXTN
G5 SAVE MODULE REGISTERS
1
1
3
H2 ERET OPTION
IGNORE
RETRY
2
IJGCADD3
H3 GET LOWER HEAD LIMIT FROM DTF TABLE
IJGCEXTN
H4 SAVE MODULE REGISTERS
H5 RESTORE USER REGISTERS
J1 IJGCRIRI J1 SVC J RETRY READ
IJGCRET
J2 WAS ERROR DATA CHECK
YES
NO
J3 MODIFY TO SET RECORD NUMBER TO 1
4
J4 SET END OF EXTENTS INDICATOR FOR OPEN ROUTINE
J5 GET ADDRESS OF DTF TABLE
BF B4
IJGCRELA
K2 WAS ERROR UNRECOG I/O ERROR
YES
NO
K3 SVC 50 CANCEL
K4 RESTORE USER REGISTERS
K5 SVC 2 FETCH $$BOPEN
BJ B1
BJ B2

Chart BJ. SDMODFI: Without Truncation, GET Macro (Section 4 of 4)



*A5
BF-J3
BF-H5
BH K4
B1 GET ADDRESS OF DTF TABLE
C1 SVC 2 FETCH $$BOPEN
D1 SAVE USER REGISTERS
E1 RESTORE MODULE REGISTERS
F1 STORE POINTER TO SAVE AREA
G1 FILE ASSIGNED TO IGNORE
YES
NO
BF E3
IJGCIORU
H1 FEOVD = YES *B5
NO
YES
J1 FEOV SWITCH ON
NO
YES
2
K1 RESET FEOV SWITCH
IJGCRD1
BF K2
BH K5
B2 RESTORE MODULE REGISTERS
1
IJGCGETA
C2 SKIP SWITCH ON
YES
NO
C3 RESET SKIP SWITCH
BF G2
IJGCTEST
IJGCGETD
D2 SET FIRST ENTRY SWITCH ON
E2 GET DEBLOCKING CONSTANT
BF B3
IJGCGETX
2
G2 END OF EXTENT SWITCH ON
YES
NO
BH F4
IJGCSW03
H2 2 I/O AREAS
NO
YES
1
J2 EXCHANGE I/O AREA ADDRESSES IN READ DATA CCW
K2 SVC 0 READ DATA
1
A5
IJGCWLRI
B4 WRONG LENGTH RECORD
NO
YES
BH C3
IJGCADD1
*B5
SDMODFI MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.
C4 GET RESIDUAL COUNT FROM CCB
D4 IS RESIDUAL COUNT = 0
YES
NO
IJGCDECT
E4 DECR RESIDUAL COUNT BY RCD SIZE UNTIL RESULT IS 0 OR NEGATIVE
F4 RESULT ZERO OR NEGATIVE
NEGATIVE
ZERO
G4 SET ON SHORT RECORD SWITCH
BH C3
IJGCADD1
IJGCWLRE
G5 USER WLR ERROR ROUTINE SPECIFIED
NO
YES
BG B1
IJGCDATE
H5 GET ADDRESS OF USER WLR ERROR ROUTINE
BG D1
IJGCREDU

Chart BK. SDMODFO: With Truncation, PUT Macro (Section 1 of 5)

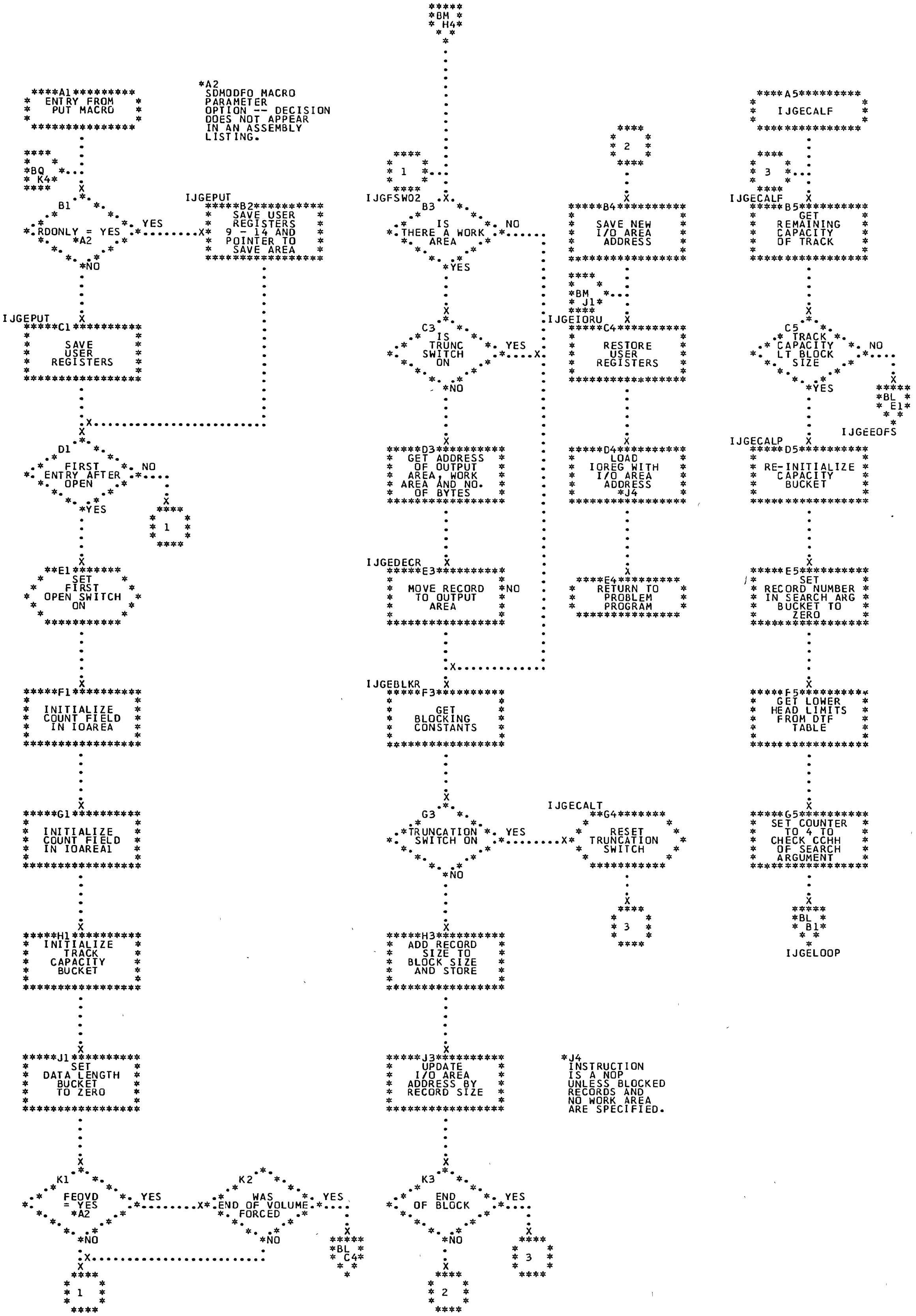

A1 ENTRY FROM PUT MACRO
*A2 SDMODFO MACRO PARAMETER OPTION -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
BQ K4
B1 RDONLY = YES *A2
YES
NO
IJGEPUT
B2 SAVE USER REGISTERS 9 - 14 AND POINTER TO SAVE AREA
IJGEPUT
C1 SAVE USER REGISTERS
D1 FIRST ENTRY AFTER OPEN
NO
YES
1
E1 SET FIRST OPEN SWITCH ON
F1 INITIALIZE COUNT FIELD IN IOAREA
G1 INITIALIZE COUNT FIELD IN IOAREA1
H1 INITIALIZE TRACK CAPACITY BUCKET
J1 SET DATA LENGTH BUCKET TO ZERO
K1 FEOVD = YES *A2
YES
NO
1
K2 WAS END OF VOLUME FORCED
YES
NO
BL C4
BM H4
1
IJGFSW02
B3 IS THERE A WORK AREA
NO
YES
C3 IS TRUNC SWITCH ON
YES
NO
D3 GET ADDRESS OF OUTPUT AREA, WORK AREA AND NO. OF BYTES
IJGEDECR
E3 MOVE RECORD TO OUTPUT AREA
*NO
IJGEBLKR
F3 GET BLOCKING CONSTANTS
G3 TRUNCATION SWITCH ON
YES
NO
IJGECALT
G4 RESET TRUNCATION SWITCH
3
H3 ADD RECORD SIZE TO BLOCK SIZE AND STORE
J3 UPDATE I/O AREA ADDRESS BY RECORD SIZE
*J4 INSTRUCTION IS A NOP UNLESS BLOCKED RECORDS AND NO WORK AREA ARE SPECIFIED.
K3 END OF BLOCK
YES
NO
3
2
2
B4 SAVE NEW I/O AREA ADDRESS
BM J1
IJGEIORU
C4 RESTORE USER REGISTERS
D4 LOAD IOREG WITH I/O AREA ADDRESS *J4
E4 RETURN TO PROBLEM PROGRAM
A5 IJGECALF
3
IJGECALF
B5 GET REMAINING CAPACITY OF TRACK
C5 TRACK CAPACITY LT BLOCK SIZE
NO
YES
BL E1
IJGEEOFS
IJGECALP
D5 RE-INITIALIZE CAPACITY BUCKET
E5 SET RECORD NUMBER IN SEARCH ARG BUCKET TO ZERO
F5 GET LOWER HEAD LIMITS FROM DTF TABLE
G5 SET COUNTER TO 4 TO CHECK CCHH OF SEARCH ARGUMENT
BL B1
IJGELOOP

Chart BL. SDMODFO: With Truncation, PUT Macro (Section 2 of 5)

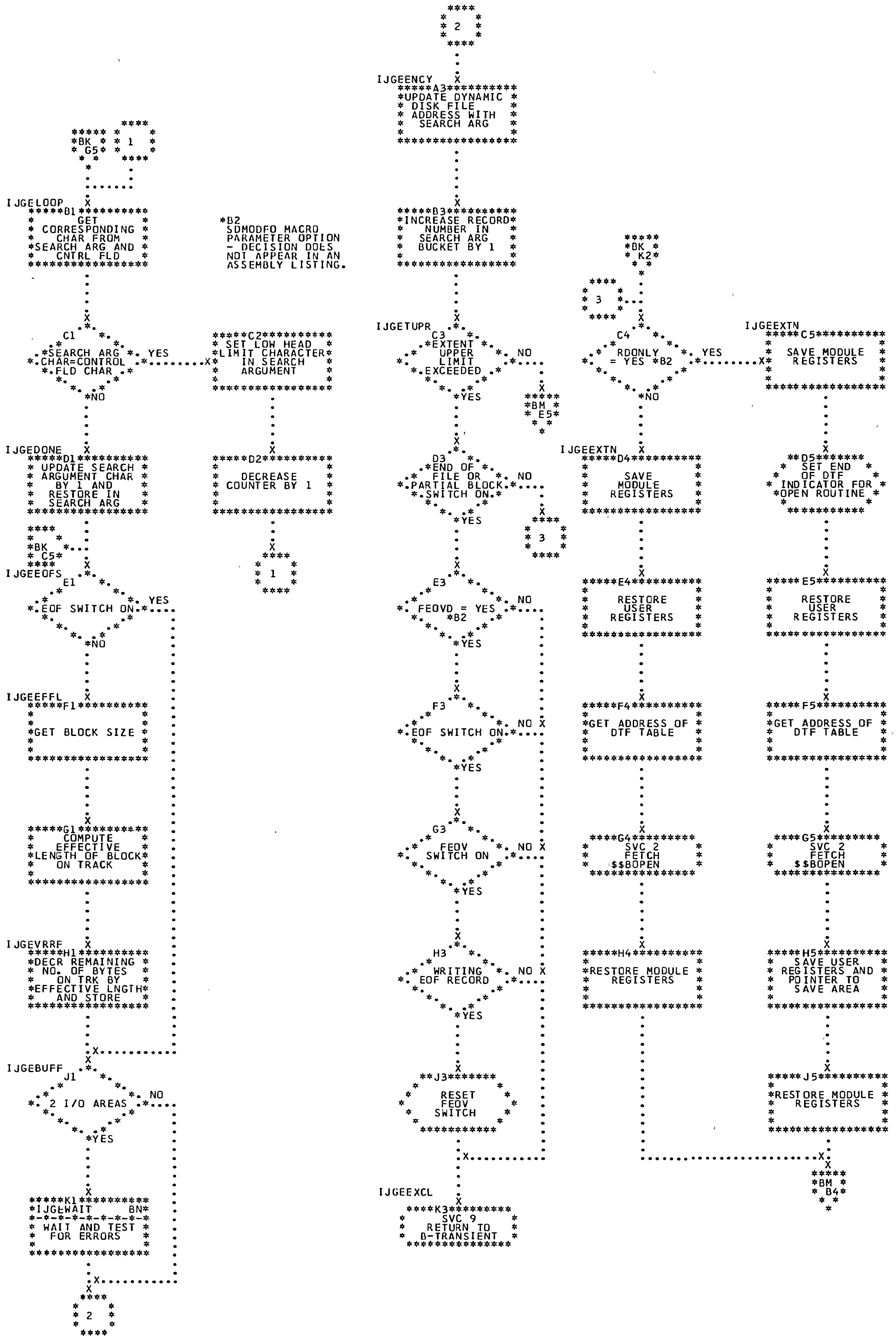
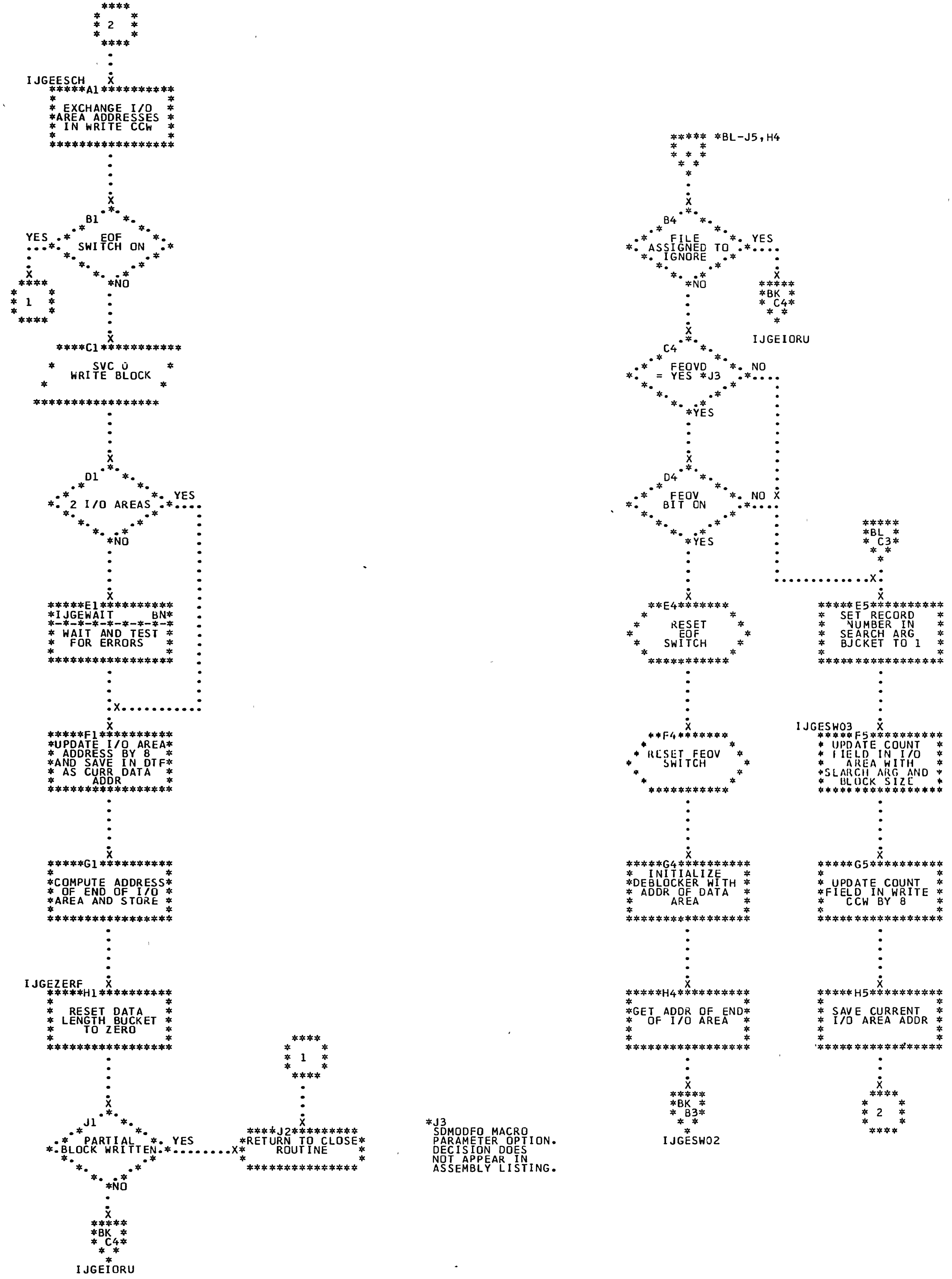
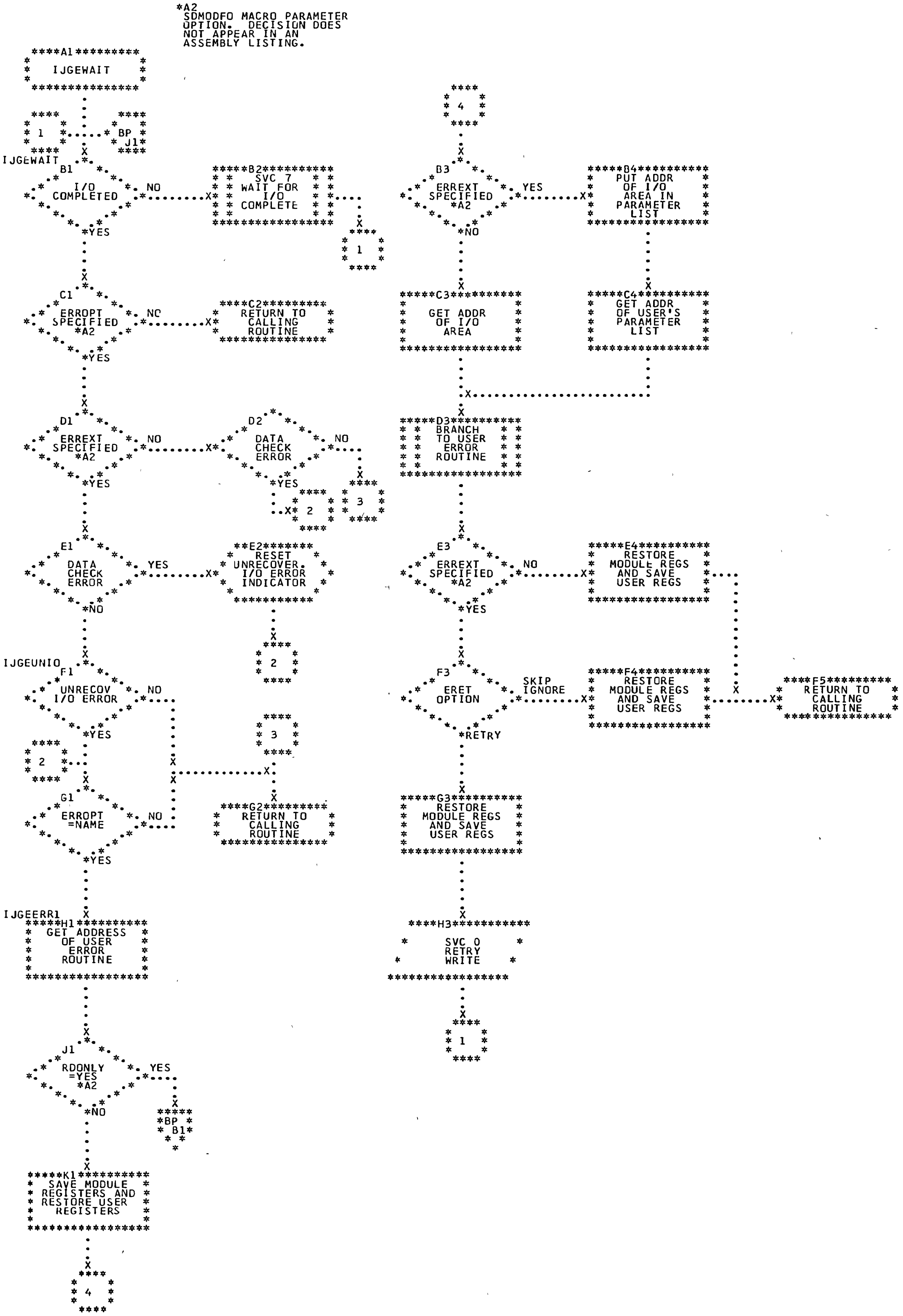

2
IJGEENCY
A3
UPDATE DYNAMIC DISK FILE ADDRESS WITH SEARCH ARG
B3
INCREASE RECORD NUMBER IN SEARCH ARG BUCKET BY 1
IJGETUPR
C3
EXTENT UPPER LIMIT EXCEEDED
NO
BM E5
YES
D3
END OF FILE OR PARTIAL BLOCK SWITCH ON
NO
3
YES
E3
FEOVD = YES *B2
NO
YES
F3
EOF SWITCH ON
NO
YES
G3
FEOV SWITCH ON
NO
YES
H3
WRITING EOF RECORD
NO
YES
J3
RESET FEOV SWITCH
IJGEEXCL
K3
SVC 9 RETURN TO B-TRANSIENT
BK G5
1
IJGELOOP
B1
GET CORRESPONDING CHAR FROM SEARCH ARG AND CNTRL FLD
*B2
SDMODFO MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
C1
SEARCH ARG CHAR=CONTROL FLD CHAR
YES
NO
C2
SET LOW HEAD LIMIT CHARACTER IN SEARCH ARGUMENT
IJGEDONE
D1
UPDATE SEARCH ARGUMENT CHAR BY 1 AND RESTORE IN SEARCH ARG
D2
DECREASE COUNTER BY 1
1
BK C5
IJGEEOFS
E1
EOF SWITCH ON
YES
NO
IJGEEFFL
F1
GET BLOCK SIZE
G1
COMPUTE EFFECTIVE LENGTH OF BLOCK ON TRACK
IJGEVRRF
H1
DECR REMAINING NO. OF BYTES ON TRK BY EFFECTIVE LNGTH AND STORE
IJGEBUFF
J1
2 I/O AREAS
NO
YES
K1
IJGEWAIT BN
WAIT AND TEST FOR ERRORS
2
BK K2
3
C4
RDONLY = YES *B2
YES
NO
IJGEEXTN
D4
SAVE MODULE REGISTERS
E4
RESTORE USER REGISTERS
F4
GET ADDRESS OF DTF TABLE
G4
SVC 2 FETCH $$BOPEN
H4
RESTORE MODULE REGISTERS
IJGEEXTN
C5
SAVE MODULE REGISTERS
D5
SET END OF DTF INDICATOR FOR OPEN ROUTINE
E5
RESTORE USER REGISTERS
F5
GET ADDRESS OF DTF TABLE
G5
SVC 2 FETCH $$BOPEN
H5
SAVE USER REGISTERS AND POINTER TO SAVE AREA
J5
RESTORE MODULE REGISTERS
BM B4

Chart BM. SDMODFO: With Truncation, PUT Macro (Section 3 of 5)



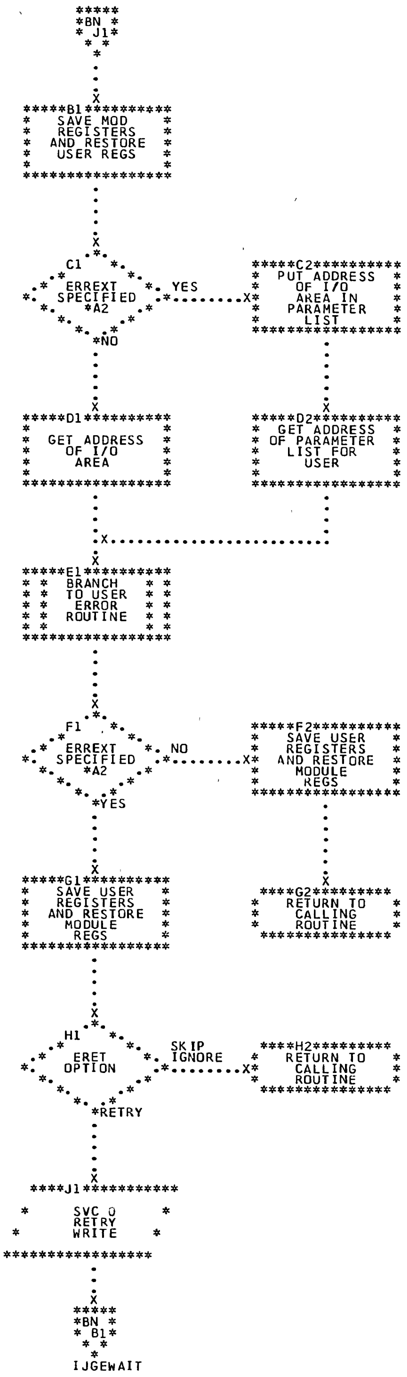
2
IJGEESCH
A1
EXCHANGE I/O AREA ADDRESSES IN WRITE CCW
B1
EOF SWITCH ON
YES
NO
1
C1
SVC 0 WRITE BLOCK
D1
2 I/O AREAS
YES
NO
E1
IJGEWAIT BN
WAIT AND TEST FOR ERRORS
F1
UPDATE I/O AREA ADDRESS BY 8 AND SAVE IN DTF AS CURR DATA ADDR
G1
COMPUTE ADDRESS OF END OF I/O AREA AND STORE
IJGEZERF
H1
RESET DATA LENGTH BUCKET TO ZERO
J1
PARTIAL BLOCK WRITTEN
YES
NO
BK C4
IJGEIORU
1
J2
RETURN TO CLOSE ROUTINE
*J3
SDMODFO MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.
*BL-J5,H4
B4
FILE ASSIGNED TO IGNORE
YES
NO
BK C4
IJGEIORU
C4
FEOVD = YES *J3
NO
YES
D4
FEOV BIT ON
NO
YES
BL C3
E4
RESET EOF SWITCH
E5
SET RECORD NUMBER IN SEARCH ARG BUCKET TO 1
F4
RESET FEOV SWITCH
IJGESW03
F5
UPDATE COUNT FIELD IN I/O AREA WITH SEARCH ARG AND BLOCK SIZE
G4
INITIALIZE DEBLOCKER WITH ADDR OF DATA AREA
G5
UPDATE COUNT FIELD IN WRITE CCW BY 8
H4
GET ADDR OF END OF I/O AREA
H5
SAVE CURRENT I/O AREA ADDR
BK B3
IJGESW02
2

Chart BN. SDMODFO: With Truncation, PUT Macro (Section 4 of 5)


*A2
SDMODFO MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
A1
IJGEWAIT
1
BP J1
IJGEWAIT
B1
I/O COMPLETED
NO
YES
B2
SVC 7 WAIT FOR I/O COMPLETE
1
C1
ERROPT SPECIFIED *A2
NO
YES
C2
RETURN TO CALLING ROUTINE
D1
ERREXT SPECIFIED *A2
NO
YES
D2
DATA CHECK ERROR
NO
YES
2
3
E1
DATA CHECK ERROR
YES
NO
E2
RESET UNRECOVER. I/O ERROR INDICATOR
2
IJGEUNIO
F1
UNRECOV I/O ERROR
NO
YES
3
2
G1
ERROPT =NAME
NO
YES
G2
RETURN TO CALLING ROUTINE
IJGEERR1
H1
GET ADDRESS OF USER ERROR ROUTINE
J1
RDONLY =YES *A2
YES
NO
BP B1
K1
SAVE MODULE REGISTERS AND RESTORE USER REGISTERS
4
4
B3
ERREXT SPECIFIED *A2
YES
NO
B4
PUT ADDR OF I/O AREA IN PARAMETER LIST
C3
GET ADDR OF I/O AREA
C4
GET ADDR OF USER'S PARAMETER LIST
D3
BRANCH TO USER ERROR ROUTINE
E3
ERREXT SPECIFIED *A2
NO
YES
E4
RESTORE MODULE REGS AND SAVE USER REGS
F3
ERET OPTION
SKIP IGNORE
RETRY
F4
RESTORE MODULE REGS AND SAVE USER REGS
F5
RETURN TO CALLING ROUTINE
G3
RESTORE MODULE REGS AND SAVE USER REGS
H3
SVC 0 RETRY WRITE
1

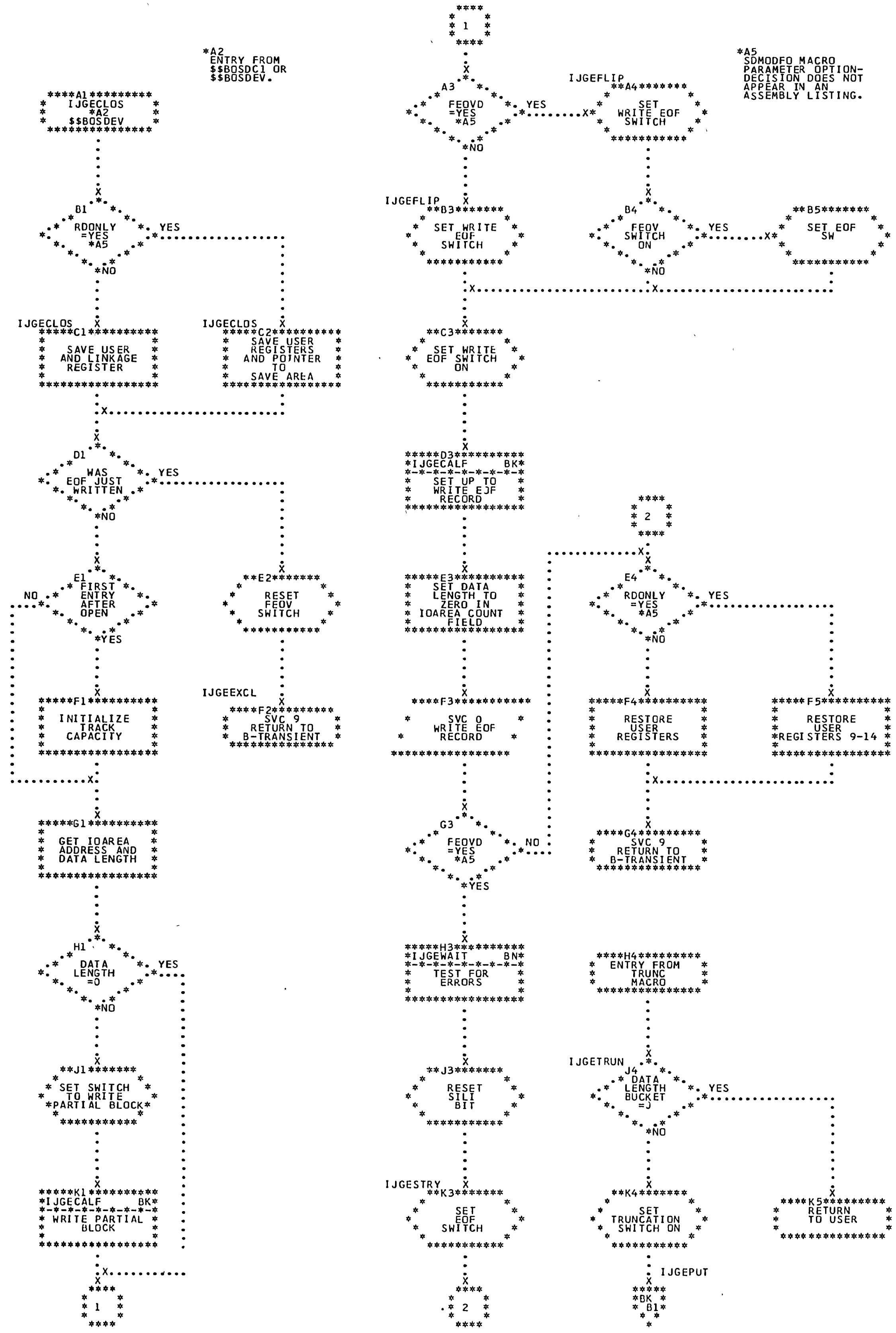
## Chart BP. SDMODFO: With Truncation, PUT Macro (Section 5 of 5)

*A2
SDMODFO MACRO PARAMETER
OPTION. DECISION DOES
NOT APPEAR IN AN
ASSEMBLY LISTING.



BN J1
B1 SAVE MOD REGISTERS AND RESTORE USER REGS
C1 ERREXT SPECIFIED *A2
YES
NO
C2 PUT ADDRESS OF I/O AREA IN PARAMETER LIST
D1 GET ADDRESS OF I/O AREA
D2 GET ADDRESS OF PARAMETER LIST FOR USER
E1 BRANCH TO USER ERROR ROUTINE
F1 ERREXT SPECIFIED *A2
NO
YES
F2 SAVE USER REGISTERS AND RESTORE MODULE REGS
G1 SAVE USER REGISTERS AND RESTORE MODULE REGS
G2 RETURN TO CALLING ROUTINE
H1 ERET OPTION
SKIP IGNORE
RETRY
H2 RETURN TO CALLING ROUTINE
J1 SVC 0 RETRY WRITE
BN B1
IJGEWAIT

# Chart BQ. SDMODFO: With Truncation, Close Routine, TRUNC Macro


*A2 ENTRY FROM $$BOSDC1 OR $$BOSDEV.
A1 IJGECLOS *A2 $$BOSDEV
B1 RDONLY =YES *A5
YES
NO
IJGECLOS C1 SAVE USER AND LINKAGE REGISTER
IJGECLOS C2 SAVE USER REGISTERS AND POINTER TO SAVE AREA
D1 WAS EOF JUST WRITTEN
YES
NO
E1 FIRST ENTRY AFTER OPEN
NO
YES
E2 RESET FEOV SWITCH
F1 INITIALIZE TRACK CAPACITY
IJGEEXCL F2 SVC 9 RETURN TO B-TRANSIENT
G1 GET IOAREA ADDRESS AND DATA LENGTH
H1 DATA LENGTH =0
YES
NO
J1 SET SWITCH TO WRITE PARTIAL BLOCK
K1 IJGECALF BK WRITE PARTIAL BLOCK
1
1
A3 FEOVD =YES *A5
YES
NO
IJGEFLIP A4 SET WRITE EOF SWITCH
*A5 SDMODFO MACRO PARAMETER OPTION-DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
IJGEFLIP B3 SET WRITE EOF SWITCH
B4 FEOV SWITCH ON
YES
NO
B5 SET EOF SW
C3 SET WRITE EOF SWITCH ON
D3 IJGECALF BK SET UP TO WRITE EOF RECORD
E3 SET DATA LENGTH TO ZERO IN IOAREA COUNT FIELD
F3 SVC 0 WRITE EOF RECORD
G3 FEOVD =YES *A5
NO
YES
H3 IJGEWAIT BN TEST FOR ERRORS
J3 RESET SILI BIT
IJGESTRY K3 SET EOF SWITCH
2
2
E4 RDONLY =YES *A5
YES
NO
F4 RESTORE USER REGISTERS
F5 RESTORE USER REGISTERS 9-14
G4 SVC 9 RETURN TO B-TRANSIENT
H4 ENTRY FROM TRUNC MACRO
IJGETRUN J4 DATA LENGTH BUCKET =J
YES
NO
K4 SET TRUNCATION SWITCH ON
K5 RETURN TO USER
IJGEPUT
BK B1

## Chart BR. SDMODFO: Without Truncation, PUT Macro (Section 1 of 4)

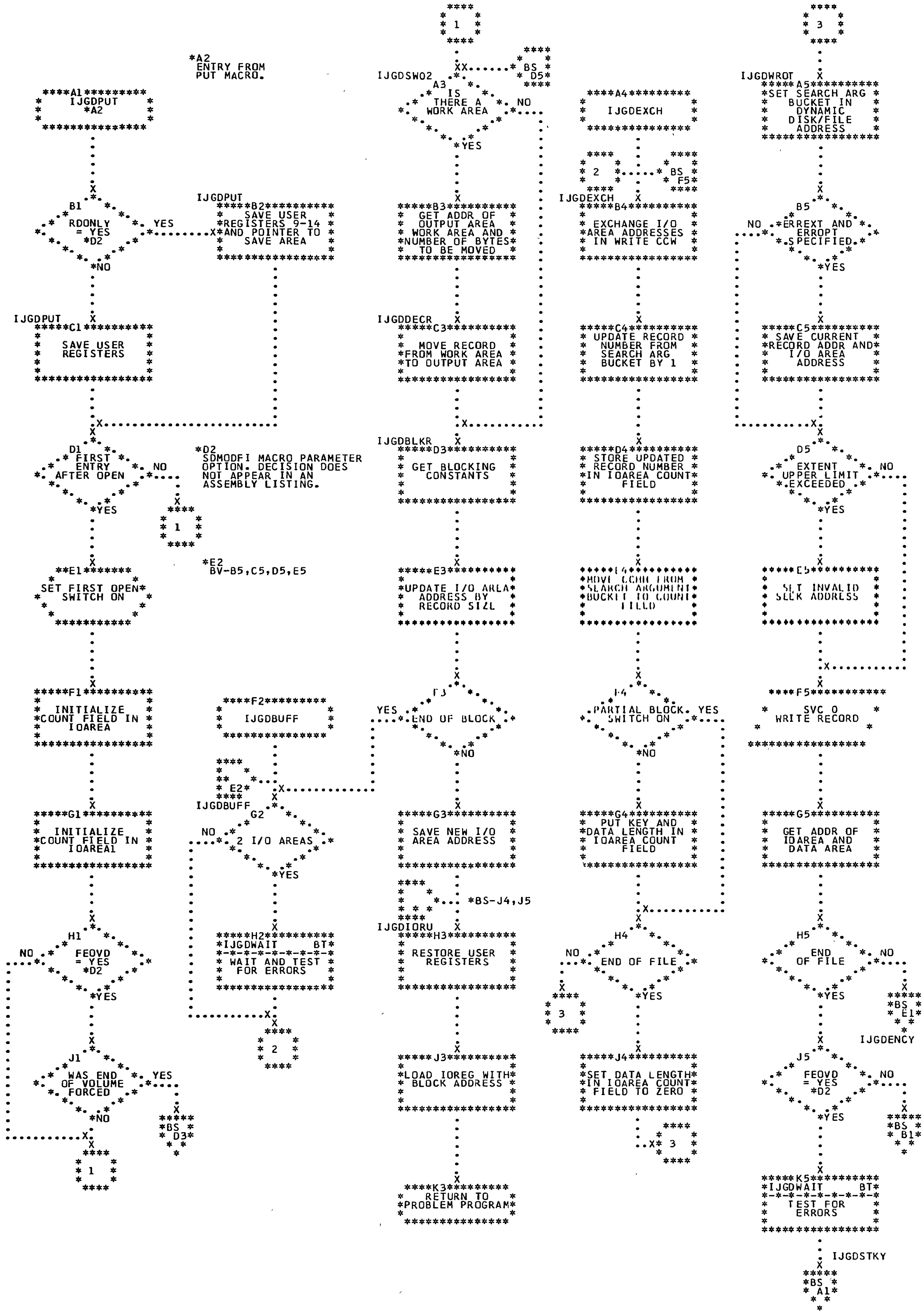
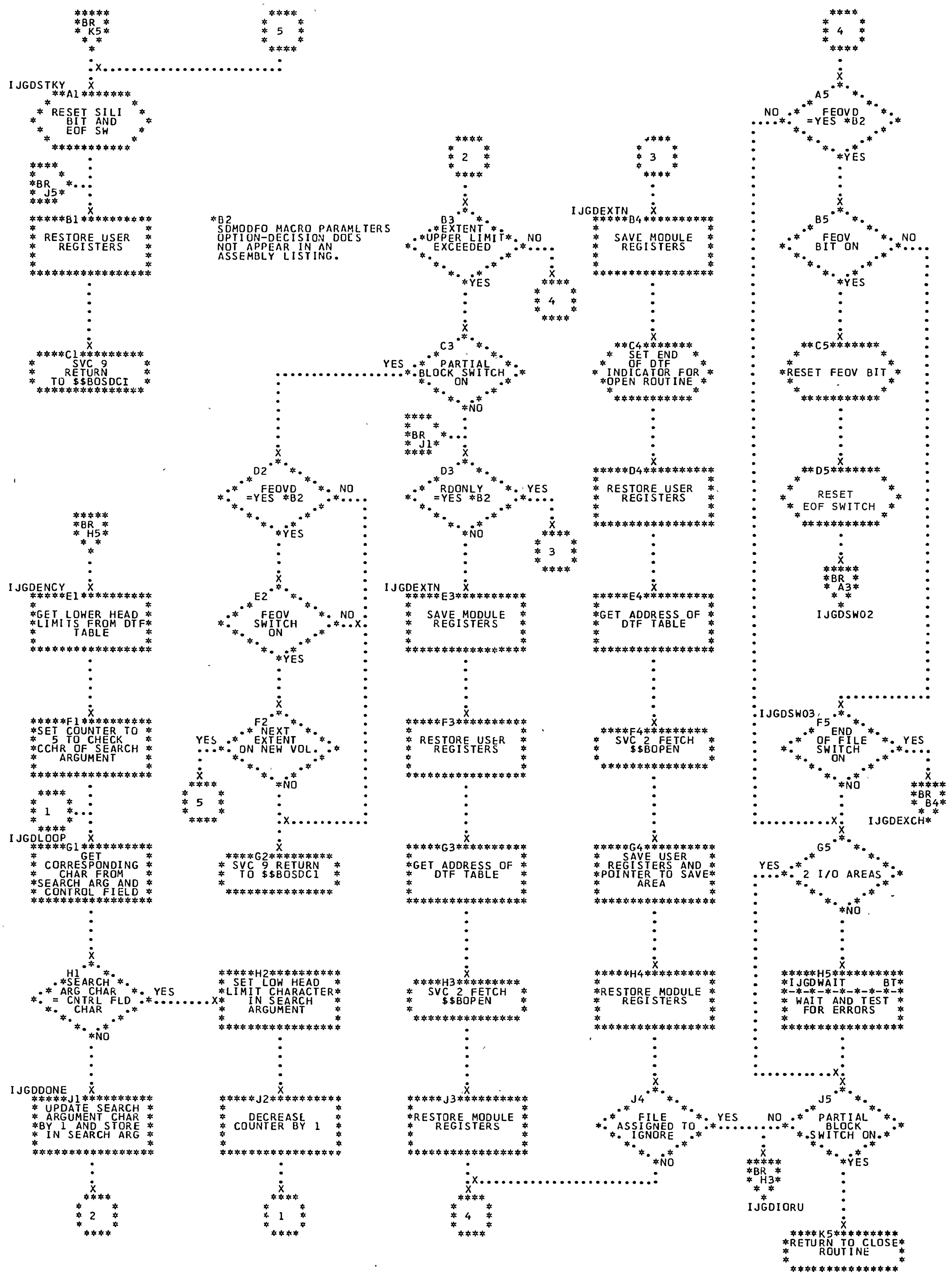

*A2 ENTRY FROM PUT MACRO.
A1 IJGDPUT *A2
B1 RDONLY = YES *D2
YES
NO
IJGDPUT
B2 SAVE USER REGISTERS 9-14 AND POINTER TO SAVE AREA
IJGDPUT
C1 SAVE USER REGISTERS
D1 FIRST ENTRY AFTER OPEN
NO
YES
1
*D2 SDMODFI MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
E1 SET FIRST OPEN SWITCH ON
*E2 BV-B5,C5,D5,E5
F1 INITIALIZE COUNT FIELD IN IOAREA
F2 IJGDBUFF
E2
IJGDBUFF
G1 INITIALIZE COUNT FIELD IN IOAREA1
G2 2 I/O AREAS
NO
YES
H1 FEOVD = YES *D2
NO
YES
H2 IJGDWAIT BT WAIT AND TEST FOR ERRORS
2
J1 WAS END OF VOLUME FORCED
YES
NO
BS D3
1
1
BS D5
IJGDSW02
A3 IS THERE A WORK AREA
NO
YES
B3 GET ADDR OF OUTPUT AREA WORK AREA AND NUMBER OF BYTES TO BE MOVED
IJGDDECR
C3 MOVE RECORD FROM WORK AREA TO OUTPUT AREA
IJGDBLKR
D3 GET BLOCKING CONSTANTS
E3 UPDATE I/O AREA ADDRESS BY RECORD SIZE
F3 END OF BLOCK
YES
NO
G3 SAVE NEW I/O AREA ADDRESS
*BS-J4,J5
IJGDIORU
H3 RESTORE USER REGISTERS
J3 LOAD IOREG WITH BLOCK ADDRESS
K3 RETURN TO PROBLEM PROGRAM
A4 IJGDEXCH
2
BS F5
IJGDEXCH
B4 EXCHANGE I/O AREA ADDRESSES IN WRITE CCW
C4 UPDATE RECORD NUMBER FROM SEARCH ARG BUCKET BY 1
D4 STORE UPDATED RECORD NUMBER IN IOAREA COUNT FIELD
E4 MOVE CCHH FROM SEARCH ARGUMENT BUCKET TO COUNT FIELD
F4 PARTIAL BLOCK SWITCH ON
YES
NO
G4 PUT KEY AND DATA LENGTH IN IOAREA COUNT FIELD
H4 END OF FILE
NO
YES
3
J4 SET DATA LENGTH IN IOAREA COUNT FIELD TO ZERO
3
3
IJGDWROT
A5 SET SEARCH ARG BUCKET IN DYNAMIC DISK/FILE ADDRESS
B5 ERREXT AND ERROPT SPECIFIED
NO
YES
C5 SAVE CURRENT RECORD ADDR AND I/O AREA ADDRESS
D5 EXTENT UPPER LIMIT EXCEEDED
NO
YES
E5 SET INVALID SEEK ADDRESS
F5 SVC 0 WRITE RECORD
G5 GET ADDR OF IOAREA AND DATA AREA
H5 END OF FILE
NO
YES
BS E1
IJGDENCY
J5 FEOVD = YES *D2
NO
YES
BS B1
K5 IJGDWAIT BT TEST FOR ERRORS
IJGDSTKY
BS A1

Chart BS. SDMODFO: Without Truncation, PUT Macro (Section 2 of 4)



## Chart BT. SDMODFO: Without Truncation, PUT Macro (Section 3 of 4)

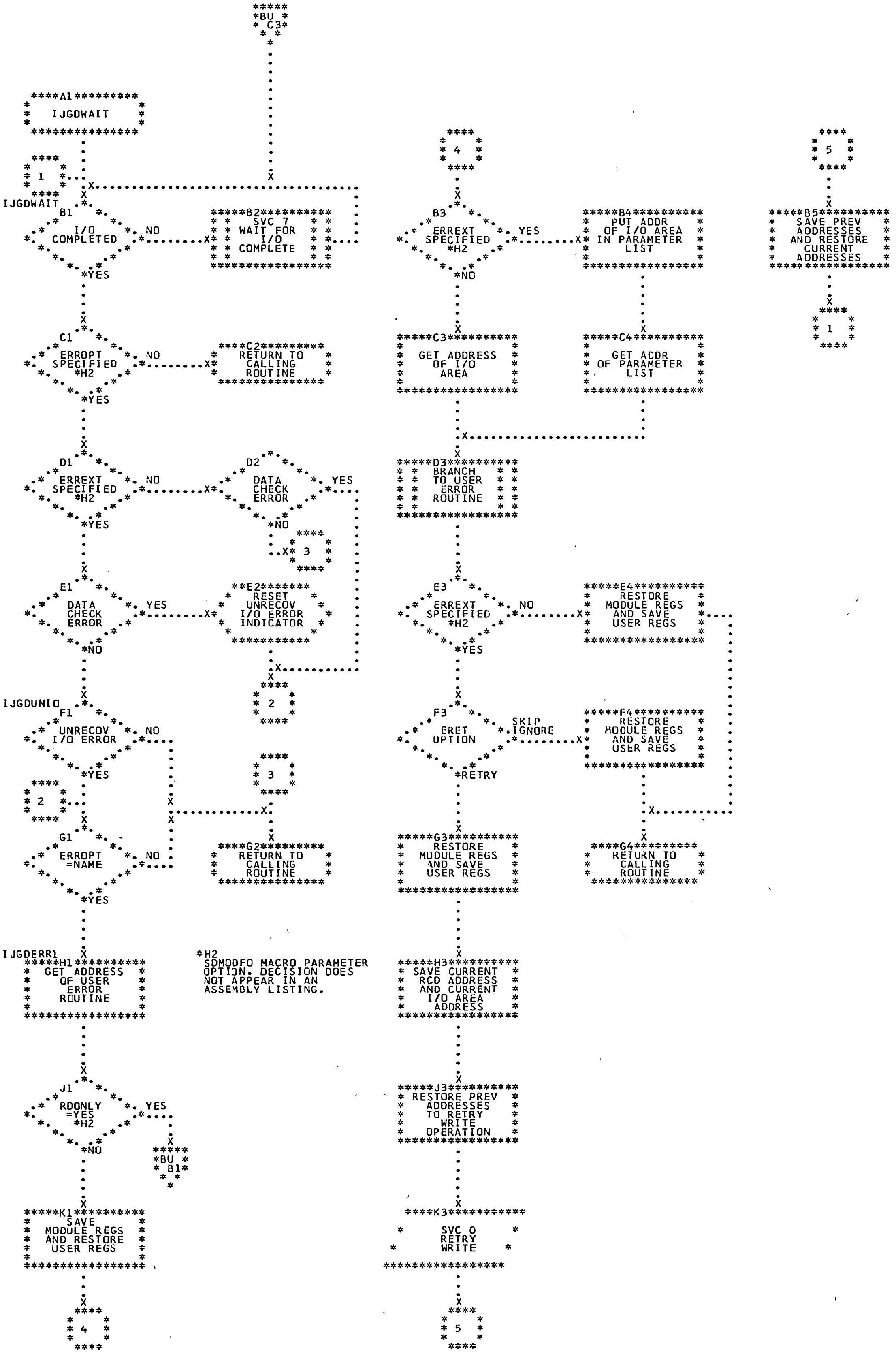
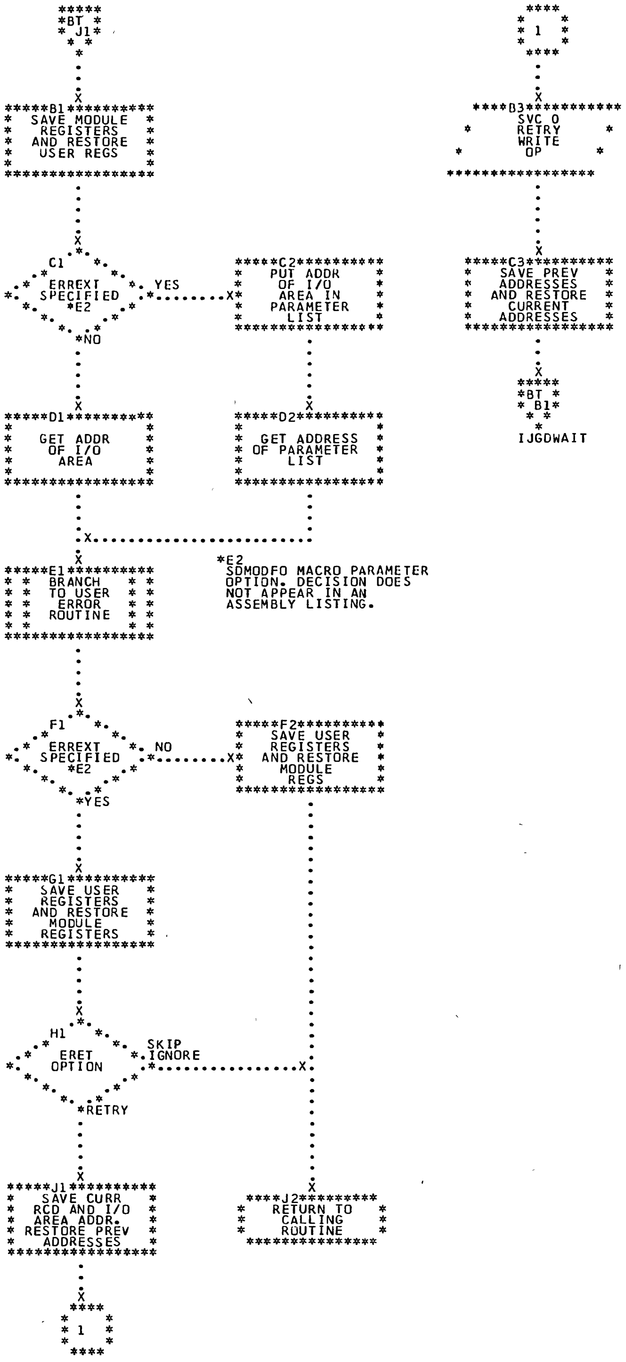

BU C3
A1
IJGDWAIT
1
IJGDWAIT B1
I/O COMPLETED
NO
YES
B2
SVC 7 WAIT FOR I/O COMPLETE
C1
ERROPT SPECIFIED *H2
NO
YES
C2
RETURN TO CALLING ROUTINE
D1
ERREXT SPECIFIED *H2
NO
YES
D2
DATA CHECK ERROR
YES
NO
3
E1
DATA CHECK ERROR
YES
NO
E2
RESET UNRECOV I/O ERROR INDICATOR
2
IJGDUNIO F1
UNRECOV I/O ERROR
NO
YES
3
2
G1
ERROPT =NAME
NO
YES
G2
RETURN TO CALLING ROUTINE
IJGDERR1 H1
GET ADDRESS OF USER ERROR ROUTINE
*H2
SDMODFO MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
J1
RDONLY =YES *H2
YES
NO
BU B1
K1
SAVE MODULE REGS AND RESTORE USER REGS
4
4
B3
ERREXT SPECIFIED *H2
YES
NO
B4
PUT ADDR OF I/O AREA IN PARAMETER LIST
C3
GET ADDRESS OF I/O AREA
C4
GET ADDR OF PARAMETER LIST
D3
BRANCH TO USER ERROR ROUTINE
E3
ERREXT SPECIFIED *H2
NO
YES
E4
RESTORE MODULE REGS AND SAVE USER REGS
F3
ERET OPTION
SKIP IGNORE
RETRY
F4
RESTORE MODULE REGS AND SAVE USER REGS
G3
RESTORE MODULE REGS AND SAVE USER REGS
G4
RETURN TO CALLING ROUTINE
H3
SAVE CURRENT RCD ADDRESS AND CURRENT I/O AREA ADDRESS
J3
RESTORE PREV ADDRESSES TO RETRY WRITE OPERATION
K3
SVC 0 RETRY WRITE
5
5
B5
SAVE PREV ADDRESSES AND RESTORE CURRENT ADDRESSES
1

Chart BU. SDMODFO: Without Truncation, PUT Macro (Section 4 of 4)


BT J1
B1 SAVE MODULE REGISTERS AND RESTORE USER REGS
C1 ERREXT SPECIFIED *E2
YES
NO
C2 PUT ADDR OF I/O AREA IN PARAMETER LIST
D1 GET ADDR OF I/O AREA
D2 GET ADDRESS OF PARAMETER LIST
E1 BRANCH TO USER ERROR ROUTINE
*E2 SDMODFO MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
F1 ERREXT SPECIFIED *E2
NO
YES
F2 SAVE USER REGISTERS AND RESTORE MODULE REGS
G1 SAVE USER REGISTERS AND RESTORE MODULE REGISTERS
H1 ERET OPTION
SKIP IGNORE
RETRY
J1 SAVE CURR RCD AND I/O AREA ADDR. RESTORE PREV ADDRESSES
J2 RETURN TO CALLING ROUTINE
1
1
B3 SVC 0 RETRY WRITE OP
C3 SAVE PREV ADDRESSES AND RESTORE CURRENT ADDRESSES
BT B1
IJGDWAIT

## Chart BV. SDMODFO: Without Truncation, Close File Routine

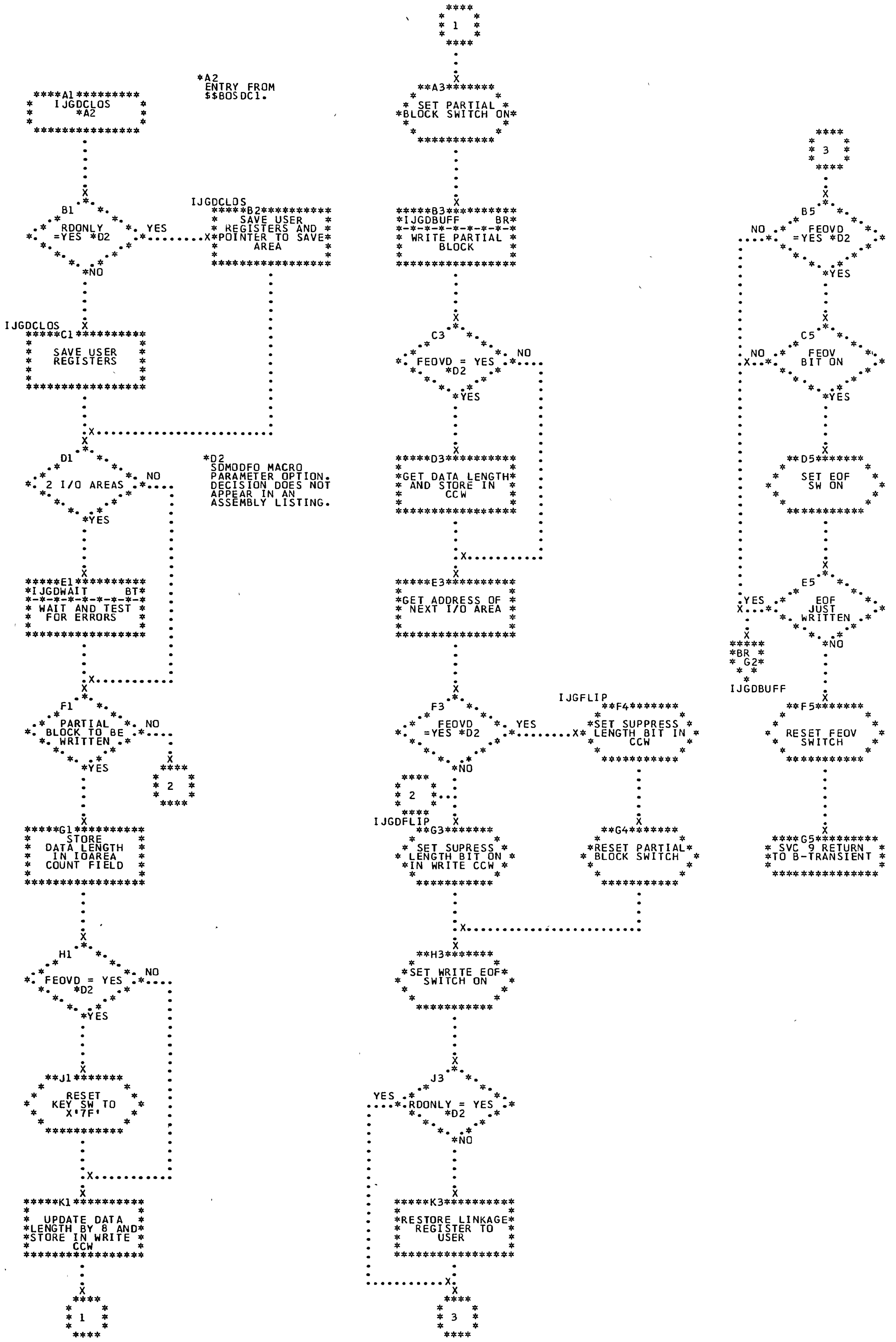

A1 IJGDCLOS *A2
*A2 ENTRY FROM $$BOSDC1.
B1 RDONLY =YES *D2
YES
NO
IJGDCLOS B2 SAVE USER REGISTERS AND POINTER TO SAVE AREA
IJGDCLOS C1 SAVE USER REGISTERS
D1 2 I/O AREAS
NO
YES
*D2 SDMODFO MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
E1 IJGDWAIT BT WAIT AND TEST FOR ERRORS
F1 PARTIAL BLOCK TO BE WRITTEN
NO
YES
2
G1 STORE DATA LENGTH IN IOAREA COUNT FIELD
H1 FEOVD = YES *D2
NO
YES
J1 RESET KEY SW TO X'7F'
K1 UPDATE DATA LENGTH BY 8 AND STORE IN WRITE CCW
1
1
A3 SET PARTIAL BLOCK SWITCH ON
B3 IJGDBUFF BR WRITE PARTIAL BLOCK
C3 FEOVD = YES *D2
NO
YES
D3 GET DATA LENGTH AND STORE IN CCW
E3 GET ADDRESS OF NEXT I/O AREA
F3 FEOVD =YES *D2
YES
NO
IJGFLIP F4 SET SUPPRESS LENGTH BIT IN CCW
2
IJGDFLIP G3 SET SUPRESS LENGTH BIT ON IN WRITE CCW
G4 RESET PARTIAL BLOCK SWITCH
H3 SET WRITE EOF SWITCH ON
J3 RDONLY = YES *D2
YES
NO
K3 RESTORE LINKAGE REGISTER TO USER
3
3
B5 FEOVD =YES *D2
NO
YES
C5 FEOV BIT ON
NO
YES
D5 SET EOF SW ON
E5 EOF JUST WRITTEN
YES
NO
BR G2 IJGDBUFF
F5 RESET FEOV SWITCH
G5 SVC 9 RETURN TO B-TRANSIENT

Chart BW. SDMODFU: With Truncation, GET Macro and CLOSE Routine (Section 1 of 10)

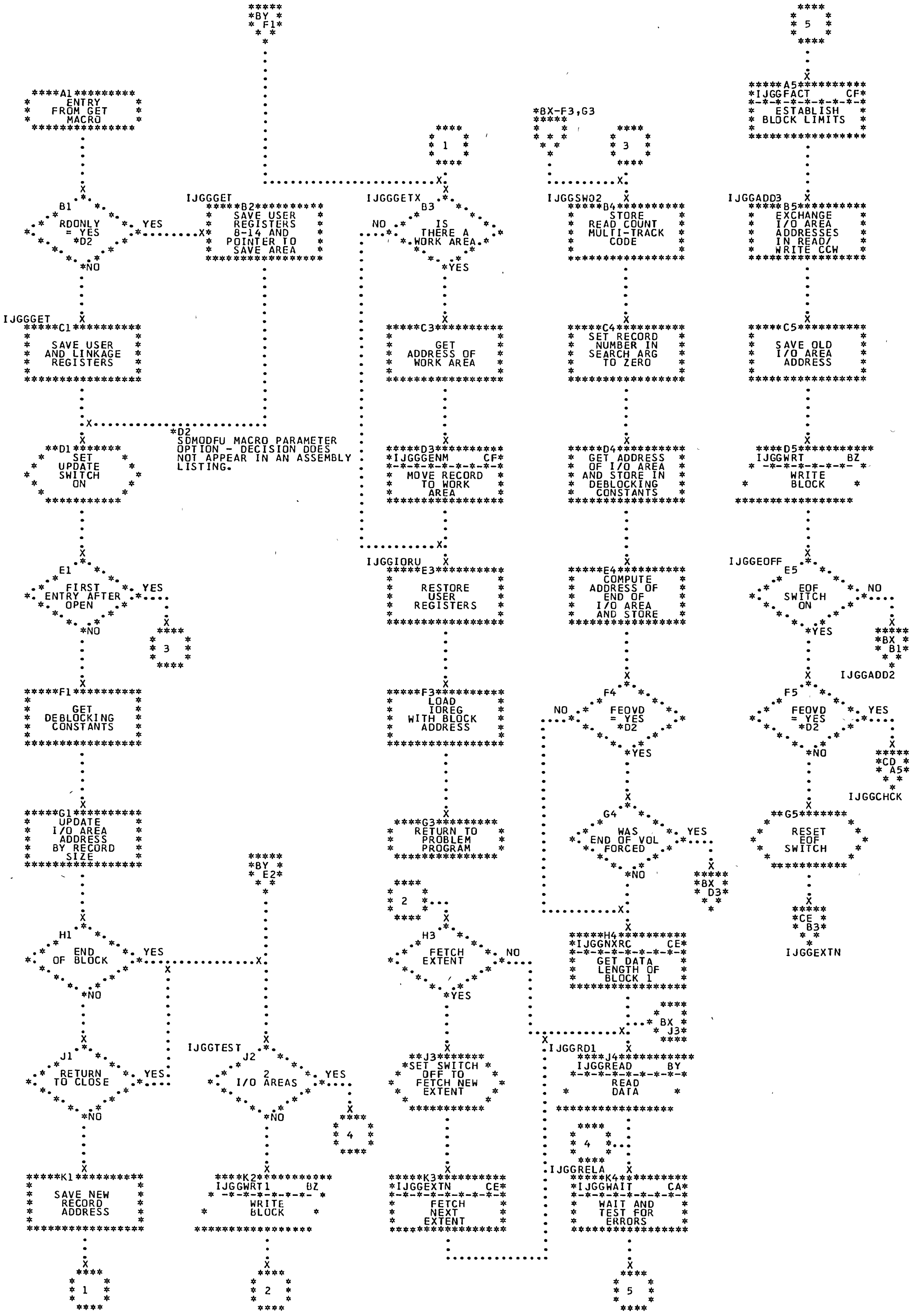

BY F1
A1 ENTRY FROM GET MACRO
B1 RDONLY = YES D2
YES
NO
IJGGGET B2 SAVE USER REGISTERS 8-14 AND POINTER TO SAVE AREA
IJGGGET C1 SAVE USER AND LINKAGE REGISTERS
D2 SDMODFU MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
D1 SET UPDATE SWITCH ON
E1 FIRST ENTRY AFTER OPEN
YES
NO
3
F1 GET DEBLOCKING CONSTANTS
G1 UPDATE I/O AREA ADDRESS BY RECORD SIZE
H1 END OF BLOCK
YES
NO
J1 RETURN TO CLOSE
YES
NO
K1 SAVE NEW RECORD ADDRESS
1
BY E2
IJGGTEST J2 2 I/O AREAS
YES
NO
4
K2 IJGGWRT1 BZ WRITE BLOCK
2
1
IJGGETX B3 IS THERE A WORK AREA
NO
YES
C3 GET ADDRESS OF WORK AREA
D3 IJGGGENM CF MOVE RECORD TO WORK AREA
IJGGIORU E3 RESTORE USER REGISTERS
F3 LOAD IOREG WITH BLOCK ADDRESS
G3 RETURN TO PROBLEM PROGRAM
2
H3 FETCH EXTENT
NO
YES
J3 SET SWITCH OFF TO FETCH NEW EXTENT
K3 IJGGEXTN CE FETCH NEXT EXTENT
BX-F3,G3
3
IJGGSW02 B4 STORE READ COUNT MULTI-TRACK CODE
C4 SET RECORD NUMBER IN SEARCH ARG TO ZERO
D4 GET ADDRESS OF I/O AREA AND STORE IN DEBLOCKING CONSTANTS
E4 COMPUTE ADDRESS OF END OF I/O AREA AND STORE
F4 FEOVD = YES D2
NO
YES
G4 WAS END OF VOL FORCED
YES
NO
BX D3
H4 IJGGNXRC CE GET DATA LENGTH OF BLOCK 1
BX J3
IJGGRD1 J4 IJGGREAD BY READ DATA
4
IJGGRELA K4 IJGGWAIT CA WAIT AND TEST FOR ERRORS
5
5
A5 IJGGFACT CF ESTABLISH BLOCK LIMITS
IJGGADD3 B5 EXCHANGE I/O AREA ADDRESSES IN READ/ WRITE CCW
C5 SAVE OLD I/O AREA ADDRESS
D5 IJGGWRT BZ WRITE BLOCK
IJGGEOFF E5 EOF SWITCH ON
NO
YES
BX B1
IJGGADD2
F5 FEOVD = YES D2
YES
NO
CD A5
IJGGCHCK
G5 RESET EOF SWITCH
CE B3
IJGGEXTN

Chart BX. SDMODFU: With Truncation, GET Macro (Section 2 of 10)

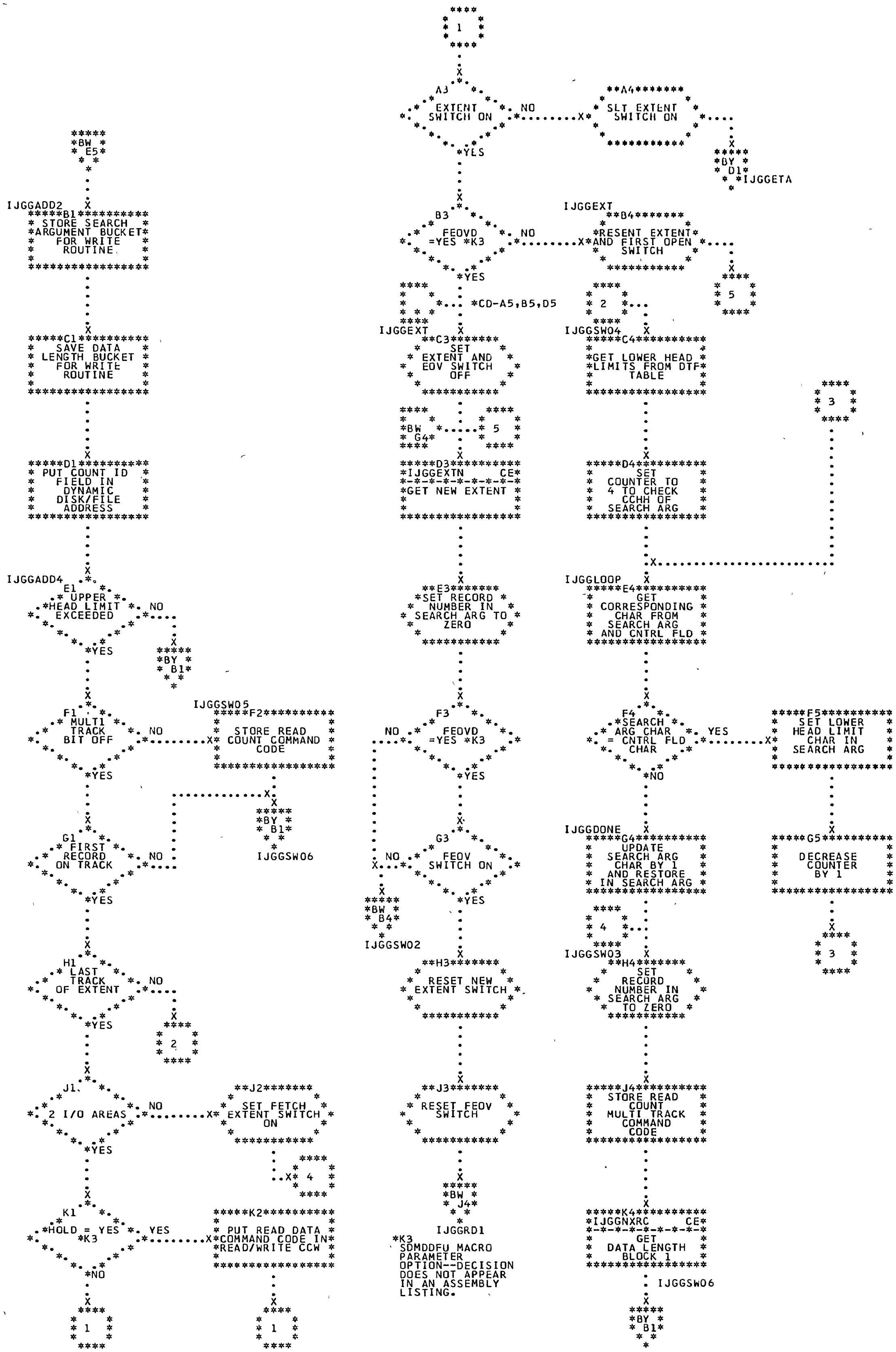

BW E5
IJGGADD2
B1 STORE SEARCH ARGUMENT BUCKET FOR WRITE ROUTINE
C1 SAVE DATA LENGTH BUCKET FOR WRITE ROUTINE
D1 PUT COUNT ID FIELD IN DYNAMIC DISK/FILE ADDRESS
IJGGADD4
E1 UPPER HEAD LIMIT EXCEEDED
NO
YES
BY B1
F1 MULTI TRACK BIT OFF
IJGGSW05
F2 STORE READ COUNT COMMAND CODE
BY B1
IJGGSW06
G1 FIRST RECORD ON TRACK
H1 LAST TRACK OF EXTENT
2
J1 2 I/O AREAS
J2 SET FETCH EXTENT SWITCH ON
4
K1 HOLD = YES *K3
K2 PUT READ DATA COMMAND CODE IN READ/WRITE CCW
1
1
1
A3 EXTENT SWITCH ON
A4 SET EXTENT SWITCH ON
BY D1
IJGGETA
B3 FEOVD =YES *K3
IJGGEXT
B4 RESENT EXTENT AND FIRST OPEN SWITCH
5
*CD-A5,B5,D5
2
IJGGEXT
C3 SET EXTENT AND EOV SWITCH OFF
IJGGSW04
C4 GET LOWER HEAD LIMITS FROM DTF TABLE
3
BW G4
5
D3 IJGGEXTN CE
GET NEW EXTENT
D4 SET COUNTER TO 4 TO CHECK CCHH OF SEARCH ARG
E3 SET RECORD NUMBER IN SEARCH ARG TO ZERO
IJGGLOOP
E4 GET CORRESPONDING CHAR FROM SEARCH ARG AND CNTRL FLD
F3 FEOVD =YES *K3
F4 SEARCH ARG CHAR = CNTRL FLD CHAR
F5 SET LOWER HEAD LIMIT CHAR IN SEARCH ARG
G3 FEOV SWITCH ON
IJGGDONE
G4 UPDATE SEARCH ARG CHAR BY 1 AND RESTORE IN SEARCH ARG
G5 DECREASE COUNTER BY 1
BW B4
IJGGSW02
4
3
IJGGSW03
H3 RESET NEW EXTENT SWITCH
H4 SET RECORD NUMBER IN SEARCH ARG TO ZERO
J3 RESET FEOV SWITCH
J4 STORE READ COUNT MULTI TRACK COMMAND CODE
BW J4
IJGGRD1
K4 IJGGNXRC CE
GET DATA LENGTH BLOCK 1
IJGGSW06
BY B1
*K3 SDMODFU MACRO PARAMETER OPTION--DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.

Chart BY. SDMODFU: With Truncation, GET Macro (Section 3 of 10)

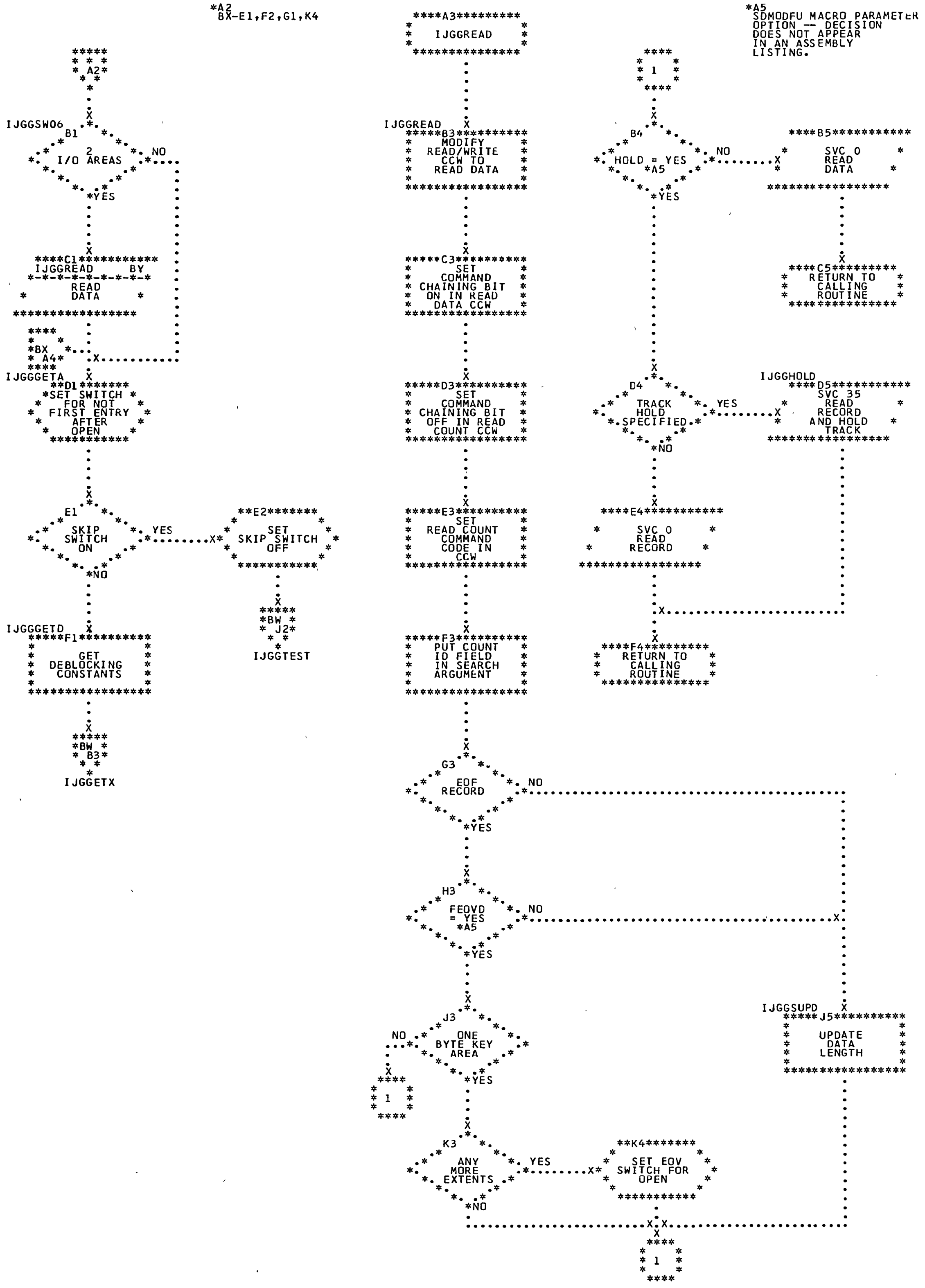


Chart BZ. SDMODFU: With Truncation, GET Macro (Section 4 of 10)

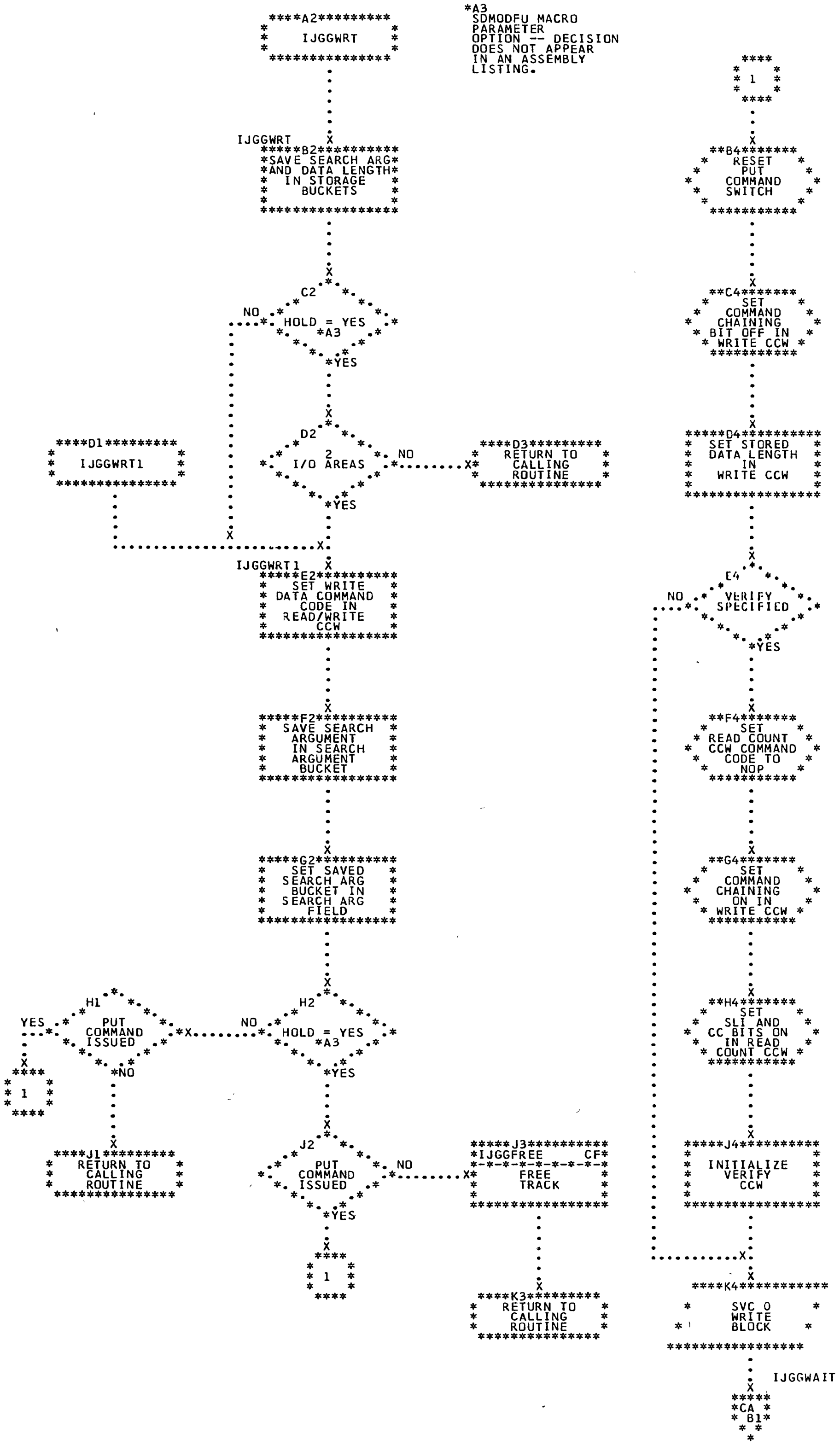
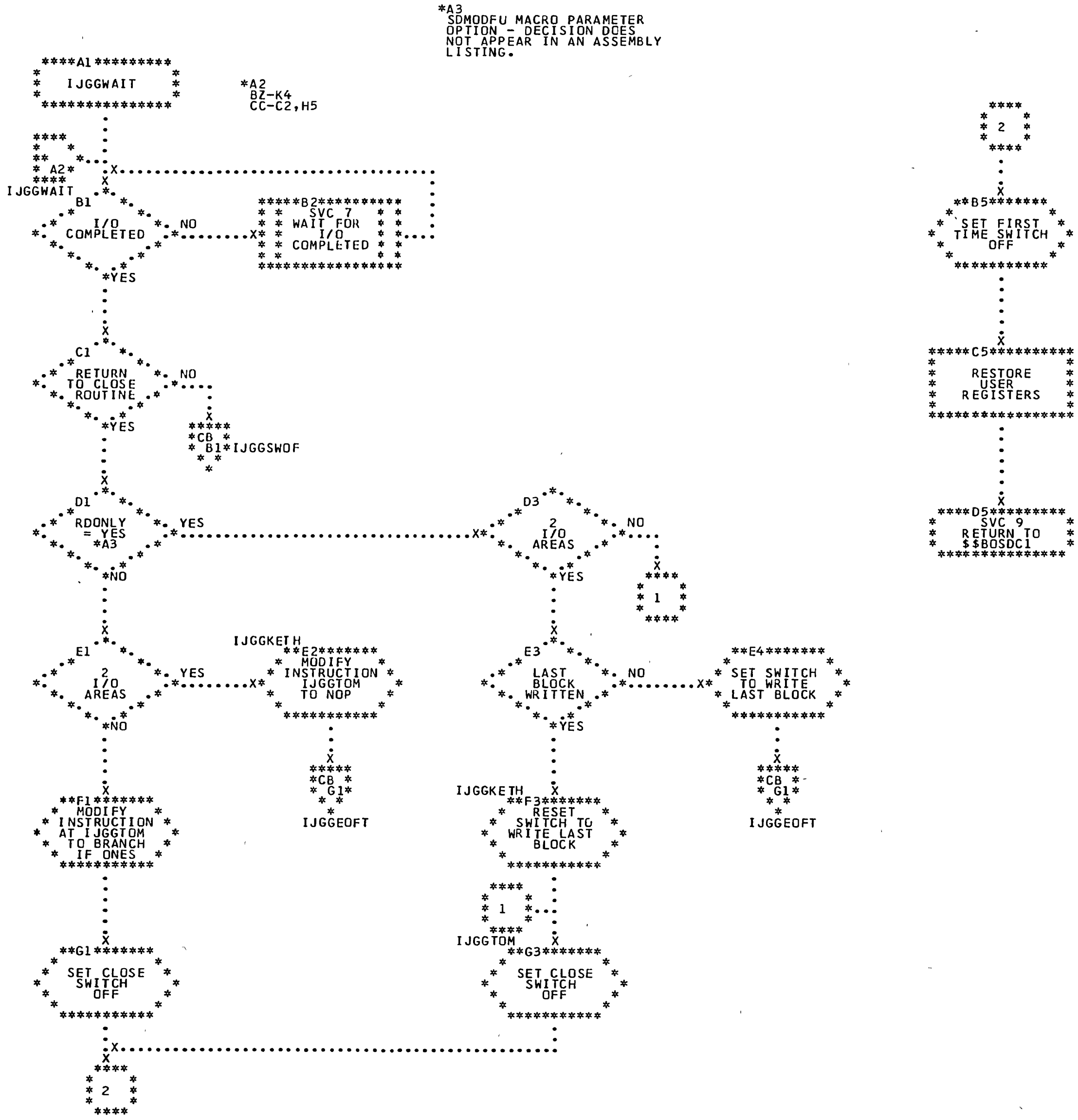

A2
IJGGWRT
*A3
SDMODFU MACRO PARAMETER OPTION -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
1
IJGGWRT
B2
SAVE SEARCH ARG AND DATA LENGTH IN STORAGE BUCKETS
B4
RESET PUT COMMAND SWITCH
C2
HOLD = YES
*A3
NO
YES
C4
SET COMMAND CHAINING BIT OFF IN WRITE CCW
D1
IJGGWRT1
D2
2 I/O AREAS
NO
YES
D3
RETURN TO CALLING ROUTINE
D4
SET STORED DATA LENGTH IN WRITE CCW
IJGGWRT1
E2
SET WRITE DATA COMMAND CODE IN READ/WRITE CCW
E4
VERIFY SPECIFIED
NO
YES
F2
SAVE SEARCH ARGUMENT IN SEARCH ARGUMENT BUCKET
F4
SET READ COUNT CCW COMMAND CODE TO NOP
G2
SET SAVED SEARCH ARG BUCKET IN SEARCH ARG FIELD
G4
SET COMMAND CHAINING ON IN WRITE CCW
H1
PUT COMMAND ISSUED
YES
NO
1
H2
HOLD = YES
*A3
NO
YES
H4
SET SLI AND CC BITS ON IN READ COUNT CCW
J1
RETURN TO CALLING ROUTINE
J2
PUT COMMAND ISSUED
NO
YES
1
J3
IJGGFREE CF
FREE TRACK
J4
INITIALIZE VERIFY CCW
K3
RETURN TO CALLING ROUTINE
K4
SVC 0 WRITE BLOCK
IJGGWAIT
CA B1

Chart CA. SDMODFU: With Truncation, GET Macro (Section 5 of 10)



## Chart CB. SDMODFU: With Truncation, GET Macro (Section 6 of 10)

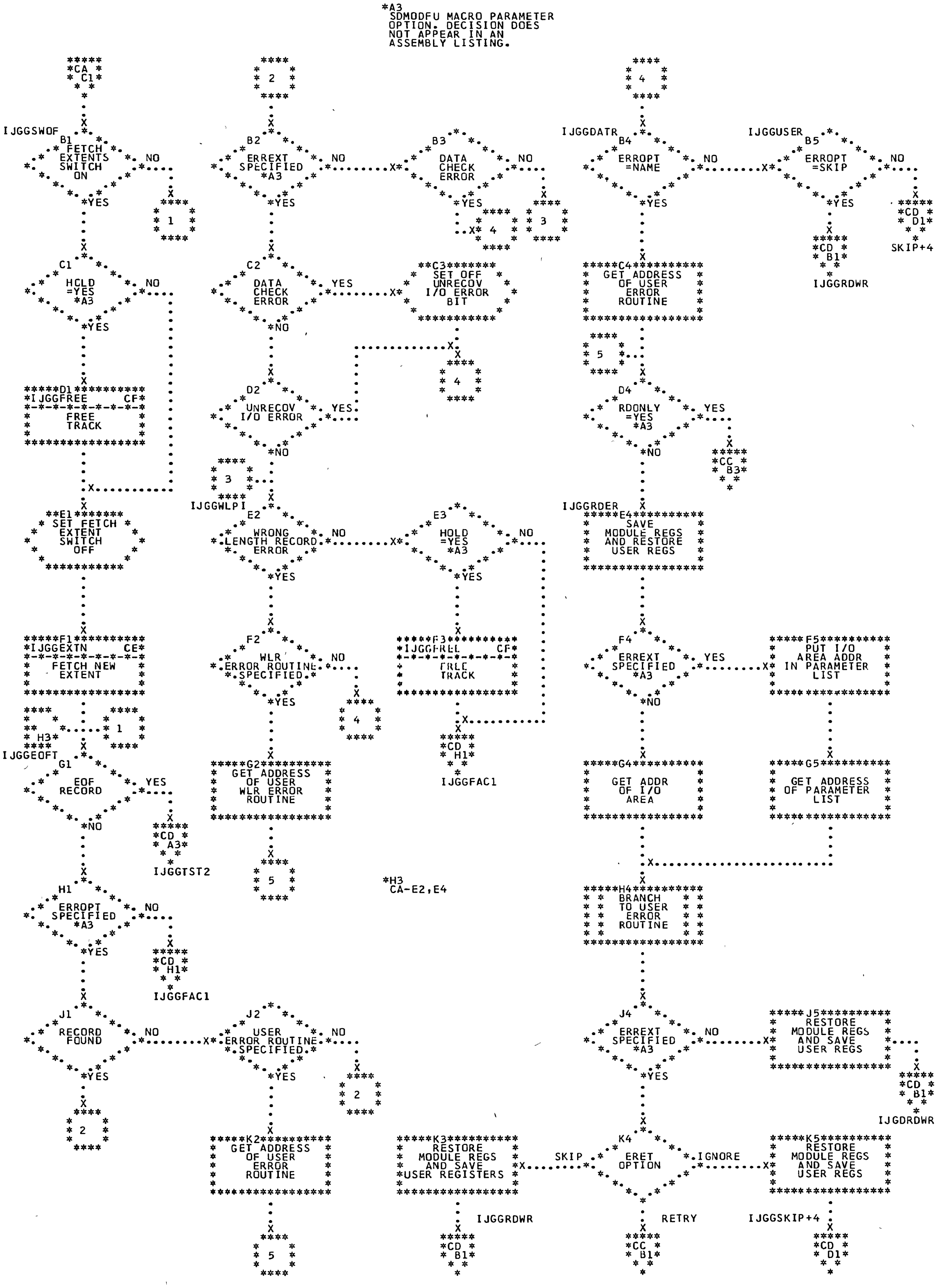
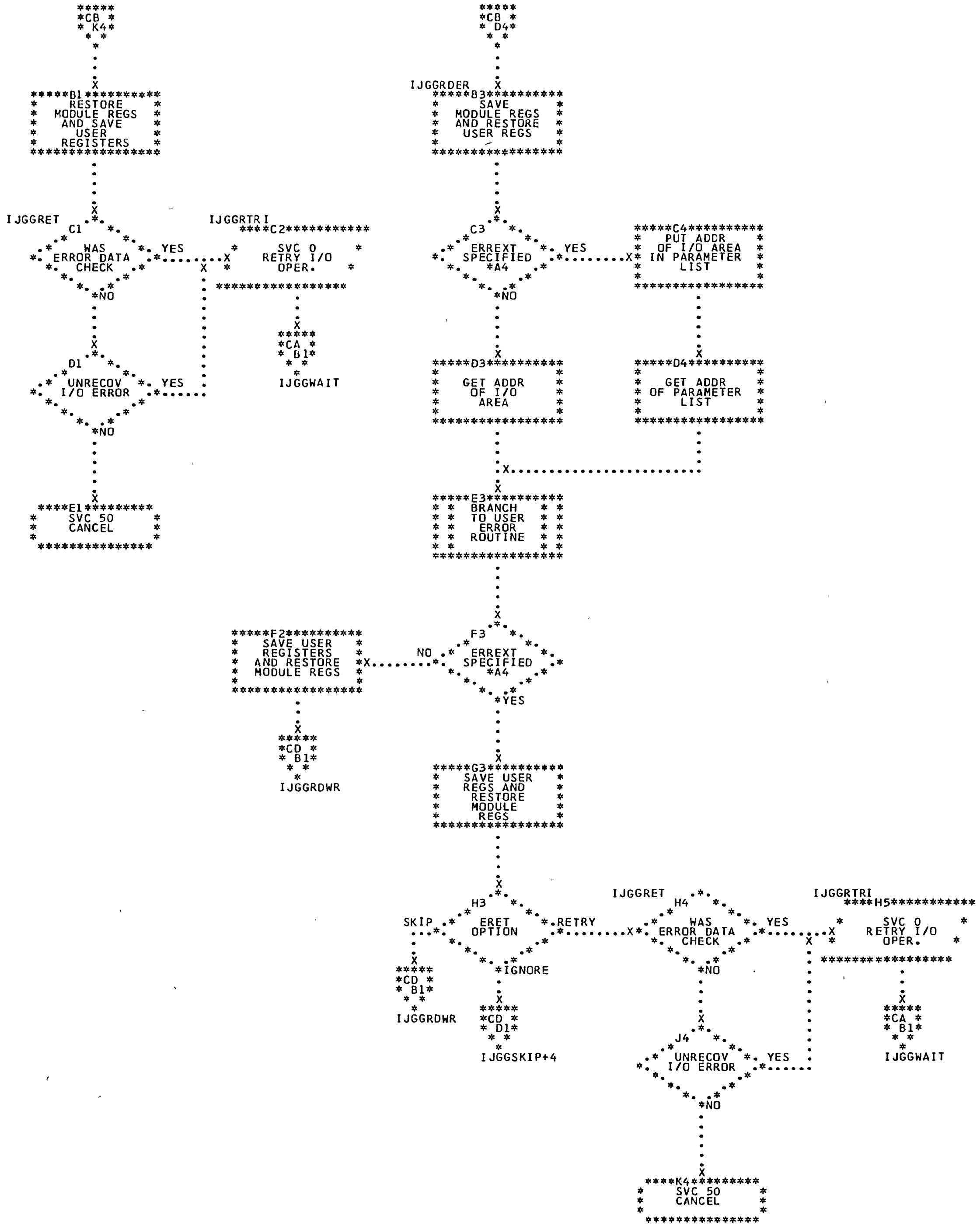


Chart CC. SDMODFU: With Truncation, GET Macro (Section 7 of 10)

*A4
SDMODFU MACRO PARAMETER OPTION.
DECISION DOES NOT APPEAR IN AN
ASSEMBLY LISTING.



## Chart CD. SDMODFU: With Truncation, GET Macro (Section 8 of 10)

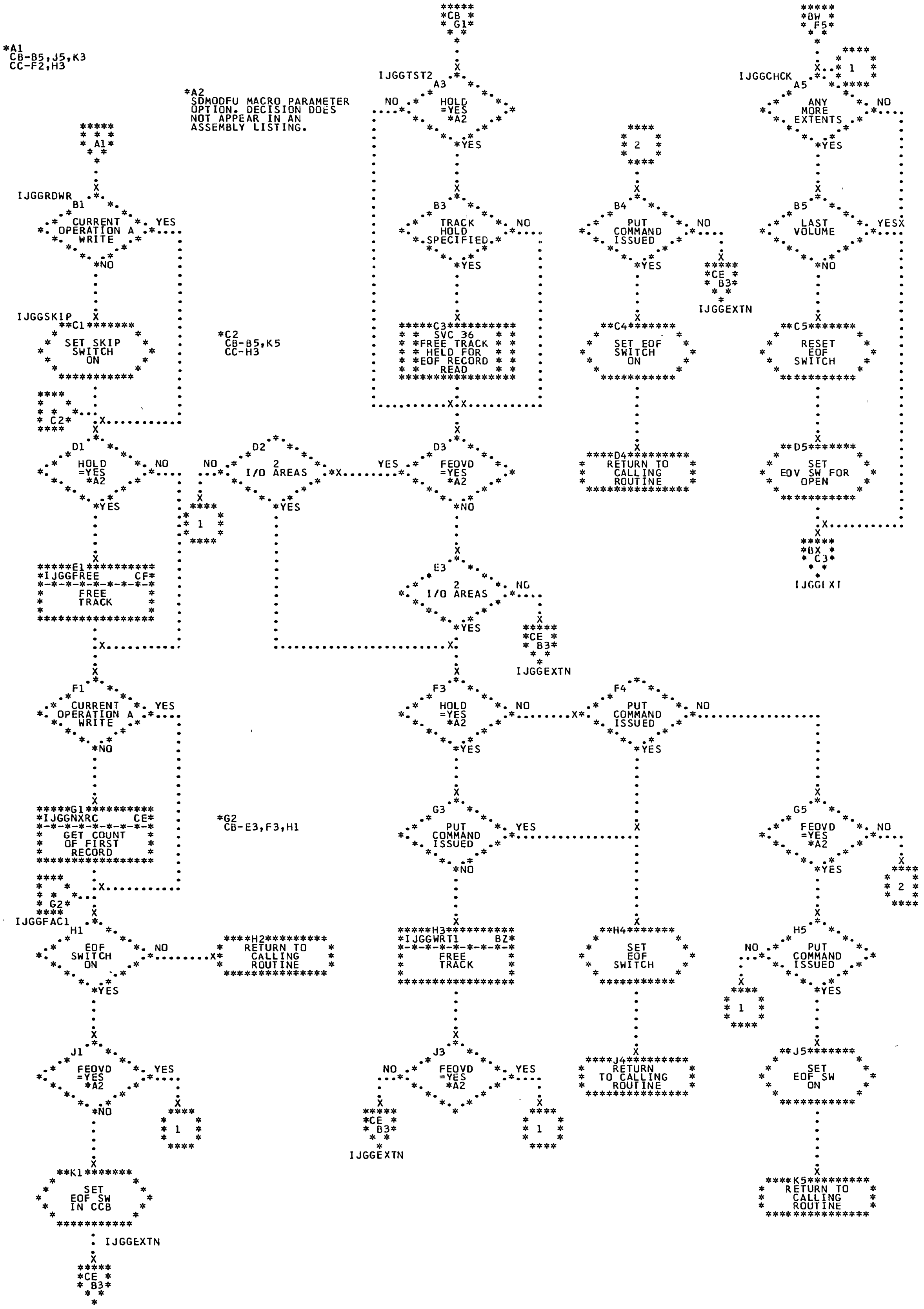


Chart CE. SDMODFU: With Truncation, GET Macro (Section 9 of 10)

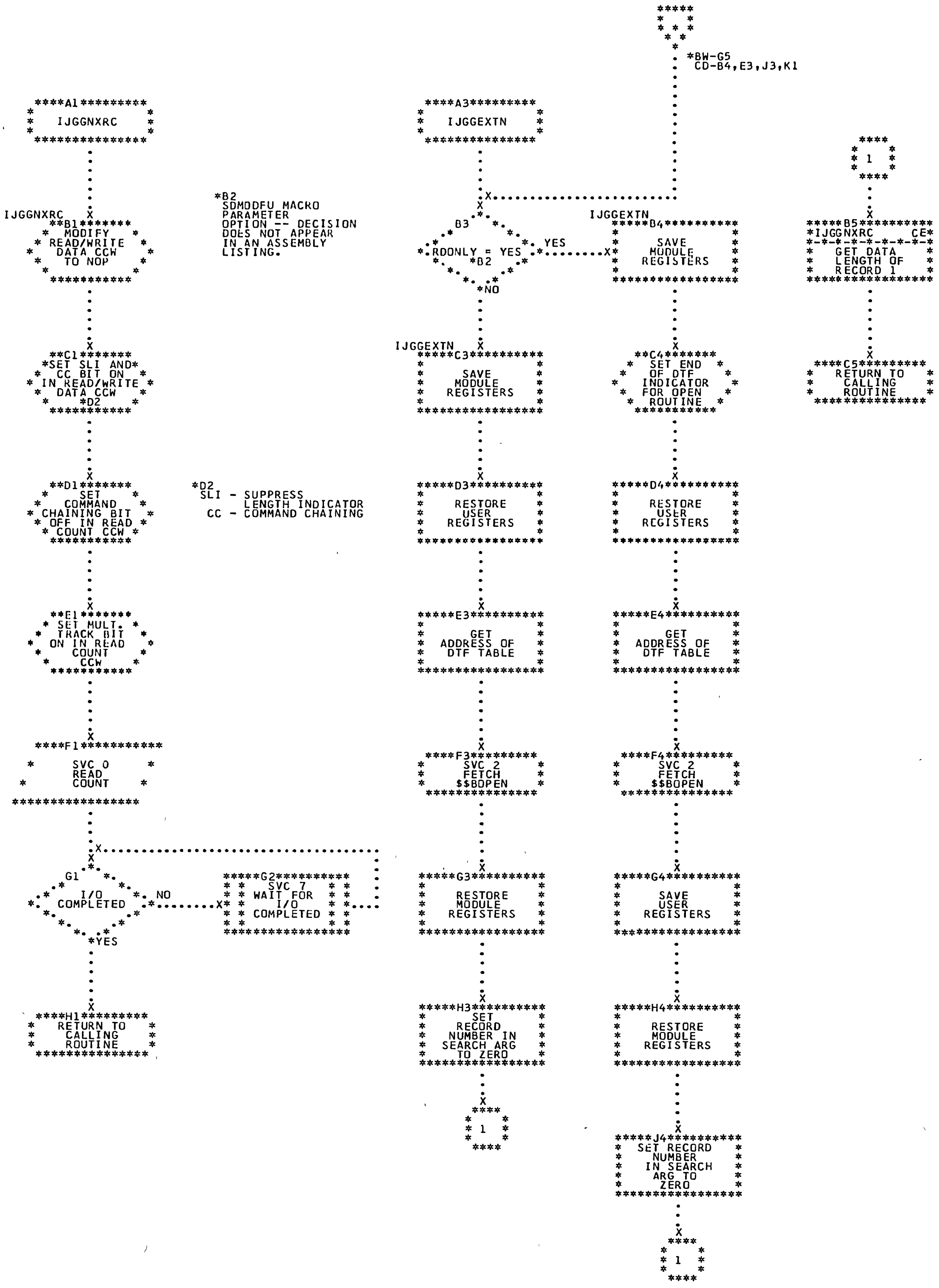

*BW-G5
CD-B4,E3,J3,K1
A1
IJGGNXRC
IJGGNXRC
B1
MODIFY READ/WRITE DATA CCW TO NOP
*B2
SDMODFU MACRO PARAMETER OPTION -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
C1
SET SLI AND CC BIT ON IN READ/WRITE DATA CCW
*D2
D1
SET COMMAND CHAINING BIT OFF IN READ COUNT CCW
*D2
SLI - SUPPRESS LENGTH INDICATOR
CC - COMMAND CHAINING
E1
SET MULT. TRACK BIT ON IN READ COUNT CCW
F1
SVC 0 READ COUNT
G1
I/O COMPLETED
NO
YES
G2
SVC 7 WAIT FOR I/O COMPLETED
H1
RETURN TO CALLING ROUTINE
A3
IJGGEXTN
B3
RDONLY = YES
*B2
YES
NO
IJGGEXTN
C3
SAVE MODULE REGISTERS
D3
RESTORE USER REGISTERS
E3
GET ADDRESS OF DTF TABLE
F3
SVC 2 FETCH $$BOPEN
G3
RESTORE MODULE REGISTERS
H3
SET RECORD NUMBER IN SEARCH ARG TO ZERO
1
IJGGEXTN
B4
SAVE MODULE REGISTERS
C4
SET END OF DTF INDICATOR FOR OPEN ROUTINE
D4
RESTORE USER REGISTERS
E4
GET ADDRESS OF DTF TABLE
F4
SVC 2 FETCH $$BOPEN
G4
SAVE USER REGISTERS
H4
RESTORE MODULE REGISTERS
J4
SET RECORD NUMBER IN SEARCH ARG TO ZERO
1
1
B5
IJGGNXRC CE
GET DATA LENGTH OF RECORD 1
C5
RETURN TO CALLING ROUTINE

Chart CF. SDMODFU: With Truncation, GET Macro (Section 10 of 10)

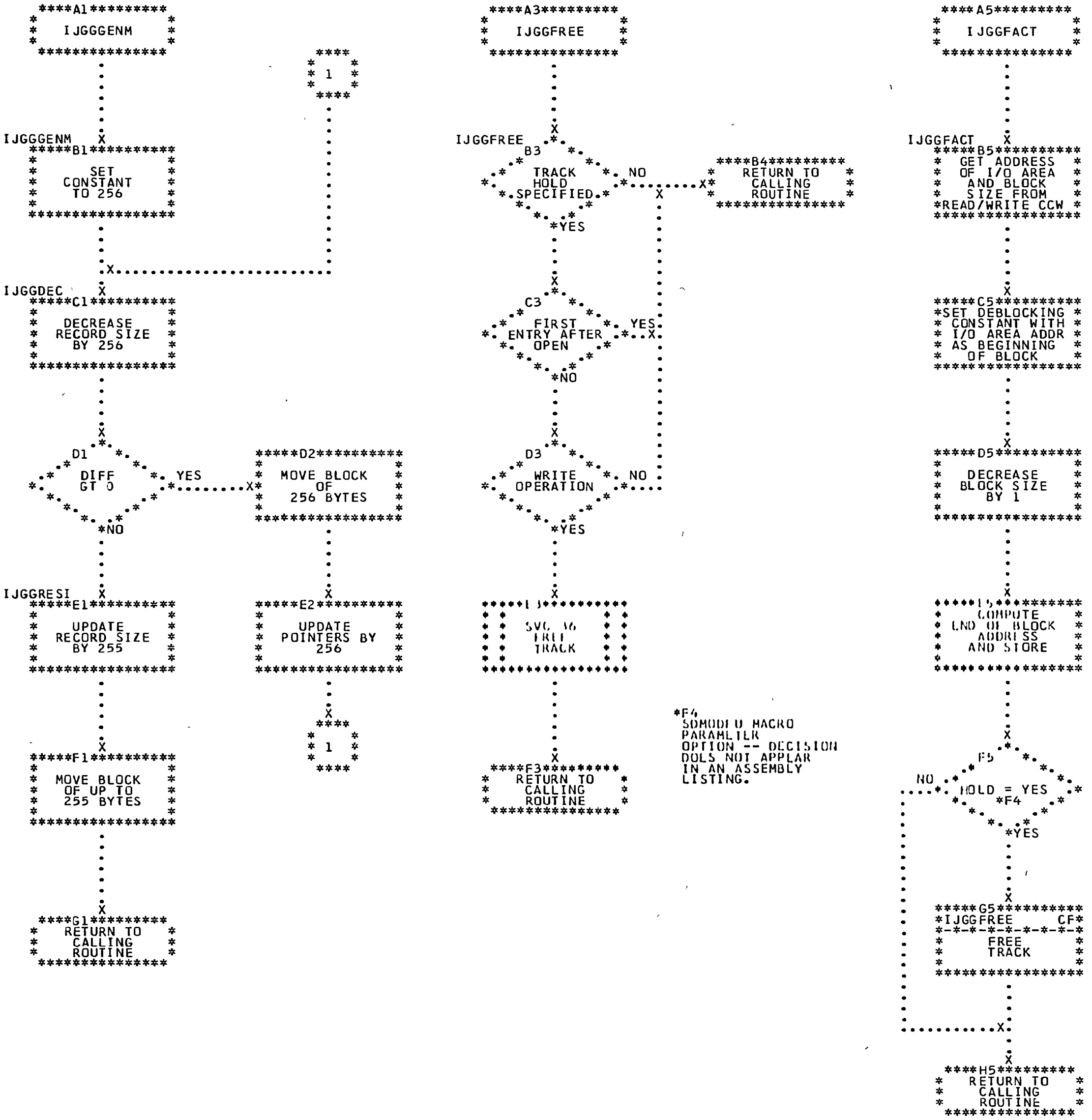
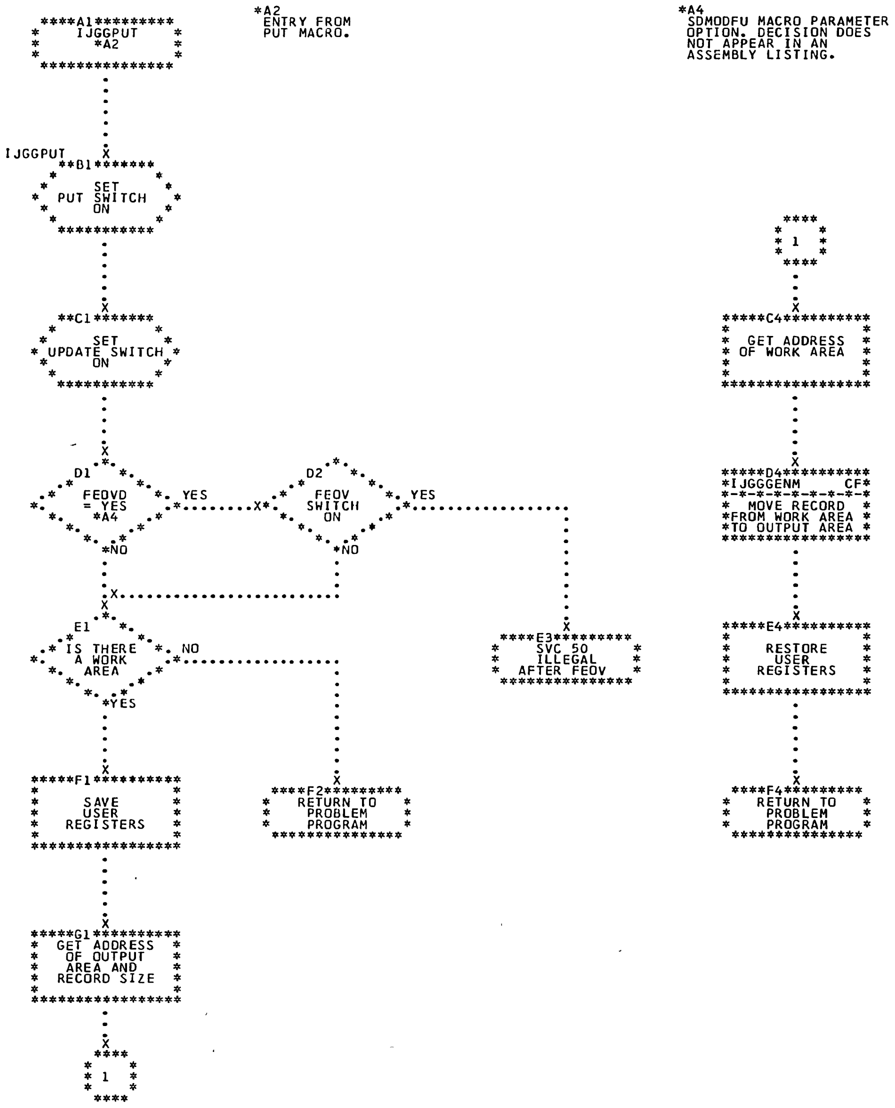

A1
IJGGGENM
IJGGGENM
B1
SET CONSTANT TO 256
1
IJGGDEC
C1
DECREASE RECORD SIZE BY 256
D1
DIFF GT 0
YES
NO
D2
MOVE BLOCK OF 256 BYTES
IJGGRESI
E1
UPDATE RECORD SIZE BY 255
E2
UPDATE POINTERS BY 256
1
F1
MOVE BLOCK OF UP TO 255 BYTES
G1
RETURN TO CALLING ROUTINE
A3
IJGGFREE
IJGGFREE
B3
TRACK HOLD SPECIFIED
NO
YES
B4
RETURN TO CALLING ROUTINE
C3
FIRST ENTRY AFTER OPEN
YES
NO
D3
WRITE OPERATION
NO
YES
E3
SVC 36 FREE TRACK
F3
RETURN TO CALLING ROUTINE
*F4
SDMODFU MACRO PARAMETER OPTION -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
A5
IJGGFACT
IJGGFACT
B5
GET ADDRESS OF I/O AREA AND BLOCK SIZE FROM READ/WRITE CCW
C5
SET DEBLOCKING CONSTANT WITH I/O AREA ADDR AS BEGINNING OF BLOCK
D5
DECREASE BLOCK SIZE BY 1
E5
COMPUTE END OF BLOCK ADDRESS AND STORE
F5
NO
HOLD = YES
*F4
YES
G5
IJGGFREE CF
FREE TRACK
H5
RETURN TO CALLING ROUTINE

Chart CG. SDMODFU: With Truncation, PUT Macro


A1 IJGGPUT *A2
*A2 ENTRY FROM PUT MACRO.
*A4 SDMODFU MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
IJGGPUT
B1 SET PUT SWITCH ON
C1 SET UPDATE SWITCH ON
D1 FEOVD = YES *A4
YES
NO
D2 FEOV SWITCH ON
YES
NO
E1 IS THERE A WORK AREA
NO
YES
F1 SAVE USER REGISTERS
G1 GET ADDRESS OF OUTPUT AREA AND RECORD SIZE
1
F2 RETURN TO PROBLEM PROGRAM
E3 SVC 50 ILLEGAL AFTER FEOV
1
C4 GET ADDRESS OF WORK AREA
D4 IJGGGENM CF
MOVE RECORD FROM WORK AREA TO OUTPUT AREA
E4 RESTORE USER REGISTERS
F4 RETURN TO PROBLEM PROGRAM

# Chart CH. SDMODFU: Without Truncation, GET Macro and CLOSE Routine (Section 1 of 9)

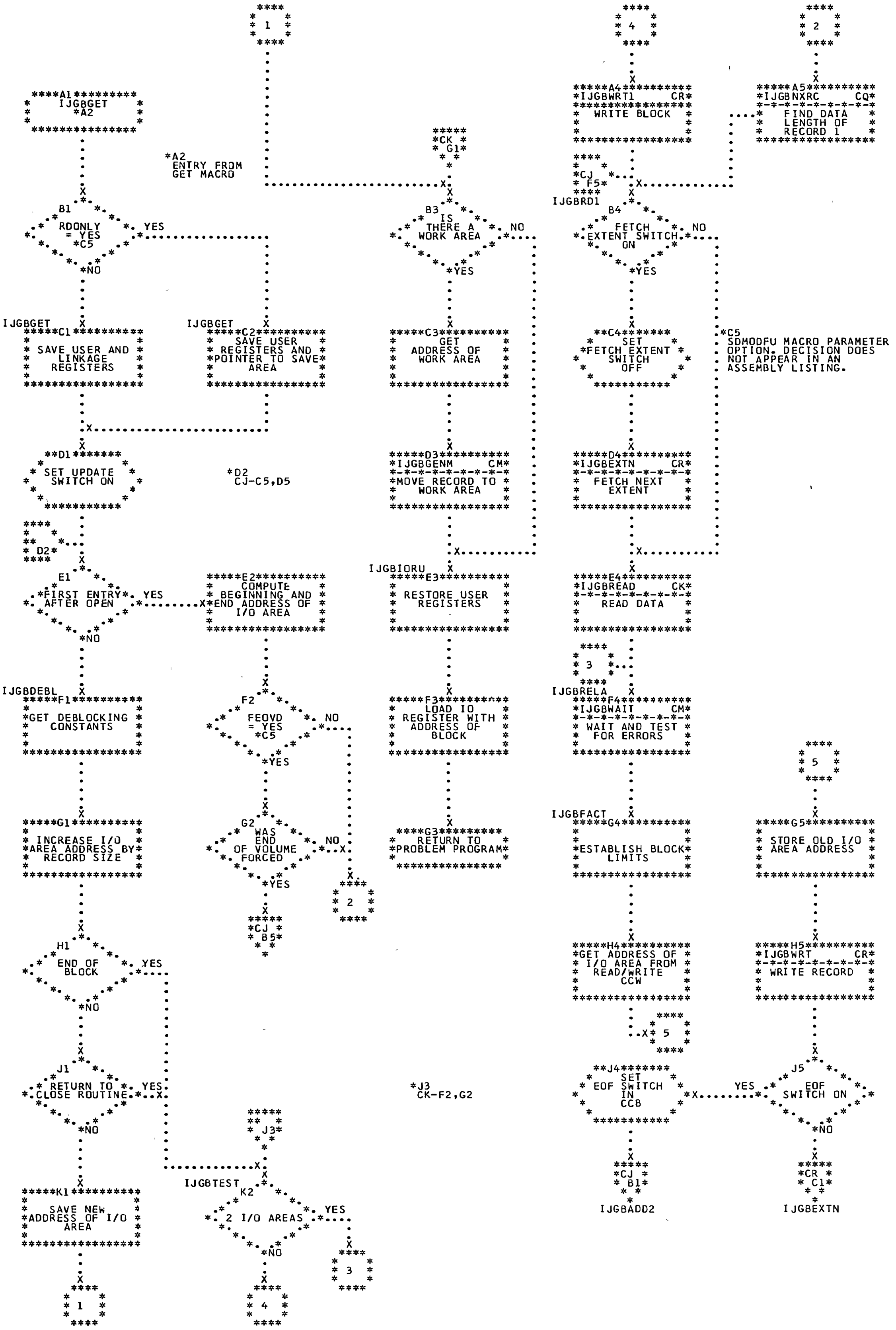


Chart CJ. SDMODFU: Without Truncation, GET Macro (Section 2 of 9)

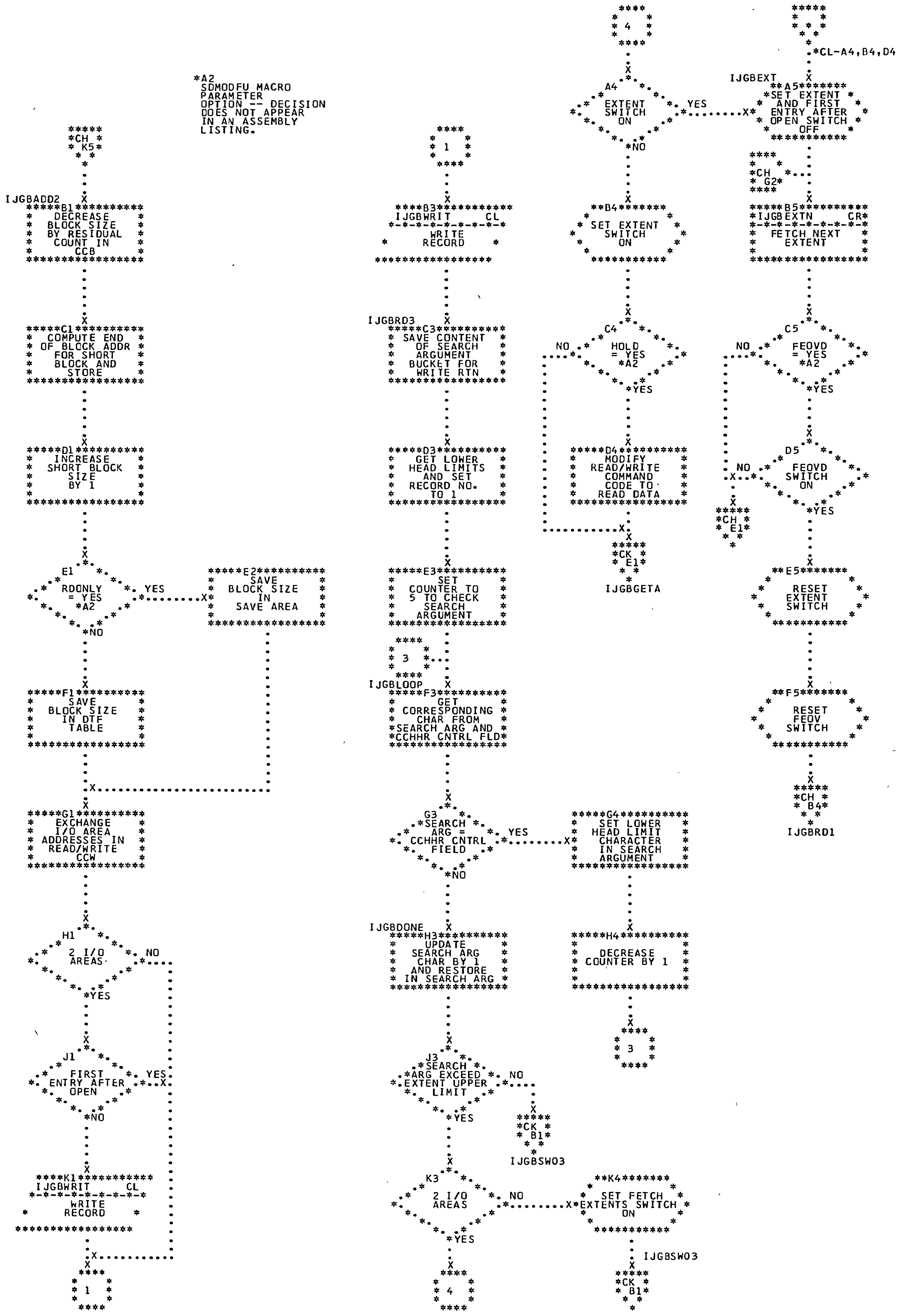

*A2
SDMODFU MACRO PARAMETER OPTION -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
CH K5
IJGBADD2
B1 DECREASE BLOCK SIZE BY RESIDUAL COUNT IN CCB
C1 COMPUTE END OF BLOCK ADDR FOR SHORT BLOCK AND STORE
D1 INCREASE SHORT BLOCK SIZE BY 1
E1 RDONLY = YES *A2
YES
NO
E2 SAVE BLOCK SIZE IN SAVE AREA
F1 SAVE BLOCK SIZE IN DTF TABLE
G1 EXCHANGE I/O AREA ADDRESSES IN READ/WRITE CCW
H1 2 I/O AREAS
NO
YES
J1 FIRST ENTRY AFTER OPEN
YES
NO
K1 IJGBWRIT CL WRITE RECORD
1
1
B3 IJGBWRIT CL WRITE RECORD
IJGBRD3
C3 SAVE CONTENT OF SEARCH ARGUMENT BUCKET FOR WRITE RTN
D3 GET LOWER HEAD LIMITS AND SET RECORD NO. TO 1
E3 SET COUNTER TO 5 TO CHECK SEARCH ARGUMENT
3
IJGBLOOP
F3 GET CORRESPONDING CHAR FROM SEARCH ARG AND CCHHR CNTRL FLD
G3 SEARCH ARG = CCHHR CNTRL FIELD
YES
NO
G4 SET LOWER HEAD LIMIT CHARACTER IN SEARCH ARGUMENT
IJGBDONE
H3 UPDATE SEARCH ARG CHAR BY 1 AND RESTORE IN SEARCH ARG
H4 DECREASE COUNTER BY 1
3
J3 SEARCH ARG EXCEED EXTENT UPPER LIMIT
NO
CK B1
IJGBSW03
YES
K3 2 I/O AREAS
NO
K4 SET FETCH EXTENTS SWITCH ON
IJGBSW03
CK B1
YES
4
4
A4 EXTENT SWITCH ON
YES
NO
B4 SET EXTENT SWITCH ON
C4 HOLD = YES *A2
NO
YES
D4 MODIFY READ/WRITE COMMAND CODE TO READ DATA
CK E1
IJGBGETA
CL-A4,B4,D4
IJGBEXT
A5 SET EXTENT AND FIRST ENTRY AFTER OPEN SWITCH OFF
CH G2
B5 IJGBEXTN CR FETCH NEXT EXTENT
C5 FEOVD = YES *A2
NO
YES
D5 FEOVD SWITCH ON
NO
CH E1
YES
E5 RESET EXTENT SWITCH
F5 RESET FEOV SWITCH
CH B4
IJGBRD1

Chart CK. SDMODFU: Without Truncation, GET Macro (Section 3 of 9)

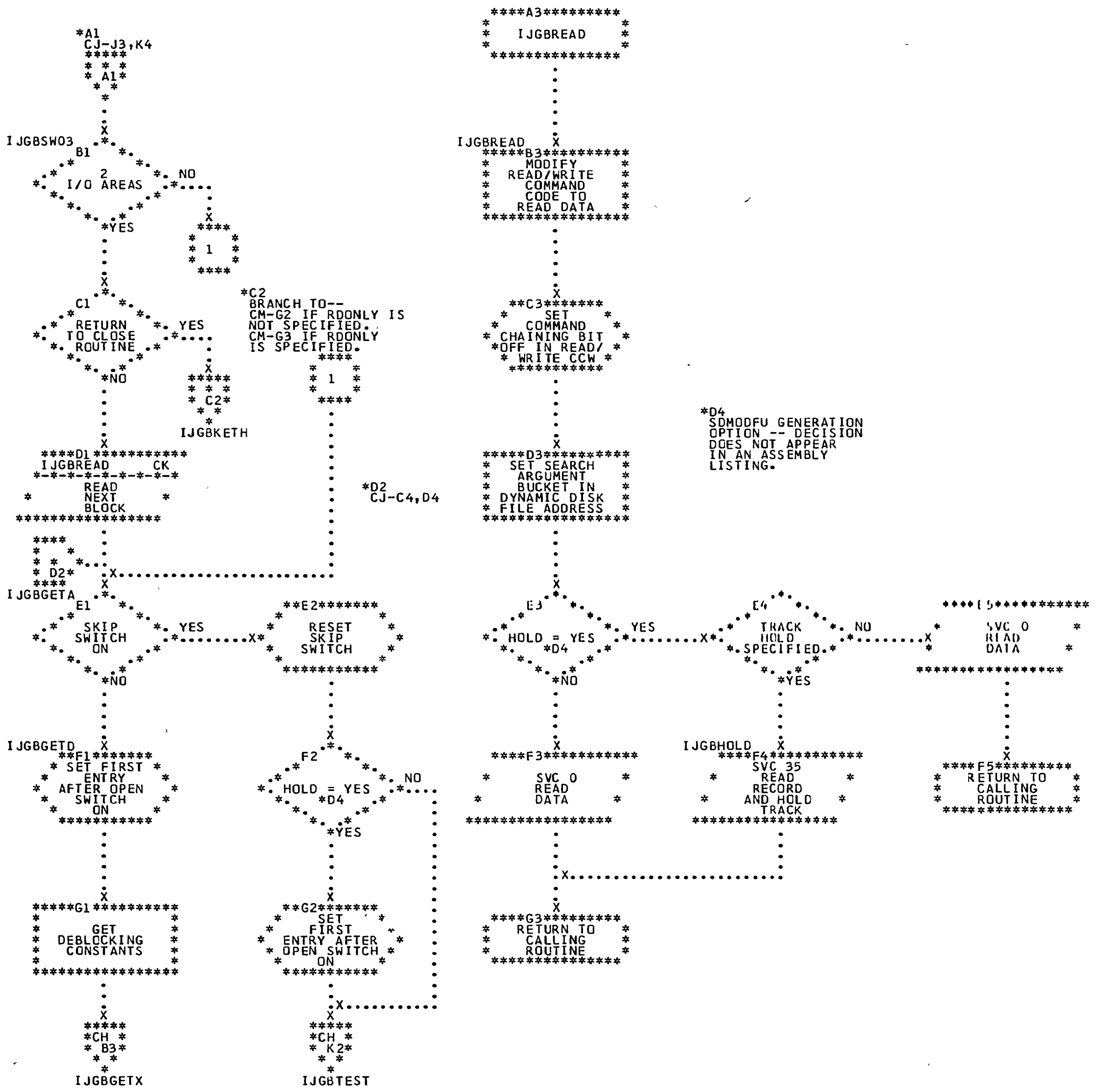

*A1
CJ-J3,K4
A1
IJGBSW03
B1
2 I/O AREAS
NO
YES
1
C1
RETURN TO CLOSE ROUTINE
YES
NO
C2
IJGBKETH
*C2
BRANCH TO--
CM-G2 IF RDONLY IS NOT SPECIFIED.
CM-G3 IF RDONLY IS SPECIFIED.
1
D1
IJGBREAD CK
READ NEXT BLOCK
*D2
CJ-C4,D4
D2
IJGBGETA
E1
SKIP SWITCH ON
YES
NO
E2
RESET SKIP SWITCH
IJGBGETD
F1
SET FIRST ENTRY AFTER OPEN SWITCH ON
F2
HOLD = YES *D4
NO
YES
G1
GET DEBLOCKING CONSTANTS
G2
SET FIRST ENTRY AFTER OPEN SWITCH ON
CH B3
IJGBGETX
CH K2
IJGBTEST
A3
IJGBREAD
IJGBREAD
B3
MODIFY READ/WRITE COMMAND CODE TO READ DATA
C3
SET COMMAND CHAINING BIT OFF IN READ/ WRITE CCW
D3
SET SEARCH ARGUMENT BUCKET IN DYNAMIC DISK FILE ADDRESS
*D4
SDMODFU GENERATION OPTION -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
E3
HOLD = YES *D4
YES
NO
E4
TRACK HOLD SPECIFIED
NO
YES
E5
SVC 0 READ DATA
F3
SVC 0 READ DATA
IJGBHOLD
F4
SVC 35 READ RECORD AND HOLD TRACK
F5
RETURN TO CALLING ROUTINE
G3
RETURN TO CALLING ROUTINE

Chart CL. SDMODFU: Without Truncation, GET Macro (Section 4 of 9)

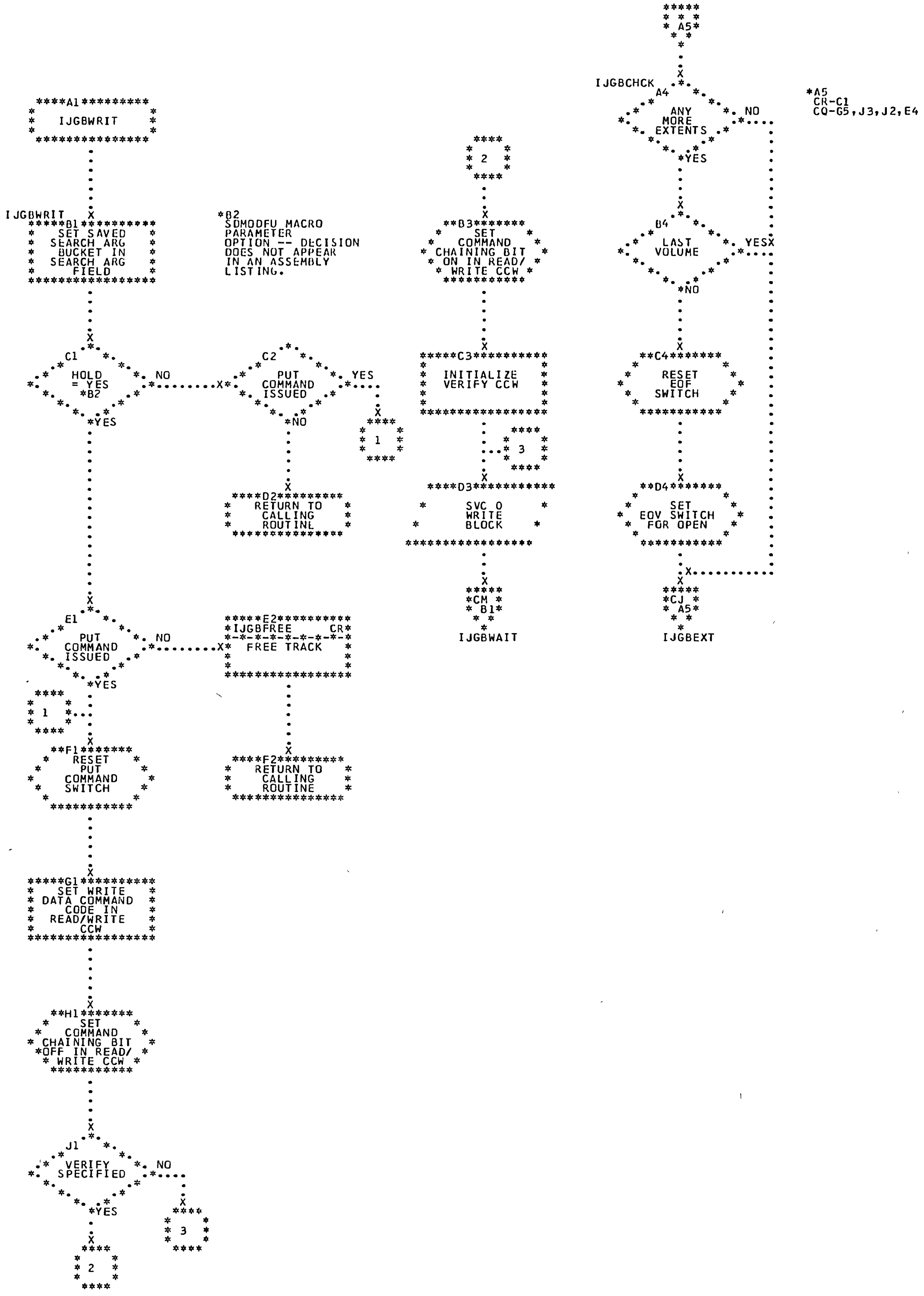

A1 IJGBWRIT
IJGBWRIT
B1 SET SAVED SEARCH ARG BUCKET IN SEARCH ARG FIELD
*B2 SDMODFU MACRO PARAMETER OPTION -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
C1 HOLD = YES *B2
NO
YES
C2 PUT COMMAND ISSUED
YES
NO
1
D2 RETURN TO CALLING ROUTINE
E1 PUT COMMAND ISSUED
NO
YES
E2 IJGBFREE CR FREE TRACK
1
F1 RESET PUT COMMAND SWITCH
F2 RETURN TO CALLING ROUTINE
G1 SET WRITE DATA COMMAND CODE IN READ/WRITE CCW
H1 SET COMMAND CHAINING BIT OFF IN READ/ WRITE CCW
J1 VERIFY SPECIFIED
NO
YES
3
2
2
B3 SET COMMAND CHAINING BIT ON IN READ/ WRITE CCW
C3 INITIALIZE VERIFY CCW
3
D3 SVC 0 WRITE BLOCK
CM B1
IJGBWAIT
A5
IJGBCHCK
A4 ANY MORE EXTENTS
NO
YES
*A5 CR-C1 CQ-G5,J3,J2,E4
B4 LAST VOLUME
YES
NO
C4 RESET EOF SWITCH
D4 SET EOV SWITCH FOR OPEN
CJ A5
IJGBEXT

Chart CM. SDMODFU: Without Truncation, GET Macro (Section 5 of 9)

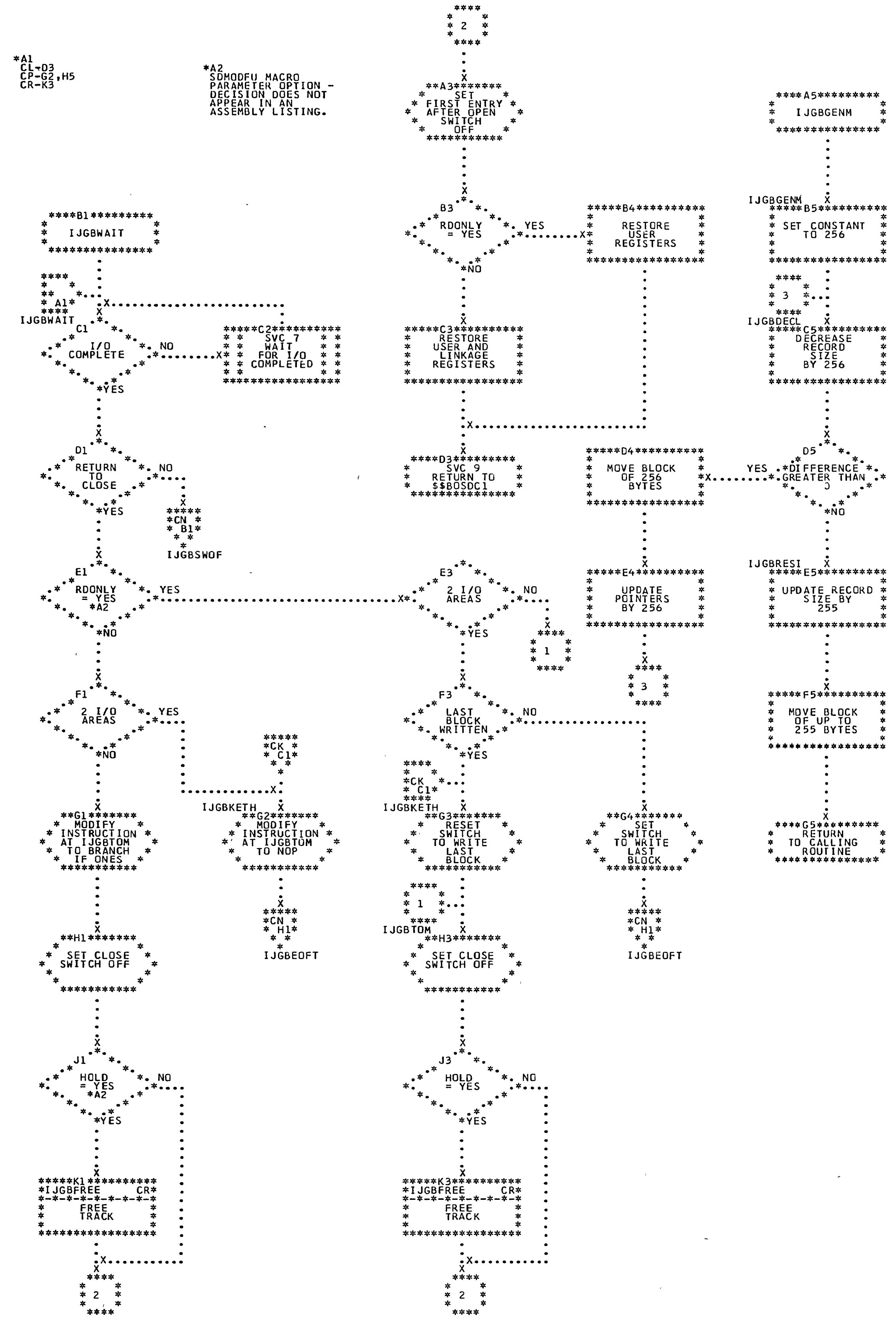


Chart CN. SDMODFU: Without Truncation, GET Macro (Section 6 of 9)

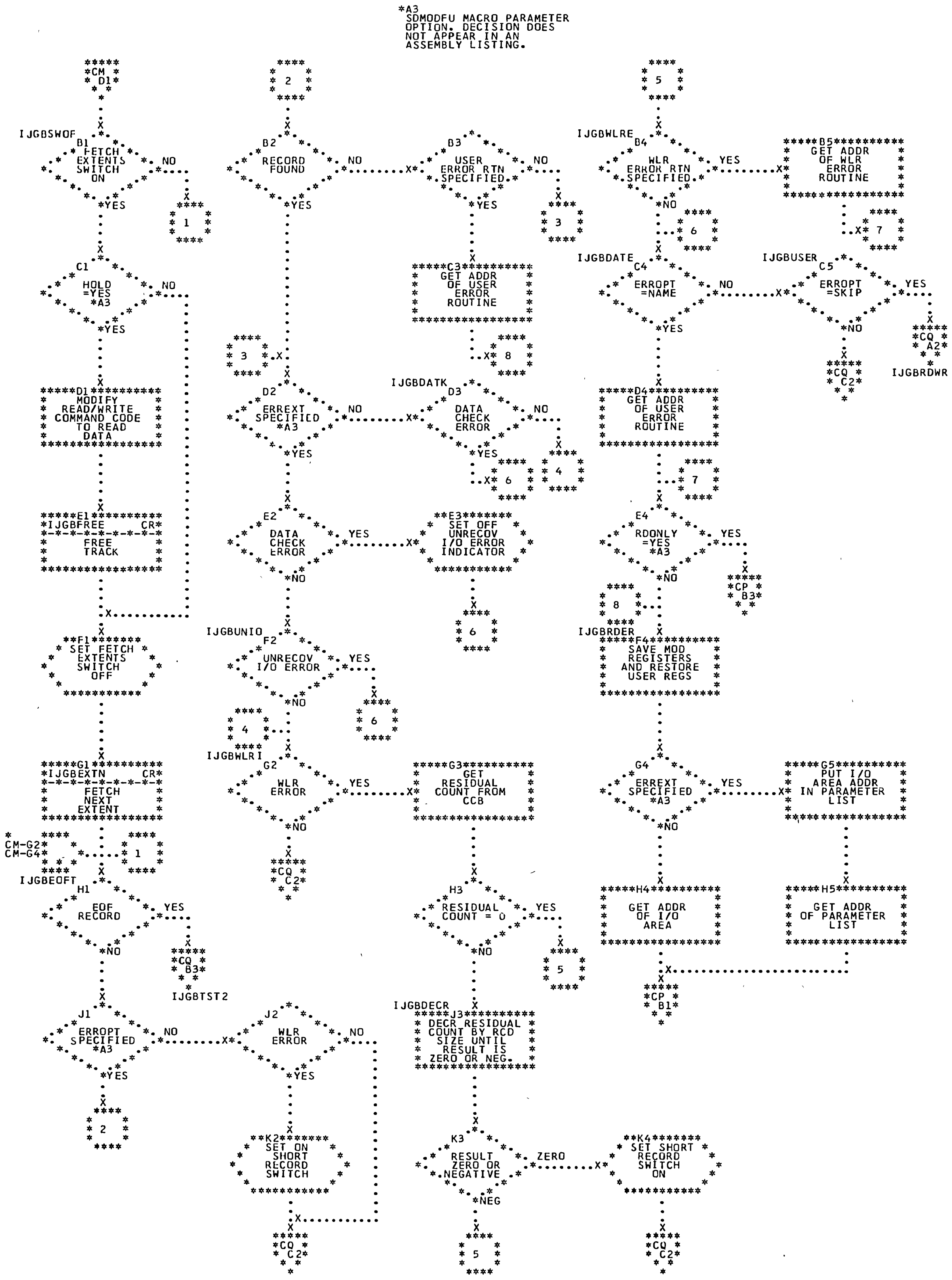

*A3
SDMODFU MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
CM D1
IJGBSWOF
B1 FETCH EXTENTS SWITCH ON
NO
1
YES
C1 HOLD =YES *A3
NO
YES
D1 MODIFY READ/WRITE COMMAND CODE TO READ DATA
E1 IJGBFREE CR
FREE TRACK
F1 SET FETCH EXTENTS SWITCH OFF
G1 IJGBEXTN CR
FETCH NEXT EXTENT
CM-G2
CM-G4
1
IJGBEOFT
H1 EOF RECORD
YES
CQ B3
IJGBTST2
NO
J1 ERROPT SPECIFIED *A3
NO
YES
2
2
B2 RECORD FOUND
NO
YES
3
D2 ERREXT SPECIFIED *A3
NO
YES
E2 DATA CHECK ERROR
YES
NO
IJGBUNIO
F2 UNRECOV I/O ERROR
YES
6
NO
4
IJGBWLRI
G2 WLR ERROR
YES
NO
CQ C2
J2 WLR ERROR
NO
YES
K2 SET ON SHORT RECORD SWITCH
CQ C2
B3 USER ERROR RTN SPECIFIED
NO
3
YES
C3 GET ADDR OF USER ERROR ROUTINE
8
IJGBDATK
D3 DATA CHECK ERROR
NO
4
YES
6
E3 SET OFF UNRECOV I/O ERROR INDICATOR
6
G3 GET RESIDUAL COUNT FROM CCB
H3 RESIDUAL COUNT = 0
YES
5
NO
IJGBDECR
J3 DECR RESIDUAL COUNT BY RCD SIZE UNTIL RESULT IS ZERO OR NEG.
K3 RESULT ZERO OR NEGATIVE
ZERO
NEG
5
5
IJGBWLRE
B4 WLR ERROR RTN SPECIFIED
YES
NO
6
IJGBDATE
C4 ERROPT =NAME
NO
YES
D4 GET ADDR OF USER ERROR ROUTINE
7
E4 RDONLY =YES *A3
YES
CP B3
NO
8
IJGBRDER
F4 SAVE MOD REGISTERS AND RESTORE USER REGS
G4 ERREXT SPECIFIED *A3
YES
NO
H4 GET ADDR OF I/O AREA
CP B1
K4 SET SHORT RECORD SWITCH ON
CQ C2
B5 GET ADDR OF WLR ERROR ROUTINE
7
IJGBUSER
C5 ERROPT =SKIP
YES
CQ A2
IJGBRDWR
NO
CQ C2
G5 PUT I/O AREA ADDR IN PARAMETER LIST
H5 GET ADDR OF PARAMETER LIST

Chart CP. SDMODFU: Without Truncation, GET Macro (Section 7 of 9)

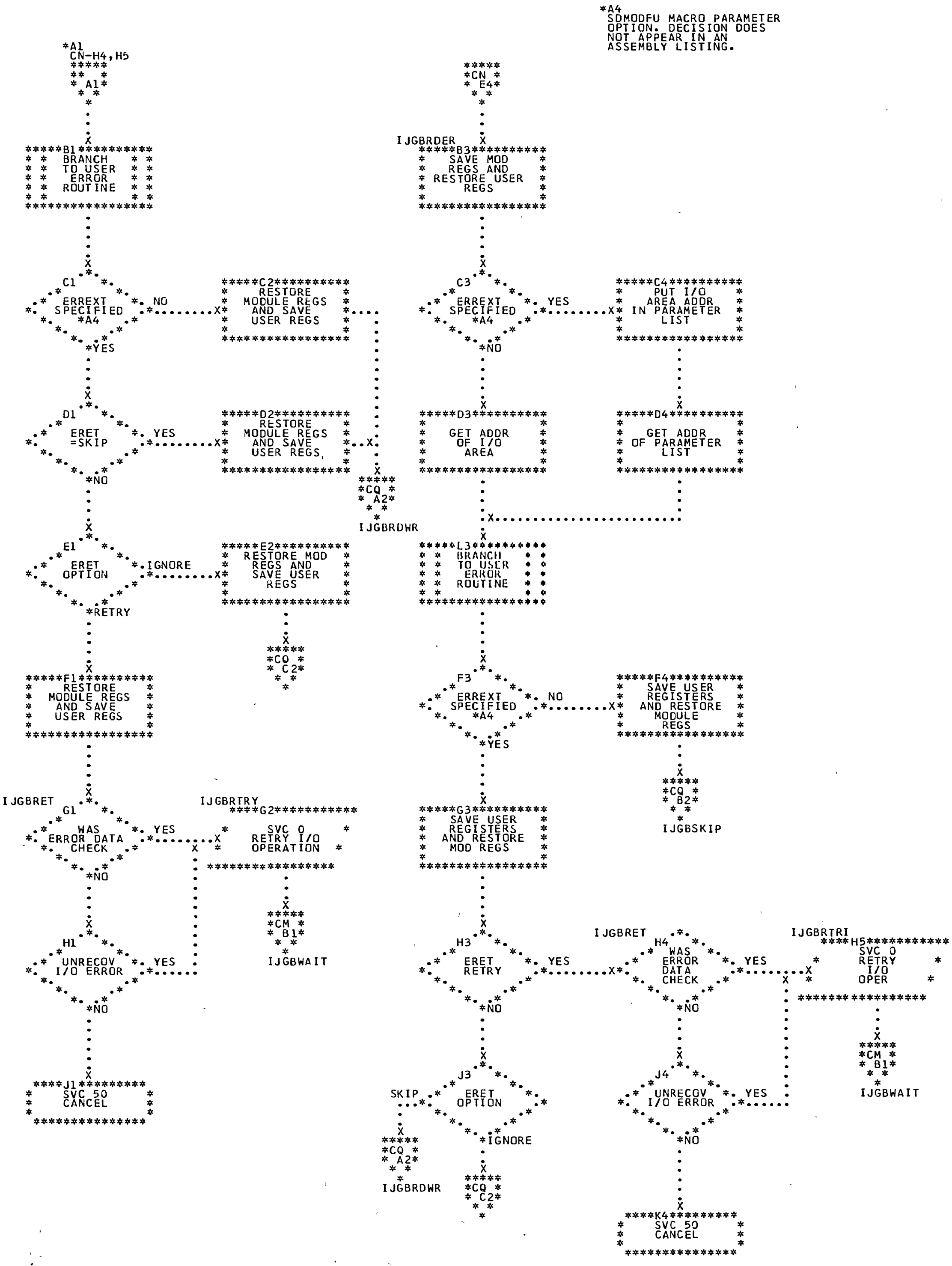

*A4
SDMODFU MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
*A1
CN-H4,H5
A1
B1
BRANCH TO USER ERROR ROUTINE
C1
ERREXT SPECIFIED *A4
NO
YES
C2
RESTORE MODULE REGS AND SAVE USER REGS
D1
ERET =SKIP
YES
NO
D2
RESTORE MODULE REGS AND SAVE USER REGS
CQ
A2
IJGBRDWR
E1
ERET OPTION
IGNORE
RETRY
E2
RESTORE MOD REGS AND SAVE USER REGS
CQ
C2
F1
RESTORE MODULE REGS AND SAVE USER REGS
IJGBRET
G1
WAS ERROR DATA CHECK
YES
NO
IJGBRTRY
G2
SVC 0 RETRY I/O OPERATION
CM
B1
IJGBWAIT
H1
UNRECOV I/O ERROR
YES
NO
J1
SVC 50 CANCEL
CN
E4
IJGBRDER
B3
SAVE MOD REGS AND RESTORE USER REGS
C3
ERREXT SPECIFIED *A4
YES
NO
C4
PUT I/O AREA ADDR IN PARAMETER LIST
D3
GET ADDR OF I/O AREA
D4
GET ADDR OF PARAMETER LIST
L3
BRANCH TO USER ERROR ROUTINE
F3
ERREXT SPECIFIED *A4
NO
YES
F4
SAVE USER REGISTERS AND RESTORE MODULE REGS
CQ
B2
IJGBSKIP
G3
SAVE USER REGISTERS AND RESTORE MOD REGS
H3
ERET RETRY
YES
NO
IJGBRET
H4
WAS ERROR DATA CHECK
YES
NO
IJGBRTRI
H5
SVC 0 RETRY I/O OPER
CM
B1
IJGBWAIT
J3
ERET OPTION
SKIP
IGNORE
CQ
A2
IJGBRDWR
CQ
C2
J4
UNRECOV I/O ERROR
YES
NO
K4
SVC 50 CANCEL

Chart CQ. SDMODFU: Without Truncation, GET Macro (Section 8 of 9)

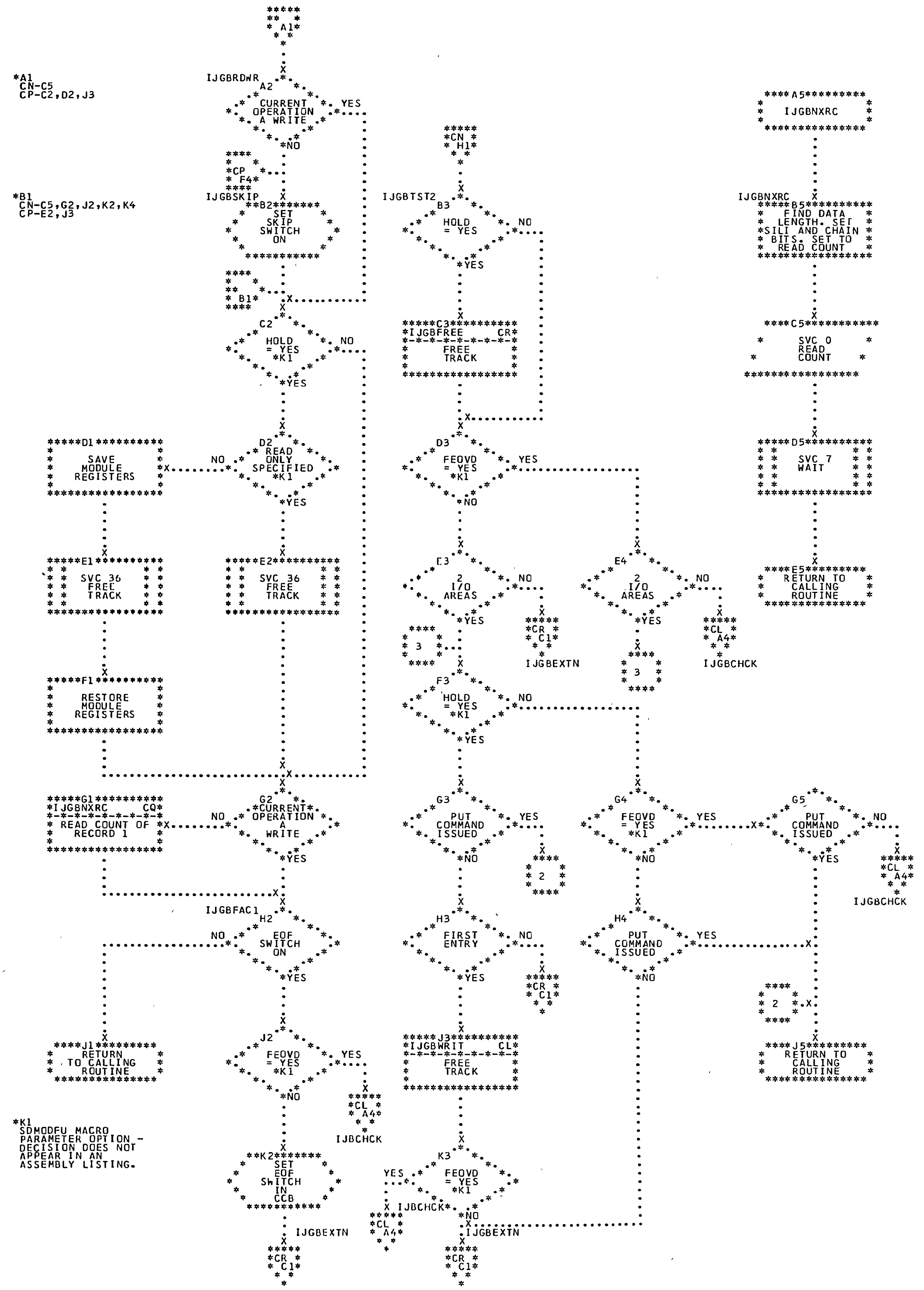
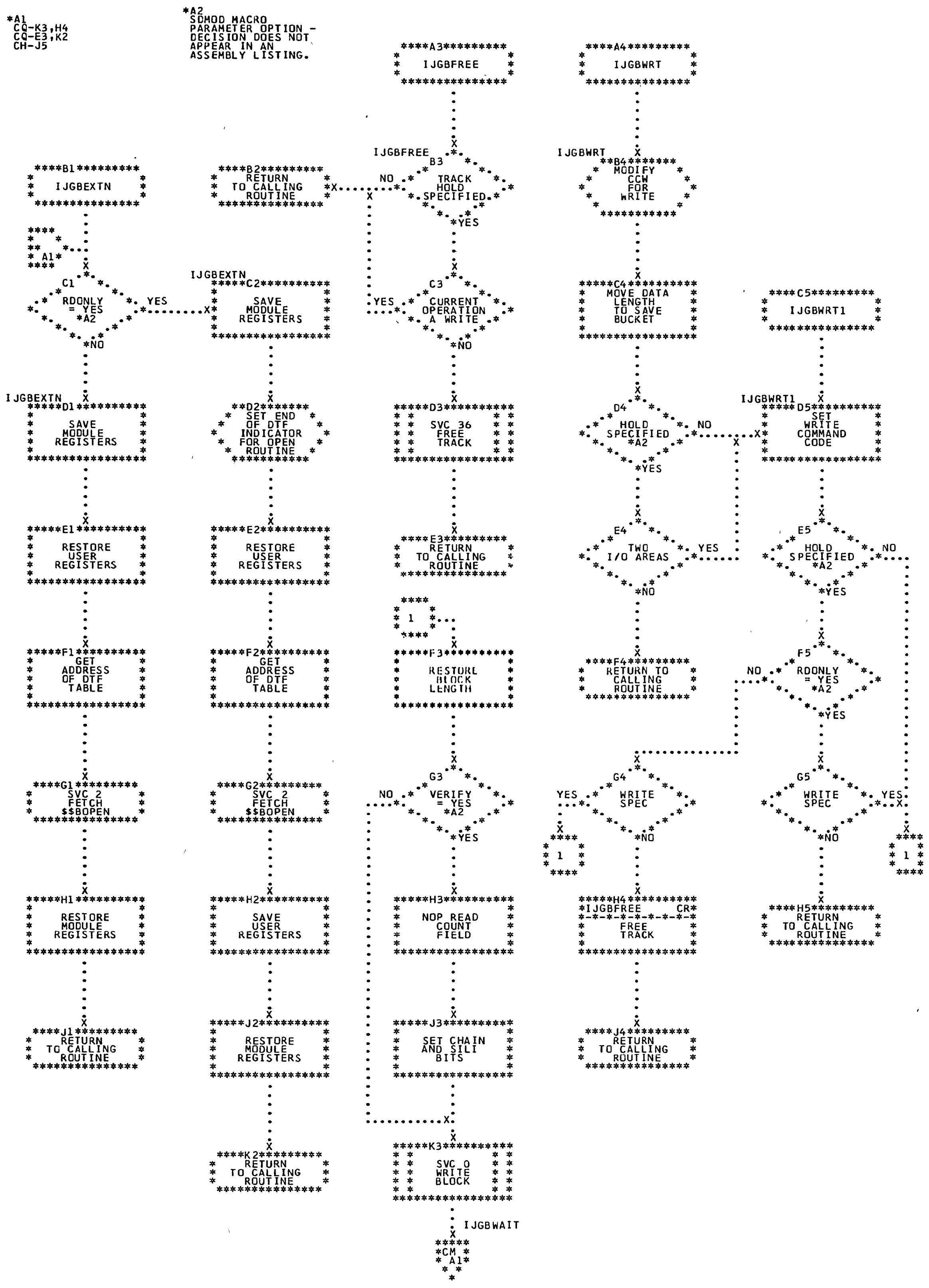
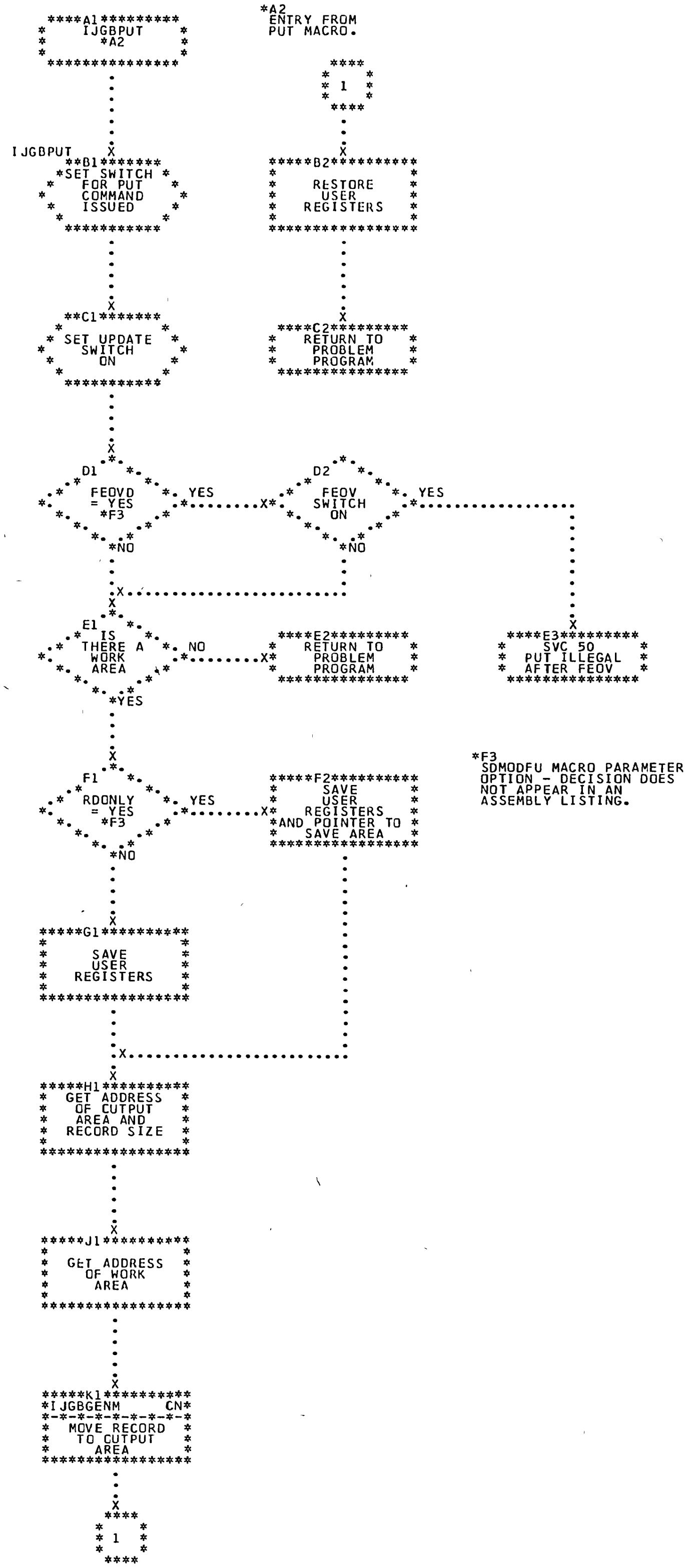


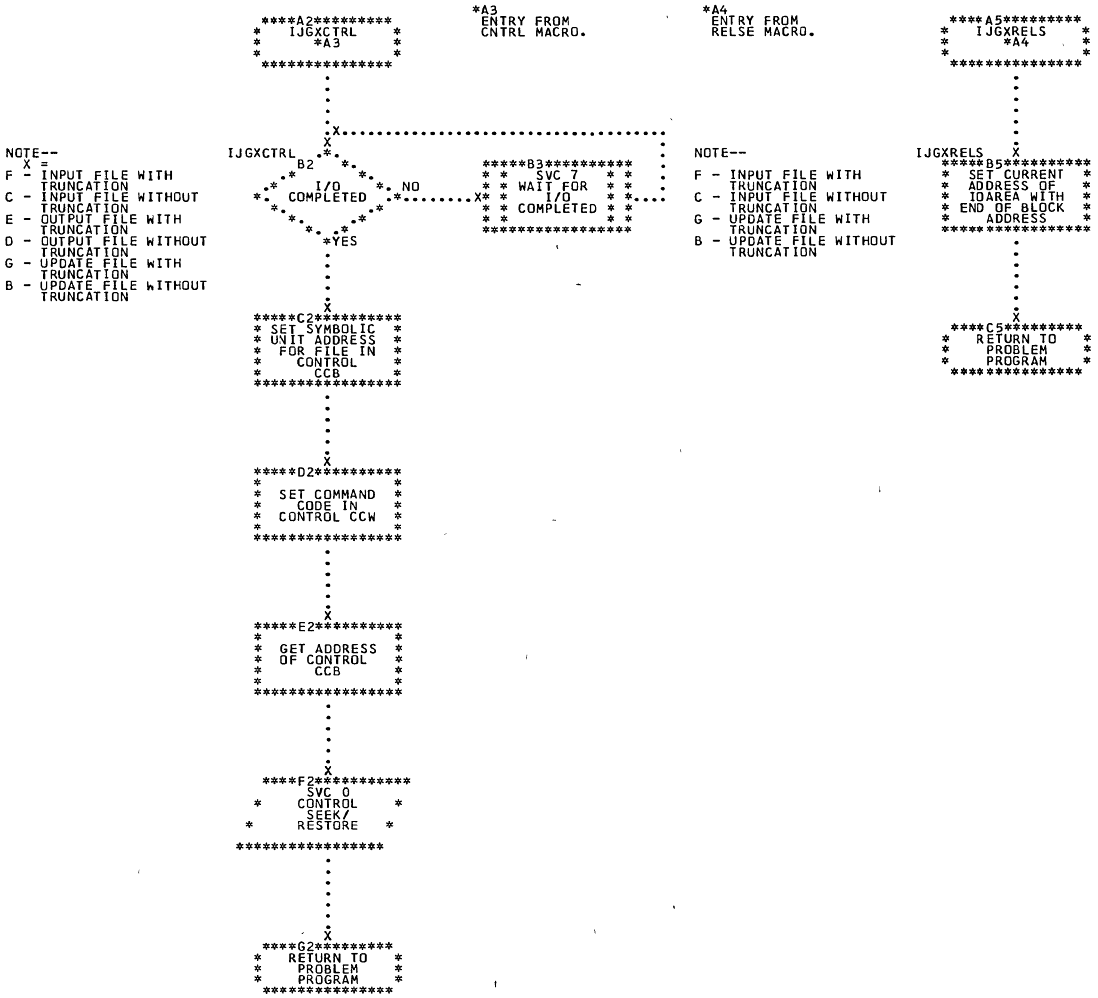
Chart CR. SDMODFU: Without Truncation, GET Macro (Section 9 of 9)


*A1
CQ-K3,H4
CQ-E3,K2
CH-J5
*A2
SDMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
A3 IJGBFREE
A4 IJGBWRT
B1 IJGBEXTN
B2 RETURN TO CALLING ROUTINE
IJGBFREE B3 TRACK HOLD SPECIFIED
NO
YES
IJGBWRT B4 MODIFY CCW FOR WRITE
A1
C1 RDONLY = YES *A2
YES
NO
IJGBEXTN C2 SAVE MODULE REGISTERS
C3 CURRENT OPERATION A WRITE
YES
NO
C4 MOVE DATA LENGTH TO SAVE BUCKET
C5 IJGBWRT1
IJGBEXTN D1 SAVE MODULE REGISTERS
D2 SET END OF DTF INDICATOR FOR OPEN ROUTINE
D3 SVC 36 FREE TRACK
D4 HOLD SPECIFIED *A2
NO
YES
IJGBWRT1 D5 SET WRITE COMMAND CODE
E1 RESTORE USER REGISTERS
E2 RESTORE USER REGISTERS
E3 RETURN TO CALLING ROUTINE
E4 TWO I/O AREAS
YES
NO
E5 HOLD SPECIFIED *A2
NO
YES
1
F1 GET ADDRESS OF DTF TABLE
F2 GET ADDRESS OF DTF TABLE
F3 RESTORE BLOCK LENGTH
F4 RETURN TO CALLING ROUTINE
F5 RDONLY = YES *A2
NO
YES
G1 SVC 2 FETCH $$BOPEN
G2 SVC 2 FETCH $$BOPEN
G3 VERIFY = YES *A2
NO
YES
G4 WRITE SPEC
YES
NO
1
G5 WRITE SPEC
YES
NO
1
H1 RESTORE MODULE REGISTERS
H2 SAVE USER REGISTERS
H3 NOP READ COUNT FIELD
H4 IJGBFREE CR FREE TRACK
H5 RETURN TO CALLING ROUTINE
J1 RETURN TO CALLING ROUTINE
J2 RESTORE MODULE REGISTERS
J3 SET CHAIN AND SILI BITS
J4 RETURN TO CALLING ROUTINE
K2 RETURN TO CALLING ROUTINE
K3 SVC 0 WRITE BLOCK
IJGBWAIT
CM A1

## Chart CS. SDMODFU: Without Truncation, PUT Macro



## Chart CT. CNTRL, RELSE Macros: Fixed-Length Record Modules



A2
IJGXCTRL
*A3
*A3
ENTRY FROM
CNTRL MACRO.
*A4
ENTRY FROM
RELSE MACRO.
A5
IJGXRELS
*A4
NOTE--
X =
F - INPUT FILE WITH TRUNCATION
C - INPUT FILE WITHOUT TRUNCATION
E - OUTPUT FILE WITH TRUNCATION
D - OUTPUT FILE WITHOUT TRUNCATION
G - UPDATE FILE WITH TRUNCATION
B - UPDATE FILE WITHOUT TRUNCATION
IJGXCTRL
B2
I/O COMPLETED
NO
YES
B3
SVC 7
WAIT FOR
I/O
COMPLETED
NOTE--
F - INPUT FILE WITH TRUNCATION
C - INPUT FILE WITHOUT TRUNCATION
G - UPDATE FILE WITH TRUNCATION
B - UPDATE FILE WITHOUT TRUNCATION
IJGXRELS
X
B5
SET CURRENT ADDRESS OF IOAREA WITH END OF BLOCK ADDRESS
C5
RETURN TO PROBLEM PROGRAM
C2
SET SYMBOLIC UNIT ADDRESS FOR FILE IN CONTROL CCB
D2
SET COMMAND CODE IN CONTROL CCW
E2
GET ADDRESS OF CONTROL CCB
F2
SVC 0
CONTROL
SEEK/
RESTORE
G2
RETURN TO PROBLEM PROGRAM
X

Chart DA. SDMODVI: GET Macro (Section 1 of 6)

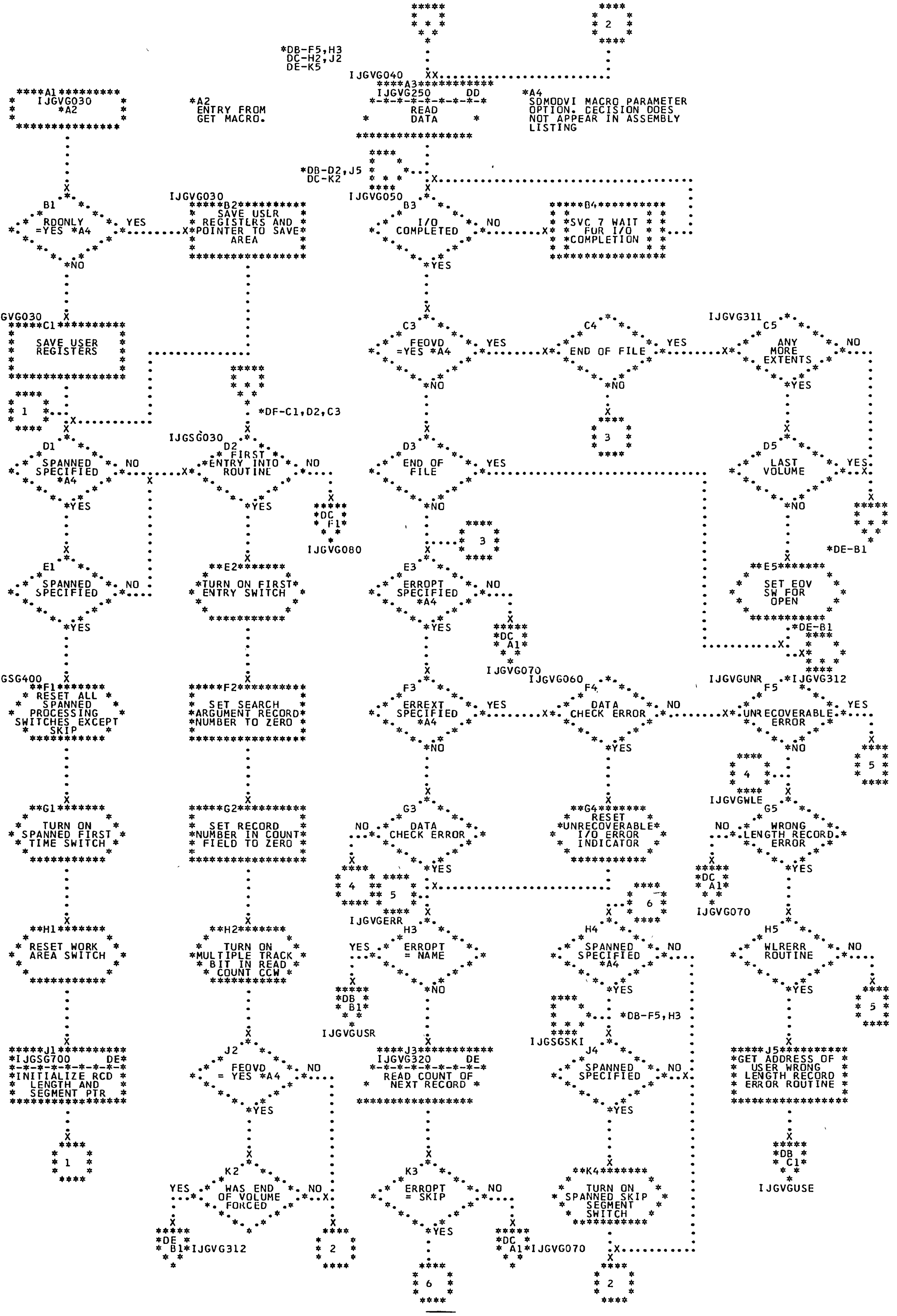
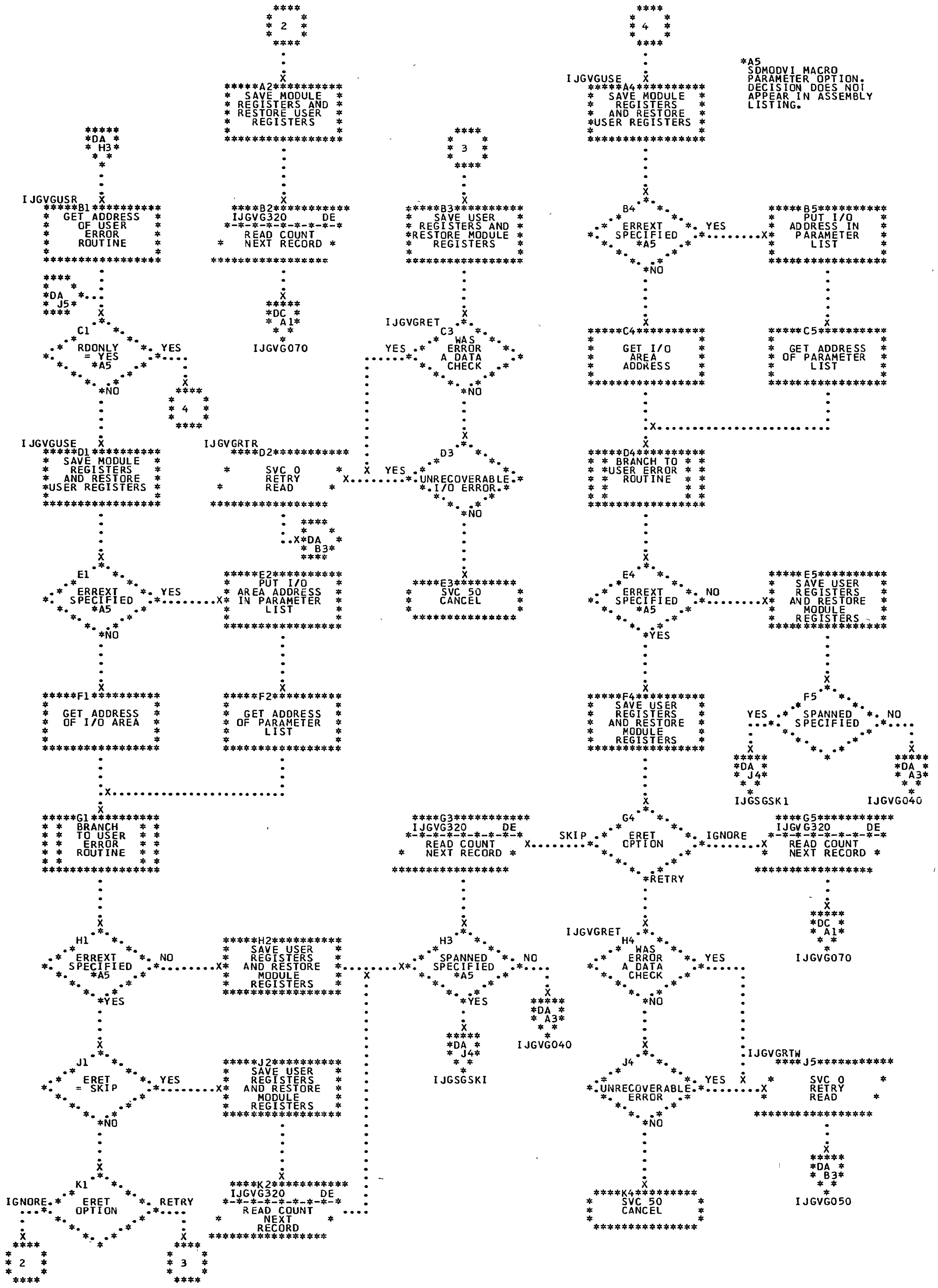


Chart DB. SDMODVI: GET Macro (Section 2 of 6)



## Chart DC. SDMODVI: GET Macro (Section 3 of 6)

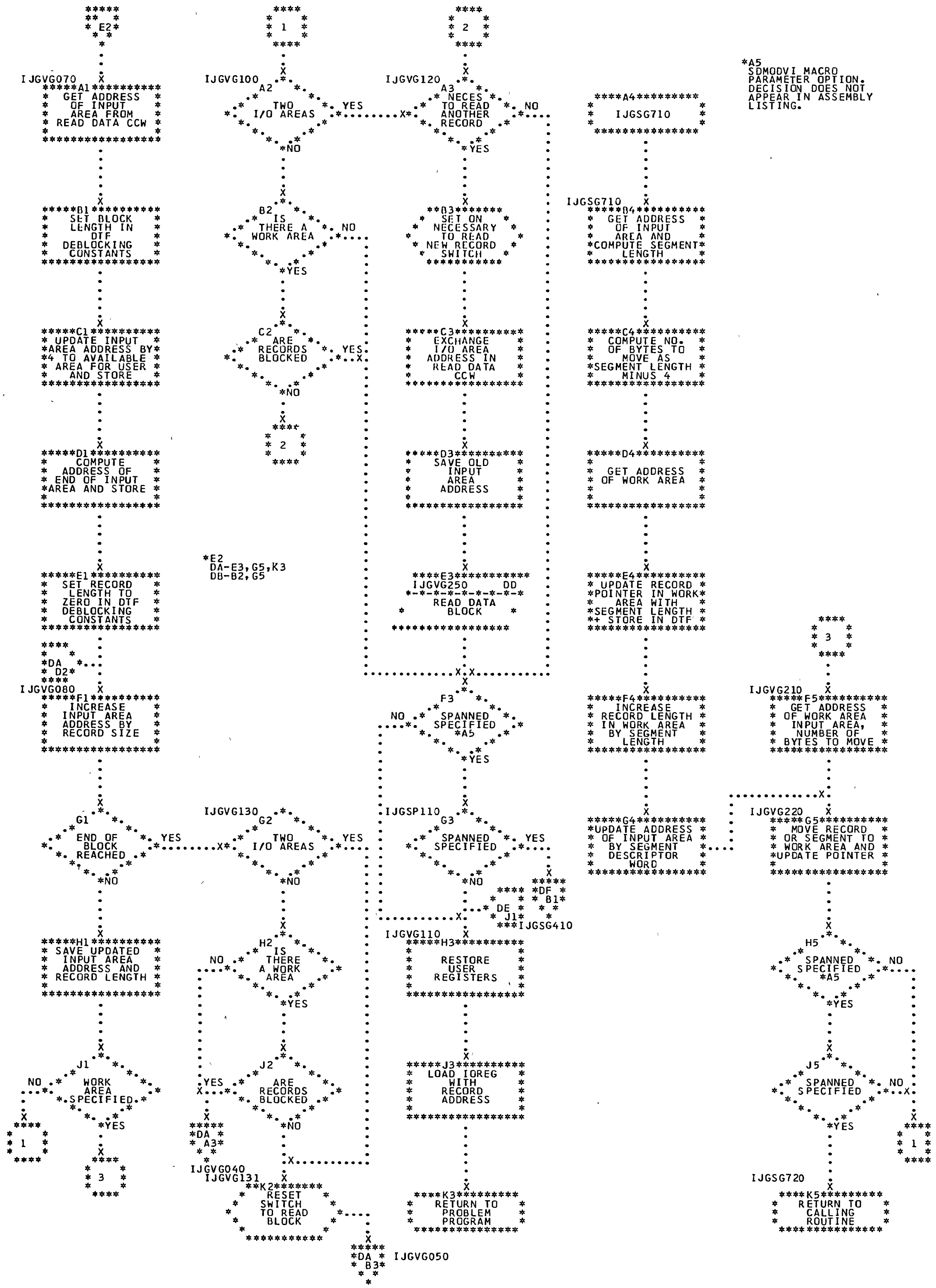

E2
IJGVG070
A1
GET ADDRESS OF INPUT AREA FROM READ DATA CCW
B1
SET BLOCK LENGTH IN DTF DEBLOCKING CONSTANTS
C1
UPDATE INPUT AREA ADDRESS BY 4 TO AVAILABLE AREA FOR USER AND STORE
D1
COMPUTE ADDRESS OF END OF INPUT AREA AND STORE
E1
SET RECORD LENGTH TO ZERO IN DTF DEBLOCKING CONSTANTS
DA D2
IJGVG080
F1
INCREASE INPUT AREA ADDRESS BY RECORD SIZE
G1
END OF BLOCK REACHED
YES
NO
H1
SAVE UPDATED INPUT AREA ADDRESS AND RECORD LENGTH
J1
WORK AREA SPECIFIED
NO
YES
1
3
1
IJGVG100
A2
TWO I/O AREAS
YES
NO
B2
IS THERE A WORK AREA
NO
YES
C2
ARE RECORDS BLOCKED
YES
NO
2
*E2
DA-E3,G5,K3
DB-B2,G5
IJGVG130
G2
TWO I/O AREAS
YES
NO
H2
IS THERE A WORK AREA
NO
YES
J2
ARE RECORDS BLOCKED
YES
NO
DA A3
IJGVG040
IJGVG131
K2
RESET SWITCH TO READ BLOCK
DA B3
IJGVG050
2
IJGVG120
A3
NECES TO READ ANOTHER RECORD
NO
YES
B3
SET ON NECESSARY TO READ NEW RECORD SWITCH
C3
EXCHANGE I/O AREA ADDRESS IN READ DATA CCW
D3
SAVE OLD INPUT AREA ADDRESS
E3
IJGVG250 DD
READ DATA BLOCK
F3
SPANNED SPECIFIED *A5
NO
YES
IJGSP110
G3
SPANNED SPECIFIED
YES
NO
DF B1
DE J1
IJGSG410
IJGVG110
H3
RESTORE USER REGISTERS
J3
LOAD IOREG WITH RECORD ADDRESS
K3
RETURN TO PROBLEM PROGRAM
A4
IJGSG710
IJGSG710
B4
GET ADDRESS OF INPUT AREA AND COMPUTE SEGMENT LENGTH
C4
COMPUTE NO. OF BYTES TO MOVE AS SEGMENT LENGTH MINUS 4
D4
GET ADDRESS OF WORK AREA
E4
UPDATE RECORD POINTER IN WORK AREA WITH SEGMENT LENGTH + STORE IN DTF
F4
INCREASE RECORD LENGTH IN WORK AREA BY SEGMENT LENGTH
G4
UPDATE ADDRESS OF INPUT AREA BY SEGMENT DESCRIPTOR WORD
*A5
SDMODVI MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.
3
IJGVG210
F5
GET ADDRESS OF WORK AREA INPUT AREA, NUMBER OF BYTES TO MOVE
IJGVG220
G5
MOVE RECORD OR SEGMENT TO WORK AREA AND UPDATE POINTER
H5
SPANNED SPECIFIED *A5
NO
YES
J5
SPANNED SPECIFIED
NO
YES
1
IJGSG720
K5
RETURN TO CALLING ROUTINE

Chart DD. SDMODVI: GET Macro (Section 4 of 6)

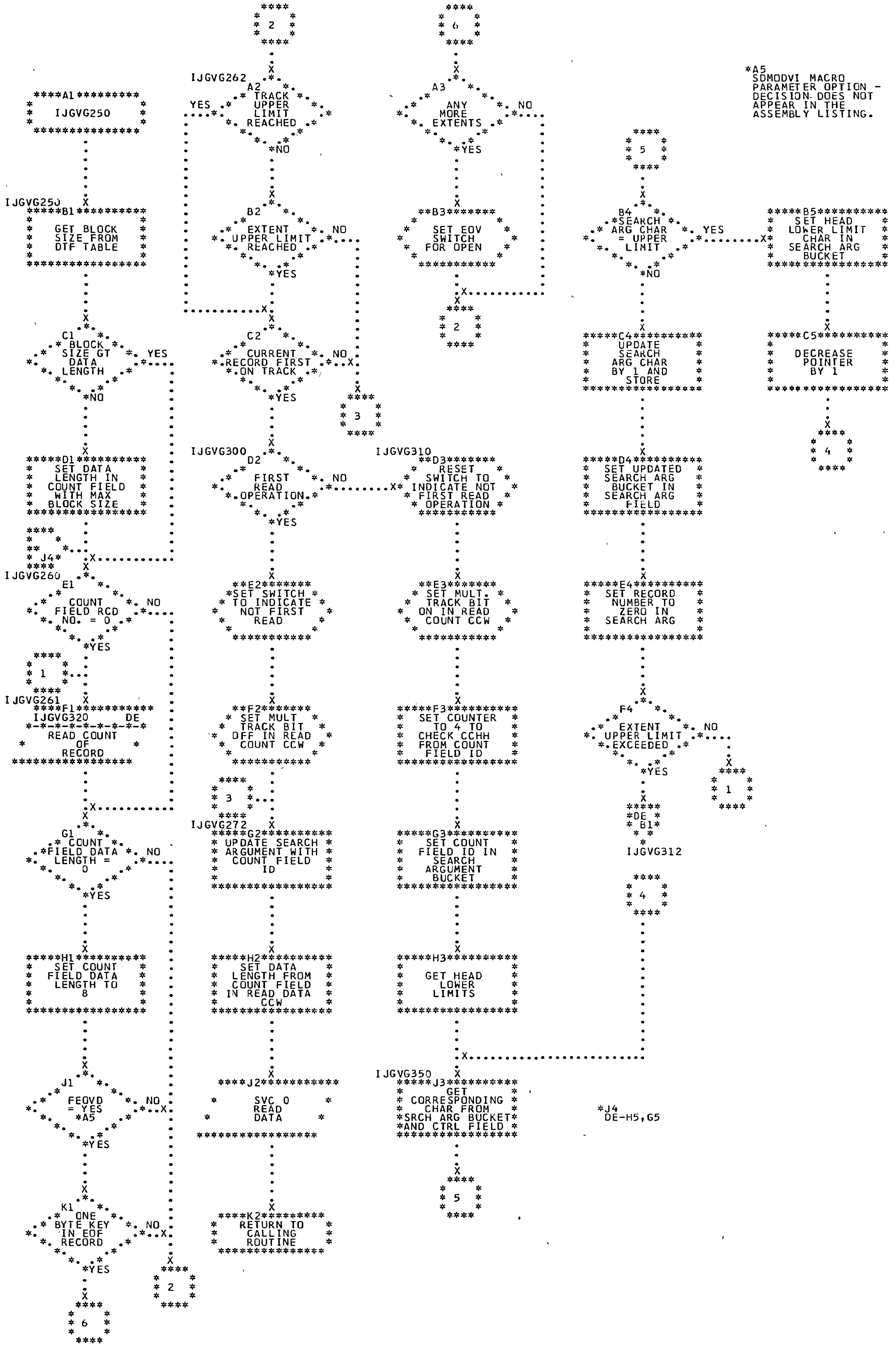
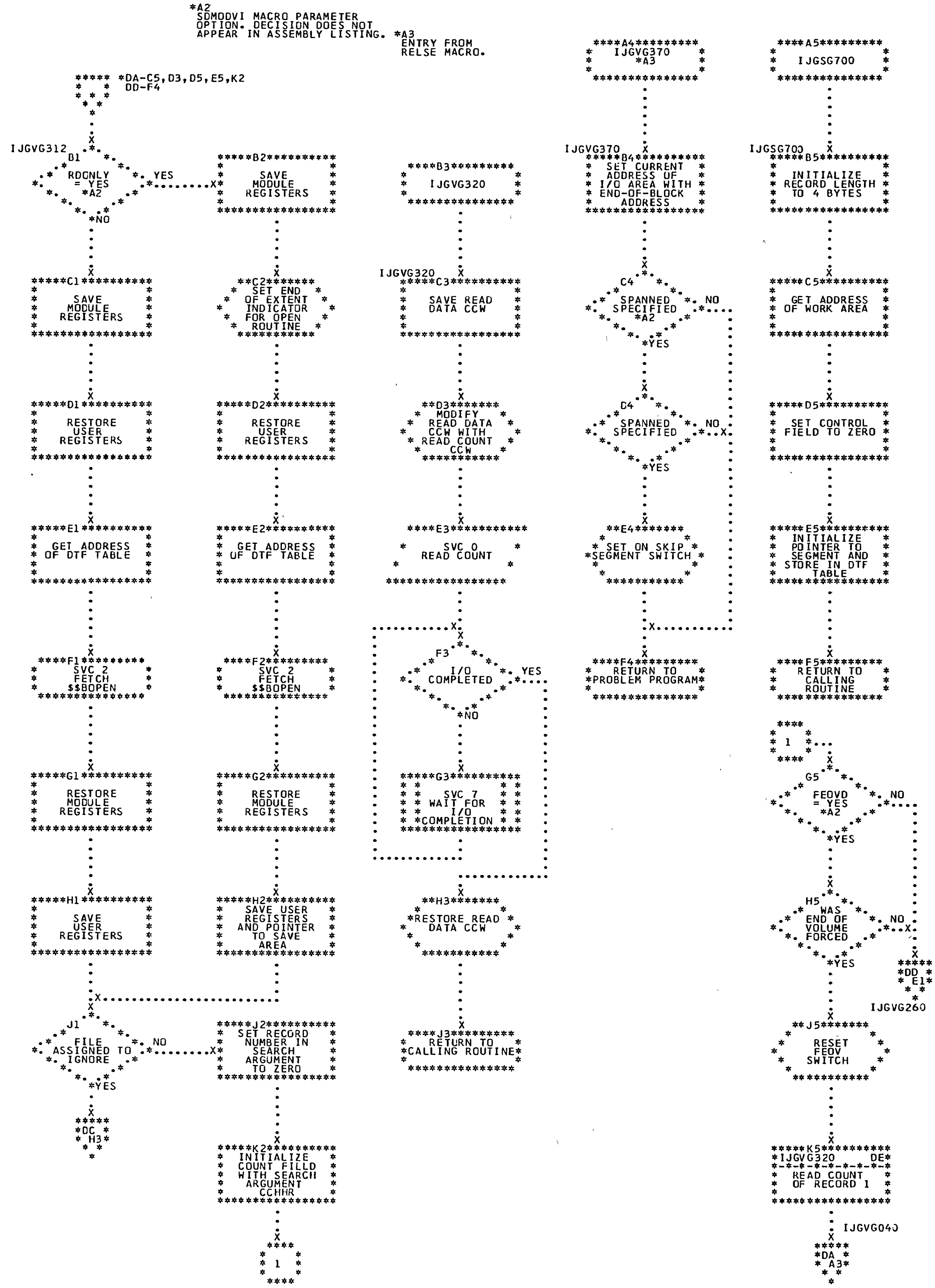
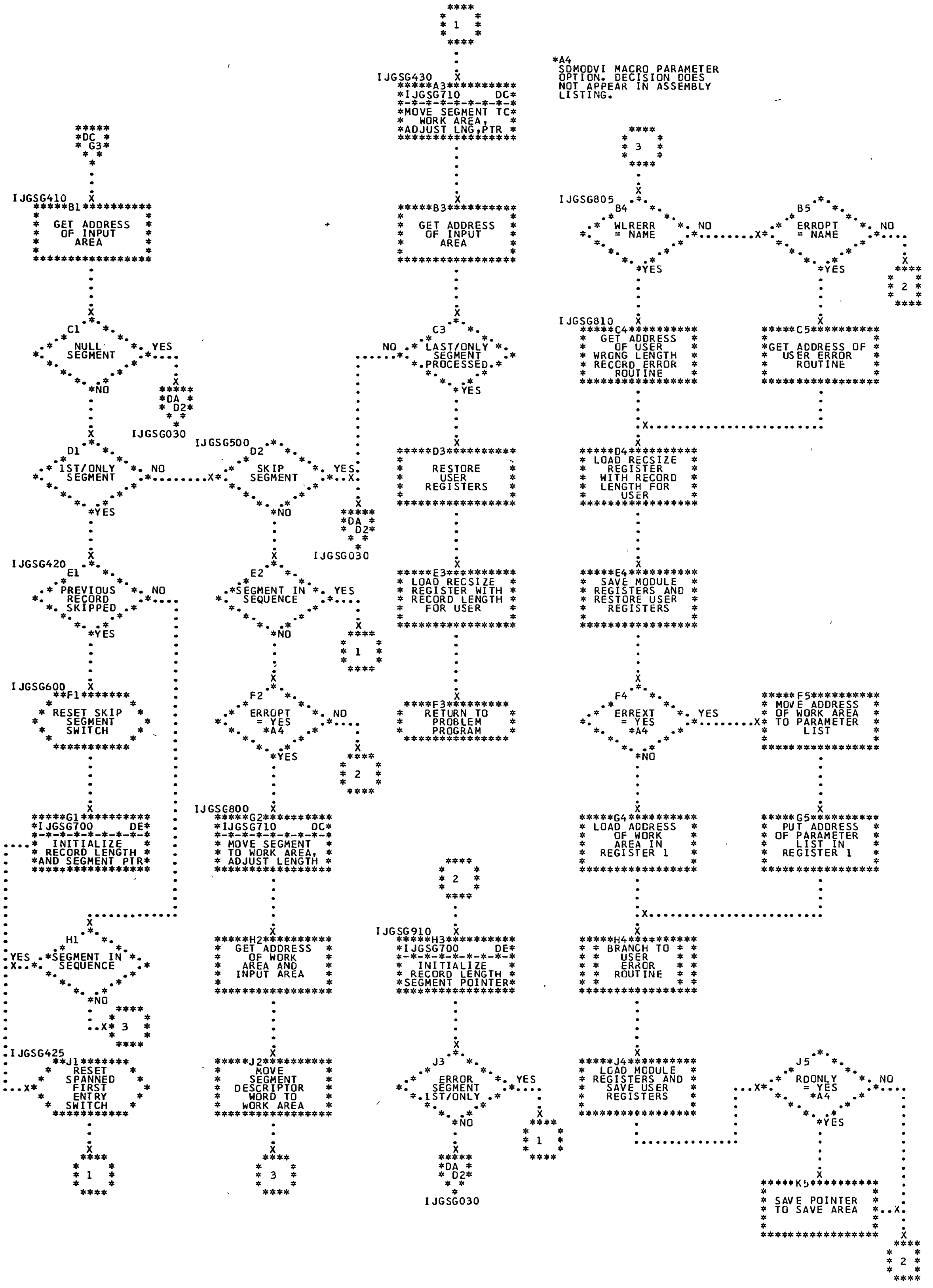

2
IJGVG262
A2
TRACK UPPER LIMIT REACHED
YES
NO
6
A3
ANY MORE EXTENTS
NO
YES
*A5
SDMODVI MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN THE ASSEMBLY LISTING.
A1
IJGVG250
IJGVG250
B1
GET BLOCK SIZE FROM DTF TABLE
B2
EXTENT UPPER LIMIT REACHED
NO
YES
B3
SET EOV SWITCH FOR OPEN
5
B4
SEARCH ARG CHAR = UPPER LIMIT
YES
NO
B5
SET HEAD LOWER LIMIT CHAR IN SEARCH ARG BUCKET
C1
BLOCK SIZE GT DATA LENGTH
YES
NO
C2
CURRENT RECORD FIRST ON TRACK
NO
YES
2
3
C4
UPDATE SEARCH ARG CHAR BY 1 AND STORE
C5
DECREASE POINTER BY 1
4
D1
SET DATA LENGTH IN COUNT FIELD WITH MAX BLOCK SIZE
IJGVG300
D2
FIRST READ OPERATION
NO
YES
IJGVG310
D3
RESET SWITCH TO INDICATE NOT FIRST READ OPERATION
D4
SET UPDATED SEARCH ARG BUCKET IN SEARCH ARG FIELD
J4
IJGVG260
E1
COUNT FIELD RCD NO. = 0
NO
YES
E2
SET SWITCH TO INDICATE NOT FIRST READ
E3
SET MULT. TRACK BIT ON IN READ COUNT CCW
E4
SET RECORD NUMBER TO ZERO IN SEARCH ARG
1
IJGVG261
F1
IJGVG320 DE
READ COUNT OF RECORD
F2
SET MULT TRACK BIT OFF IN READ COUNT CCW
F3
SET COUNTER TO 4 TO CHECK CCHH FROM COUNT FIELD ID
F4
EXTENT UPPER LIMIT EXCEEDED
NO
YES
1
3
DE B1
IJGVG312
IJGVG272
G1
COUNT FIELD DATA LENGTH = 0
NO
YES
G2
UPDATE SEARCH ARGUMENT WITH COUNT FIELD ID
G3
SET COUNT FIELD ID IN SEARCH ARGUMENT BUCKET
4
H1
SET COUNT FIELD DATA LENGTH TO 8
H2
SET DATA LENGTH FROM COUNT FIELD IN READ DATA CCW
H3
GET HEAD LOWER LIMITS
J1
FEOVD = YES *A5
NO
YES
J2
SVC 0 READ DATA
IJGVG350
J3
GET CORRESPONDING CHAR FROM SRCH ARG BUCKET AND CTRL FIELD
*J4
DE-H5,G5
5
K1
ONE BYTE KEY IN EOF RECORD
NO
YES
K2
RETURN TO CALLING ROUTINE
2
6

Chart DE. SDMODVI: GET Macro (Section 5 of 6)


*A2 SDMODVI MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.
*A3 ENTRY FROM RELSE MACRO.
*DA-C5,D3,D5,E5,K2 DD-F4
IJGVG312
B1 RDONLY = YES *A2
YES
NO
B2 SAVE MODULE REGISTERS
C1 SAVE MODULE REGISTERS
C2 SET END OF EXTENT INDICATOR FOR OPEN ROUTINE
D1 RESTORE USER REGISTERS
D2 RESTORE USER REGISTERS
E1 GET ADDRESS OF DTF TABLE
E2 GET ADDRESS OF DTF TABLE
F1 SVC 2 FETCH $$BOPEN
F2 SVC 2 FETCH $$BOPEN
G1 RESTORE MODULE REGISTERS
G2 RESTORE MODULE REGISTERS
H1 SAVE USER REGISTERS
H2 SAVE USER REGISTERS AND POINTER TO SAVE AREA
J1 FILE ASSIGNED TO IGNORE
NO
YES
DC H3
J2 SET RECORD NUMBER IN SEARCH ARGUMENT TO ZERO
K2 INITIALIZE COUNT FIELD WITH SEARCH ARGUMENT CCHHR
1
B3 IJGVG320
IJGVG320
C3 SAVE READ DATA CCW
D3 MODIFY READ DATA CCW WITH READ COUNT CCW
E3 SVC 0 READ COUNT
F3 I/O COMPLETED
YES
NO
G3 SVC 7 WAIT FOR I/O *COMPLETION
H3 RESTORE READ DATA CCW
J3 RETURN TO CALLING ROUTINE
A4 IJGVG370 *A3
IJGVG370
B4 SET CURRENT ADDRESS OF I/O AREA WITH END-OF-BLOCK ADDRESS
C4 SPANNED SPECIFIED *A2
NO
YES
D4 SPANNED SPECIFIED
NO
YES
E4 SET ON SKIP SEGMENT SWITCH
F4 RETURN TO PROBLEM PROGRAM
A5 IJGSG700
IJGSG700
B5 INITIALIZE RECORD LENGTH TO 4 BYTES
C5 GET ADDRESS OF WORK AREA
D5 SET CONTROL FIELD TO ZERO
E5 INITIALIZE POINTER TO SEGMENT AND STORE IN DTF TABLE
F5 RETURN TO CALLING ROUTINE
1
G5 FEOVD = YES *A2
NO
YES
H5 WAS END OF VOLUME FORCED
NO
YES
DD E1
IJGVG260
J5 RESET FEOV SWITCH
K5 IJGVG320 DE READ COUNT OF RECORD 1
IJGVG040
DA A3

## Chart DF. SDMODVI: Get Macro (Section 6 of 6)



*A4
SDMODVI MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.
DC G3
IJGSG410
B1
GET ADDRESS OF INPUT AREA
C1
NULL SEGMENT
YES
NO
DA D2
IJGSG030
D1
1ST/ONLY SEGMENT
NO
YES
IJGSG420
E1
PREVIOUS RECORD SKIPPED
NO
YES
IJGSG600
F1
RESET SKIP SEGMENT SWITCH
G1
IJGSG700 DE
INITIALIZE RECORD LENGTH AND SEGMENT PTR
H1
SEGMENT IN SEQUENCE
YES
NO
3
IJGSG425
J1
RESET SPANNED FIRST ENTRY SWITCH
1
IJGSG500
D2
SKIP SEGMENT
YES
NO
DA D2
IJGSG030
E2
SEGMENT IN SEQUENCE
YES
NO
1
F2
ERROPT = YES *A4
NO
YES
2
IJGSG800
G2
IJGSG710 DC
MOVE SEGMENT TO WORK AREA, ADJUST LENGTH
H2
GET ADDRESS OF WORK AREA AND INPUT AREA
J2
MOVE SEGMENT DESCRIPTOR WORD TO WORK AREA
3
1
IJGSG430
A3
IJGSG710 DC
MOVE SEGMENT TC WORK AREA, ADJUST LNG,PTR
B3
GET ADDRESS OF INPUT AREA
C3
LAST/ONLY SEGMENT PROCESSED
NO
YES
D3
RESTORE USER REGISTERS
E3
LOAD RECSIZE REGISTER WITH RECORD LENGTH FOR USER
F3
RETURN TO PROBLEM PROGRAM
2
IJGSG910
H3
IJGSG700 DE
INITIALIZE RECORD LENGTH SEGMENT POINTER
J3
ERROR SEGMENT 1ST/ONLY
YES
NO
1
DA D2
IJGSG030
3
IJGSG805
B4
WLRERR = NAME
NO
YES
IJGSG810
C4
GET ADDRESS OF USER WRONG LENGTH RECORD ERROR ROUTINE
D4
LOAD RECSIZE REGISTER WITH RECORD LENGTH FOR USER
E4
SAVE MODULE REGISTERS AND RESTORE USER REGISTERS
F4
ERREXT = YES *A4
YES
NO
G4
LOAD ADDRESS OF WORK AREA IN REGISTER 1
H4
BRANCH TO USER ERROR ROUTINE
J4
LOAD MODULE REGISTERS AND SAVE USER REGISTERS
B5
ERROPT = NAME
NO
YES
2
C5
GET ADDRESS OF USER ERROR ROUTINE
F5
MOVE ADDRESS OF WORK AREA TO PARAMETER LIST
G5
PUT ADDRESS OF PARAMETER LIST IN REGISTER 1
J5
RDONLY = YES *A4
NO
YES
K5
SAVE POINTER TO SAVE AREA
2

## Chart DG. SDMODVO: PUT Macro (Section 1 of 11)

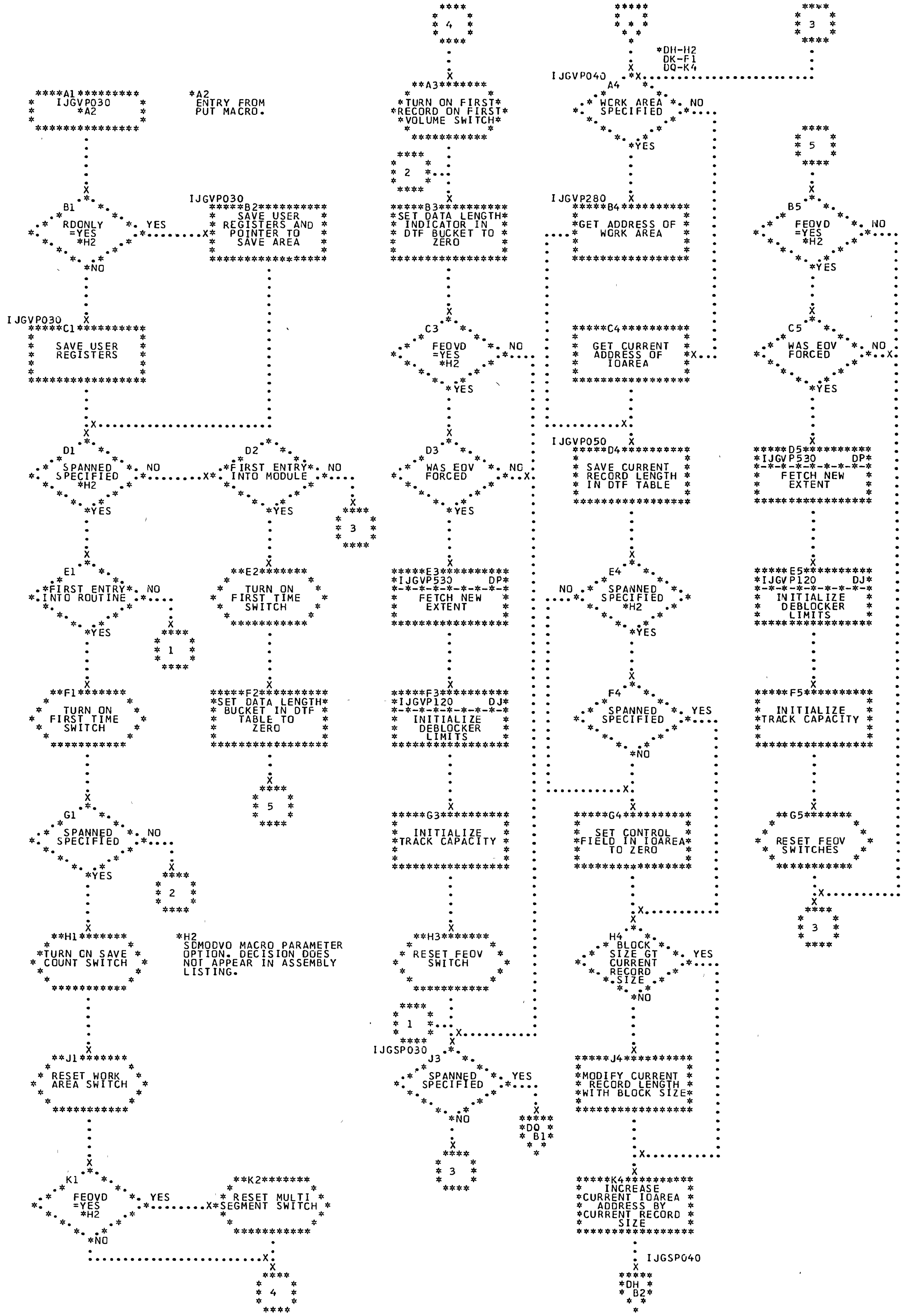

A1 IJGVP030 *A2
*A2 ENTRY FROM PUT MACRO.
B1 RDONLY =YES *H2
YES
NO
IJGVP030 B2 SAVE USER REGISTERS AND POINTER TO SAVE AREA
IJGVP030 C1 SAVE USER REGISTERS
D1 SPANNED SPECIFIED *H2
NO
YES
D2 FIRST ENTRY INTO MODULE
NO
YES
3
E1 FIRST ENTRY INTO ROUTINE
NO
YES
1
E2 TURN ON FIRST TIME SWITCH
F1 TURN ON FIRST TIME SWITCH
F2 SET DATA LENGTH BUCKET IN DTF TABLE TO ZERO
5
G1 SPANNED SPECIFIED
NO
YES
2
H1 TURN CN SAVE COUNT SWITCH
*H2 SDMODVO MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.
J1 RESET WORK AREA SWITCH
K1 FEOVD =YES *H2
YES
NO
K2 RESET MULTI SEGMENT SWITCH
4
4
A3 TURN ON FIRST RECORD ON FIRST VOLUME SWITCH
2
B3 SET DATA LENGTH INDICATOR IN DTF BUCKET TO ZERO
C3 FEOVD =YES *H2
NO
YES
D3 WAS EOV FORCED
NO
YES
E3 IJGVP530 DP FETCH NEW EXTENT
F3 IJGVP120 DJ INITIALIZE DEBLOCKER LIMITS
G3 INITIALIZE TRACK CAPACITY
H3 RESET FEOV SWITCH
1
IJGSP030 J3 SPANNED SPECIFIED
YES
NO
DQ B1
3
*DH-H2 DK-F1 DQ-K4
IJGVP040 A4 WCRK AREA SPECIFIED
NO
YES
IJGVP280 B4 GET ADDRESS OF WORK AREA
C4 GET CURRENT ADDRESS OF IOAREA
IJGVP050 D4 SAVE CURRENT RECORD LENGTH IN DTF TABLE
E4 SPANNED SPECIFIED *H2
NO
YES
F4 SPANNED SPECIFIED
YES
NO
G4 SET CONTROL FIELD IN IOAREA TO ZERO
H4 BLOCK SIZE GT CURRENT RECORD SIZE
YES
NO
J4 MODIFY CURRENT RECORD LENGTH WITH BLOCK SIZE
K4 INCREASE CURRENT IOAREA ADDRESS BY CURRENT RECORD SIZE
IJGSP040
DH B2
3
5
B5 FEOVD =YES *H2
NO
YES
C5 WAS EOV FORCED
NO
YES
D5 IJGVP530 DP FETCH NEW EXTENT
E5 IJGVP120 DJ INITIALIZE DEBLOCKER LIMITS
F5 INITIALIZE TRACK CAPACITY
G5 RESET FEOV SWITCHES
3

Chart DH. SDMODVO: Put Macro (Section 2 of 11)

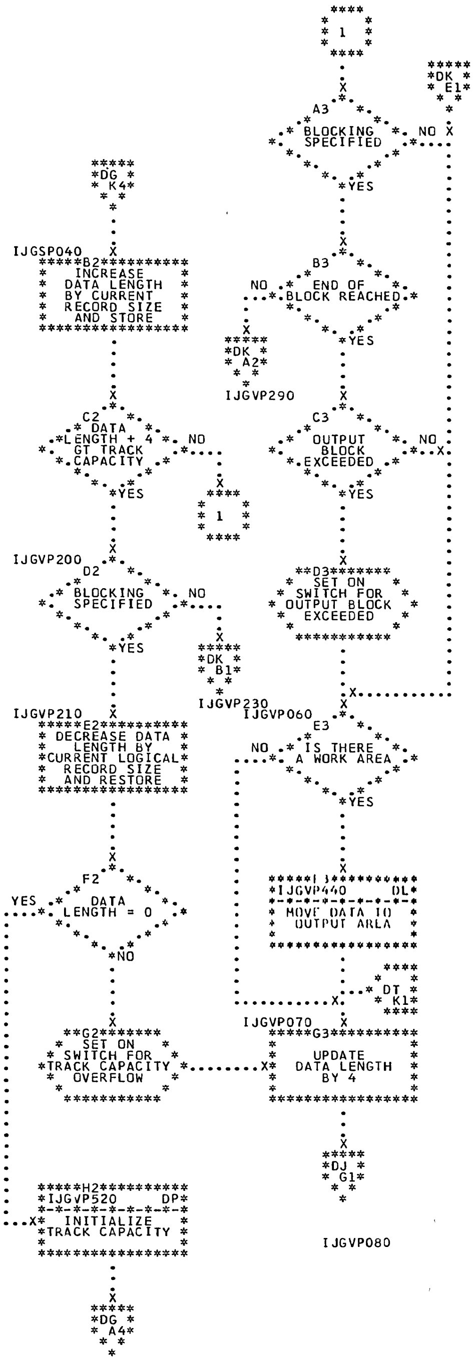
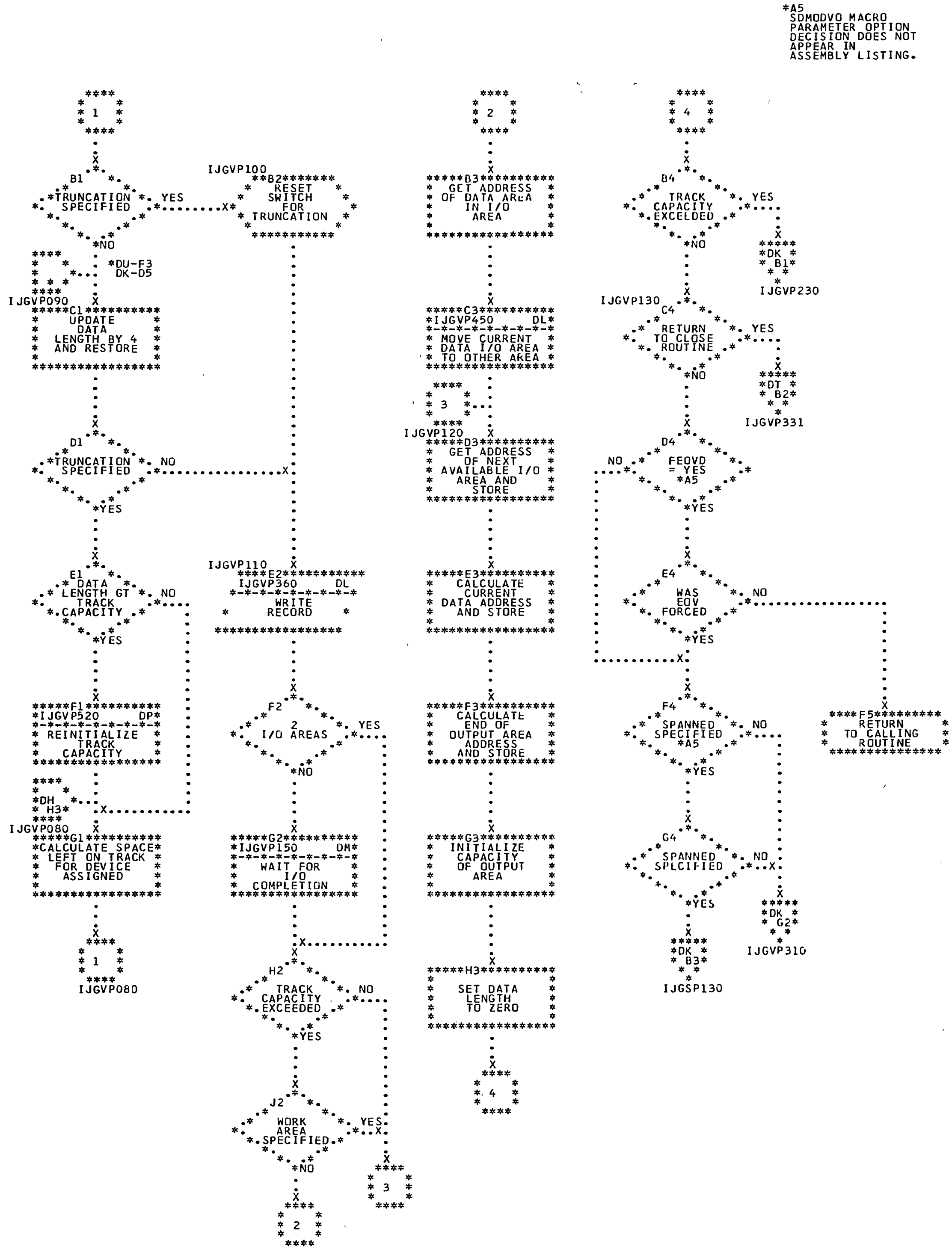
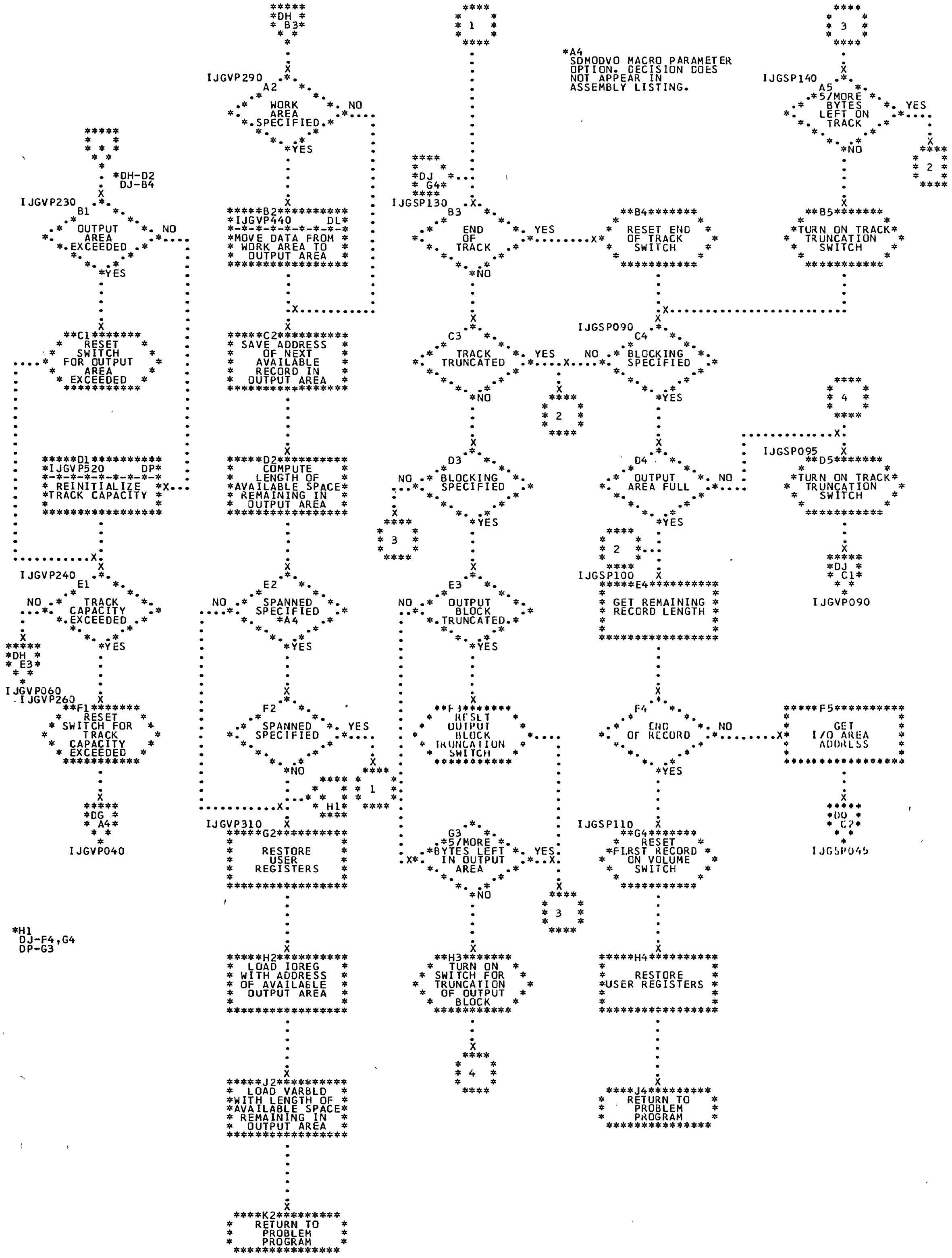
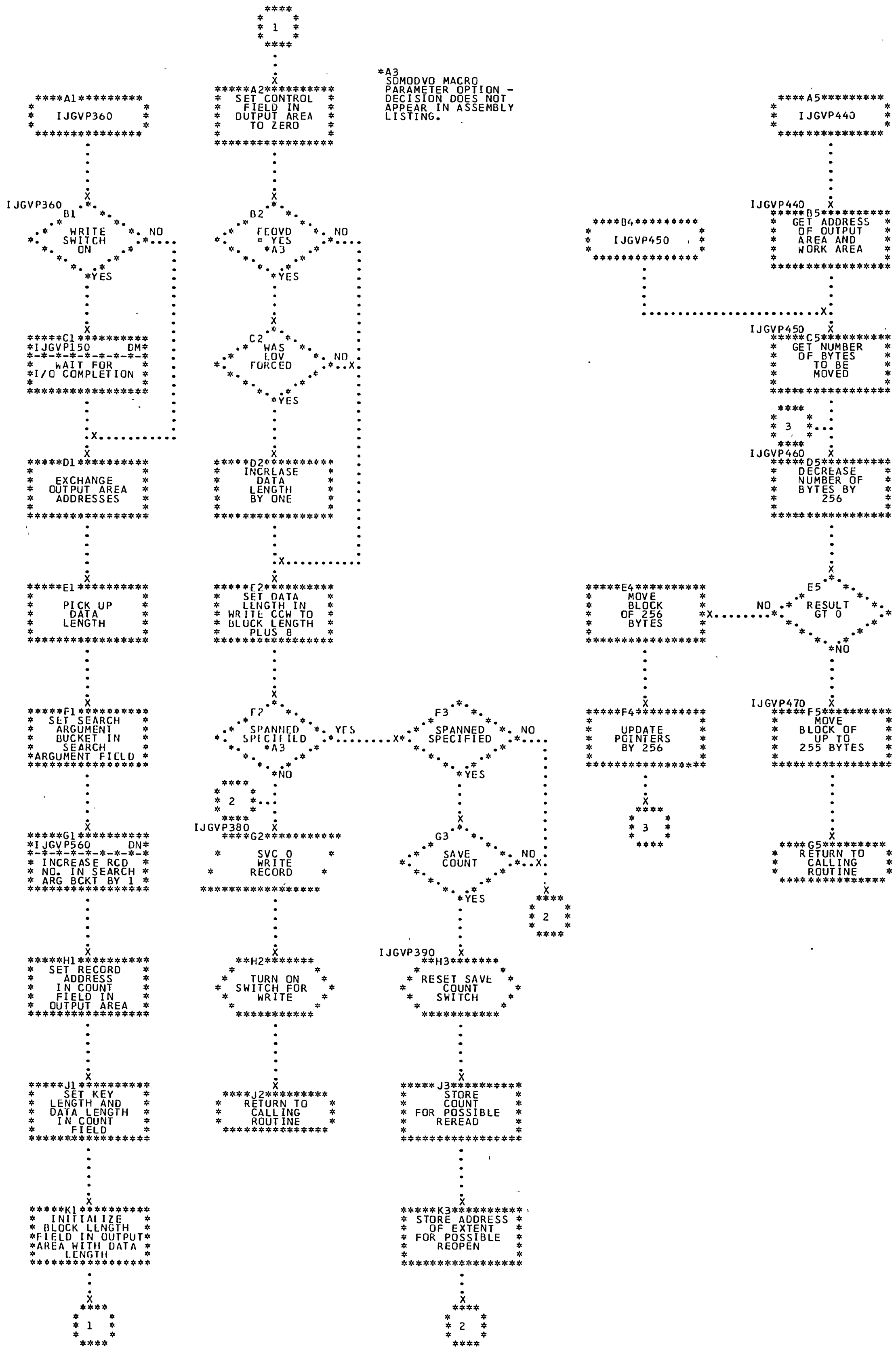


Chart DJ. SDMODVO: PUT Macro (Section 3 of 11)



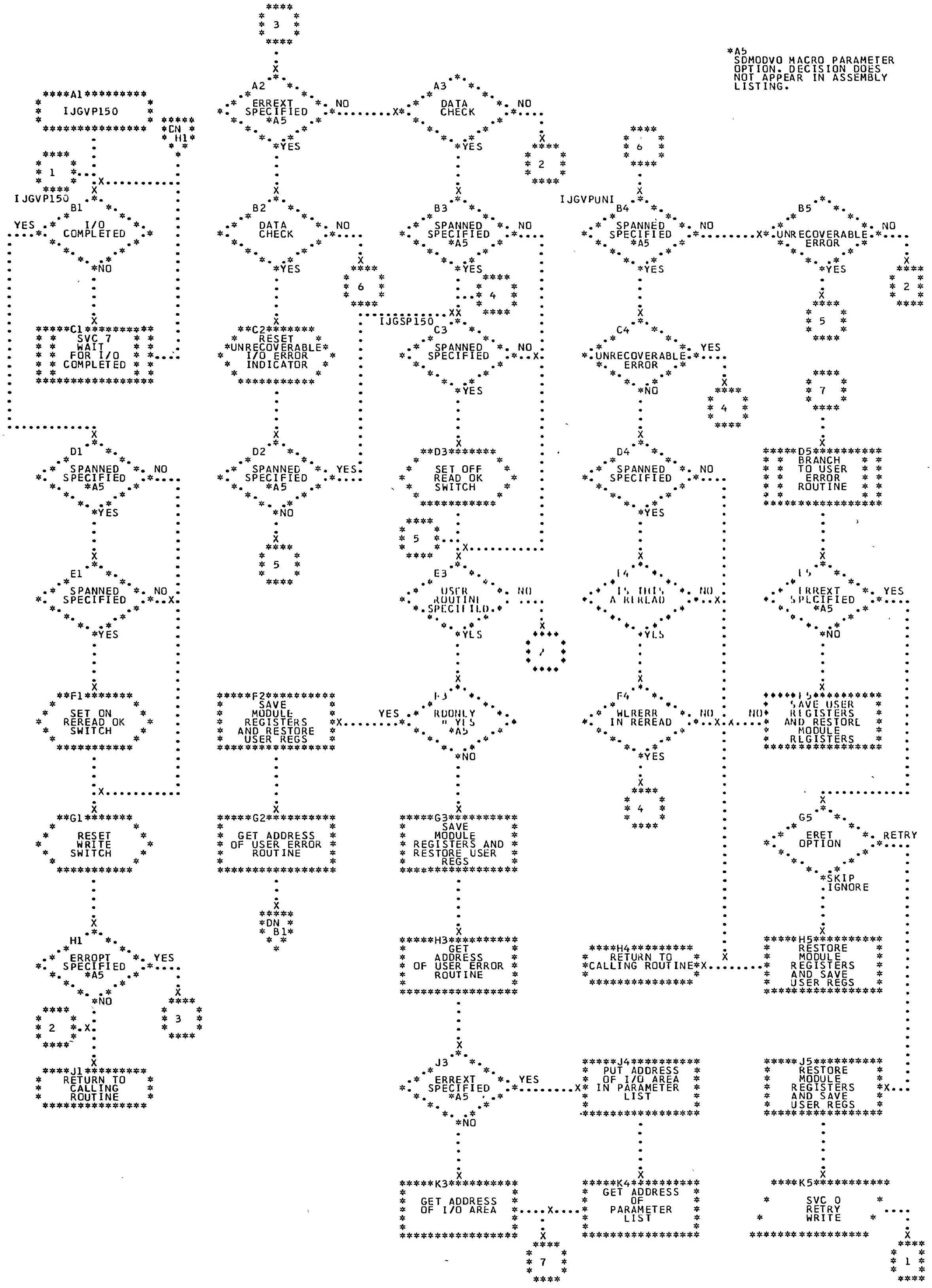

DH B3
1
3
*A4 SDMODVO MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.
IJGVP290 A2 WORK AREA SPECIFIED NO YES
IJGSP140 A5 5/MORE BYTES LEFT ON TRACK YES NO
2
*DH-D2 DJ-B4
IJGVP230 B1 OUTPUT AREA EXCEEDED NO YES
B2 IJGVP440 DL MOVE DATA FROM WORK AREA TO OUTPUT AREA
DJ G4
IJGSP130 B3 END OF TRACK YES NO
B4 RESET END OF TRACK SWITCH
B5 TURN ON TRACK TRUNCATION SWITCH
C1 RESET SWITCH FOR OUTPUT AREA EXCEEDED
C2 SAVE ADDRESS OF NEXT AVAILABLE RECORD IN OUTPUT AREA
C3 TRACK TRUNCATED YES NO
IJGSP090 C4 BLOCKING SPECIFIED NO YES
2
4
D1 IJGVP520 DP REINITIALIZE TRACK CAPACITY
D2 COMPUTE LENGTH OF AVAILABLE SPACE REMAINING IN OUTPUT AREA
D3 BLOCKING SPECIFIED NO YES
3
D4 OUTPUT AREA FULL NO YES
IJGSP095 D5 TURN ON TRACK TRUNCATION SWITCH
2
DJ C1
IJGVP090
IJGVP240 E1 TRACK CAPACITY EXCEEDED NO YES
DH E3
E2 SPANNED SPECIFIED *A4 NO YES
E3 OUTPUT BLOCK TRUNCATED NO YES
IJGSP100 E4 GET REMAINING RECORD LENGTH
IJGVP060 IJGVP260 F1 RESET SWITCH FOR TRACK CAPACITY EXCEEDED
F2 SPANNED SPECIFIED YES NO
F3 RESET OUTPUT BLOCK TRUNCATION SWITCH
F4 END OF RECORD NO YES
F5 GET I/O AREA ADDRESS
1
H1
DG A4
IJGVP040
DO C2
IJGSP045
IJGVP310 G2 RESTORE USER REGISTERS
G3 5/MORE BYTES LEFT IN OUTPUT AREA YES NO
3
IJGSP110 G4 RESET FIRST RECORD ON VOLUME SWITCH
*H1 DJ-F4,G4 DP-G3
H2 LOAD IOREG WITH ADDRESS OF AVAILABLE OUTPUT AREA
H3 TURN ON SWITCH FOR TRUNCATION OF OUTPUT BLOCK
4
H4 RESTORE USER REGISTERS
J2 LOAD VARBLD WITH LENGTH OF AVAILABLE SPACE REMAINING IN OUTPUT AREA
J4 RETURN TO PROBLEM PROGRAM
K2 RETURN TO PROBLEM PROGRAM

## Chart DL. SDMODVO: PUT Macro (Section 5 of 11)


1
A1
IJGVP360
IJGVP360
B1
WRITE SWITCH ON
NO
YES
C1
IJGVP150 DM
WAIT FOR I/O COMPLETION
D1
EXCHANGE OUTPUT AREA ADDRESSES
E1
PICK UP DATA LENGTH
F1
SET SEARCH ARGUMENT BUCKET IN SEARCH ARGUMENT FIELD
G1
IJGVP560 DN
INCREASE RCD NO. IN SEARCH ARG BCKT BY 1
H1
SET RECORD ADDRESS IN COUNT FIELD IN OUTPUT AREA
J1
SET KEY LENGTH AND DATA LENGTH IN COUNT FIELD
K1
INITIALIZE BLOCK LENGTH FIELD IN OUTPUT AREA WITH DATA LENGTH
1
A2
SET CONTROL FIELD IN OUTPUT AREA TO ZERO
B2
FEOVD = YES *A3
NO
YES
C2
WAS EOV FORCED
NO
YES
D2
INCREASE DATA LENGTH BY ONE
E2
SET DATA LENGTH IN WRITE CCW TO BLOCK LENGTH PLUS 8
F2
SPANNED SPECIFIED *A3
YES
NO
2
IJGVP380
G2
SVC 0 WRITE RECORD
H2
TURN ON SWITCH FOR WRITE
J2
RETURN TO CALLING ROUTINE
F3
SPANNED SPECIFIED
NO
YES
G3
SAVE COUNT
NO
YES
2
IJGVP390
H3
RESET SAVE COUNT SWITCH
J3
STORE COUNT FOR POSSIBLE REREAD
K3
STORE ADDRESS OF EXTENT FOR POSSIBLE REOPEN
2
B4
IJGVP450
E4
MOVE BLOCK OF 256 BYTES
F4
UPDATE POINTERS BY 256
3
A5
IJGVP440
IJGVP440
B5
GET ADDRESS OF OUTPUT AREA AND WORK AREA
IJGVP450
C5
GET NUMBER OF BYTES TO BE MOVED
3
IJGVP460
D5
DECREASE NUMBER OF BYTES BY 256
E5
RESULT GT 0
NO
IJGVP470
F5
MOVE BLOCK OF UP TO 255 BYTES
G5
RETURN TO CALLING ROUTINE

*A3 SDMODVO MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.

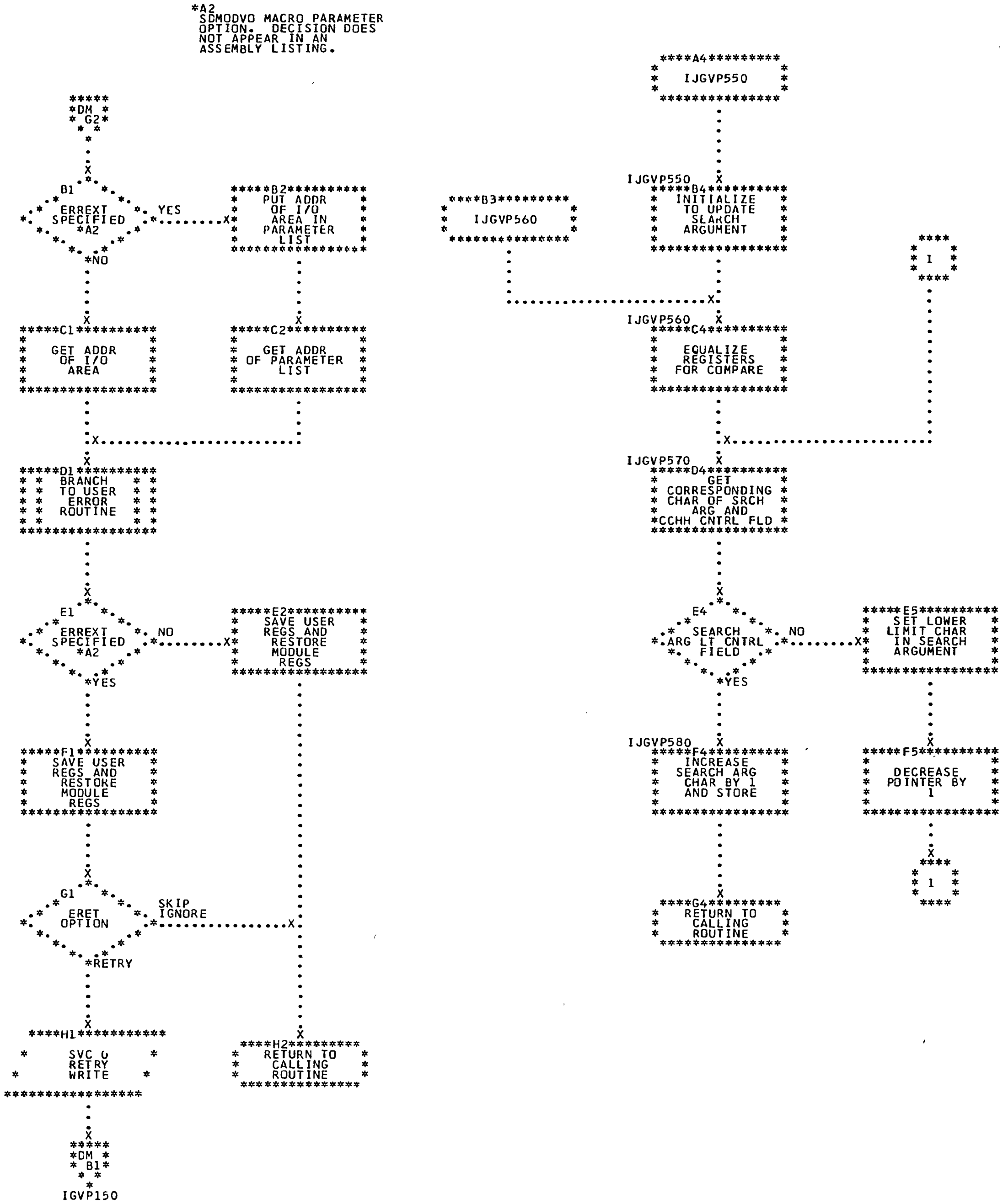
Chart DM. SDMODVO: PUT Macro (Section 6 of 11)



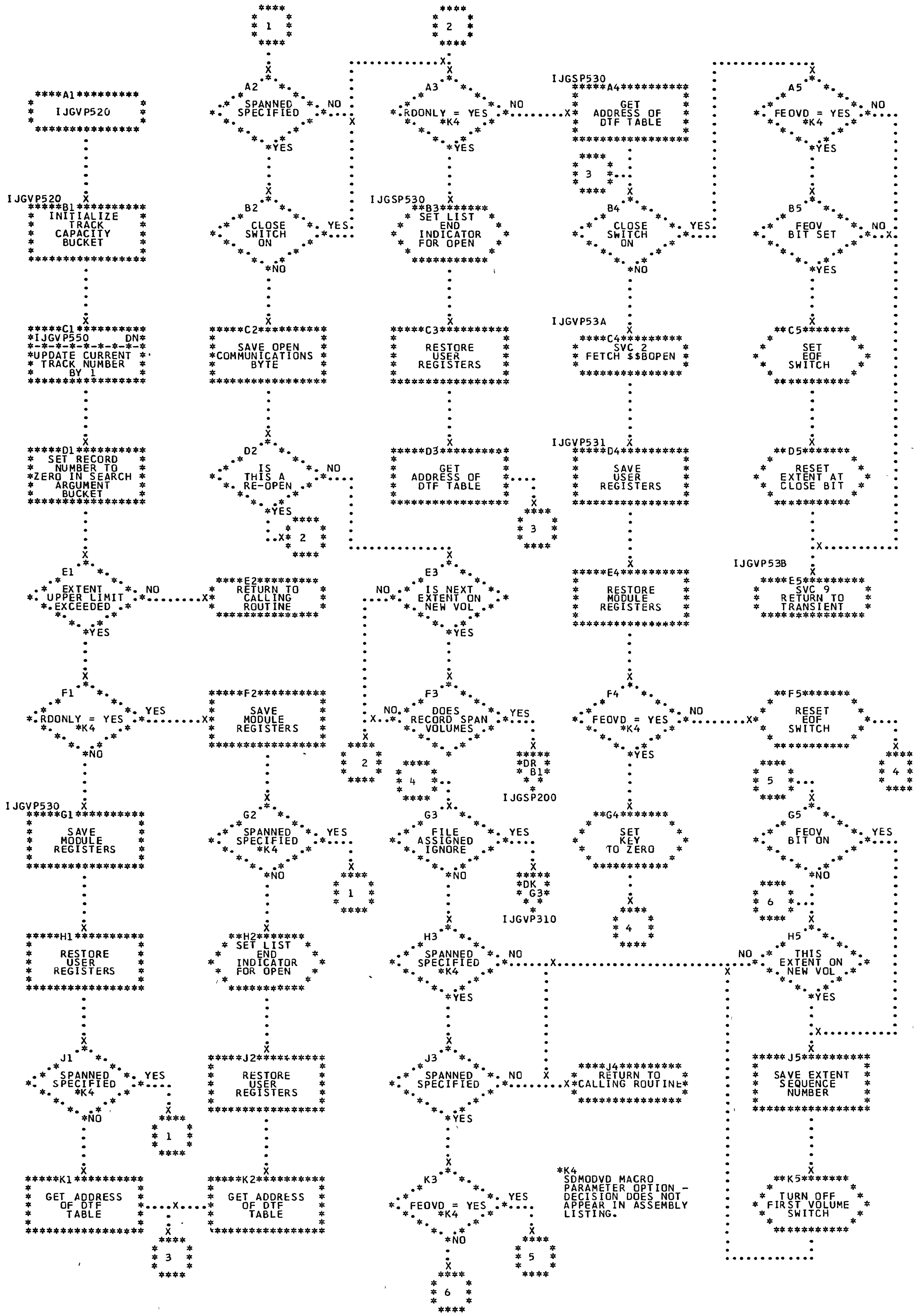
*A5
SDMODVO MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.

3
A1 IJGVP150
CN H1
1
IJGVP150 B1 I/O COMPLETED
YES
NO
C1 SVC 7 WAIT FOR I/O COMPLETED
D1 SPANNED SPECIFIED *A5
NO
YES
E1 SPANNED SPECIFIED
NO
YES
F1 SET ON REREAD OK SWITCH
G1 RESET WRITE SWITCH
H1 ERROPT SPECIFIED *A5
YES
NO
2
3
J1 RETURN TO CALLING ROUTINE
A2 ERREXT SPECIFIED *A5
NO
YES
B2 DATA CHECK
NO
YES
6
C2 RESET *UNRECOVERABLE* I/O ERROR INDICATOR
D2 SPANNED SPECIFIED *A5
YES
NO
5
F2 SAVE MODULE REGISTERS AND RESTORE USER REGS
G2 GET ADDRESS OF USER ERROR ROUTINE
DN B1
A3 DATA CHECK
NO
YES
2
B3 SPANNED SPECIFIED *A5
NO
YES
4
IJGSP150 C3 SPANNED SPECIFIED
NO
YES
D3 SET OFF READ OK SWITCH
5
E3 USER ROUTINE SPECIFIED
NO
YES
7
F3 RDONLY = YES *A5
YES
NO
G3 SAVE MODULE REGISTERS AND RESTORE USER REGS
H3 GET ADDRESS OF USER ERROR ROUTINE
J3 ERREXT SPECIFIED *A5
YES
NO
K3 GET ADDRESS OF I/O AREA
7
6
IJGVPUNI B4 SPANNED SPECIFIED *A5
YES
C4 UNRECOVERABLE ERROR
YES
NO
4
D4 SPANNED SPECIFIED
NO
YES
E4 IS THIS A REREAD
NO
YES
F4 WLRERR IN REREAD
NO
YES
4
H4 RETURN TO CALLING ROUTINE
J4 PUT ADDRESS OF I/O AREA IN PARAMETER LIST
K4 GET ADDRESS OF PARAMETER LIST
B5 UNRECOVERABLE ERROR
NO
YES
2
5
7
D5 BRANCH TO USER ERROR ROUTINE
E5 ERREXT SPECIFIED *A5
YES
NO
F5 SAVE USER REGISTERS AND RESTORE MODULE REGISTERS
NO
G5 ERET OPTION
RETRY
SKIP .IGNORE
H5 RESTORE MODULE REGISTERS AND SAVE USER REGS
J5 RESTORE MODULE REGISTERS AND SAVE USER REGS
K5 SVC 0 RETRY WRITE
1

## Chart DN. SDMODVO: PUT Macro (Section 7 of 11)


*A2
SDMODVO MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
DM G2
B1 ERREXT SPECIFIED *A2
YES
NO
B2 PUT ADDR OF I/O AREA IN PARAMETER LIST
C1 GET ADDR OF I/O AREA
C2 GET ADDR OF PARAMETER LIST
D1 BRANCH TO USER ERROR ROUTINE
E1 ERREXT SPECIFIED *A2
NO
YES
E2 SAVE USER REGS AND RESTORE MODULE REGS
F1 SAVE USER REGS AND RESTORE MODULE REGS
G1 ERET OPTION
SKIP IGNORE
RETRY
H1 SVC 0 RETRY WRITE
H2 RETURN TO CALLING ROUTINE
DM B1
IGVP150
A4 IJGVP550
B3 IJGVP560
IJGVP550
B4 INITIALIZE TO UPDATE SEARCH ARGUMENT
1
IJGVP560
C4 EQUALIZE REGISTERS FOR COMPARE
IJGVP570
D4 GET CORRESPONDING CHAR OF SRCH ARG AND CCHH CNTRL FLD
E4 SEARCH ARG LT CNTRL FIELD
NO
YES
E5 SET LOWER LIMIT CHAR IN SEARCH ARGUMENT
IJGVP580
F4 INCREASE SEARCH ARG CHAR BY 1 AND STORE
F5 DECREASE POINTER BY 1
1
G4 RETURN TO CALLING ROUTINE

# Chart DP. SDMODVO: PUT Macro (Section 8 of 11)



*K4
SDMODVD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.

Chart DQ.  SDMODVO:  PUT Macro (Section 9 of 11)

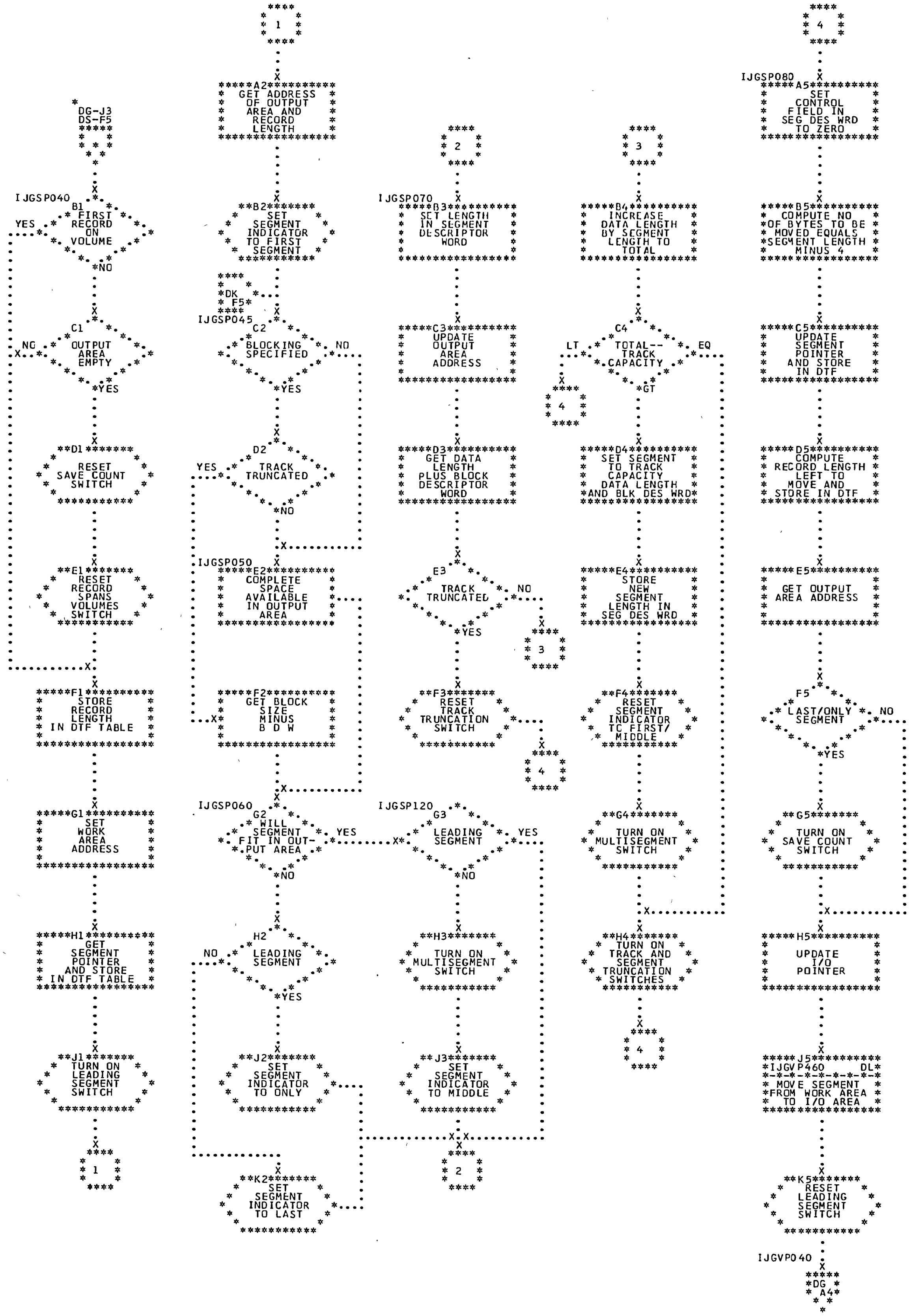
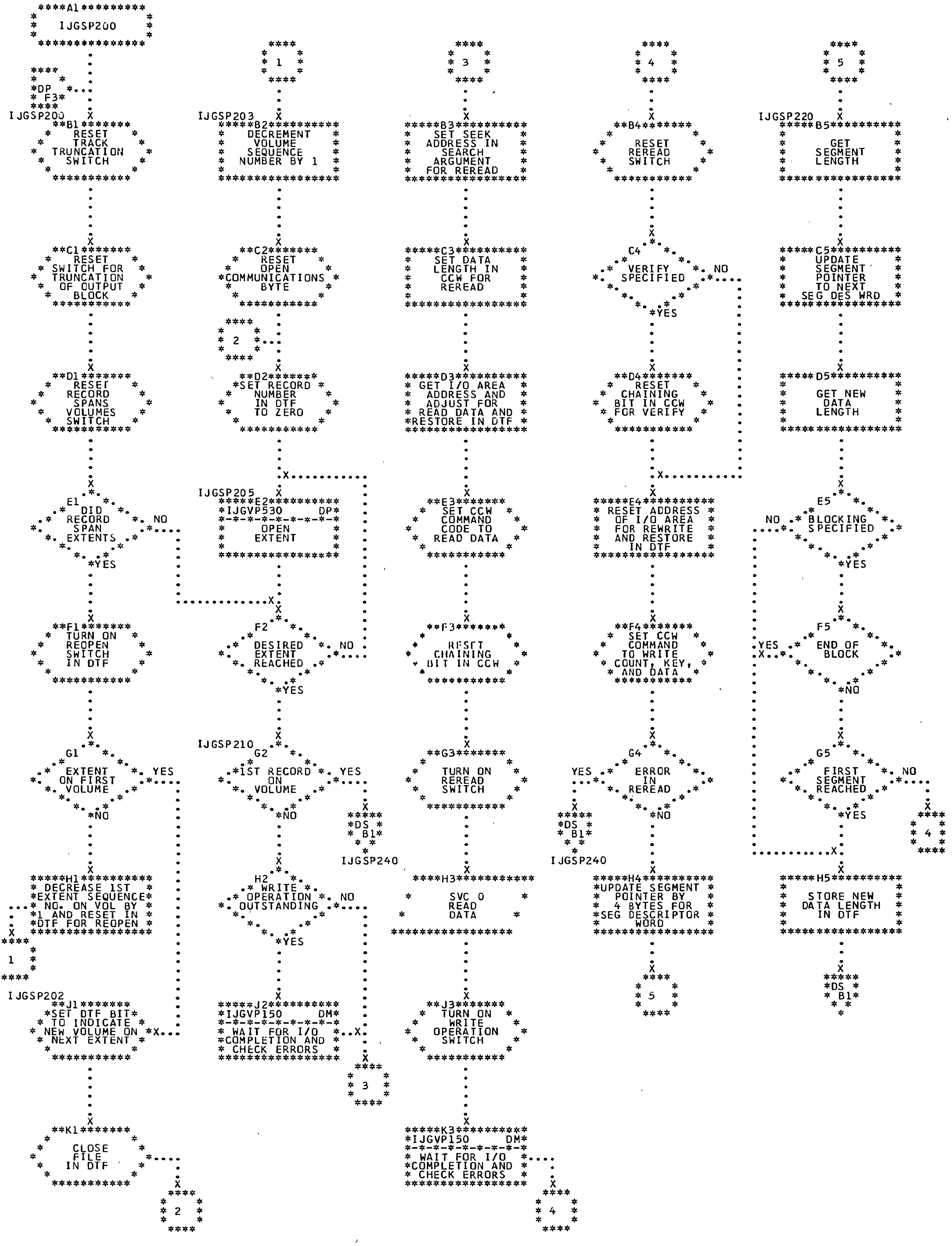
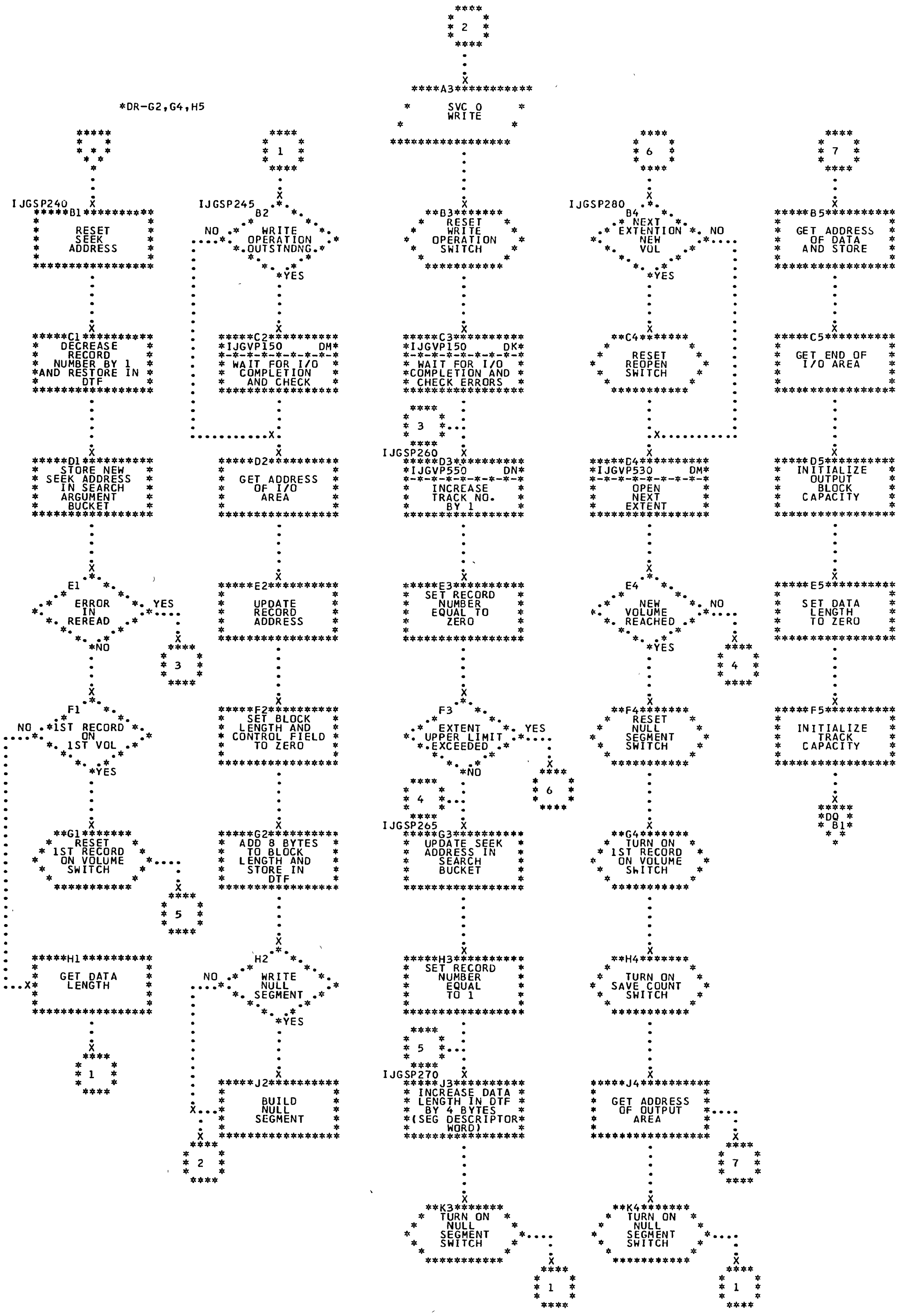

DG-J3
DS-F5
IJGSP040
B1
FIRST RECORD ON VOLUME
YES
NO
C1
OUTPUT AREA EMPTY
NO
YES
D1
RESET SAVE COUNT SWITCH
E1
RESET RECORD SPANS VOLUMES SWITCH
F1
STORE RECORD LENGTH IN DTF TABLE
G1
SET WORK AREA ADDRESS
H1
GET SEGMENT POINTER AND STORE IN DTF TABLE
J1
TURN ON LEADING SEGMENT SWITCH
1
1
A2
GET ADDRESS OF OUTPUT AREA AND RECORD LENGTH
B2
SET SEGMENT INDICATOR TO FIRST SEGMENT
DK F5
IJGSP045
C2
BLOCKING SPECIFIED
NO
YES
D2
TRACK TRUNCATED
YES
NO
IJGSP050
E2
COMPLETE SPACE AVAILABLE IN OUTPUT AREA
F2
GET BLOCK SIZE MINUS B D W
IJGSP060
G2
WILL SEGMENT FIT IN OUT- PUT AREA
YES
NO
H2
LEADING SEGMENT
NO
YES
J2
SET SEGMENT INDICATOR TO ONLY
K2
SET SEGMENT INDICATOR TO LAST
2
IJGSP070
B3
SET LENGTH IN SEGMENT DESCRIPTOR WORD
C3
UPDATE OUTPUT AREA ADDRESS
D3
GET DATA LENGTH PLUS BLOCK DESCRIPTOR WORD
E3
TRACK TRUNCATED
NO
YES
3
F3
RESET TRACK TRUNCATION SWITCH
4
IJGSP120
G3
LEADING SEGMENT
YES
NO
H3
TURN ON MULTISEGMENT SWITCH
J3
SET SEGMENT INDICATOR TO MIDDLE
2
3
B4
INCREASE DATA LENGTH BY SEGMENT LENGTH TO TOTAL
C4
TOTAL-- TRACK CAPACITY
LT
EQ
GT
4
D4
SET SEGMENT TO TRACK CAPACITY DATA LENGTH AND BLK DES WRD
E4
STORE NEW SEGMENT LENGTH IN SEG DES WRD
F4
RESET SEGMENT INDICATOR TC FIRST/ MIDDLE
G4
TURN ON MULTISEGMENT SWITCH
H4
TURN ON TRACK AND SEGMENT TRUNCATION SWITCHES
4
4
IJGSP080
A5
SET CONTROL FIELD IN SEG DES WRD TO ZERO
B5
COMPUTE NO OF BYTES TO BE MOVED EQUALS SEGMENT LENGTH MINUS 4
C5
UPDATE SEGMENT POINTER AND STORE IN DTF
D5
COMPUTE RECORD LENGTH LEFT TO MOVE AND STORE IN DTF
E5
GET OUTPUT AREA ADDRESS
F5
LAST/ONLY SEGMENT
NO
YES
G5
TURN ON SAVE COUNT SWITCH
H5
UPDATE I/O POINTER
J5
IJGVP460 DL
MOVE SEGMENT FROM WORK AREA TO I/O AREA
K5
RESET LEADING SEGMENT SWITCH
IJGVP040
DG A4

Chart DR. SDMODVO: PUT Macro (Section 10 of 11)

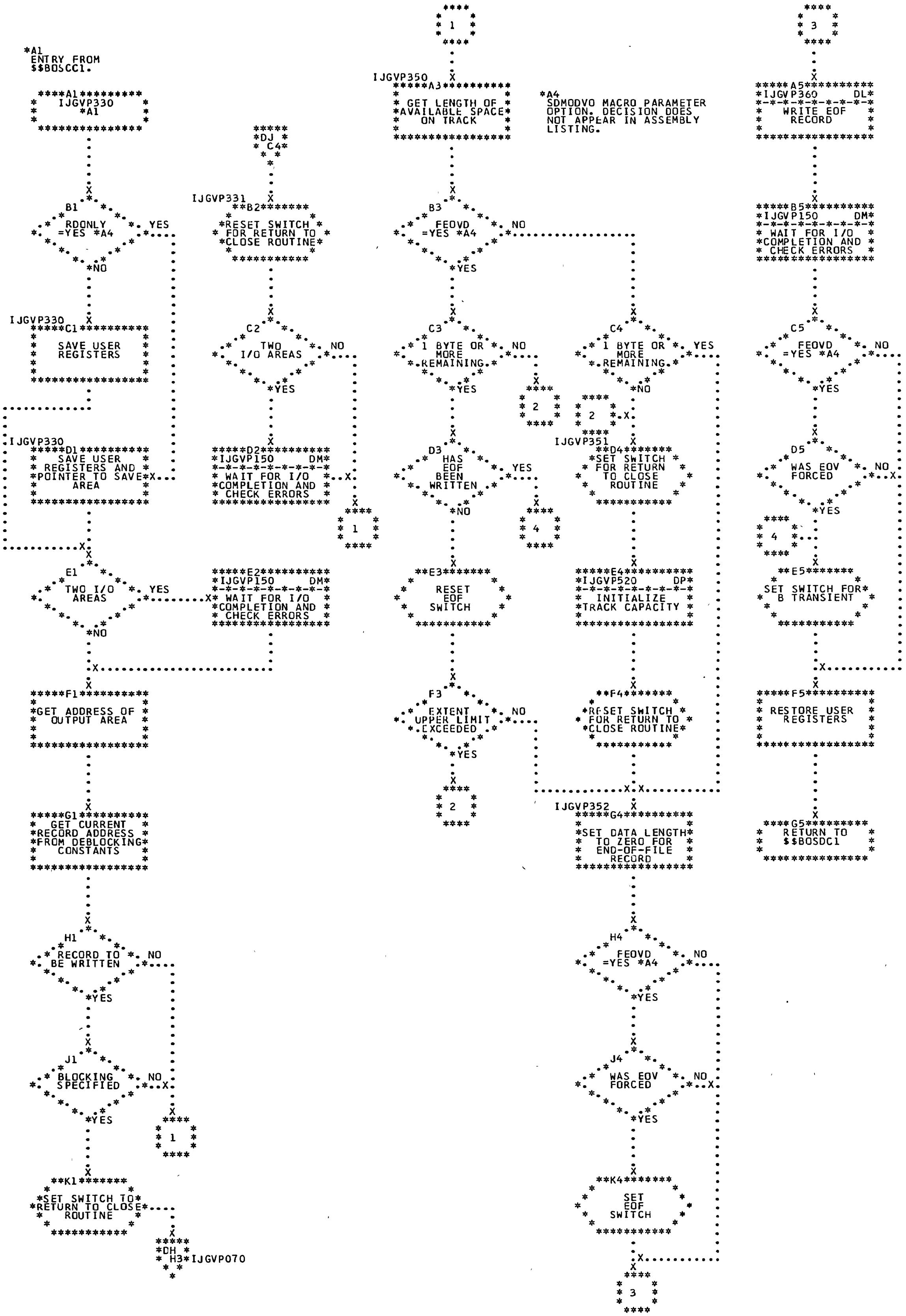

A1
IJGSP200
DP F3
IJGSP200
B1
RESET TRACK TRUNCATION SWITCH
C1
RESET SWITCH FOR TRUNCATION OF OUTPUT BLOCK
D1
RESET RECORD SPANS VOLUMES SWITCH
E1
DID RECORD SPAN EXTENTS
NO
YES
F1
TURN ON REOPEN SWITCH IN DTF
G1
EXTENT ON FIRST VOLUME
YES
NO
H1
DECREASE 1ST EXTENT SEQUENCE NO. ON VOL BY 1 AND RESET IN DTF FOR REOPEN
1
IJGSP202
J1
SET DTF BIT TO INDICATE NEW VOLUME ON NEXT EXTENT
K1
CLOSE FILE IN DTF
2
1
IJGSP203
B2
DECREMENT VOLUME SEQUENCE NUMBER BY 1
C2
RESET OPEN COMMUNICATIONS BYTE
2
D2
SET RECORD NUMBER IN DTF TO ZERO
IJGSP205
E2
IJGVP530 DP
OPEN EXTENT
F2
DESIRED EXTENT REACHED
NO
YES
IJGSP210
G2
1ST RECORD ON VOLUME
YES
NO
DS B1
IJGSP240
H2
WRITE OPERATION OUTSTANDING
NO
YES
J2
IJGVP150 DM
WAIT FOR I/O COMPLETION AND CHECK ERRORS
3
3
B3
SET SEEK ADDRESS IN SEARCH ARGUMENT FOR REREAD
C3
SET DATA LENGTH IN CCW FOR REREAD
D3
GET I/O AREA ADDRESS AND ADJUST FOR READ DATA AND RESTORE IN DTF
E3
SET CCW COMMAND CODE TO READ DATA
F3
RESET CHAINING BIT IN CCW
G3
TURN ON REREAD SWITCH
H3
SVC 0 READ DATA
J3
TURN ON WRITE OPERATION SWITCH
K3
IJGVP150 DM
WAIT FOR I/O COMPLETION AND CHECK ERRORS
4
4
B4
RESET REREAD SWITCH
C4
VERIFY SPECIFIED
NO
YES
D4
RESET CHAINING BIT IN CCW FOR VERIFY
E4
RESET ADDRESS OF I/O AREA FOR REWRITE AND RESTORE IN DTF
F4
SET CCW COMMAND TO WRITE COUNT, KEY, AND DATA
G4
ERROR IN REREAD
YES
NO
DS B1
IJGSP240
H4
UPDATE SEGMENT POINTER BY 4 BYTES FOR SEG DESCRIPTOR WORD
5
5
IJGSP220
B5
GET SEGMENT LENGTH
C5
UPDATE SEGMENT POINTER TO NEXT SEG DES WRD
D5
GET NEW DATA LENGTH
E5
BLOCKING SPECIFIED
NO
YES
F5
END OF BLOCK
YES
NO
G5
FIRST SEGMENT REACHED
NO
YES
4
H5
STORE NEW DATA LENGTH IN DTF
DS B1

## Chart DS. SDMODVO: PUT Macro (Section 11 of 11)

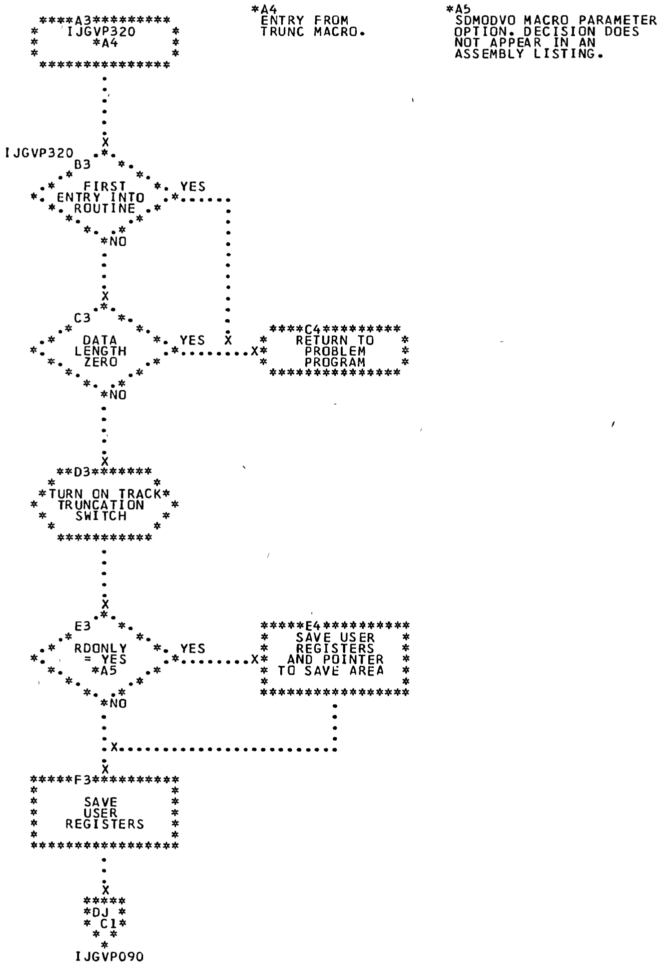


2
A3
SVC 0 WRITE
*DR-G2,G4,H5
IJGSP240
B1
RESET SEEK ADDRESS
C1
DECREASE RECORD NUMBER BY 1 AND RESTORE IN DTF
D1
STORE NEW SEEK ADDRESS IN SEARCH ARGUMENT BUCKET
E1
ERROR IN REREAD
YES
NO
3
F1
1ST RECORD ON 1ST VOL
NO
YES
G1
RESET 1ST RECORD ON VOLUME SWITCH
5
H1
GET DATA LENGTH
1
1
IJGSP245
B2
WRITE OPERATION OUTSTNDNG.
NO
YES
C2
IJGVP150 DM
WAIT FOR I/O COMPLETION AND CHECK
D2
GET ADDRESS OF I/O AREA
E2
UPDATE RECORD ADDRESS
F2
SET BLOCK LENGTH AND CONTROL FIELD TO ZERO
G2
ADD 8 BYTES TO BLOCK LENGTH AND STORE IN DTF
H2
WRITE NULL SEGMENT
NO
YES
J2
BUILD NULL SEGMENT
2
B3
RESET WRITE OPERATION SWITCH
C3
IJGVP150 DK
WAIT FOR I/O COMPLETION AND CHECK ERRORS
3
IJGSP260
D3
IJGVP550 DN
INCREASE TRACK NO. BY 1
E3
SET RECORD NUMBER EQUAL TO ZERO
F3
EXTENT UPPER LIMIT EXCEEDED
YES
NO
6
4
IJGSP265
G3
UPDATE SEEK ADDRESS IN SEARCH BUCKET
H3
SET RECORD NUMBER EQUAL TO 1
5
IJGSP270
J3
INCREASE DATA LENGTH IN DTF BY 4 BYTES (SEG DESCRIPTOR WORD)
K3
TURN ON NULL SEGMENT SWITCH
1
6
IJGSP280
B4
NEXT EXTENTION NEW VOL
NO
YES
C4
RESET REOPEN SWITCH
D4
IJGVP530 DM
OPEN NEXT EXTENT
E4
NEW VOLUME REACHED
NO
YES
4
F4
RESET NULL SEGMENT SWITCH
G4
TURN ON 1ST RECORD ON VOLUME SWITCH
H4
TURN ON SAVE COUNT SWITCH
J4
GET ADDRESS OF OUTPUT AREA
7
K4
TURN ON NULL SEGMENT SWITCH
1
7
B5
GET ADDRESS OF DATA AND STORE
C5
GET END OF I/O AREA
D5
INITIALIZE OUTPUT BLOCK CAPACITY
E5
SET DATA LENGTH TO ZERO
F5
INITIALIZE TRACK CAPACITY
DQ B1

## Chart DT. SDMODVO: Close Routine


*A1
ENTRY FROM
$$BOSCC1.
A1
IJGVP330
*A1
B1
RDONLY =YES *A4
YES
NO
IJGVP330
C1
SAVE USER REGISTERS
IJGVP330
D1
SAVE USER REGISTERS AND POINTER TO SAVE AREA
E1
TWO I/O AREAS
YES
NO
F1
GET ADDRESS OF OUTPUT AREA
G1
GET CURRENT RECORD ADDRESS FROM DEBLOCKING CONSTANTS
H1
RECORD TO BE WRITTEN
NO
YES
J1
BLOCKING SPECIFIED
NO
YES
1
K1
SET SWITCH TO RETURN TO CLOSE ROUTINE
DH
H3
IJGVP070
DJ
C4
IJGVP331
B2
RESET SWITCH FOR RETURN TO CLOSE ROUTINE
C2
TWO I/O AREAS
NO
YES
D2
IJGVP150 DM
WAIT FOR I/O COMPLETION AND CHECK ERRORS
1
E2
IJGVP150 DM
WAIT FOR I/O COMPLETION AND CHECK ERRORS
1
IJGVP350
A3
GET LENGTH OF AVAILABLE SPACE ON TRACK
*A4
SDMODVO MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.
B3
FEOVD =YES *A4
NO
YES
C3
1 BYTE OR MORE REMAINING
NO
YES
2
D3
HAS EOF BEEN WRITTEN
YES
NO
4
E3
RESET EOF SWITCH
F3
EXTENT UPPER LIMIT EXCEEDED
NO
YES
2
C4
1 BYTE OR MORE REMAINING
YES
NO
2
IJGVP351
D4
SET SWITCH FOR RETURN TO CLOSE ROUTINE
E4
IJGVP520 DP
INITIALIZE TRACK CAPACITY
F4
RESET SWITCH FOR RETURN TO CLOSE ROUTINE
IJGVP352
G4
SET DATA LENGTH TO ZERO FOR END-OF-FILE RECORD
H4
FEOVD =YES *A4
NO
YES
J4
WAS EOV FORCED
NO
YES
K4
SET EOF SWITCH
3
3
A5
IJGVP360 DL
WRITE EOF RECORD
B5
IJGVP150 DM
WAIT FOR I/O COMPLETION AND CHECK ERRORS
C5
FEOVD =YES *A4
NO
YES
D5
WAS EOV FORCED
NO
YES
4
E5
SET SWITCH FOR B TRANSIENT
F5
RESTORE USER REGISTERS
G5
RETURN TO $$BOSDC1

## Chart DU. SDMODVO: TRUNC Macro



*A4
ENTRY FROM
TRUNC MACRO.

*A5
SDMODVO MACRO PARAMETER
OPTION. DECISION DOES
NOT APPEAR IN AN
ASSEMBLY LISTING.

Chart DV. SDMODVU: GET Macro (Section 1 of 11)

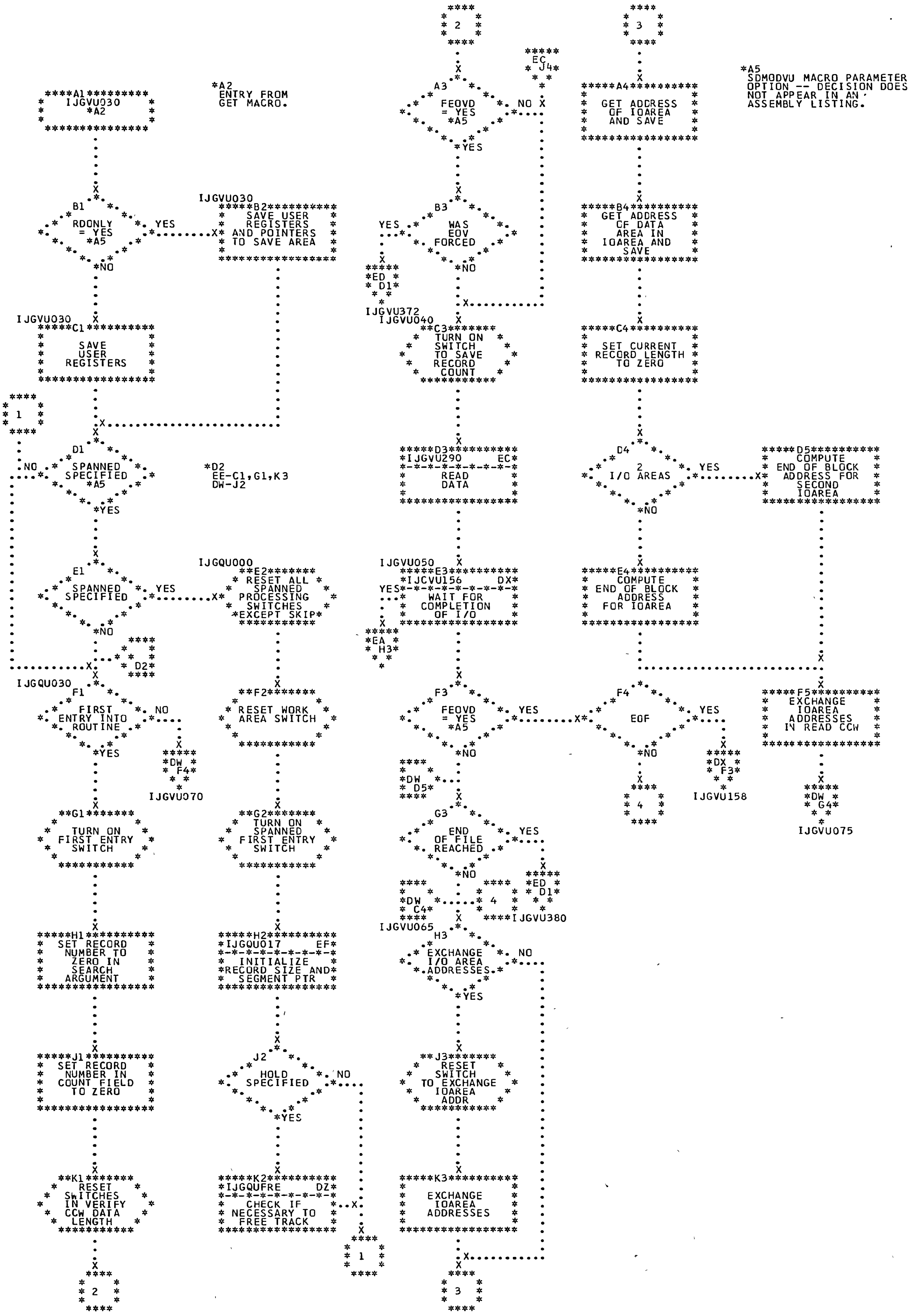
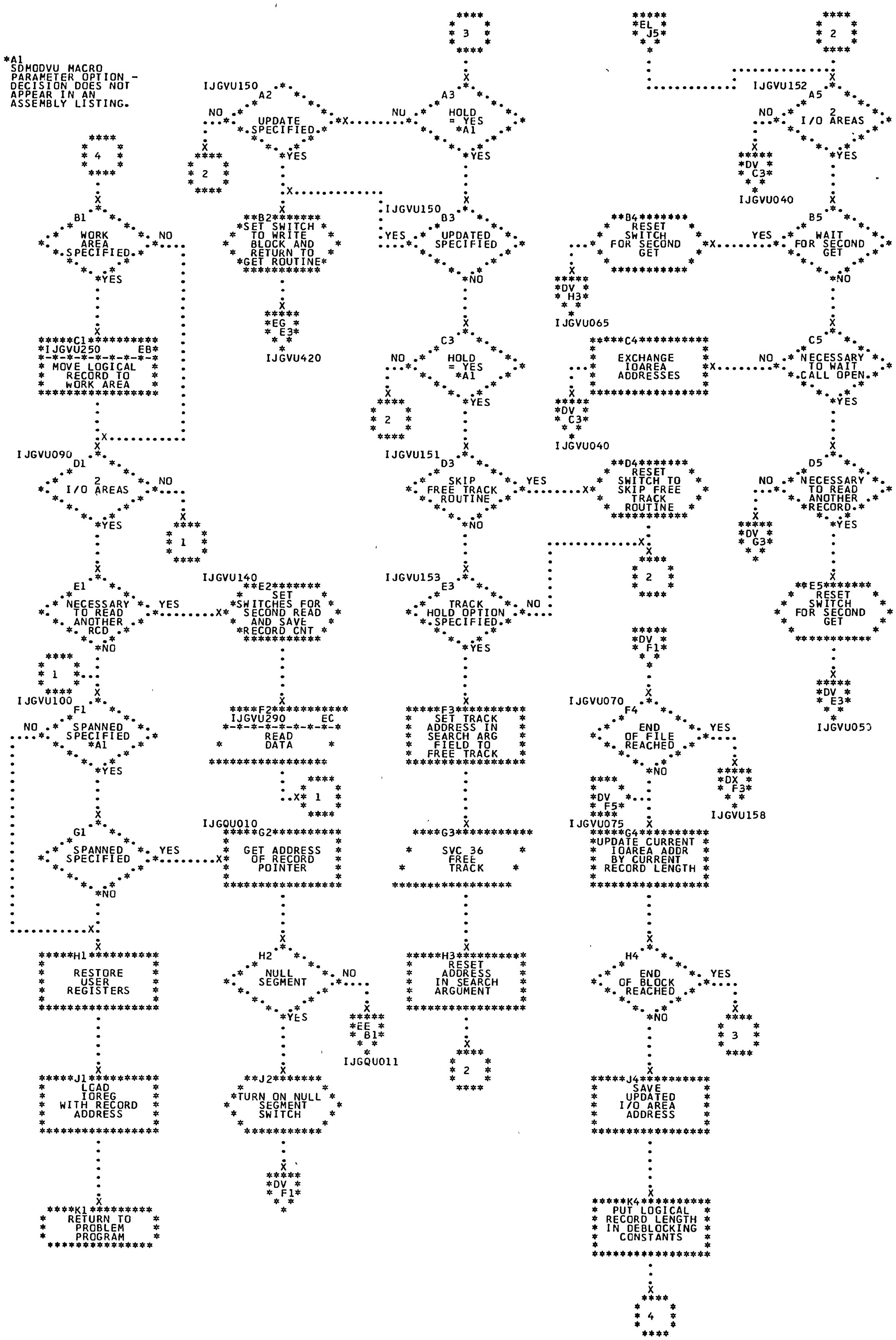
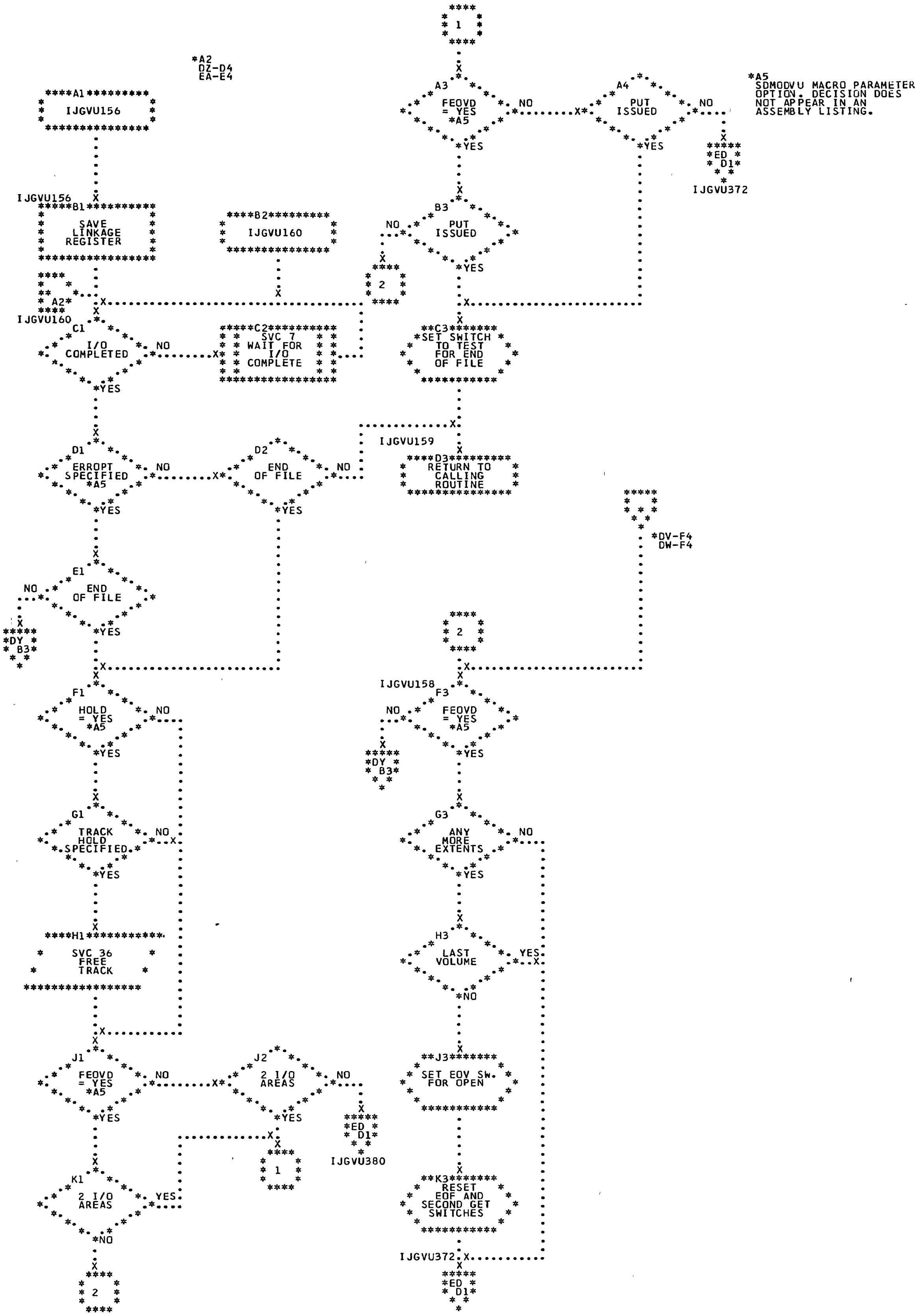

A1 IJGVU030 *A2
*A2 ENTRY FROM GET MACRO.
B1 RDONLY = YES *A5
YES
NO
IJGVU030 B2 SAVE USER REGISTERS AND POINTERS TO SAVE AREA
IJGVU030 C1 SAVE USER REGISTERS
1
D1 SPANNED SPECIFIED *A5
NO
YES
*D2 EE-C1,G1,K3 DW-J2
E1 SPANNED SPECIFIED
YES
NO
IJGQU000 E2 RESET ALL SPANNED PROCESSING SWITCHES EXCEPT SKIP
D2
IJGQU030 F1 FIRST ENTRY INTO ROUTINE
NO
YES
DW F4 IJGVU070
F2 RESET WORK AREA SWITCH
G1 TURN ON FIRST ENTRY SWITCH
G2 TURN ON SPANNED FIRST ENTRY SWITCH
H1 SET RECORD NUMBER TO ZERO IN SEARCH ARGUMENT
H2 IJGQU017 EF INITIALIZE RECORD SIZE AND SEGMENT PTR
J1 SET RECORD NUMBER IN COUNT FIELD TO ZERO
J2 HOLD SPECIFIED
NO
YES
K1 RESET SWITCHES IN VERIFY CCW DATA LENGTH
K2 IJGQUFRE DZ CHECK IF NECESSARY TO FREE TRACK
2
1
2
EC J4
A3 FEOVD = YES *A5
NO
YES
B3 WAS EOV FORCED
YES
NO
ED D1 IJGVU372
IJGVU040 C3 TURN ON SWITCH TO SAVE RECORD COUNT
D3 IJGVU290 EC READ DATA
IJGVU050 E3 IJCVU156 DX WAIT FOR COMPLETION OF I/O
YES
EA H3
F3 FEOVD = YES *A5
YES
NO
DW D5
G3 END OF FILE REACHED
YES
NO
ED D1 IJGVU380
DW C4
4
IJGVU065 H3 EXCHANGE I/O AREA ADDRESSES
NO
YES
J3 RESET SWITCH TO EXCHANGE IOAREA ADDR
K3 EXCHANGE IOAREA ADDRESSES
3
3
A4 GET ADDRESS OF IOAREA AND SAVE
B4 GET ADDRESS CF DATA AREA IN IOAREA AND SAVE
C4 SET CURRENT RECORD LENGTH TO ZERO
D4 2 I/O AREAS
YES
NO
E4 COMPUTE END OF BLOCK ADDRESS FOR IOAREA
F4 EOF
YES
NO
DX F3 IJGVU158
4
D5 COMPUTE END OF BLOCK ADDRESS FOR SECOND IOAREA
F5 EXCHANGE IOAREA ADDRESSES IN READ CCW
DW G4 IJGVU075
*A5 SDMODVU MACRO PARAMETER OPTION -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.

Chart DW. SDMODVU: GET Macro (Section 2 of 11)


*A1 SDMODVU MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
4
B1 WORK AREA SPECIFIED
NO
YES
C1 IJGVU250 EB
MOVE LOGICAL RECORD TO WORK AREA
IJGVU090 D1
2 I/O AREAS
NO
YES
1
E1 NECESSARY TO READ ANOTHER RCD
YES
NO
1
IJGVU100 F1
SPANNED SPECIFIED *A1
NO
YES
G1 SPANNED SPECIFIED
YES
NO
H1 RESTORE USER REGISTERS
J1 LOAD IOREG WITH RECORD ADDRESS
K1 RETURN TO PROBLEM PROGRAM
IJGVU150 A2
UPDATE SPECIFIED
NO
YES
2
B2 SET SWITCH TO WRITE BLOCK AND RETURN TO GET ROUTINE
EG E3
IJGVU420
IJGVU140
E2 SET SWITCHES FOR SECOND READ AND SAVE RECORD CNT
F2 IJGVU290 EC
READ DATA
1
IJGQU010
G2 GET ADDRESS OF RECORD POINTER
H2 NULL SEGMENT
NO
YES
EE B1
IJGQU011
J2 TURN ON NULL SEGMENT SWITCH
DV F1
3
A3 HOLD = YES *A1
NO
YES
IJGVU150 B3
UPDATED SPECIFIED
YES
NO
C3 HOLD = YES *A1
NO
YES
2
IJGVU151 D3
SKIP FREE TRACK ROUTINE
YES
NO
IJGVU153 E3
TRACK HOLD OPTION SPECIFIED
NO
YES
F3 SET TRACK ADDRESS IN SEARCH ARG FIELD TO FREE TRACK
G3 SVC 36 FREE TRACK
H3 RESET ADDRESS IN SEARCH ARGUMENT
2
EL J5
B4 RESET SWITCH FOR SECOND GET
DV H3
IJGVU065
C4 EXCHANGE IOAREA ADDRESSES
DV C3
IJGVU040
D4 RESET SWITCH TO SKIP FREE TRACK ROUTINE
2
DV F1
IJGVU070 F4
END OF FILE REACHED
YES
NO
DX F3
IJGVU158
DV F5
IJGVU075
G4 UPDATE CURRENT IOAREA ADDR BY CURRENT RECORD LENGTH
H4 END OF BLOCK REACHED
YES
NO
3
J4 SAVE UPDATED I/O AREA ADDRESS
K4 PUT LOGICAL RECORD LENGTH IN DEBLOCKING CONSTANTS
4
2
IJGVU152 A5
2 I/O AREAS
NO
YES
DV C3
IJGVU040
B5 WAIT FOR SECOND GET
YES
NO
C5 NECESSARY TO WAIT CALL OPEN.
NO
YES
D5 NECESSARY TO READ ANOTHER RECORD.
NO
YES
DV G3
E5 RESET SWITCH FOR SECOND GET
DV E3
IJGVU053

## Chart DX. SDMODVU: GET Macro (Section 3 of 11)



## Chart DY. SDMODVU: GET Macro (Section 4 of 11)

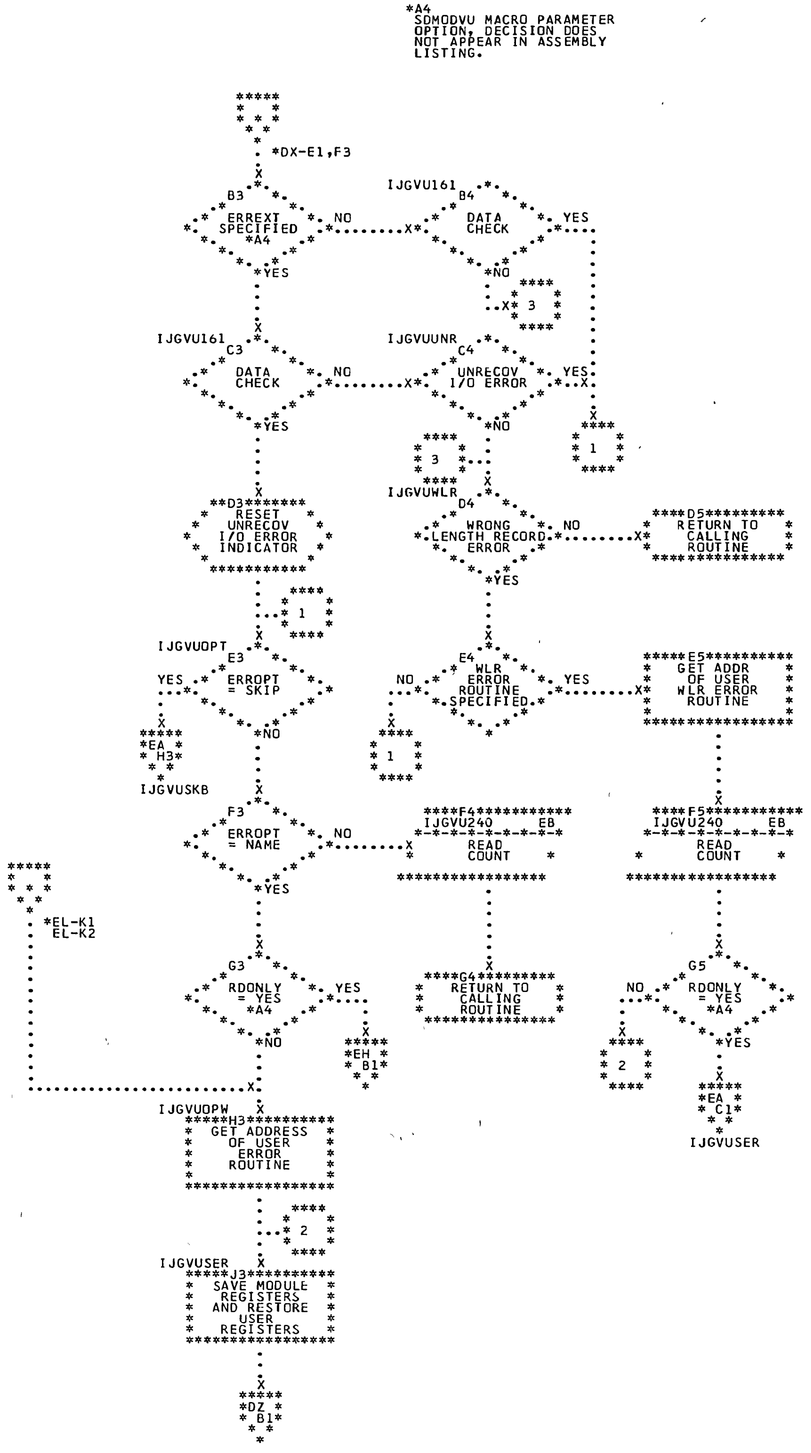


Chart DZ. SDMODVU: GET Macro (Section 5 of 11)

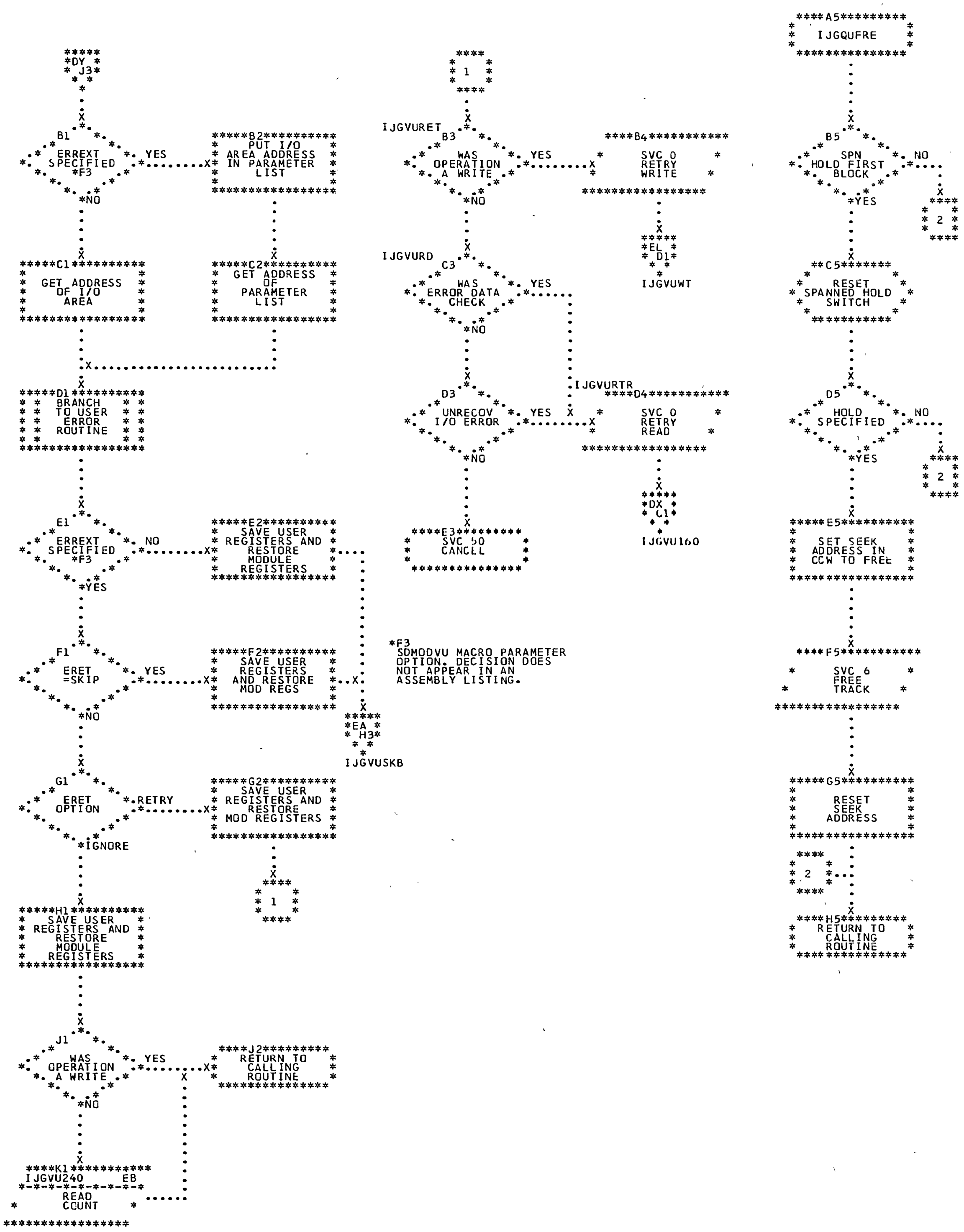
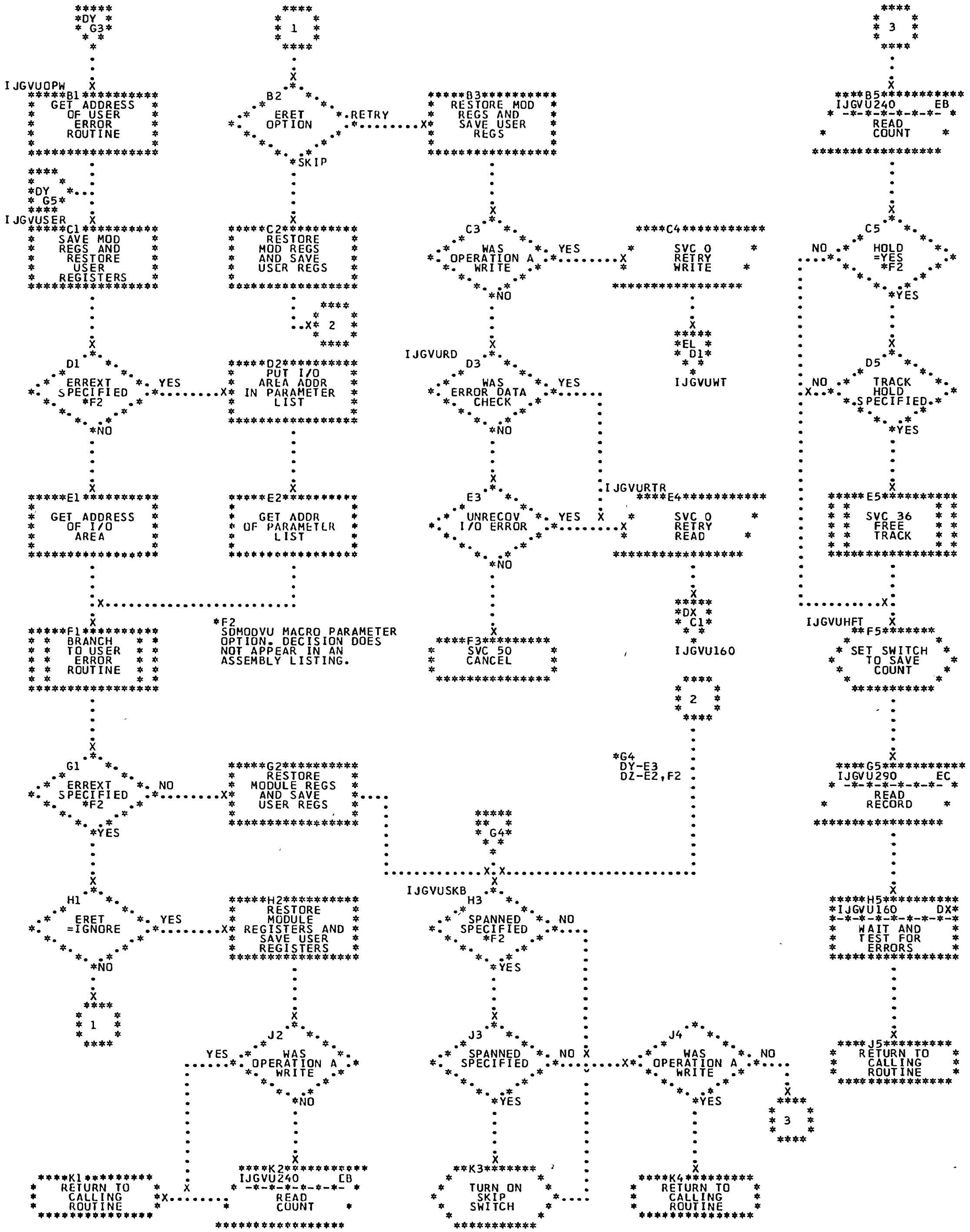

DY J3
B1 ERREXT SPECIFIED *F3
YES
NO
B2 PUT I/O AREA ADDRESS IN PARAMETER LIST
C1 GET ADDRESS OF I/O AREA
C2 GET ADDRESS OF PARAMETER LIST
D1 BRANCH TO USER ERROR ROUTINE
E1 ERREXT SPECIFIED *F3
NO
YES
E2 SAVE USER REGISTERS AND RESTORE MODULE REGISTERS
F1 ERET =SKIP
YES
NO
F2 SAVE USER REGISTERS AND RESTORE MOD REGS
EA H3
IJGVUSKB
G1 ERET OPTION
RETRY
IGNORE
G2 SAVE USER REGISTERS AND RESTORE MOD REGISTERS
1
H1 SAVE USER REGISTERS AND RESTORE MODULE REGISTERS
J1 WAS OPERATION A WRITE
YES
NO
J2 RETURN TO CALLING ROUTINE
K1 IJGVU240 EB READ COUNT
1
IJGVURET
B3 WAS OPERATION A WRITE
YES
NO
B4 SVC 0 RETRY WRITE
EL D1
IJGVUWT
IJGVURD
C3 WAS ERROR DATA CHECK
YES
NO
IJGVURTR
D3 UNRECOV I/O ERROR
YES
NO
D4 SVC 0 RETRY READ
DX C1
IJGVU160
E3 SVC 50 CANCEL
*F3
SDMODVU MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
A5 IJGQUFRE
B5 SPN HOLD FIRST BLOCK
NO
YES
2
C5 RESET SPANNED HOLD SWITCH
D5 HOLD SPECIFIED
NO
YES
2
E5 SET SEEK ADDRESS IN CCW TO FREE
F5 SVC 6 FREE TRACK
G5 RESET SEEK ADDRESS
2
H5 RETURN TO CALLING ROUTINE

Chart EA. SDMODVU: GET Macro (Section 6 of 11)



*F2
SDMODVU MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.

*G4
DY-E3
DZ-E2,F2

## Chart EB. SDMODVU: GET Macro (Section 7 of 11)

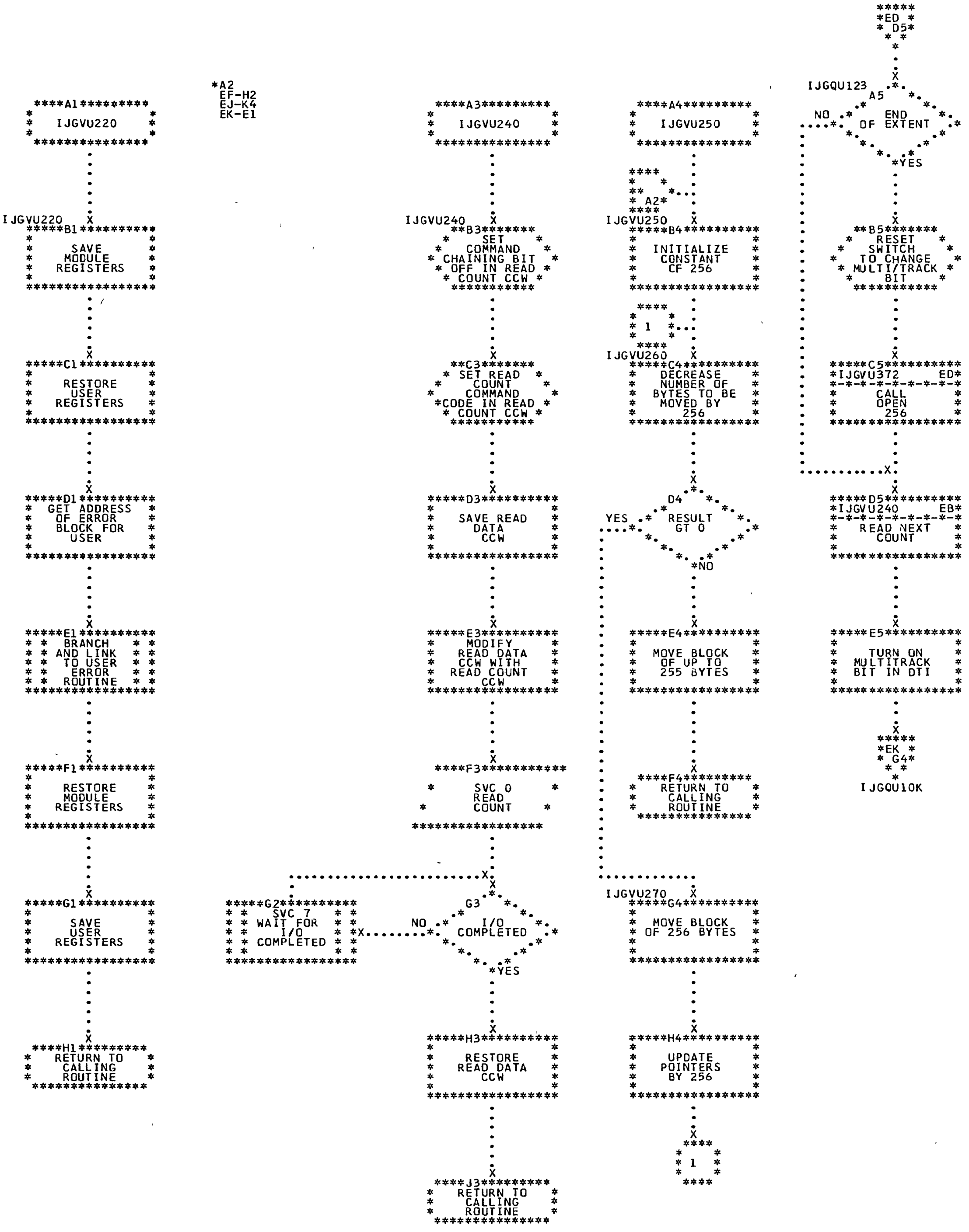
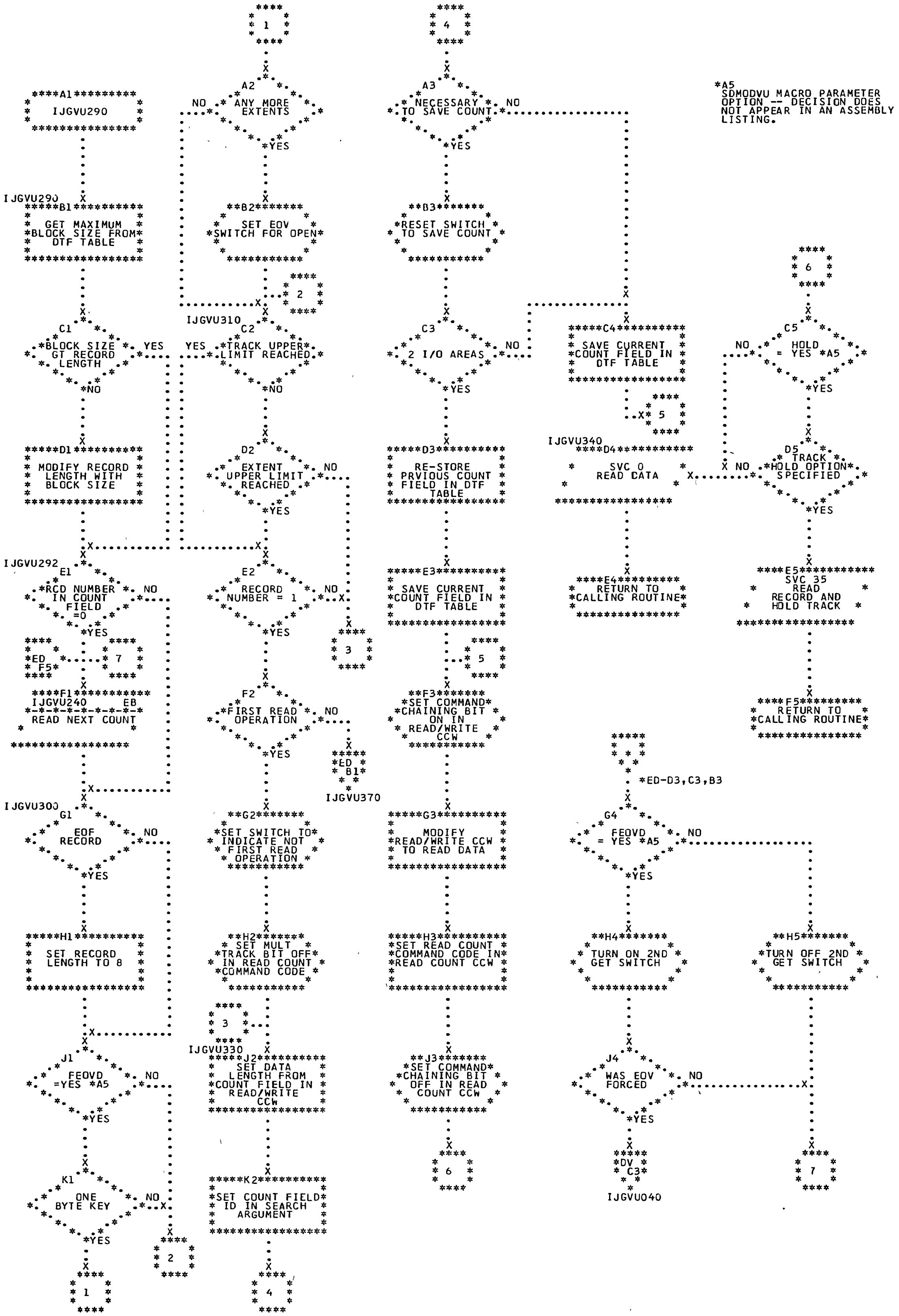
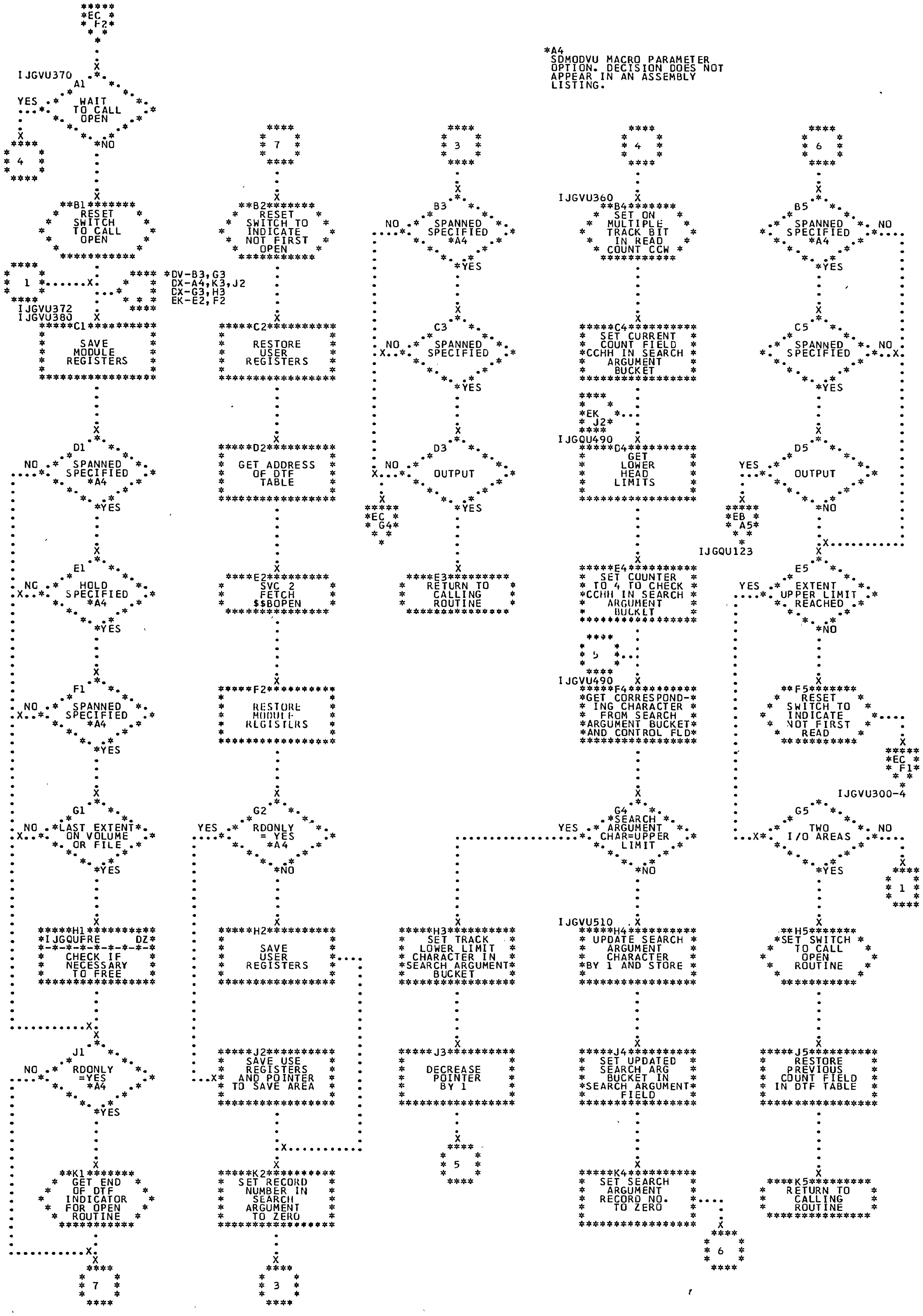
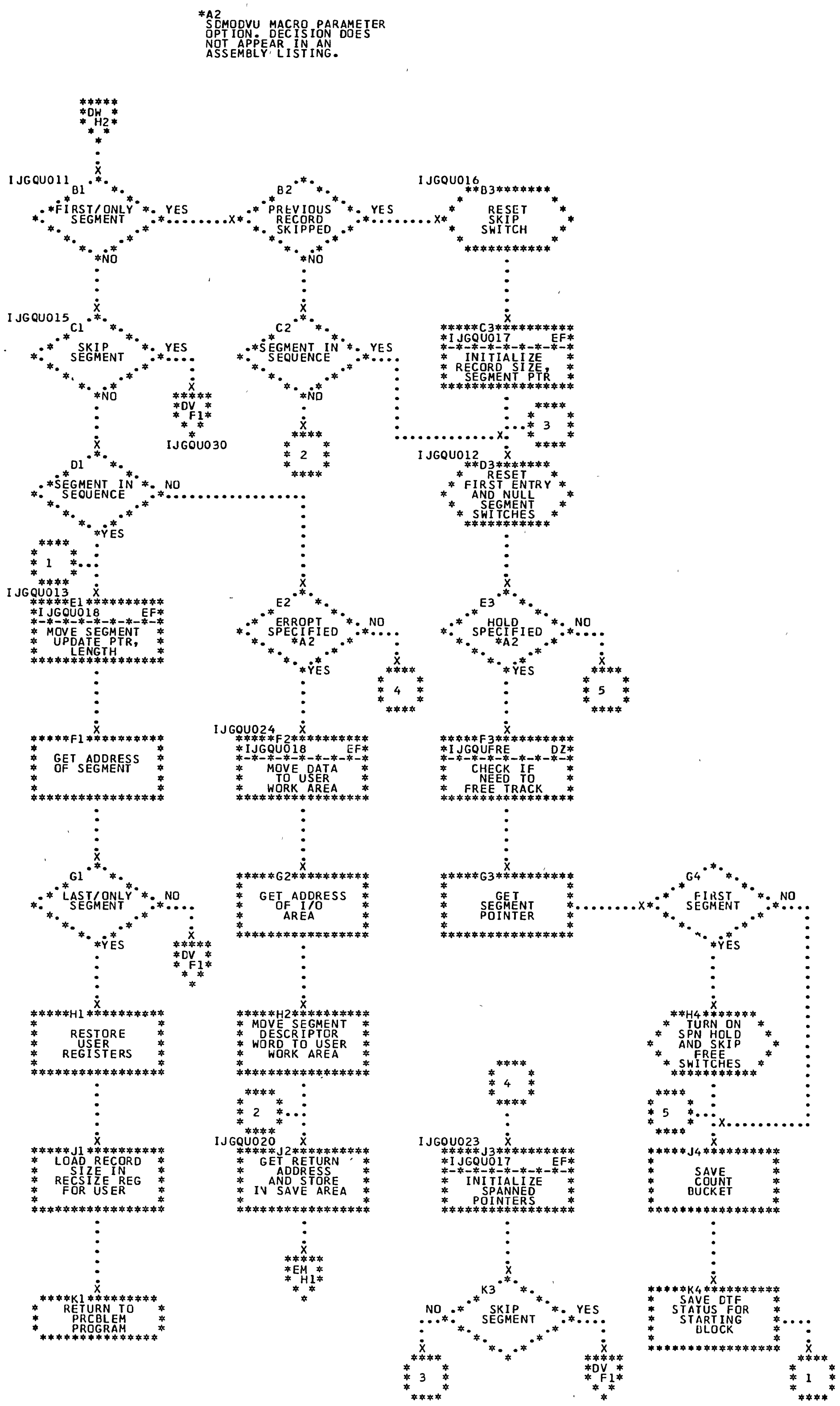

ED D5
IJGQU123 A5
END OF EXTENT
NO
YES
A1
IJGVU220
*A2
EF-H2
EJ-K4
EK-E1
A3
IJGVU240
A4
IJGVU250
A2
IJGVU220 B1
SAVE MODULE REGISTERS
IJGVU240 B3
SET COMMAND CHAINING BIT OFF IN READ COUNT CCW
IJGVU250 B4
INITIALIZE CONSTANT CF 256
B5
RESET SWITCH TO CHANGE MULTI/TRACK BIT
1
C1
RESTORE USER REGISTERS
C3
SET READ COUNT COMMAND CODE IN READ COUNT CCW
IJGVU260 C4
DECREASE NUMBER OF BYTES TO BE MOVED BY 256
C5
IJGVU372 ED
CALL OPEN 256
D1
GET ADDRESS OF ERROR BLOCK FOR USER
D3
SAVE READ DATA CCW
D4
RESULT GT 0
YES
NO
D5
IJGVU240 EB
READ NEXT COUNT
E1
BRANCH AND LINK TO USER ERROR ROUTINE
E3
MODIFY READ DATA CCW WITH READ COUNT CCW
E4
MOVE BLOCK OF UP TO 255 BYTES
E5
TURN ON MULTITRACK BIT IN DTI
EK G4
IJGQU10K
F1
RESTORE MODULE REGISTERS
F3
SVC 0 READ COUNT
F4
RETURN TO CALLING ROUTINE
G1
SAVE USER REGISTERS
G2
SVC 7 WAIT FOR I/O COMPLETED
G3
I/O COMPLETED
NO
YES
IJGVU270 G4
MOVE BLOCK OF 256 BYTES
H1
RETURN TO CALLING ROUTINE
H3
RESTORE READ DATA CCW
H4
UPDATE POINTERS BY 256
1
J3
RETURN TO CALLING ROUTINE

Chart EC. SDMODVU: GET Macro (Section 8 of 11)


1
A1
IJGVU290
IJGVU290 B1
GET MAXIMUM BLOCK SIZE FROM DTF TABLE
C1
BLOCK SIZE GT RECORD LENGTH
YES
NO
D1
MODIFY RECORD LENGTH WITH BLOCK SIZE
IJGVU292 E1
RCD NUMBER IN COUNT FIELD =0
NO
YES
ED F5
7
F1
IJGVU240 EB
READ NEXT COUNT
IJGVU300 G1
EOF RECORD
NO
YES
H1
SET RECORD LENGTH TO 8
J1
FEOVD =YES *A5
NO
YES
K1
ONE BYTE KEY
NO
YES
2
1
A2
ANY MORE EXTENTS
NO
YES
B2
SET EOV SWITCH FOR OPEN
2
IJGVU310 C2
TRACK UPPER LIMIT REACHED
YES
NO
D2
EXTENT UPPER LIMIT REACHED
NO
YES
E2
RECORD NUMBER = 1
NO
YES
3
F2
FIRST READ OPERATION
NO
YES
ED B1
IJGVU370
G2
SET SWITCH TO INDICATE NOT FIRST READ OPERATION
H2
SET MULT TRACK BIT OFF IN READ COUNT COMMAND CODE
3
IJGVU330 J2
SET DATA LENGTH FROM COUNT FIELD IN READ/WRITE CCW
K2
SET COUNT FIELD ID IN SEARCH ARGUMENT
4
4
A3
NECESSARY TO SAVE COUNT
NO
YES
B3
RESET SWITCH TO SAVE COUNT
C3
2 I/O AREAS
NO
YES
D3
RE-STORE PRVIOUS COUNT FIELD IN DTF TABLE
E3
SAVE CURRENT COUNT FIELD IN DTF TABLE
5
F3
SET COMMAND CHAINING BIT ON IN READ/WRITE CCW
G3
MODIFY READ/WRITE CCW TO READ DATA
H3
SET READ COUNT COMMAND CODE IN READ COUNT CCW
J3
SET COMMAND CHAINING BIT OFF IN READ COUNT CCW
6
C4
SAVE CURRENT COUNT FIELD IN DTF TABLE
5
IJGVU340 D4
SVC 0 READ DATA
E4
RETURN TO CALLING ROUTINE
ED-D3,C3,B3
G4
FEOVD = YES *A5
NO
YES
H4
TURN ON 2ND GET SWITCH
J4
WAS EOV FORCED
NO
YES
DV C3
IJGVU040
*A5
SDMODVU MACRO PARAMETER OPTION -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
6
C5
HOLD = YES *A5
NO
YES
D5
TRACK HOLD OPTION SPECIFIED
NO
YES
E5
SVC 35 READ RECORD AND HOLD TRACK
F5
RETURN TO CALLING ROUTINE
H5
TURN OFF 2ND GET SWITCH
7


*A4
SDMODVU MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
EC F2
IJGVU370
A1 WAIT TO CALL OPEN
YES
NO
4
B1 RESET SWITCH TO CALL OPEN
1
IJGVU372
IJGVU380
*DV-B3,G3 DX-A4,K3,J2 CX-G3,H3 EK-E2,F2
C1 SAVE MODULE REGISTERS
D1 SPANNED SPECIFIED *A4
E1 HOLD SPECIFIED *A4
F1 SPANNED SPECIFIED *A4
G1 LAST EXTENT ON VOLUME OR FILE
H1 IJGQUFRE DZ CHECK IF NECESSARY TO FREE
J1 RDONLY =YES *A4
K1 GET END OF DTF INDICATOR FOR OPEN ROUTINE
7
7
B2 RESET SWITCH TO INDICATE NOT FIRST OPEN
C2 RESTORE USER REGISTERS
D2 GET ADDRESS OF DTF TABLE
E2 SVC 2 FETCH $$BOPEN
F2 RESTORE MODULE REGISTERS
G2 RDONLY = YES *A4
H2 SAVE USER REGISTERS
J2 SAVE USE REGISTERS AND POINTER TO SAVE AREA
K2 SET RECORD NUMBER IN SEARCH ARGUMENT TO ZERO
3
3
B3 SPANNED SPECIFIED *A4
C3 SPANNED SPECIFIED
D3 OUTPUT
EC G4
E3 RETURN TO CALLING ROUTINE
H3 SET TRACK LOWER LIMIT CHARACTER IN SEARCH ARGUMENT BUCKET
J3 DECREASE POINTER BY 1
5
4
IJGVU360
B4 SET ON MULTIPLE TRACK BIT IN READ COUNT CCW
C4 SET CURRENT COUNT FIELD CCHH IN SEARCH ARGUMENT BUCKET
EK J2
IJGQU490
D4 GET LOWER HEAD LIMITS
E4 SET COUNTER TO 4 TO CHECK CCHH IN SEARCH ARGUMENT BUCKET
5
IJGVU490
F4 GET CORRESPONDING CHARACTER FROM SEARCH ARGUMENT BUCKET AND CONTROL FLD
G4 SEARCH ARGUMENT CHAR=UPPER LIMIT
IJGVU510
H4 UPDATE SEARCH ARGUMENT CHARACTER BY 1 AND STORE
J4 SET UPDATED SEARCH ARG BUCKET IN SEARCH ARGUMENT FIELD
K4 SET SEARCH ARGUMENT RECORD NO. TO ZERO
6
6
B5 SPANNED SPECIFIED *A4
C5 SPANNED SPECIFIED
D5 OUTPUT
EB A5
IJGQU123
E5 EXTENT UPPER LIMIT REACHED
F5 RESET SWITCH TO INDICATE NOT FIRST READ
EC F1
IJGVU300-4
G5 TWO I/O AREAS
1
H5 SET SWITCH TO CALL OPEN ROUTINE
J5 RESTORE PREVIOUS COUNT FIELD IN DTF TABLE
K5 RETURN TO CALLING ROUTINE

## Chart EE. SDMODVU: GET Macro (Section 10 of 11)



## Chart EF. SDMODVU: GET Macro (Section 11 of 11)

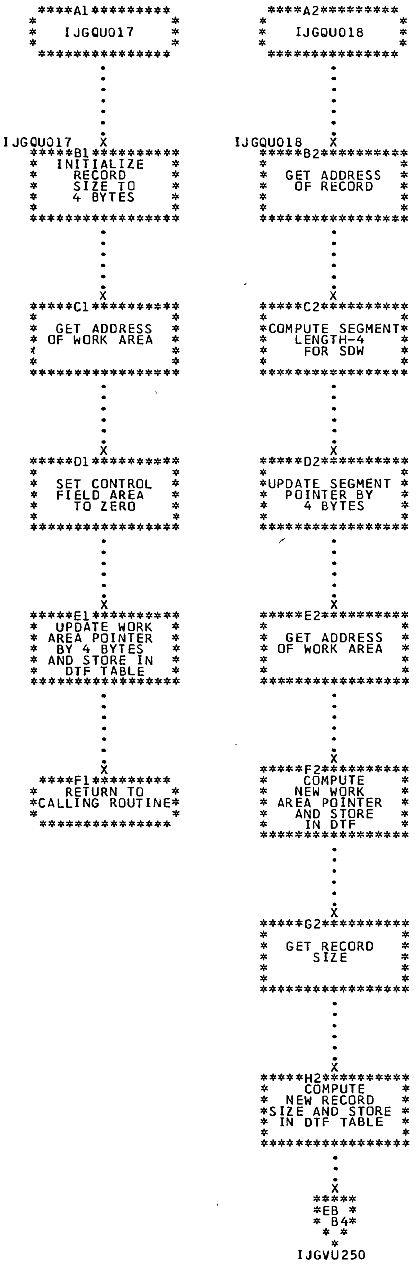

A1
IJGQU017
IJGQU017
B1
INITIALIZE RECORD SIZE TO 4 BYTES
C1
GET ADDRESS OF WORK AREA
D1
SET CONTROL FIELD AREA TO ZERO
E1
UPDATE WORK AREA POINTER BY 4 BYTES AND STORE IN DTF TABLE
F1
RETURN TO CALLING ROUTINE
A2
IJGQU018
IJGQU018
B2
GET ADDRESS OF RECORD
C2
COMPUTE SEGMENT LENGTH-4 FOR SDW
D2
UPDATE SEGMENT POINTER BY 4 BYTES
E2
GET ADDRESS OF WORK AREA
F2
COMPUTE NEW WORK AREA POINTER AND STORE IN DTF
G2
GET RECORD SIZE
H2
COMPUTE NEW RECORD SIZE AND STORE IN DTF TABLE
EB
B4
IJGVU250

Chart EG. SDMODVU: PUT Macro (Section 1 of 6)

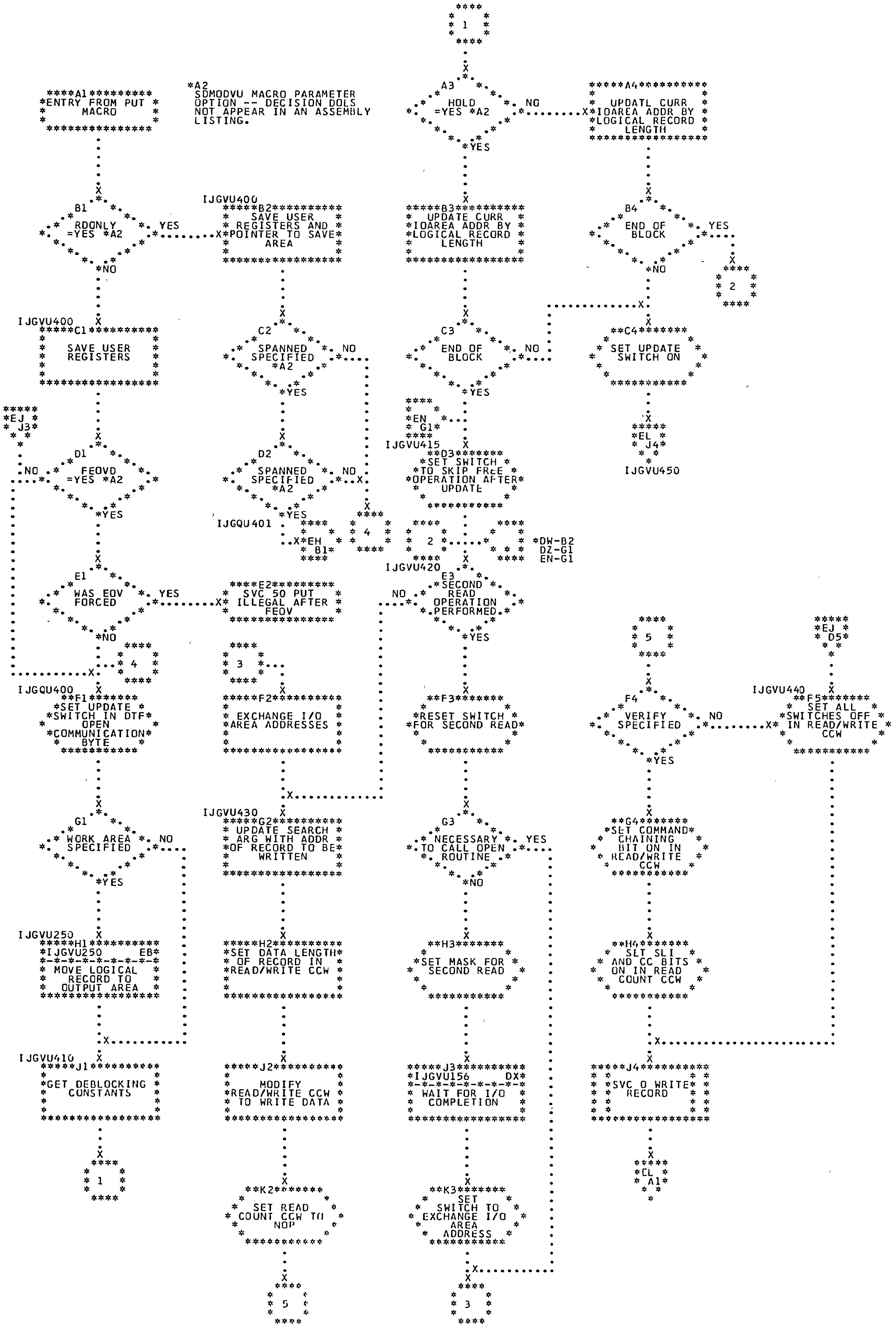

1
A1 ENTRY FROM PUT MACRO
A2 SDMODVU MACRO PARAMETER OPTION -- DECISION DOLS NOT APPEAR IN AN ASSEMBLY LISTING.
A3 HOLD =YES *A2
NO
YES
A4 UPDATL CURR IOAREA ADDR BY LOGICAL RECORD LENGTH
B1 RDONLY =YES *A2
YES
NO
IJGVU400
B2 SAVE USER REGISTERS AND POINTER TO SAVE AREA
B3 UPDATE CURR IOAREA ADDR BY LOGICAL RECORD LENGTH
B4 END OF BLOCK
YES
NO
2
IJGVU400
C1 SAVE USER REGISTERS
C2 SPANNED SPECIFIED *A2
NO
YES
C3 END OF BLOCK
NO
YES
C4 SET UPDATE SWITCH ON
EN G1
EJ J3
D1 FEOVD =YES *A2
NO
YES
D2 SPANNED SPECIFIED *A2
NO
YES
IJGVU415
D3 SET SWITCH TO SKIP FREE OPERATION AFTER UPDATE
EL J4
IJGVU450
IJGQU401
EH B1
4
2
DW-B2
DZ-G1
EN-G1
E1 WAS EOV FORCED
YES
NO
E2 SVC 50 PUT ILLEGAL AFTER FEOV
IJGVU420
E3 SECOND READ OPERATION PERFORMED
NO
YES
4
3
5
EJ D5
IJGQU400
F1 SET UPDATE SWITCH IN DTF OPEN COMMUNICATION BYTE
F2 EXCHANGE I/O AREA ADDRESSES
F3 RESET SWITCH FOR SECOND READ
F4 VERIFY SPECIFIED
NO
YES
IJGVU440
F5 SET ALL SWITCHES OFF IN READ/WRITE CCW
G1 WORK AREA SPECIFIED
NO
YES
IJGVU430
G2 UPDATE SEARCH ARG WITH ADDR OF RECORD TO BE WRITTEN
G3 NECESSARY TO CALL OPEN ROUTINE
YES
NO
G4 SET COMMAND CHAINING BIT ON IN READ/WRITE CCW
IJGVU250
H1 IJGVU250 EB MOVE LOGICAL RECORD TO OUTPUT AREA
H2 SET DATA LENGTH OF RECORD IN READ/WRITE CCW
H3 SET MASK FOR SECOND READ
H4 SLT SLI AND CC BITS ON IN READ COUNT CCW
IJGVU410
J1 GET DEBLOCKING CONSTANTS
J2 MODIFY READ/WRITE CCW TO WRITE DATA
J3 IJGVU156 DX WAIT FOR I/O COMPLETION
J4 SVC 0 WRITE RECORD
1
CL A1
K2 SET READ COUNT CCW TO NOP
K3 SET SWITCH TO EXCHANGE I/O AREA ADDRESS
5
3

Chart EH. SDMODVU: PUT Macro (Section 2 of 6)

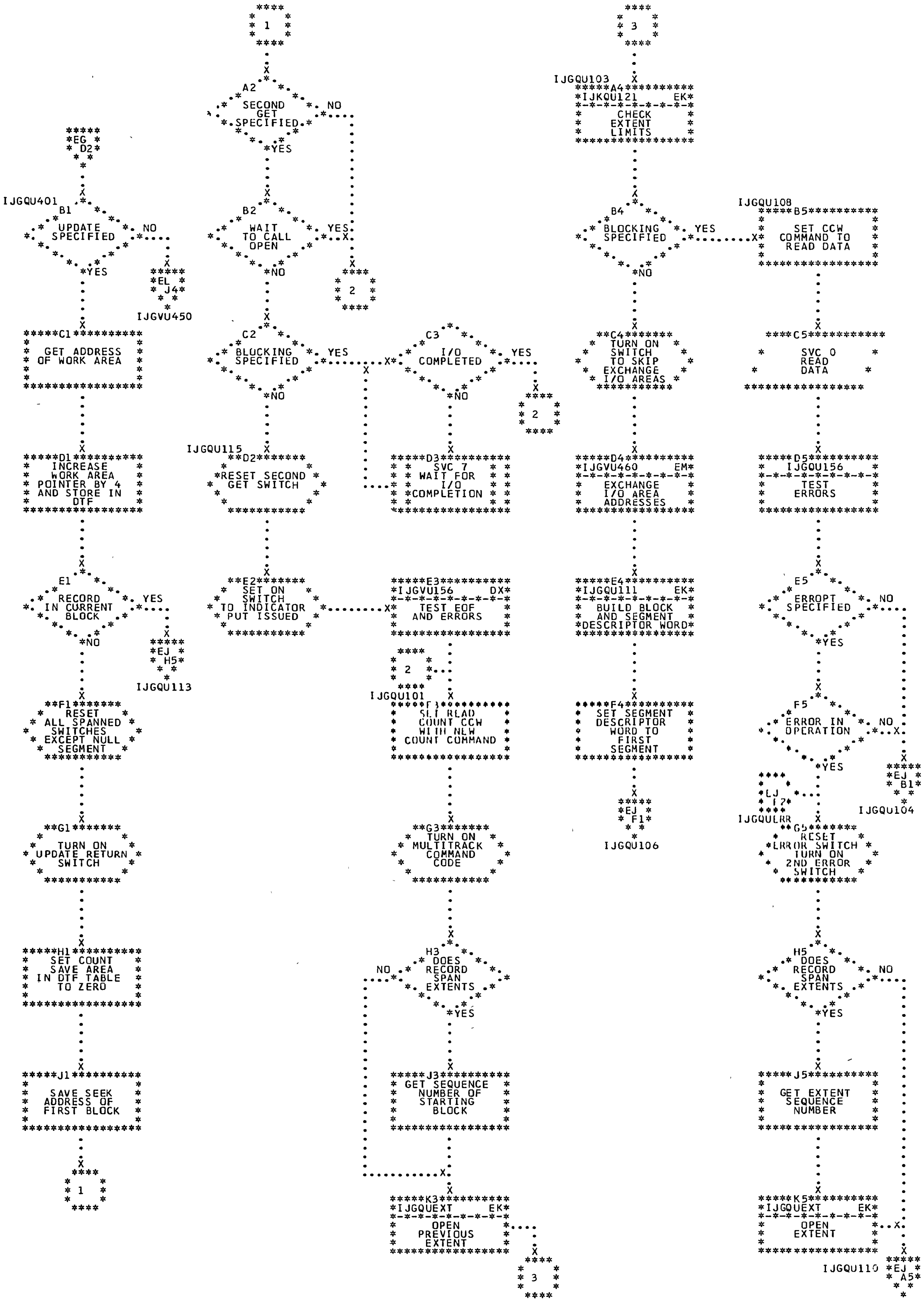


Chart EJ. SDMODVU: PUT Macro (Section 3 of 6)

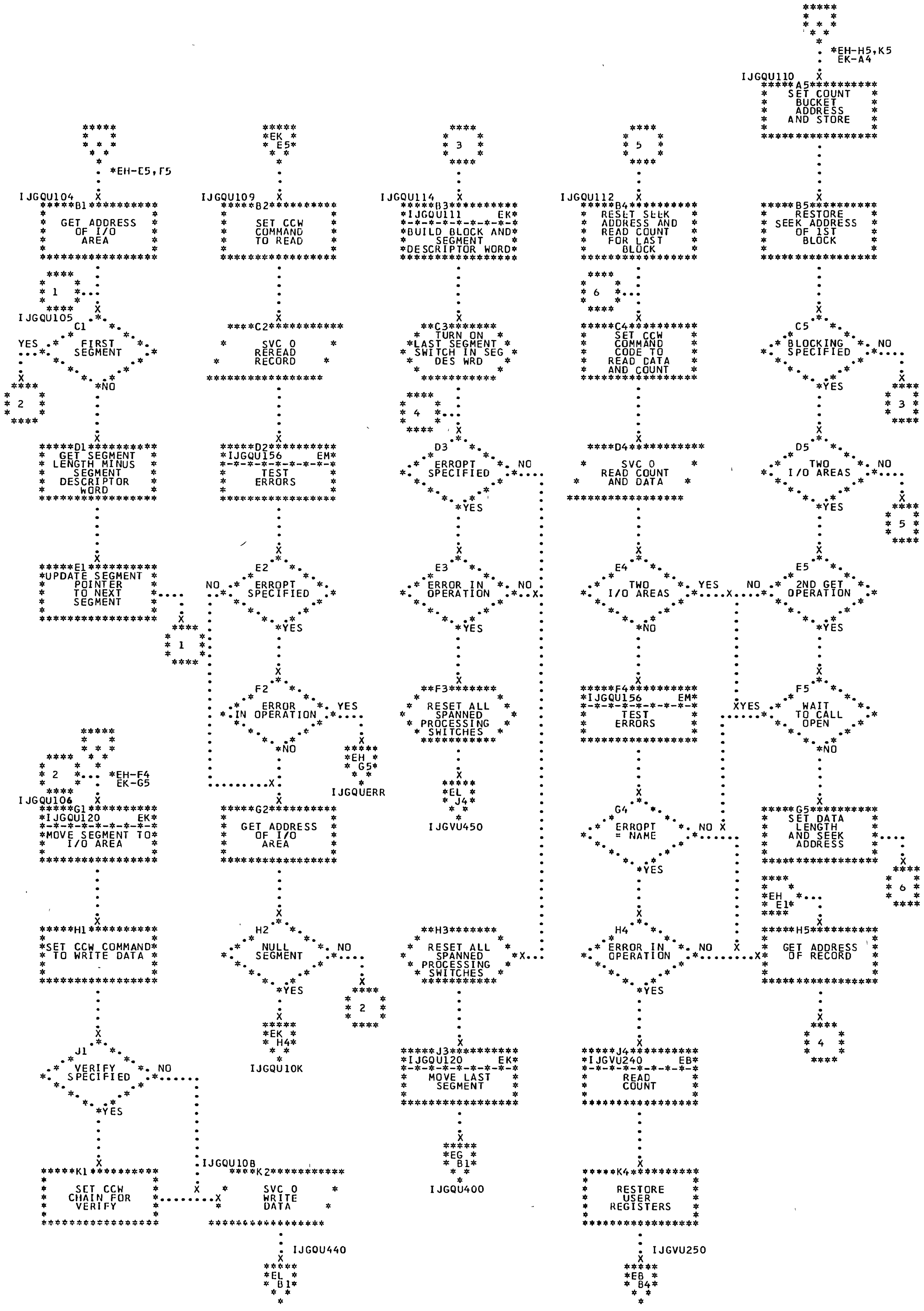
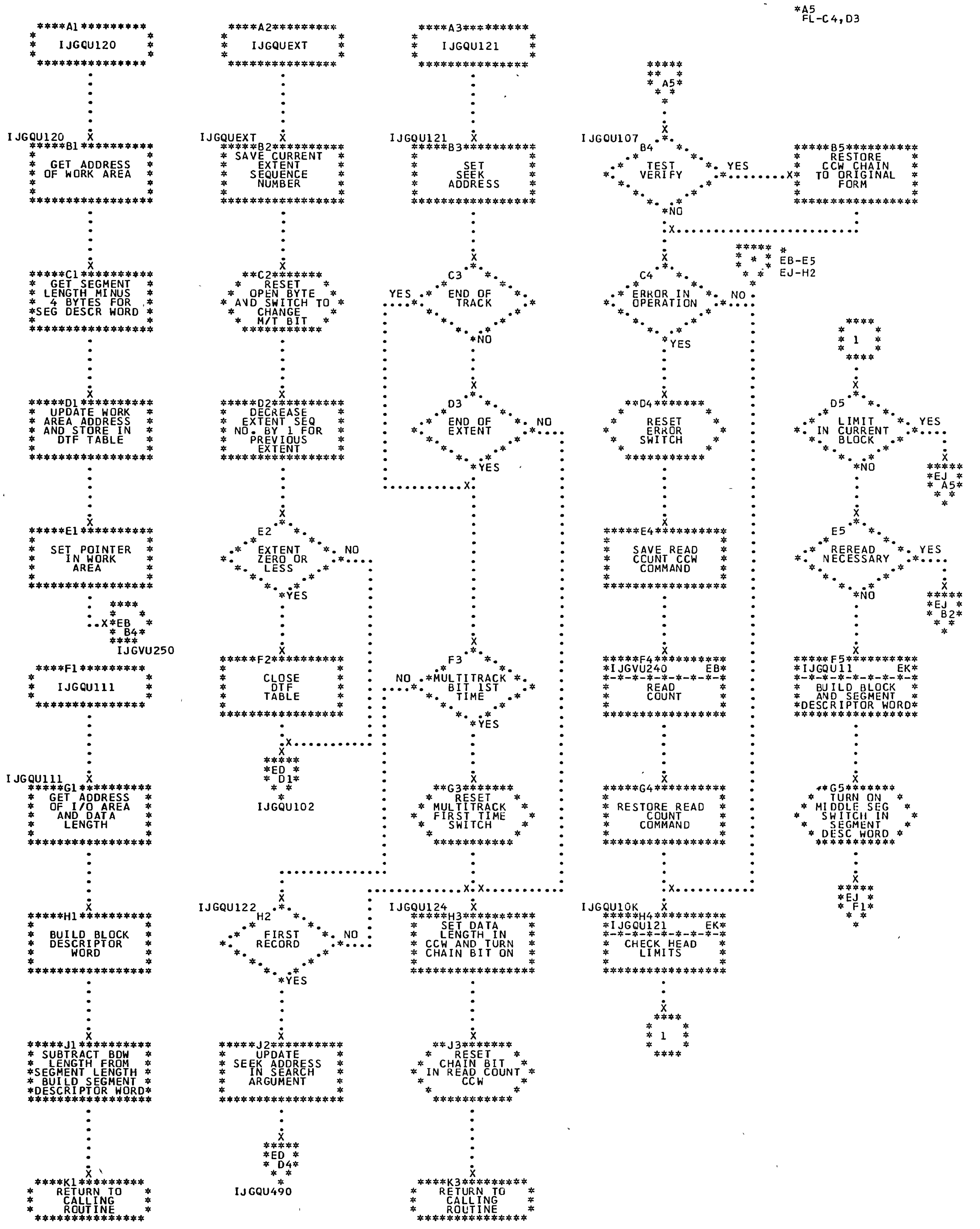
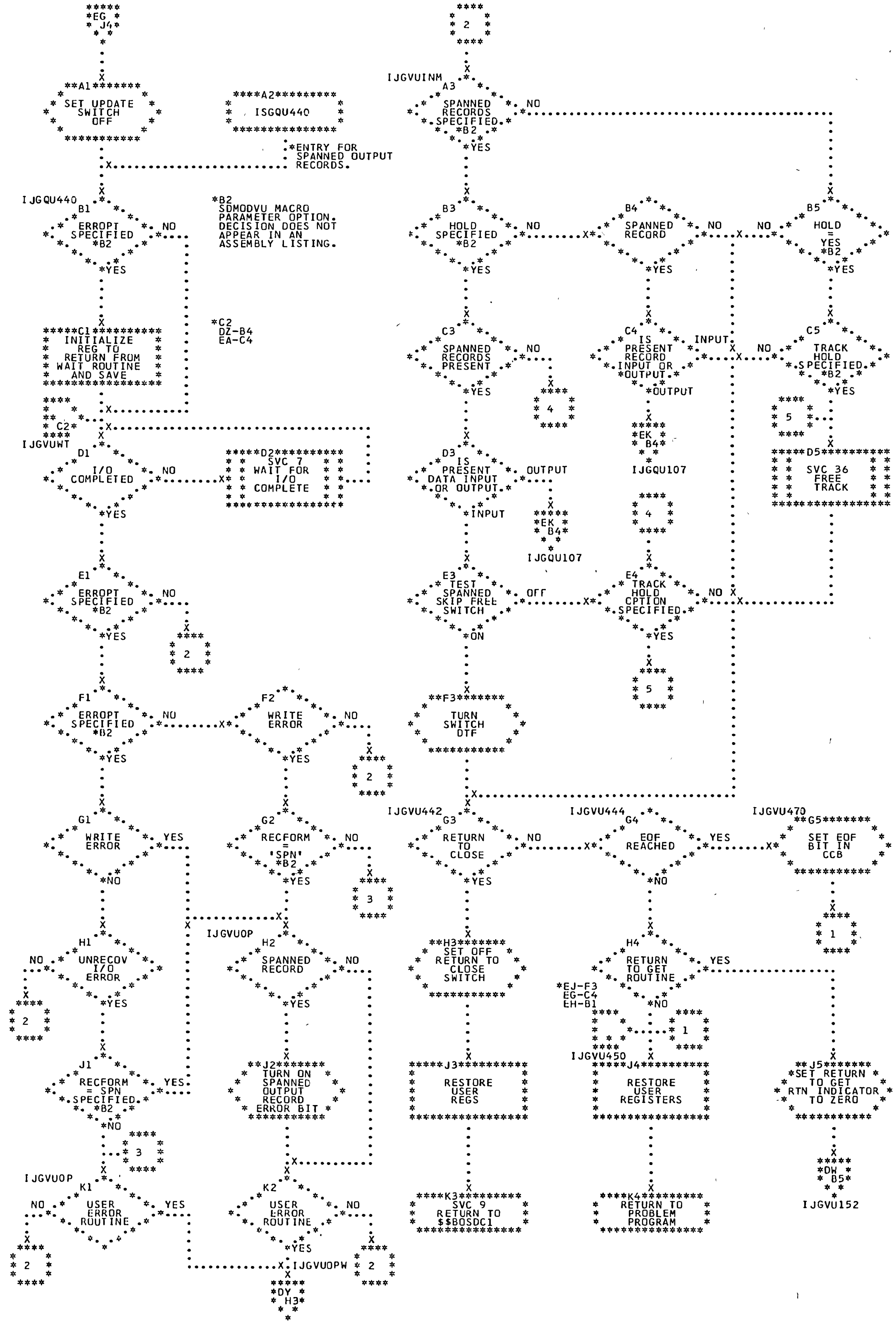

*EH-C5,F5
IJGQU104
B1 GET ADDRESS OF I/O AREA
1
IJGQU105
C1 FIRST SEGMENT
YES
NO
2
D1 GET SEGMENT LENGTH MINUS SEGMENT DESCRIPTOR WORD
E1 UPDATE SEGMENT POINTER TO NEXT SEGMENT
1
2
*EH-F4 EK-G5
IJGQU106
G1 IJGQU120 EK MOVE SEGMENT TO I/O AREA
H1 SET CCW COMMAND TO WRITE DATA
J1 VERIFY SPECIFIED
NO
YES
K1 SET CCW CHAIN FOR VERIFY
EK E5
IJGQU109
B2 SET CCW COMMAND TO READ
C2 SVC 0 REREAD RECORD
D2 IJGQU156 EM TEST ERRORS
E2 ERROPT SPECIFIED
NO
YES
F2 ERROR IN OPERATION
YES
NO
EH G5
IJGQUERR
G2 GET ADDRESS OF I/O AREA
H2 NULL SEGMENT
NO
YES
2
EK H4
IJGQU10K
IJGQU108
K2 SVC 0 WRITE DATA
IJGQU440
EL B1
3
IJGQU114
B3 IJGQU111 EK BUILD BLOCK AND SEGMENT DESCRIPTOR WORD
C3 TURN ON LAST SEGMENT SWITCH IN SEG DES WRD
4
D3 ERROPT SPECIFIED
NO
YES
E3 ERROR IN OPERATION
NO
YES
F3 RESET ALL SPANNED PROCESSING SWITCHES
EL J4
IJGVU450
H3 RESET ALL SPANNED PROCESSING SWITCHES
J3 IJGQU120 EK MOVE LAST SEGMENT
EG B1
IJGQU400
5
IJGQU112
B4 RESET SEEK ADDRESS AND READ COUNT FOR LAST BLOCK
6
C4 SET CCW COMMAND CODE TO READ DATA AND COUNT
D4 SVC 0 READ COUNT AND DATA
E4 TWO I/O AREAS
YES
NO
F4 IJGQU156 EM TEST ERRORS
G4 ERROPT = NAME
NO
YES
H4 ERROR IN OPERATION
NO
YES
J4 IJGVU240 EB READ COUNT
K4 RESTORE USER REGISTERS
IJGVU250
EB B4
*EH-H5,K5 EK-A4
IJGQU110
A5 SET COUNT BUCKET ADDRESS AND STORE
B5 RESTORE SEEK ADDRESS OF 1ST BLOCK
C5 BLOCKING SPECIFIED
NO
YES
3
D5 TWO I/O AREAS
NO
YES
5
E5 2ND GET OPERATION
NO
YES
F5 WAIT TO CALL OPEN
YES
NO
G5 SET DATA LENGTH AND SEEK ADDRESS
6
EH E1
H5 GET ADDRESS OF RECORD
4

Chart EK. SDMODVU: PUT Macro (Section 4 of 6)

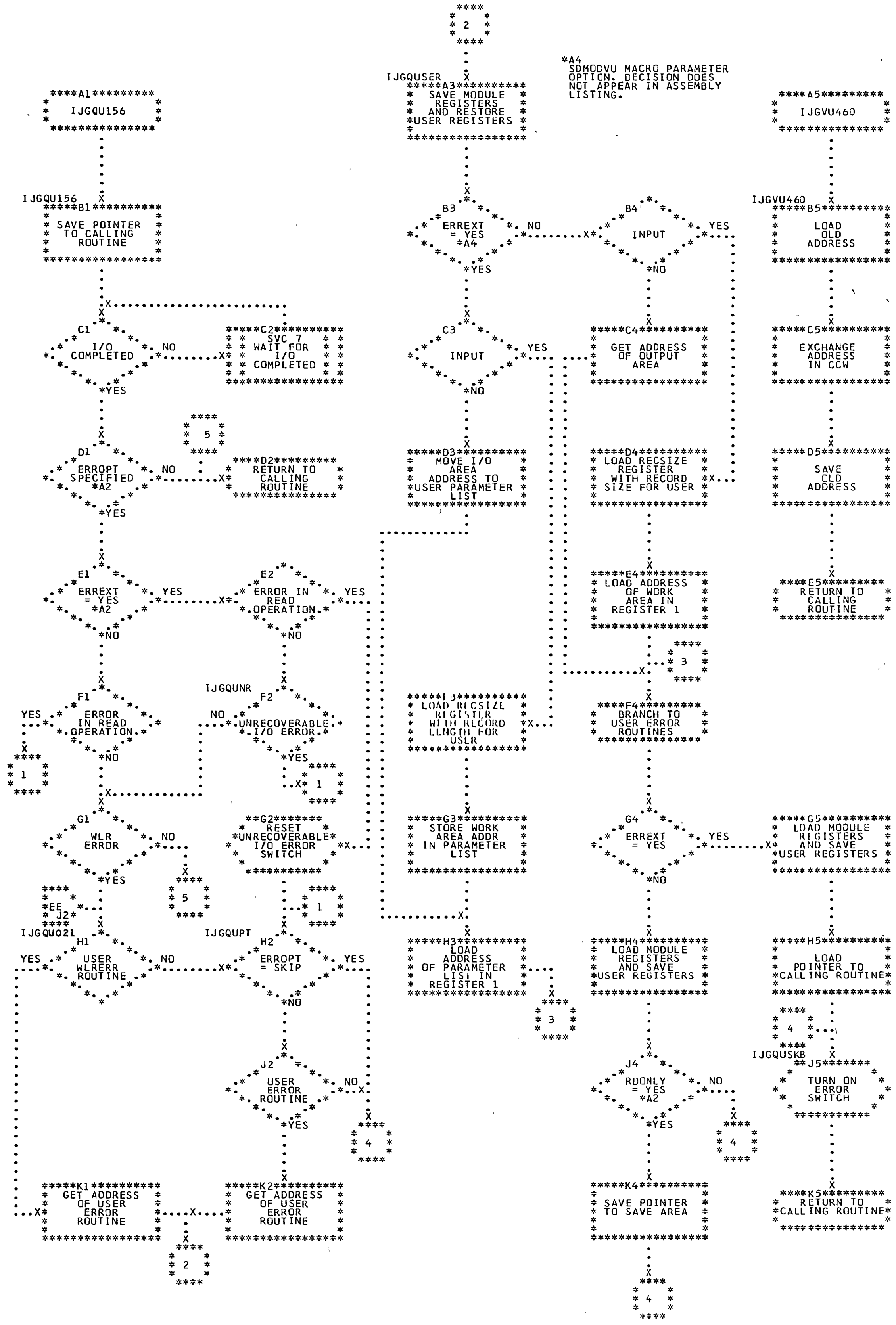

A1 IJGQU120
IJGQU120 B1 GET ADDRESS OF WORK AREA
C1 GET SEGMENT LENGTH MINUS 4 BYTES FOR SEG DESCR WORD
D1 UPDATE WORK AREA ADDRESS AND STORE IN DTF TABLE
E1 SET POINTER IN WORK AREA
EB B4
IJGVU250
F1 IJGQU111
IJGQU111 G1 GET ADDRESS OF I/O AREA AND DATA LENGTH
H1 BUILD BLOCK DESCRIPTOR WORD
J1 SUBTRACT BDW LENGTH FROM SEGMENT LENGTH TO BUILD SEGMENT DESCRIPTOR WORD
K1 RETURN TO CALLING ROUTINE
A2 IJGQUEXT
IJGQUEXT B2 SAVE CURRENT EXTENT SEQUENCE NUMBER
C2 RESET OPEN BYTE AND SWITCH TO CHANGE M/T BIT
D2 DECREASE EXTENT SEQ NO. BY 1 FOR PREVIOUS EXTENT
E2 EXTENT ZERO OR LESS
NO
YES
F2 CLOSE DTF TABLE
ED D1
IJGQU102
IJGQU122 H2 FIRST RECORD
NO
YES
J2 UPDATE SEEK ADDRESS IN SEARCH ARGUMENT
ED D4
IJGQU490
A3 IJGQU121
IJGQU121 B3 SET SEEK ADDRESS
C3 END OF TRACK
YES
NO
D3 END OF EXTENT
NO
YES
F3 MULTITRACK BIT 1ST TIME
NO
YES
G3 RESET MULTITRACK FIRST TIME SWITCH
IJGQU124 H3 SET DATA LENGTH IN CCW AND TURN CHAIN BIT ON
J3 RESET CHAIN BIT IN READ COUNT CCW
K3 RETURN TO CALLING ROUTINE
*A5
FL-C4,D3
A5
IJGQU107 B4 TEST VERIFY
YES
NO
B5 RESTORE CCW CHAIN TO ORIGINAL FORM
C4 ERKOR IN OPERATION
NO
YES
EB-E5
EJ-H2
D4 RESET ERROR SWITCH
E4 SAVE READ CCUNT CCW COMMAND
F4 IJGVU240 EB READ COUNT
G4 RESTORE READ COUNT COMMAND
IJGQU10K H4 IJGQU121 EK CHECK HEAD LIMITS
1
1
D5 LIMIT IN CURRENT BLOCK
YES
NO
EJ A5
E5 REREAD NECESSARY
YES
NO
EJ B2
F5 IJGQU11 EK BUILD BLOCK AND SEGMENT DESCRIPTOR WORD
G5 TURN ON MIDDLE SEG SWITCH IN SEGMENT DESC WORD
EJ F1

## Chart EL. SDMODVU: PUT Macro (Section 5 of 6)

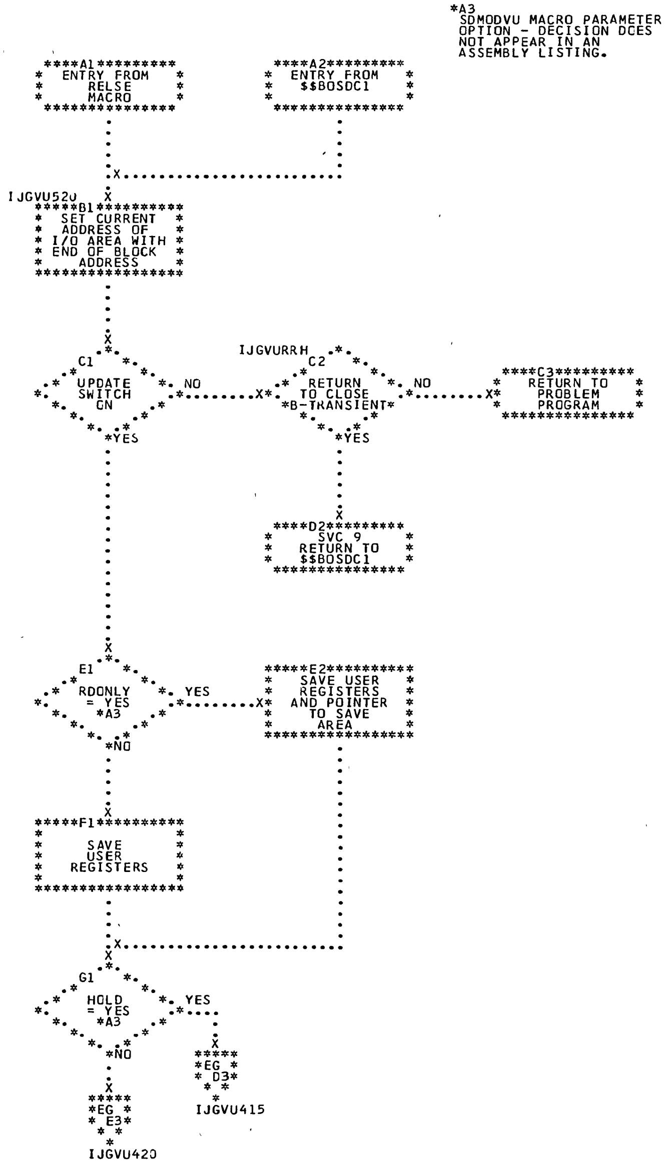


EG J4
A1 SET UPDATE SWITCH OFF
A2 ISGQU440
ENTRY FOR SPANNED OUTPUT RECORDS.
IJGQU440
B1 ERROPT SPECIFIED *B2
NO
YES
*B2 SDMODVU MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
C1 INITIALIZE REG TO RETURN FROM WAIT ROUTINE AND SAVE
*C2 DZ-B4 EA-C4
C2
IJGVUWT
D1 I/O COMPLETED
D2 SVC 7 WAIT FOR I/O COMPLETE
E1 ERROPT SPECIFIED *B2
2
F1 ERROPT SPECIFIED *B2
F2 WRITE ERROR
G1 WRITE ERROR
G2 RECFORM = 'SPN' *B2
3
IJGVUOP
H1 UNRECOV I/O ERROR
H2 SPANNED RECORD
J1 RECFORM = SPN SPECIFIED *B2
J2 TURN ON SPANNED OUTPUT RECORD ERROR BIT
IJGVUOP
K1 USER ERROR ROUTINE
K2 USER ERROR ROUTINE
IJGVUOPW
DY H3
2
IJGVUINM
A3 SPANNED RECORDS SPECIFIED *B2
B3 HOLD SPECIFIED *B2
B4 SPANNED RECORD
B5 HOLD = YES *B2
C3 SPANNED RECORDS PRESENT
4
C4 IS PRESENT RECORD INPUT OR OUTPUT
INPUT
OUTPUT
EK B4
IJGQU107
C5 TRACK HOLD SPECIFIED *B2
5
D3 IS PRESENT DATA INPUT OR OUTPUT
D5 SVC 36 FREE TRACK
E3 TEST SPANNED SKIP FREE SWITCH
OFF
ON
E4 TRACK HOLD OPTION SPECIFIED
F3 TURN SWITCH OFF
IJGVU442
G3 RETURN TO CLOSE
IJGVU444
G4 EOF REACHED
IJGVU470
G5 SET EOF BIT IN CCB
1
H3 SET OFF RETURN TO CLOSE SWITCH
H4 RETURN TO GET ROUTINE
*EJ-F3 EG-C4 EH-B1
IJGVU450
J3 RESTORE USER REGS
J4 RESTORE USER REGISTERS
J5 SET RETURN TO GET RTN INDICATOR TO ZERO
DW B5
IJGVU152
K3 SVC 9 RETURN TO $$BOSDC1
K4 RETURN TO PROBLEM PROGRAM

## Chart EM. SDMODVU: PUT Macro (Section 6 of 6)


2
IJGQUSER
A3
SAVE MODULE REGISTERS AND RESTORE USER REGISTERS
*A4
SDMODVU MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.
A1
IJGQU156
A5
IJGVU460
IJGQU156
B1
SAVE POINTER TO CALLING ROUTINE
B3
ERREXT = YES *A4
NO
YES
B4
INPUT
YES
NO
IJGVU460
B5
LOAD OLD ADDRESS
C1
I/O COMPLETED
NO
YES
C2
SVC 7 WAIT FOR I/O COMPLETED
C3
INPUT
YES
NO
C4
GET ADDRESS OF OUTPUT AREA
C5
EXCHANGE ADDRESS IN CCW
5
D1
ERROPT SPECIFIED *A2
NO
YES
D2
RETURN TO CALLING ROUTINE
D3
MOVE I/O AREA ADDRESS TO USER PARAMETER LIST
D4
LOAD RECSIZE REGISTER WITH RECORD SIZE FOR USER
D5
SAVE OLD ADDRESS
E1
ERREXT = YES *A2
YES
NO
E2
ERROR IN READ OPERATION
YES
NO
E4
LOAD ADDRESS OF WORK AREA IN REGISTER 1
E5
RETURN TO CALLING ROUTINE
3
F1
ERROR IN READ OPERATION
YES
NO
1
IJGQUNR
F2
UNRECOVERABLE I/O ERROR
NO
YES
1
F3
LOAD RECSIZE REGISTER WITH RECORD LENGTH FOR USER
F4
BRANCH TO USER ERROR ROUTINES
G1
WLR ERROR
NO
YES
5
G2
RESET UNRECOVERABLE I/O ERROR SWITCH
1
G3
STORE WORK AREA ADDR IN PARAMETER LIST
G4
ERREXT = YES
YES
NO
G5
LOAD MODULE REGISTERS AND SAVE USER REGISTERS
EE J2
IJGQU021
H1
USER WLRERR ROUTINE
YES
NO
IJGQUPT
H2
ERROPT = SKIP
YES
NO
H3
LOAD ADDRESS OF PARAMETER LIST IN REGISTER 1
3
H4
LOAD MODULE REGISTERS AND SAVE USER REGISTERS
H5
LOAD POINTER TO CALLING ROUTINE
4
IJGQUSKB
J5
TURN ON ERROR SWITCH
J2
USER ERROR ROUTINE
NO
YES
4
J4
RDONLY = YES *A2
NO
YES
4
K1
GET ADDRESS OF USER ERROR ROUTINE
K2
GET ADDRESS OF USER ERROR ROUTINE
2
K4
SAVE POINTER TO SAVE AREA
K5
RETURN TO CALLING ROUTINE
4

## Chart EN. SDMODVU: Close Routine, RELSE Macro


*A3
SDMODVU MACRO PARAMETER OPTION - DECISION DCES NOT APPEAR IN AN ASSEMBLY LISTING.
A1
ENTRY FROM RELSE MACRO
A2
ENTRY FROM $$BOSDC1
IJGVU520
B1
SET CURRENT ADDRESS OF I/O AREA WITH END OF BLOCK ADDRESS
C1
UPDATE SWITCH CN
NO
YES
IJGVURRH
C2
RETURN TO CLOSE B-TRANSIENT
NO
YES
C3
RETURN TO PROBLEM PROGRAM
D2
SVC 9 RETURN TO $$BOSDC1
E1
RDONLY = YES *A3
YES
NO
E2
SAVE USER REGISTERS AND POINTER TO SAVE AREA
F1
SAVE USER REGISTERS
G1
HOLD = YES *A3
YES
NO
EG D3
IJGVU415
EG E3
IJGVU420

## Chart EP. CNTRL Macro, Variable-Length Record Modules

NOTE--
X =
G - INPUT FILE
P - OUTPUT FILE
U - INPUT FILE
WITH UPDATING

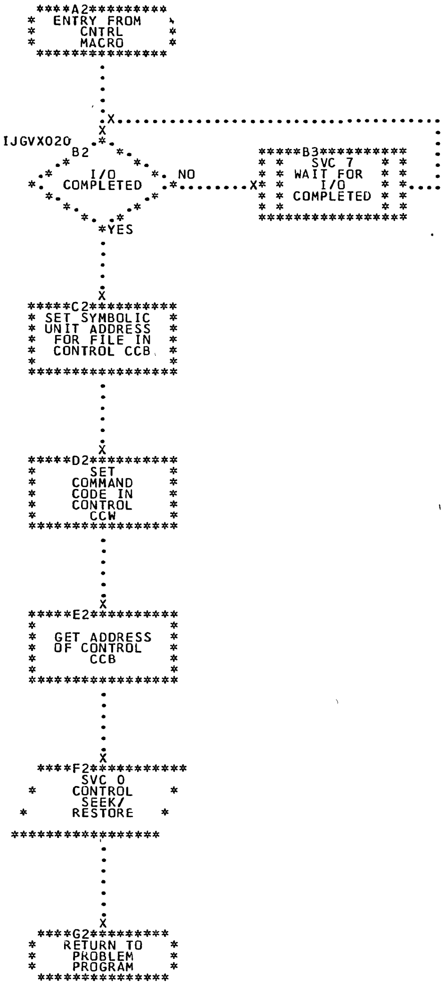
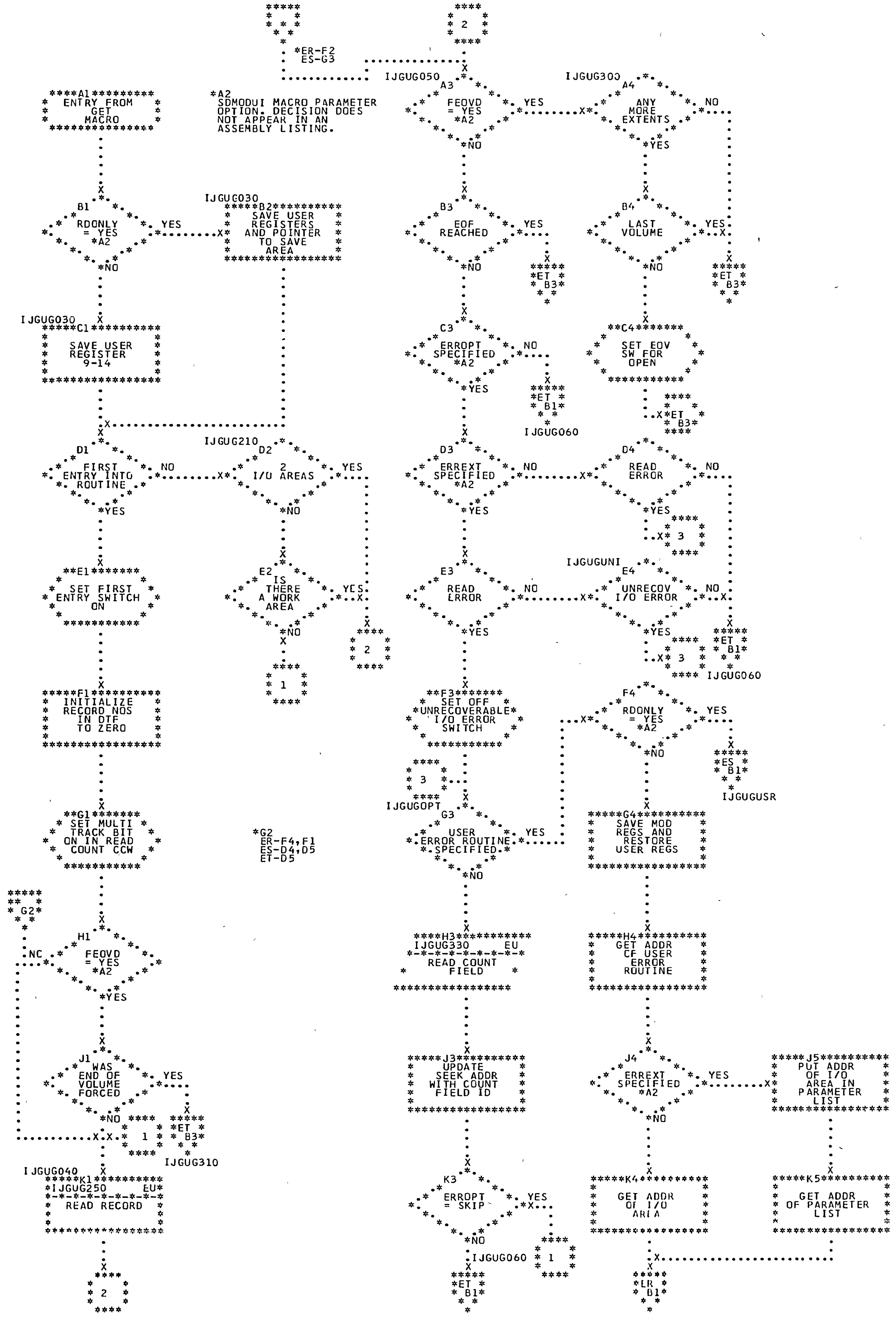
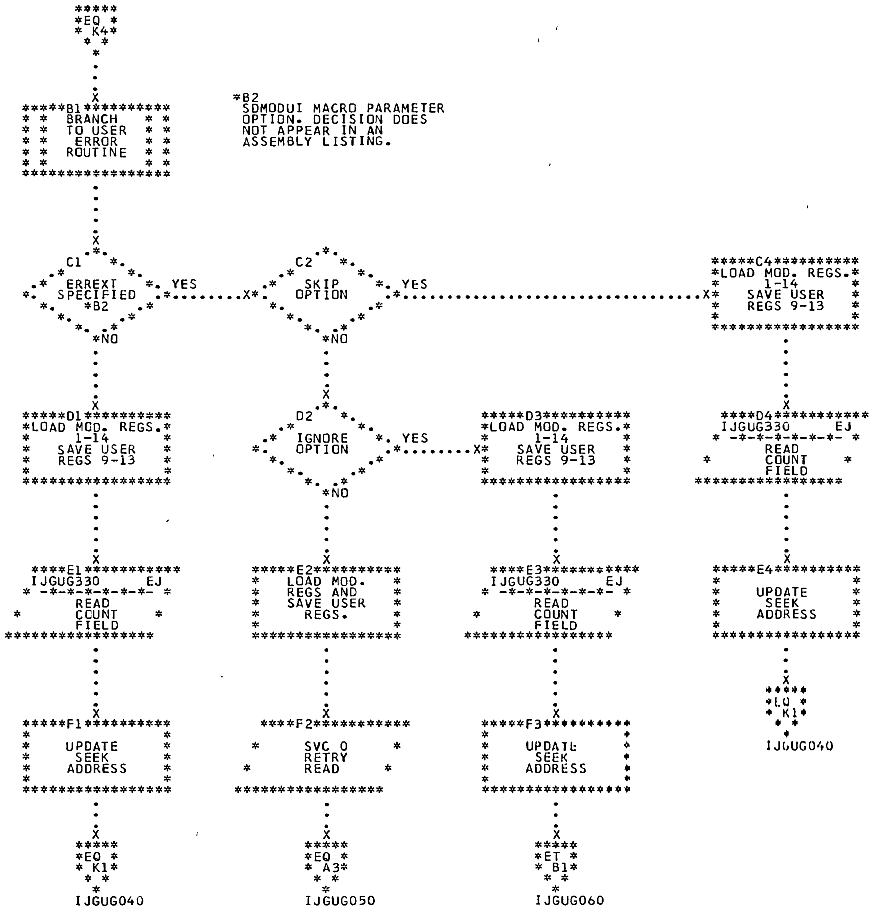

A2
ENTRY FROM CNTRL MACRO
IJGVX020
B2
I/O COMPLETED
NO
YES
B3
SVC 7 WAIT FOR I/O COMPLETED
C2
SET SYMBOLIC UNIT ADDRESS FOR FILE IN CONTROL CCB
D2
SET COMMAND CODE IN CONTROL CCW
E2
GET ADDRESS OF CONTROL CCB
F2
SVC 0 CONTROL SEEK/ RESTORE
G2
RETURN TO PROBLEM PROGRAM
X

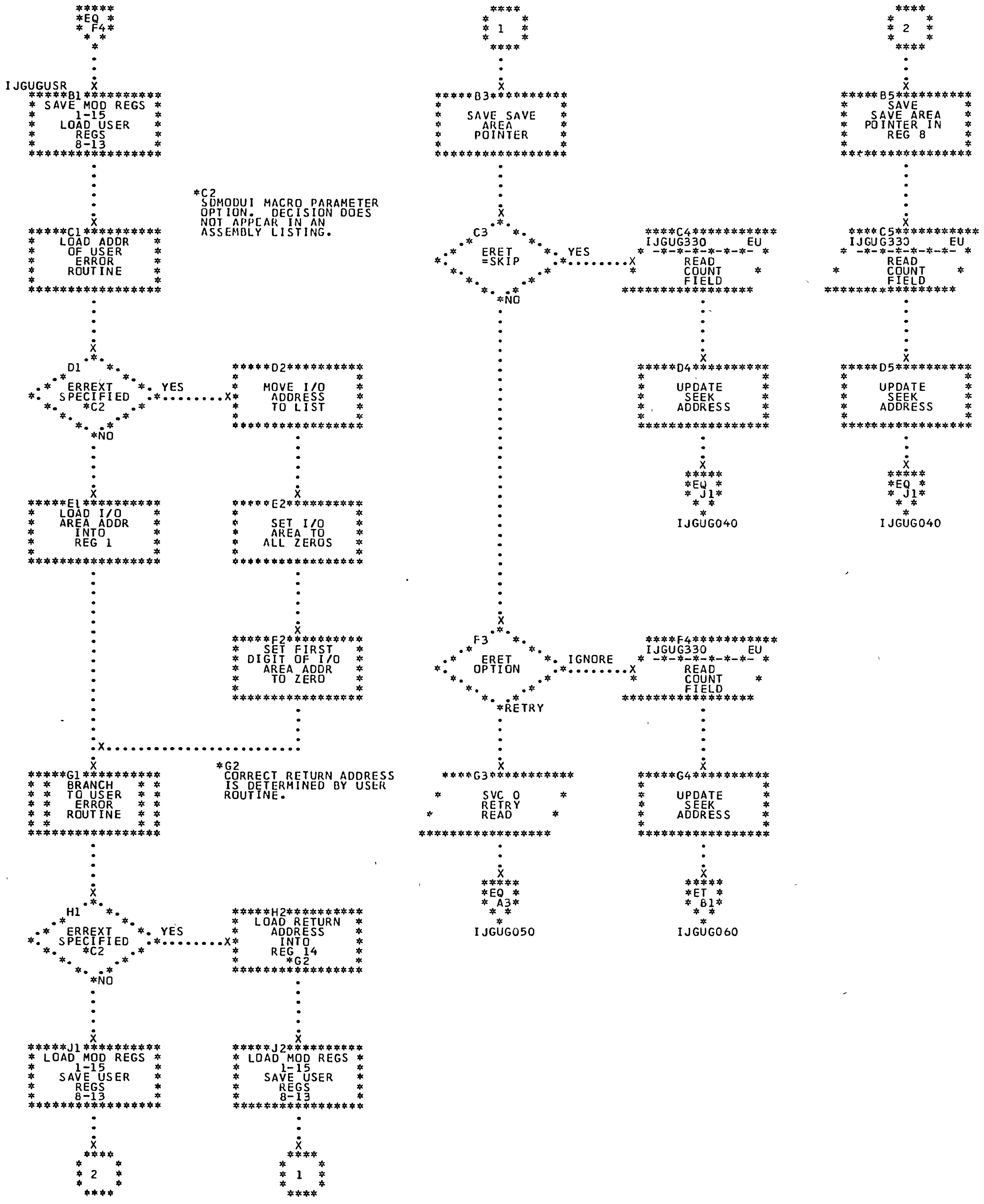
Chart EQ. SDMODUI: GET Macro (Section 1 of 5)


A1 ENTRY FROM GET MACRO
*A2 SDMODUI MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
*ER-F2 ES-G3
2
IJGUG050 A3 FEOVD = YES *A2
YES
NO
IJGUG300 A4 ANY MORE EXTENTS
NO
YES
B1 RDONLY = YES *A2
YES
NO
IJGUG030 B2 SAVE USER REGISTERS AND POINTER TO SAVE AREA
B3 EOF REACHED
YES
ET B3
NO
B4 LAST VOLUME
YES
ET B3
NO
IJGUG030 C1 SAVE USER REGISTER 9-14
C3 ERROPT SPECIFIED *A2
NO
ET B1
IJGUG060
YES
C4 SET EOV SW FOR OPEN
ET B3
D1 FIRST ENTRY INTO ROUTINE
NO
YES
IJGUG210 D2 2 I/O AREAS
YES
NO
D3 ERREXT SPECIFIED *A2
NO
YES
D4 READ ERROR
NO
YES
3
E1 SET FIRST ENTRY SWITCH ON
E2 IS THERE A WORK AREA
YES
NO
2
1
E3 READ ERROR
NO
YES
IJGUGUNI E4 UNRECOV I/O ERROR
NO
YES
3
ET B1
IJGUG060
F1 INITIALIZE RECORD NOS IN DTF TO ZERO
F3 SET OFF UNRECOVERABLE I/O ERROR SWITCH
F4 RDONLY = YES *A2
YES
ES B1
IJGUGUSR
NO
3
IJGUGOPT G3 USER ERROR ROUTINE SPECIFIED
YES
NO
G1 SET MULTI TRACK BIT ON IN READ COUNT CCW
*G2 ER-F4,F1 ES-D4,D5 ET-D5
G4 SAVE MOD REGS AND RESTORE USER REGS
G2
H1 FEOVD = YES *A2
NC
YES
H3 IJGUG330 EU READ COUNT FIELD
H4 GET ADDR CF USER ERROR ROUTINE
J1 WAS END OF VOLUME FORCED
YES
NO
1
ET B3
IJGUG310
J3 UPDATE SEEK ADDR WITH COUNT FIELD ID
J4 ERREXT SPECIFIED *A2
YES
NO
J5 PUT ADDR OF I/O AREA IN PARAMETER LIST
IJGUG040 K1 IJGUG250 EU READ RECORD
K3 ERROPT = SKIP
YES
NO
1
IJGUG060
ET B1
K4 GET ADDR OF I/O ARIA
K5 GET ADDR OF PARAMETER LIST
LR B1
2

## Chart ER. SDMODUI: GET Macro (Section 2 of 5)


EQ K4
B1 BRANCH TO USER ERROR ROUTINE
*B2 SDMODUI MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
C1 ERREXT SPECIFIED *B2
YES
NO
C2 SKIP OPTION
YES
NO
C4 LOAD MOD. REGS. 1-14 SAVE USER REGS 9-13
D1 LOAD MOD. REGS. 1-14 SAVE USER REGS 9-13
D2 IGNORE OPTION
YES
NO
D3 LOAD MOD. REGS. 1-14 SAVE USER REGS 9-13
D4 IJGUG330 EJ READ COUNT FIELD
E1 IJGUG330 EJ READ COUNT FIELD
E2 LOAD MOD. REGS AND SAVE USER REGS.
E3 IJGUG330 EJ READ COUNT FIELD
E4 UPDATE SEEK ADDRESS
EQ K1
IJGUG040
F1 UPDATE SEEK ADDRESS
F2 SVC 0 RETRY READ
F3 UPDATE SEEK ADDRESS
EQ K1
IJGUG040
EQ A3
IJGUG050
ET B1
IJGUG060

## Chart ES. SDMODUI: GET Macro (Section 3 of 5)


EQ F4
IJGUGUSR
B1 SAVE MOD REGS 1-15 LOAD USER REGS 8-13
*C2 SDMODUI MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
C1 LOAD ADDR OF USER ERROR ROUTINE
D1 ERREXT SPECIFIED *C2
YES
NO
D2 MOVE I/O ADDRESS TO LIST
E1 LOAD I/O AREA ADDR INTO REG 1
E2 SET I/O AREA TO ALL ZEROS
F2 SET FIRST DIGIT OF I/O AREA ADDR TO ZERO
G1 BRANCH TO USER ERROR ROUTINE
*G2 CORRECT RETURN ADDRESS IS DETERMINED BY USER ROUTINE.
H1 ERREXT SPECIFIED *C2
YES
NO
H2 LOAD RETURN ADDRESS INTO REG 14 *G2
J1 LOAD MOD REGS 1-15 SAVE USER REGS 8-13
2
J2 LOAD MOD REGS 1-15 SAVE USER REGS 8-13
1
1
B3 SAVE SAVE AREA POINTER
C3 ERET =SKIP
YES
NO
C4 IJGUG330 EU READ COUNT FIELD
D4 UPDATE SEEK ADDRESS
EQ J1
IJGUG040
F3 ERET OPTION
IGNORE
RETRY
F4 IJGUG330 EU READ COUNT FIELD
G3 SVC 0 RETRY READ
EQ A3
IJGUG050
G4 UPDATE SEEK ADDRESS
ET B1
IJGUG060
2
B5 SAVE SAVE AREA POINTER IN REG 8
C5 IJGUG330 EU READ COUNT FIELD
D5 UPDATE SEEK ADDRESS
EQ J1
IJGUG040

## Chart ET. SDMODUI: GET Macro (Section 4 of 5)

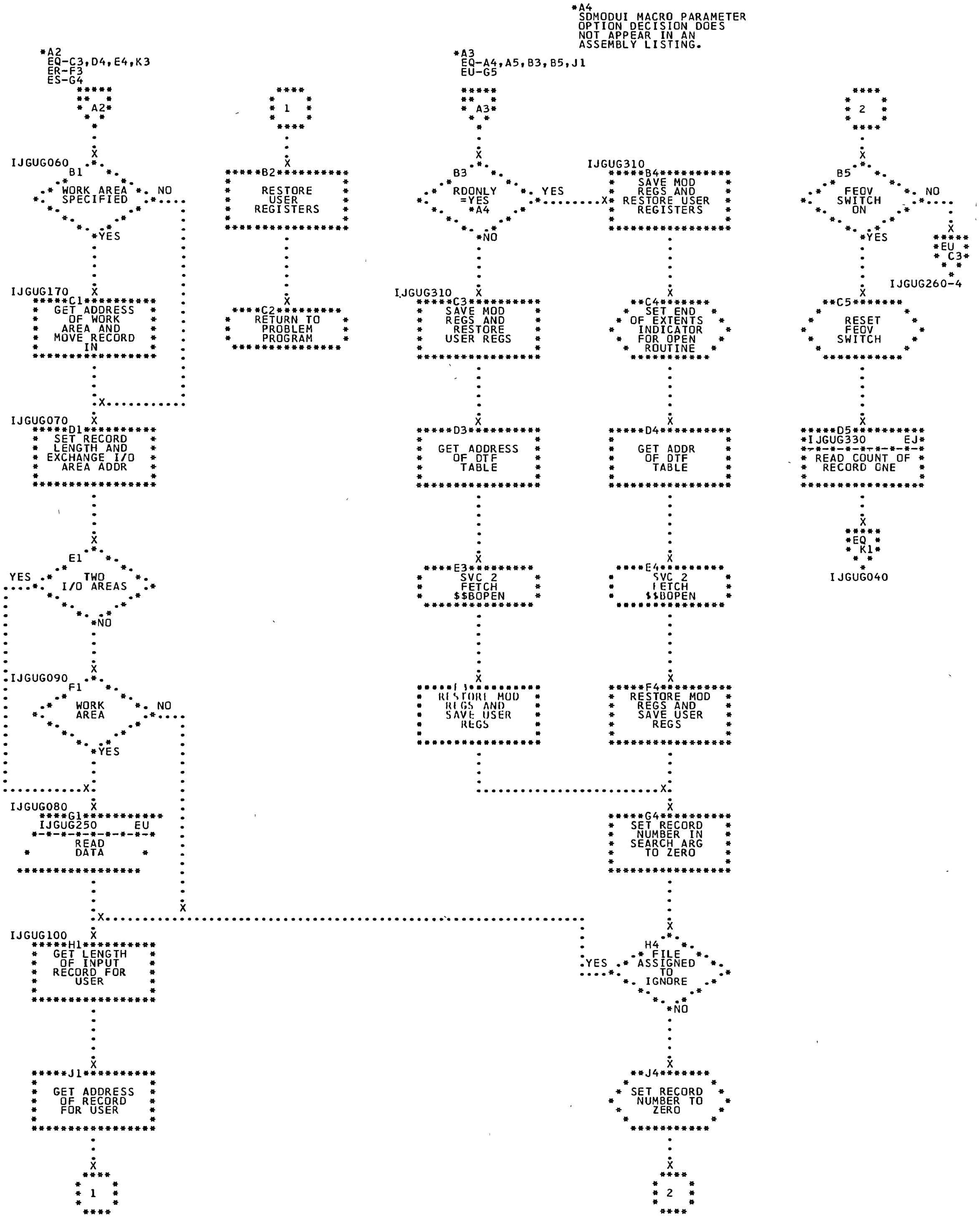
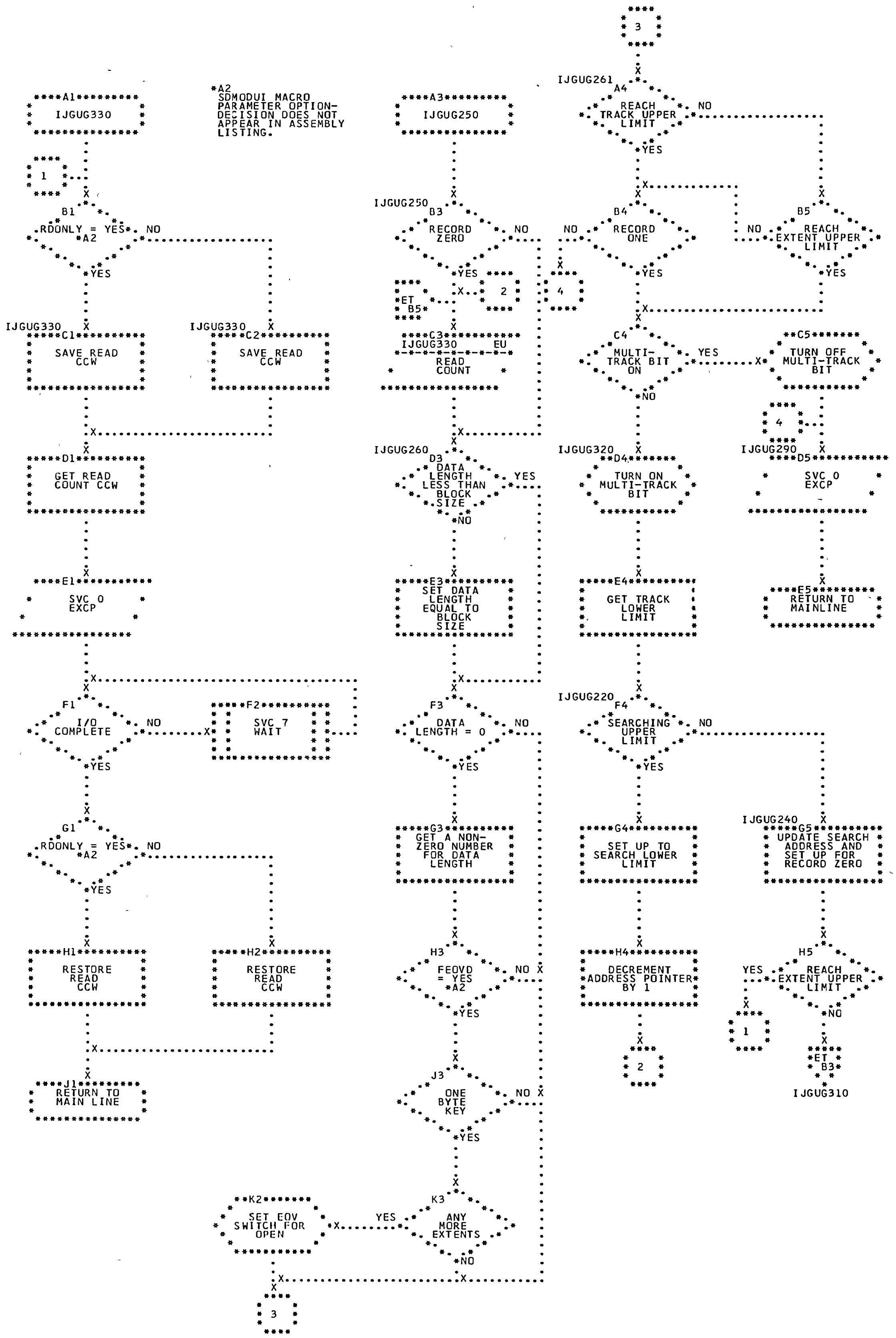

*A4
SDMODUI MACRO PARAMETER OPTION DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
*A2
EQ-C3,D4,E4,K3
ER-F3
ES-G4
A2
1
*A3
EQ-A4,A5,B3,B5,J1
EU-G5
A3
2
IJGUG060
B1
WORK AREA SPECIFIED
NO
YES
B2
RESTORE USER REGISTERS
B3
RDONLY =YES *A4
YES
NO
IJGUG310
B4
SAVE MOD REGS AND RESTORE USER REGISTERS
B5
FEOV SWITCH ON
NO
YES
EU
C3
IJGUG260-4
IJGUG170
C1
GET ADDRESS OF WORK AREA AND MOVE RECORD IN
C2
RETURN TO PROBLEM PROGRAM
IJGUG310
C3
SAVE MOD REGS AND RESTORE USER REGS
C4
SET END OF EXTENTS INDICATOR FOR OPEN ROUTINE
C5
RESET FEOV SWITCH
IJGUG070
D1
SET RECORD LENGTH AND EXCHANGE I/O AREA ADDR
D3
GET ADDRESS OF DTF TABLE
D4
GET ADDR OF DTF TABLE
D5
IJGUG330 EJ
READ COUNT OF RECORD ONE
EQ
K1
IJGUG040
E1
YES
TWO I/O AREAS
NO
E3
SVC 2 FETCH $$BOPEN
E4
SVC 2 FETCH $$BOPEN
IJGUG090
F1
WORK AREA
NO
YES
F3
RESTORE MOD REGS AND SAVE USER REGS
F4
RESTORE MOD REGS AND SAVE USER REGS
IJGUG080
G1
IJGUG250 EU
READ DATA
G4
SET RECORD NUMBER IN SEARCH ARG TO ZERO
IJGUG100
H1
GET LENGTH OF INPUT RECORD FOR USER
H4
YES
FILE ASSIGNED TO IGNORE
NO
J1
GET ADDRESS OF RECORD FOR USER
J4
SET RECORD NUMBER TO ZERO
1
2

Chart EU. SDMODUI: GET Macro (Section 5 of 5)



A1 IJGUG330
A2 SDMODUI MACRO PARAMETER OPTION-DECISION DOES NOT APPEAR IN ASSEMBLY LISTING.
1
B1 RDONLY = YES* A2
NO
YES
IJGUG330 C1 SAVE READ CCW
IJGUG330 C2 SAVE READ CCW
D1 GET READ COUNT CCW
E1 SVC 0 EXCP
F1 I/O COMPLETE
NO
YES
F2 SVC 7 WAIT
G1 RDONLY = YES* A2
NO
YES
H1 RESTORE READ CCW
H2 RESTORE READ CCW
J1 RETURN TO MAIN LINE
A3 IJGUG250
IJGUG250 B3 RECORD ZERO
NO
YES
2
ET B5
C3 IJGUG330 EU READ COUNT
IJGUG260 D3 DATA LENGTH LESS THAN BLOCK SIZE
YES
NO
E3 SET DATA LENGTH EQUAL TO BLOCK SIZE
F3 DATA LENGTH = 0
NO
YES
G3 GET A NON-ZERO NUMBER FOR DATA LENGTH
H3 FEOVD = YES *A2
NO
YES
J3 ONE BYTE KEY
NO
YES
K3 ANY MORE EXTENTS
YES
NO
K2 SET EOV SWITCH FOR OPEN
3
3
IJGUG261 A4 REACH TRACK UPPER LIMIT
NO
YES
B4 RECORD ONE
NO
YES
4
B5 REACH EXTENT UPPER LIMIT
NO
YES
C4 MULTI-TRACK BIT ON
YES
NO
C5 TURN OFF MULTI-TRACK BIT
4
IJGUG320 D4 TURN ON MULTI-TRACK BIT
IJGUG290 D5 SVC 0 EXCP
E4 GET TRACK LOWER LIMIT
E5 RETURN TO MAINLINE
IJGUG220 F4 SEARCHING UPPER LIMIT
NO
YES
G4 SET UP TO SEARCH LOWER LIMIT
IJGUG240 G5 UPDATE SEARCH ADDRESS AND SET UP FOR RECORD ZERO
H4 DECREMENT ADDRESS POINTER BY 1
2
H5 REACH EXTENT UPPER LIMIT
YES
NO
1
ET B3
IJGUG310

## Chart EV. SDMODUO: PUT Macro (Section 1 of 5)

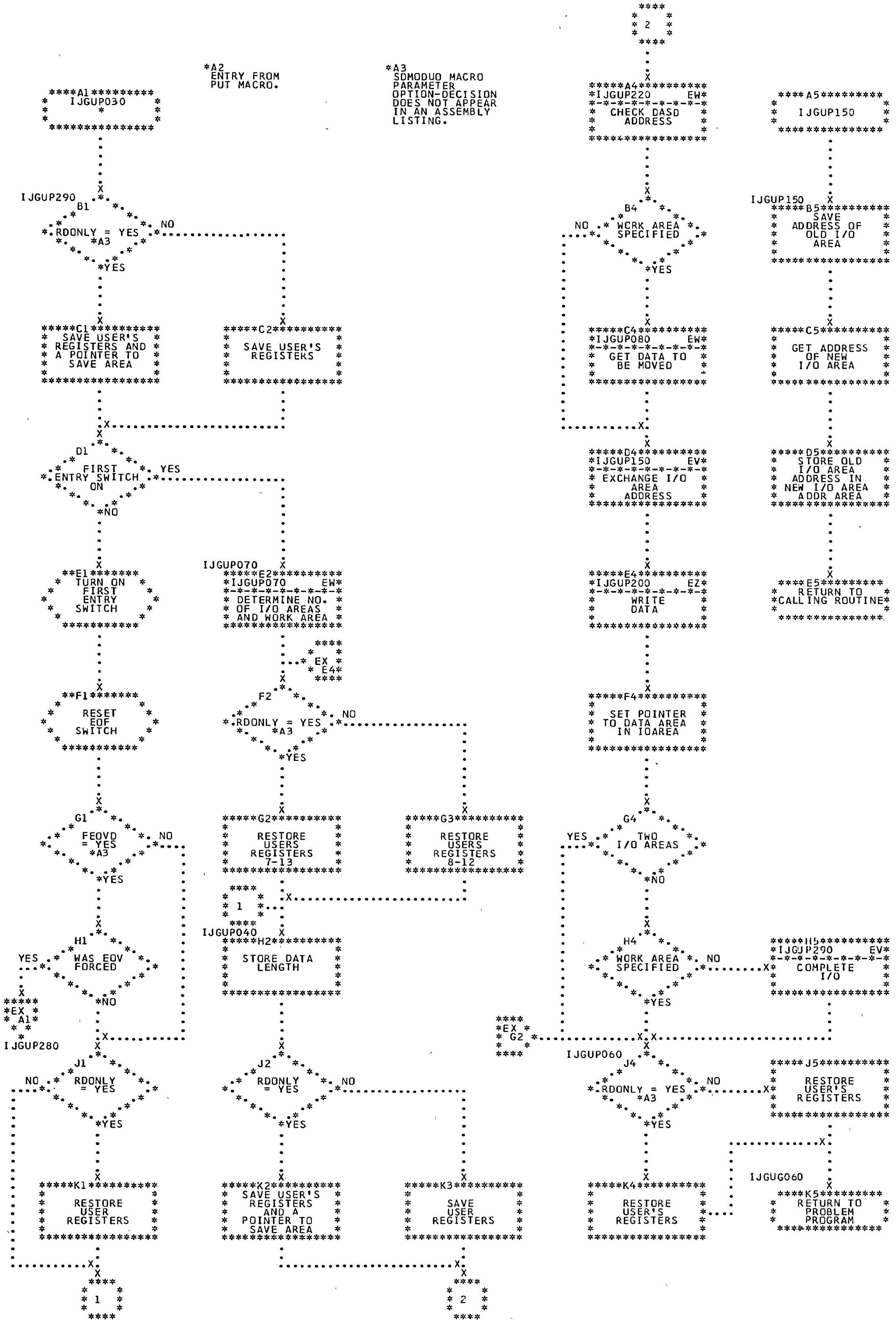
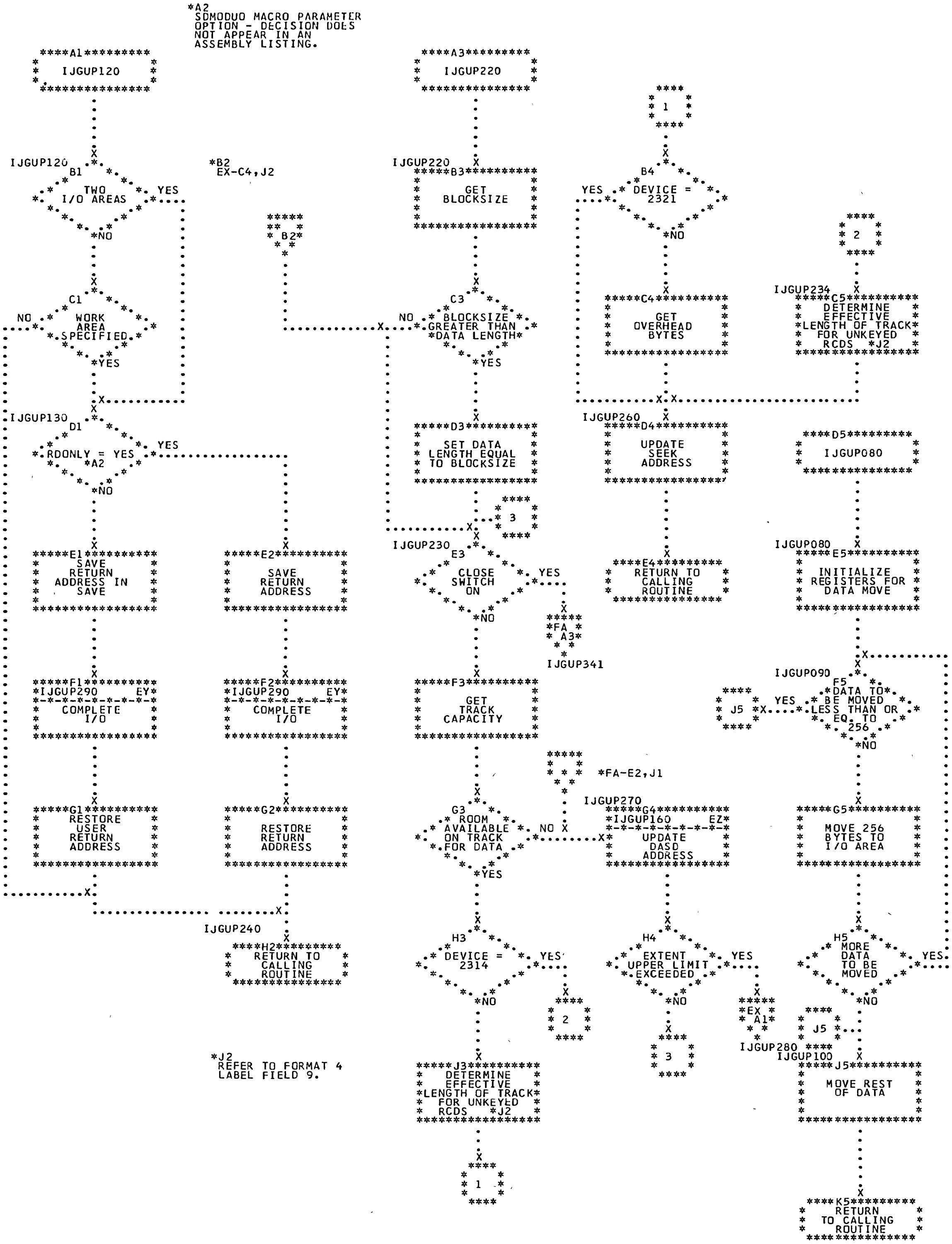

2
A1 IJGUP030
A2 ENTRY FROM PUT MACRO.
A3 SDMODUO MACRO PARAMETER OPTION-DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
A4 IJGUP220 EW CHECK DASD ADDRESS
A5 IJGUP150
IJGUP290 B1 RDONLY = YES A3
NO
YES
B4 WORK AREA SPECIFIED
NO
YES
IJGUP150 B5 SAVE ADDRESS OF OLD I/O AREA
C1 SAVE USER'S REGISTERS AND A POINTER TO SAVE AREA
C2 SAVE USER'S REGISTERS
C4 IJGUP080 EW GET DATA TO BE MOVED
C5 GET ADDRESS OF NEW I/O AREA
D1 FIRST ENTRY SWITCH ON
YES
NO
D4 IJGUP150 EV EXCHANGE I/O AREA ADDRESS
D5 STORE OLD I/O AREA ADDRESS IN NEW I/O AREA ADDR AREA
E1 TURN ON FIRST ENTRY SWITCH
IJGUP070 E2 IJGUP070 EW DETERMINE NO. OF I/O AREAS AND WORK AREA
EX E4
E4 IJGUP200 EZ WRITE DATA
E5 RETURN TO CALLING ROUTINE
F1 RESET EOF SWITCH
F2 RDONLY = YES A3
NO
YES
F4 SET POINTER TO DATA AREA IN IOAREA
G1 FEOVD = YES A3
NO
YES
G2 RESTORE USERS REGISTERS 7-13
G3 RESTORE USERS REGISTERS 8-12
G4 TWO I/O AREAS
YES
NO
1
IJGUP040 H2 STORE DATA LENGTH
H1 WAS EOV FORCED
YES
NO
EX A1
H4 WORK AREA SPECIFIED
NO
YES
H5 IJGUP290 EV COMPLETE I/O
EX G2
IJGUP280 J1 RDONLY = YES
NO
YES
J2 RDONLY = YES
NO
YES
IJGUP060 J4 RDONLY = YES A3
NO
YES
J5 RESTORE USER'S REGISTERS
IJGUG060
K1 RESTORE USER REGISTERS
K2 SAVE USER'S REGISTERS AND A POINTER TO SAVE AREA
K3 SAVE USER REGISTERS
K4 RESTORE USER'S REGISTERS
K5 RETURN TO PROBLEM PROGRAM
1
2

Chart EW. SDMODUO: PUT Macro (Section 2 of 5)



# Chart EX. SDMODUO: PUT Macro (Section 3 of 5)

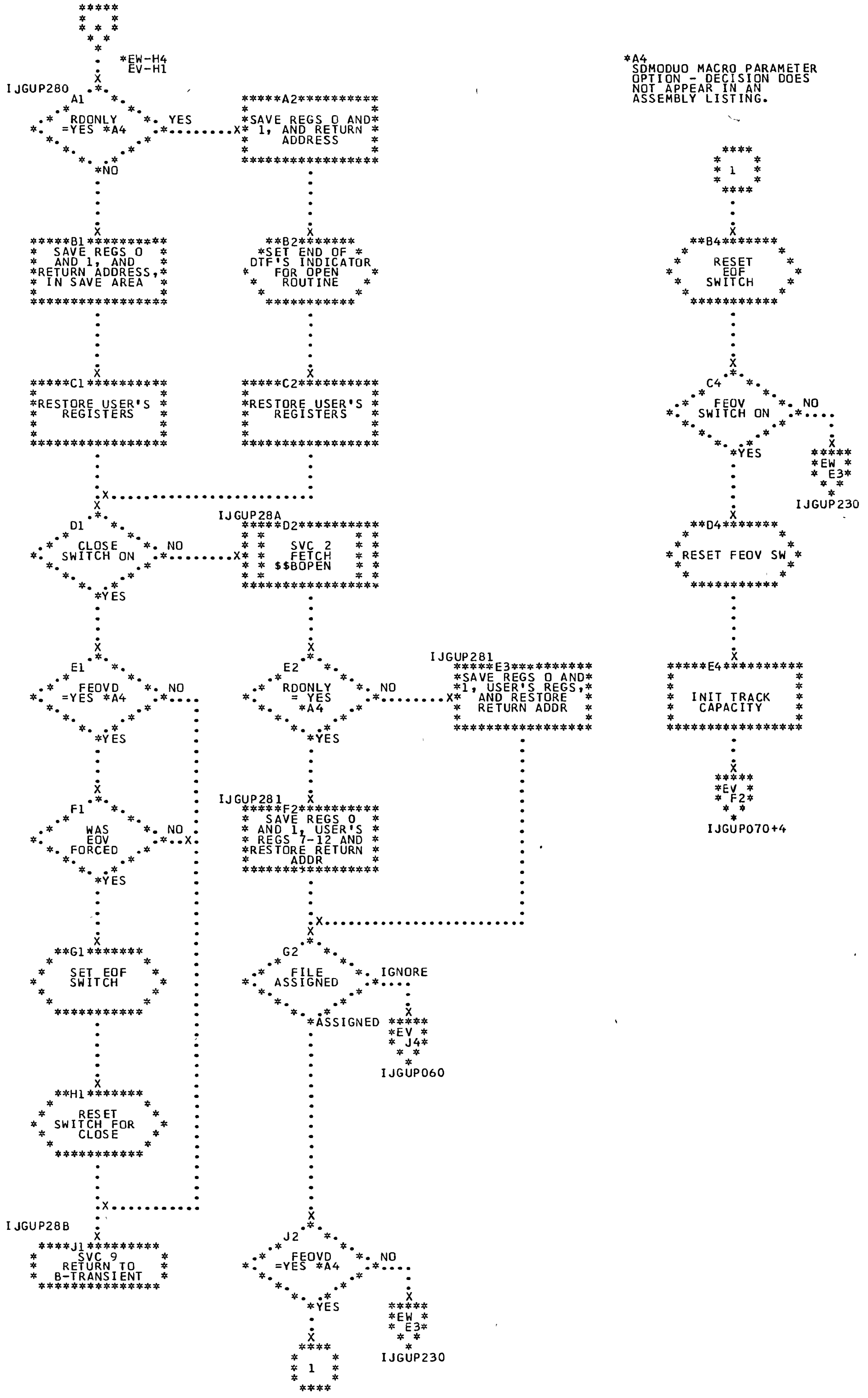


Chart EY. SDMODUO: PUT Macro (Section 4 of 5)

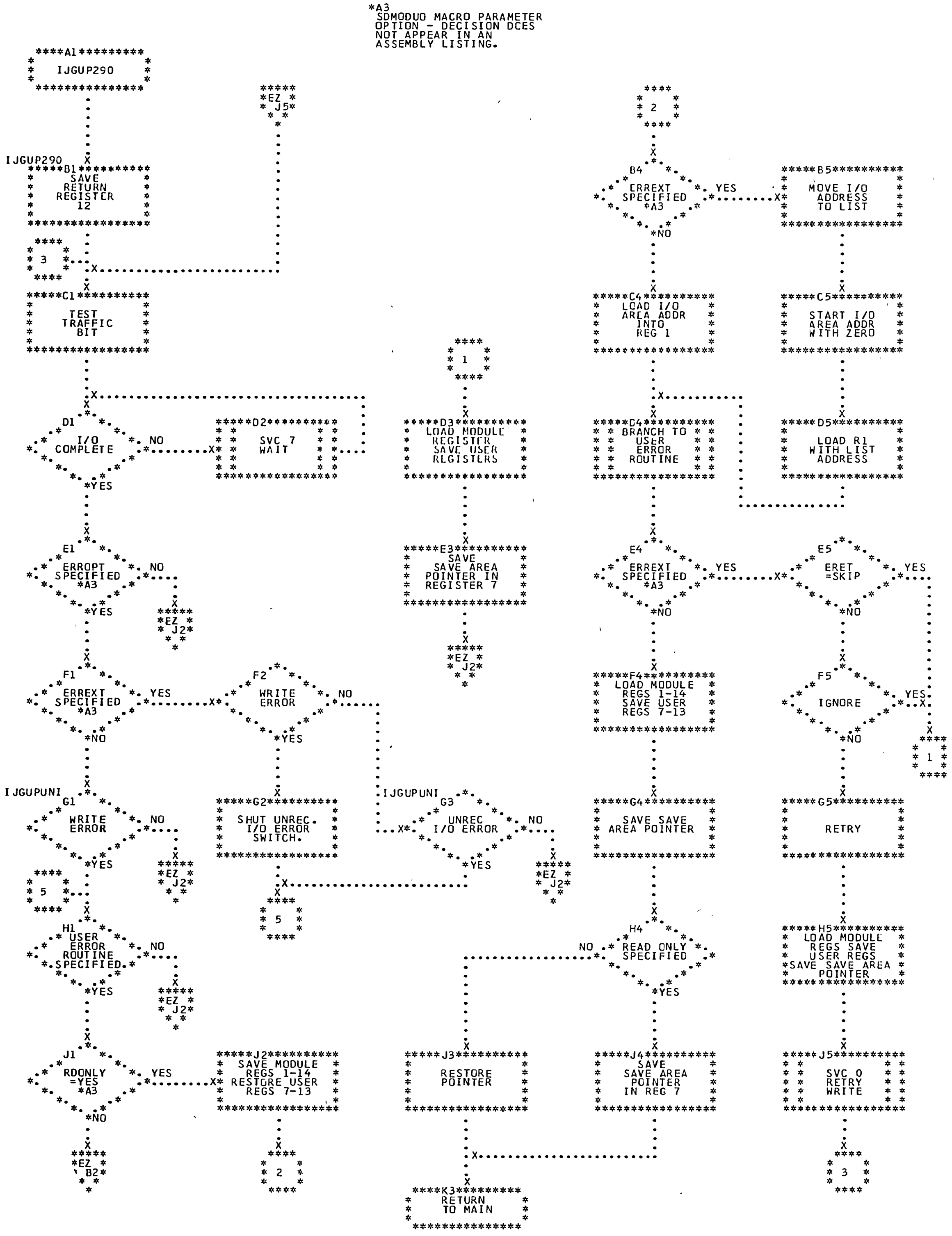
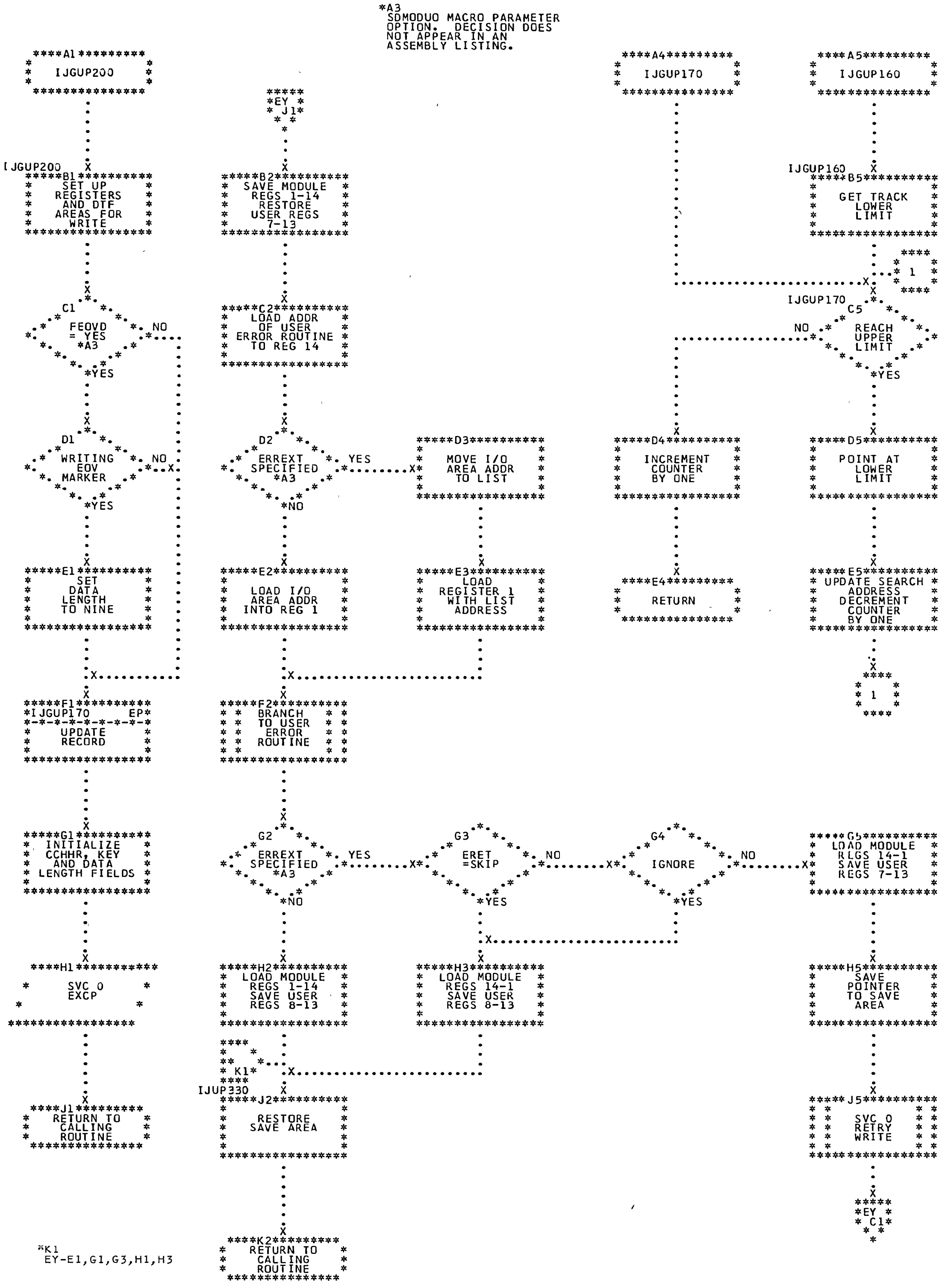


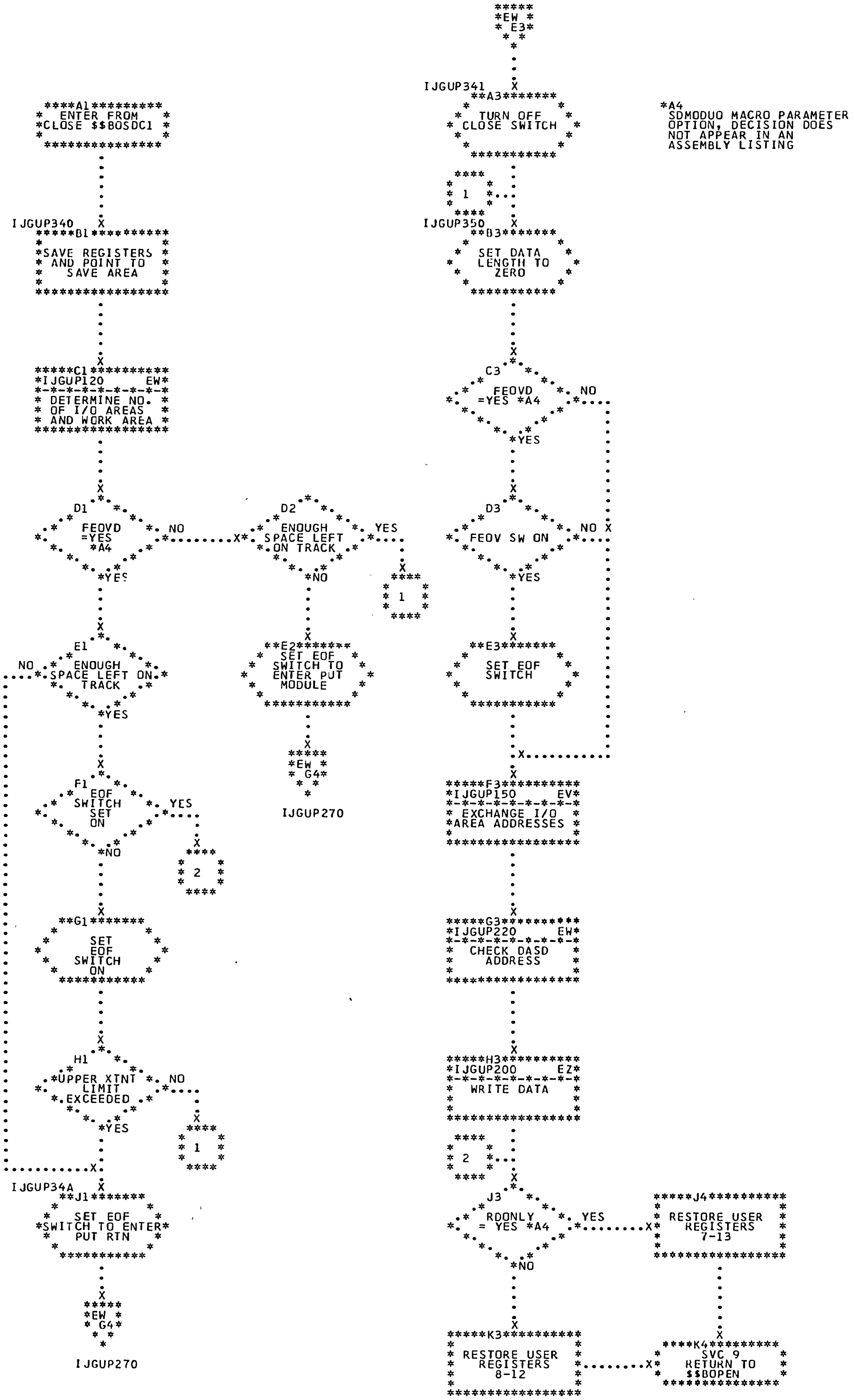
Chart EZ. SDMODUO: PUT Macro (Section 5 of 5)


*A3
SDMODUO MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.

*K1
EY-E1,G1,G3,H1,H3

A1 IJGUP200
IJGUP200
B1 SET UP REGISTERS AND DTF AREAS FOR WRITE
C1 FEOVD = YES *A3
NO
YES
D1 WRITING EOV MARKER
NO
YES
E1 SET DATA LENGTH TO NINE
F1 IJGUP170 EP UPDATE RECORD
G1 INITIALIZE CCHHR, KEY AND DATA LENGTH FIELDS
H1 SVC 0 EXCP
J1 RETURN TO CALLING ROUTINE
EY J1
B2 SAVE MODULE REGS 1-14 RESTORE USER REGS 7-13
C2 LOAD ADDR OF USER ERROR ROUTINE TO REG 14
D2 ERREXT SPECIFIED *A3
YES
NO
E2 LOAD I/O AREA ADDR INTO REG 1
F2 BRANCH TO USER ERROR ROUTINE
G2 ERREXT SPECIFIED *A3
YES
NO
H2 LOAD MODULE REGS 1-14 SAVE USER REGS 8-13
K1
IJUP330
J2 RESTORE SAVE AREA
K2 RETURN TO CALLING ROUTINE
D3 MOVE I/O AREA ADDR TO LIST
E3 LOAD REGISTER 1 WITH LIST ADDRESS
G3 ERET =SKIP
NO
YES
H3 LOAD MODULE REGS 14-1 SAVE USER REGS 8-13
A4 IJGUP170
D4 INCREMENT COUNTER BY ONE
E4 RETURN
G4 IGNORE
NO
YES
A5 IJGUP160
IJGUP160
B5 GET TRACK LOWER LIMIT
1
IJGUP170
C5 REACH UPPER LIMIT
NO
YES
D5 POINT AT LOWER LIMIT
E5 UPDATE SEARCH ADDRESS DECREMENT COUNTER BY ONE
1
G5 LOAD MODULE REGS 14-1 SAVE USER REGS 7-13
H5 SAVE POINTER TO SAVE AREA
J5 SVC 0 RETRY WRITE
EY C1

## Chart FA. SDMODUO: Close Routine


EW E3
IJGUP341
A3 TURN OFF CLOSE SWITCH
A4 SDMODUO MACRO PARAMETER OPTION, DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING
A1 ENTER FROM CLOSE $$BOSDC1
1
IJGUP340
B1 SAVE REGISTERS AND POINT TO SAVE AREA
IJGUP350
B3 SET DATA LENGTH TO ZERO
C1 IJGUP120 EW DETERMINE NO. OF I/O AREAS AND WORK AREA
C3 FEOVD =YES *A4
NO
YES
D1 FEOVD =YES *A4
NO
D2 ENOUGH SPACE LEFT ON TRACK
YES
NO
1
D3 FEOV SW ON
NO
YES
YES
E1 ENOUGH SPACE LEFT ON TRACK
NO
YES
E2 SET EOF SWITCH TO ENTER PUT MODULE
E3 SET EOF SWITCH
EW G4
IJGUP270
F1 EOF SWITCH SET ON
YES
NO
2
F3 IJGUP150 EV EXCHANGE I/O AREA ADDRESSES
G1 SET EOF SWITCH ON
G3 IJGUP220 EW CHECK DASD ADDRESS
H1 UPPER XTNT LIMIT EXCEEDED
NO
YES
1
H3 IJGUP200 EZ WRITE DATA
2
IJGUP34A
J1 SET EOF SWITCH TO ENTER PUT RTN
J3 RDONLY = YES *A4
YES
NO
J4 RESTORE USER REGISTERS 7-13
EW G4
IJGUP270
K3 RESTORE USER REGISTERS 8-12
K4 SVC 9 RETURN TO $$BOPEN

## Chart FB. SDMODUU: GET Macro (Section 1 of 2)

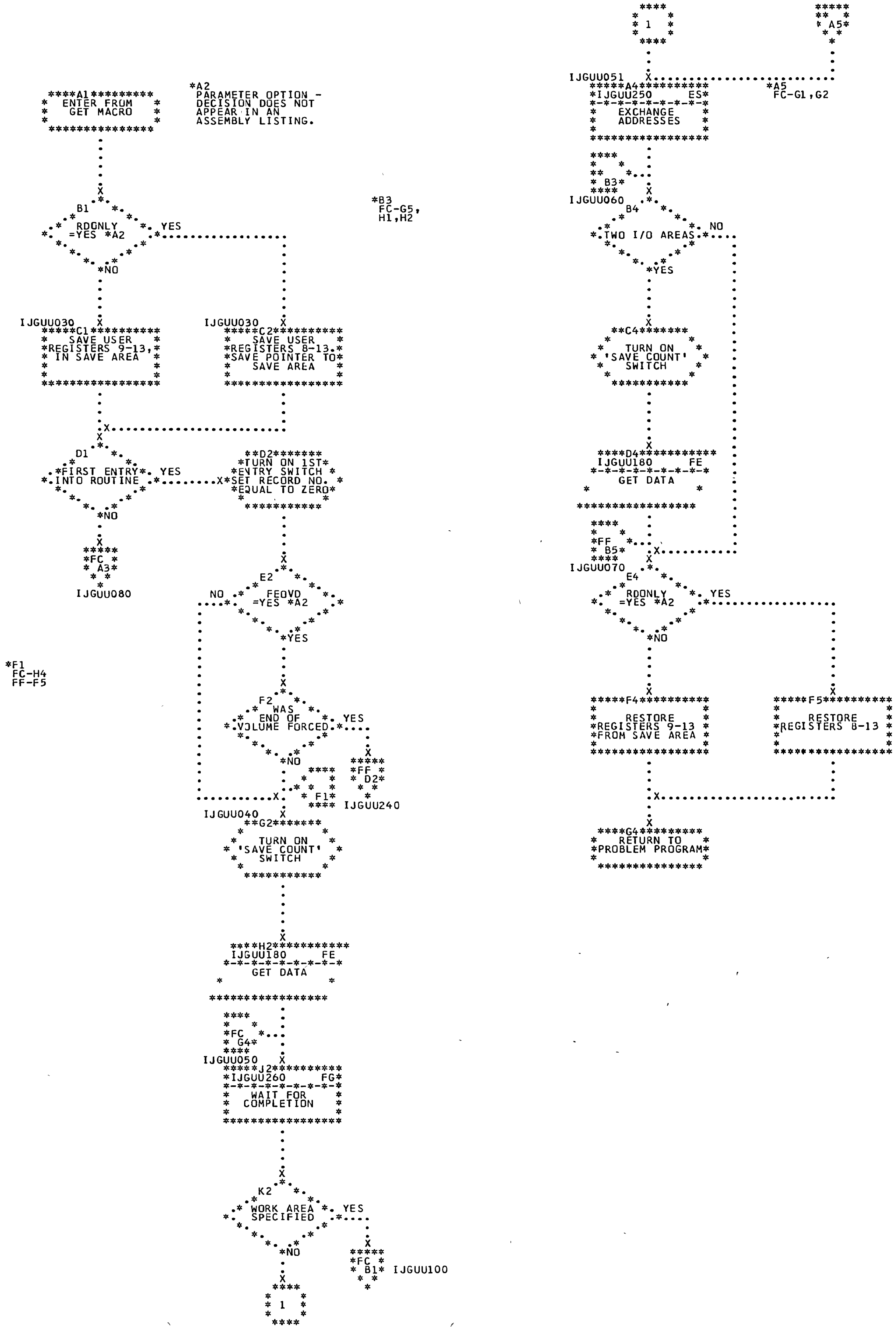
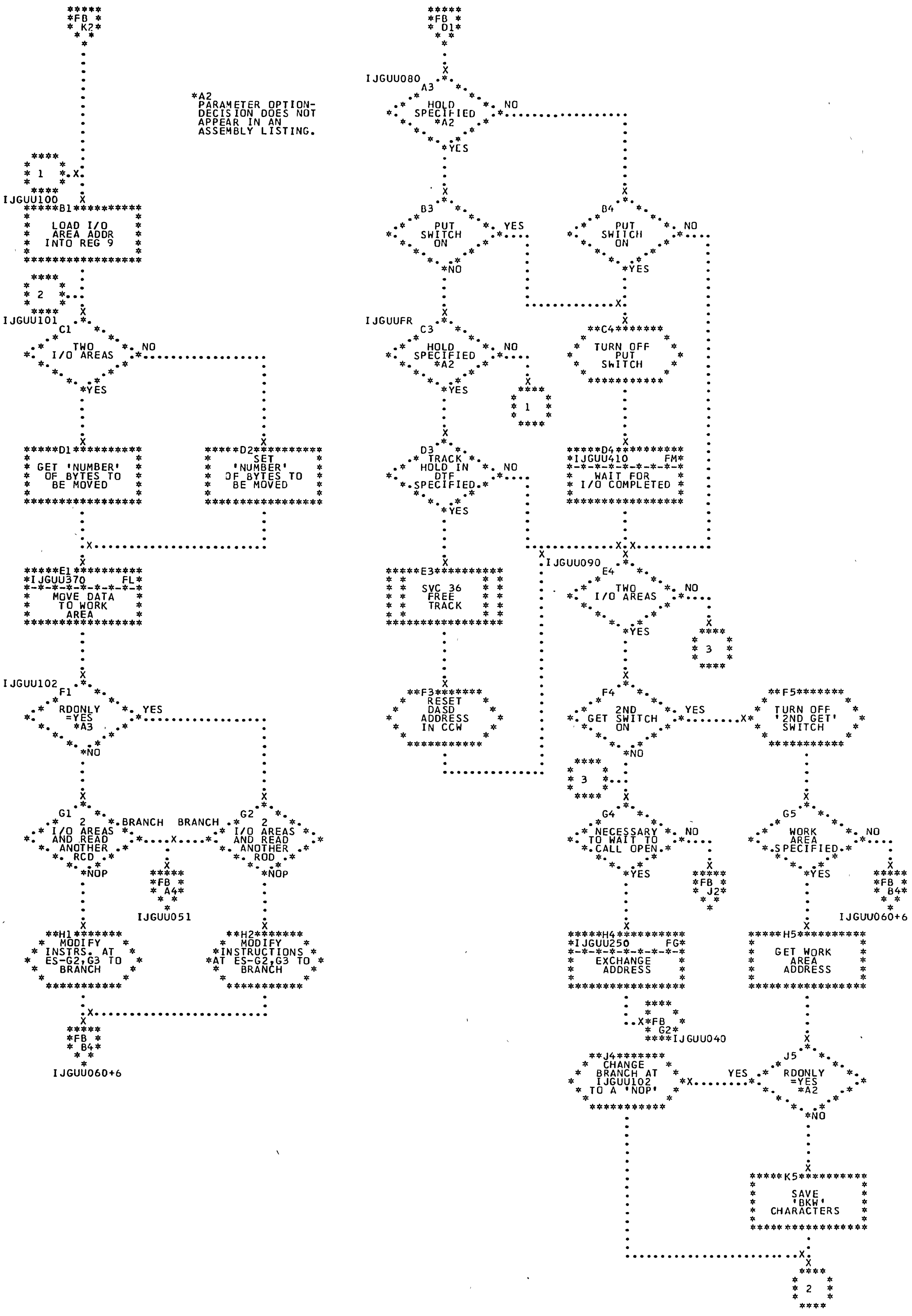
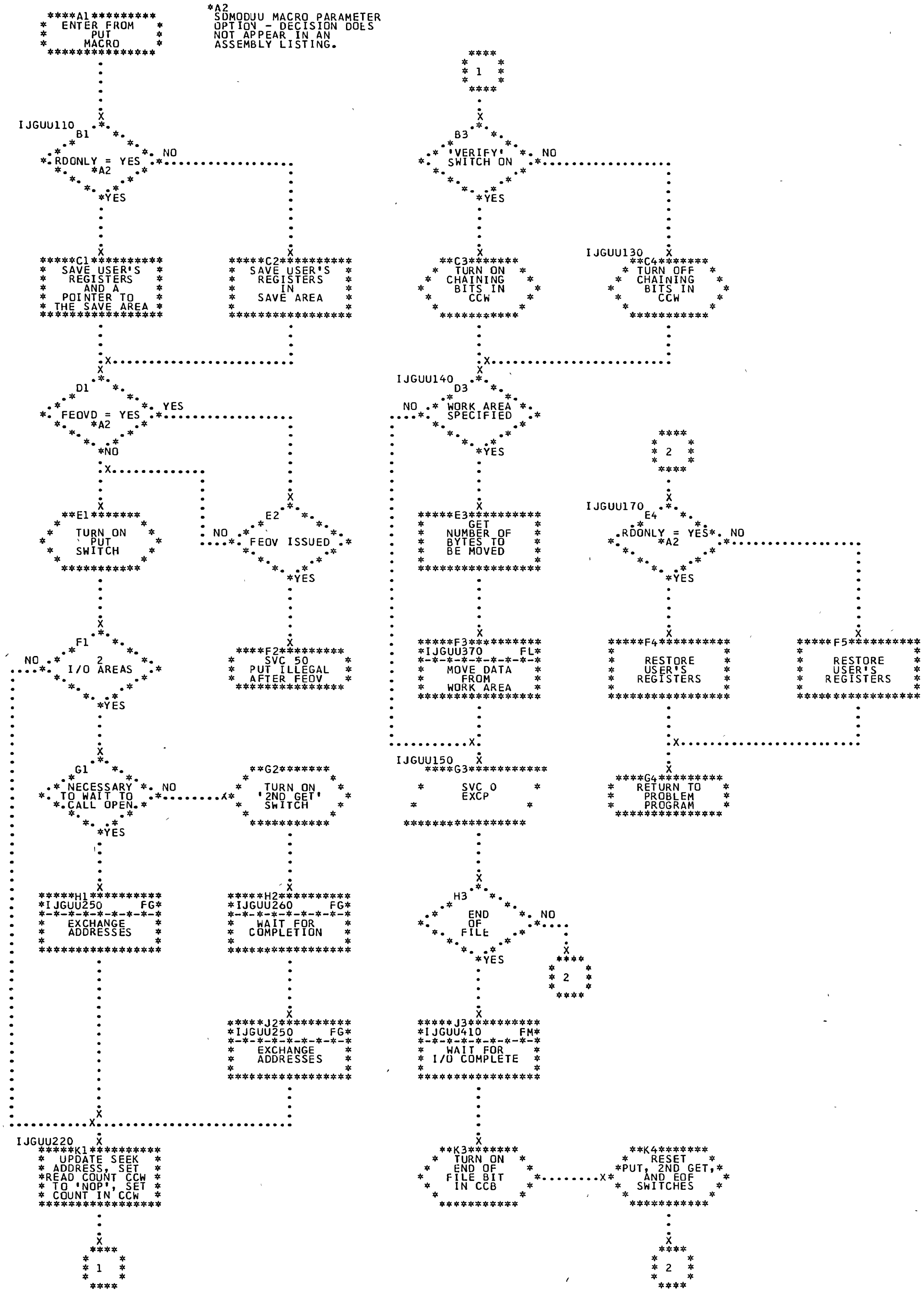

A1 ENTER FROM GET MACRO
*A2 PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
*B3 FC-G5, H1,H2
B1 RDONLY =YES *A2
YES
NO
IJGUU030 C1 SAVE USER REGISTERS 9-13, IN SAVE AREA
IJGUU030 C2 SAVE USER REGISTERS 8-13. SAVE POINTER TO SAVE AREA
D1 FIRST ENTRY INTO ROUTINE
YES
D2 TURN ON 1ST ENTRY SWITCH SET RECORD NO. EQUAL TO ZERO
NO
FC A3
IJGUU080
E2 FEOVD =YES *A2
NO
YES
*F1 FC-H4 FF-F5
F2 WAS END OF VOLUME FORCED
YES
NO
FF D2
IJGUU240
F1
IJGUU040 G2 TURN ON 'SAVE COUNT' SWITCH
H2 IJGUU180 FE GET DATA
FC G4
IJGUU050 J2 IJGUU260 FG WAIT FOR COMPLETION
K2 WORK AREA SPECIFIED
YES
NO
FC B1 IJGUU100
1
1
A5
IJGUU051 A4 IJGUU250 ES EXCHANGE ADDRESSES
*A5 FC-G1,G2
B3
IJGUU060 B4 TWO I/O AREAS
NO
YES
C4 TURN ON 'SAVE COUNT' SWITCH
D4 IJGUU180 FE GET DATA
FF B5
IJGUU070 E4 RDONLY =YES *A2
YES
NO
F4 RESTORE REGISTERS 9-13 FROM SAVE AREA
F5 RESTORE REGISTERS 8-13
G4 RETURN TO PROBLEM PROGRAM

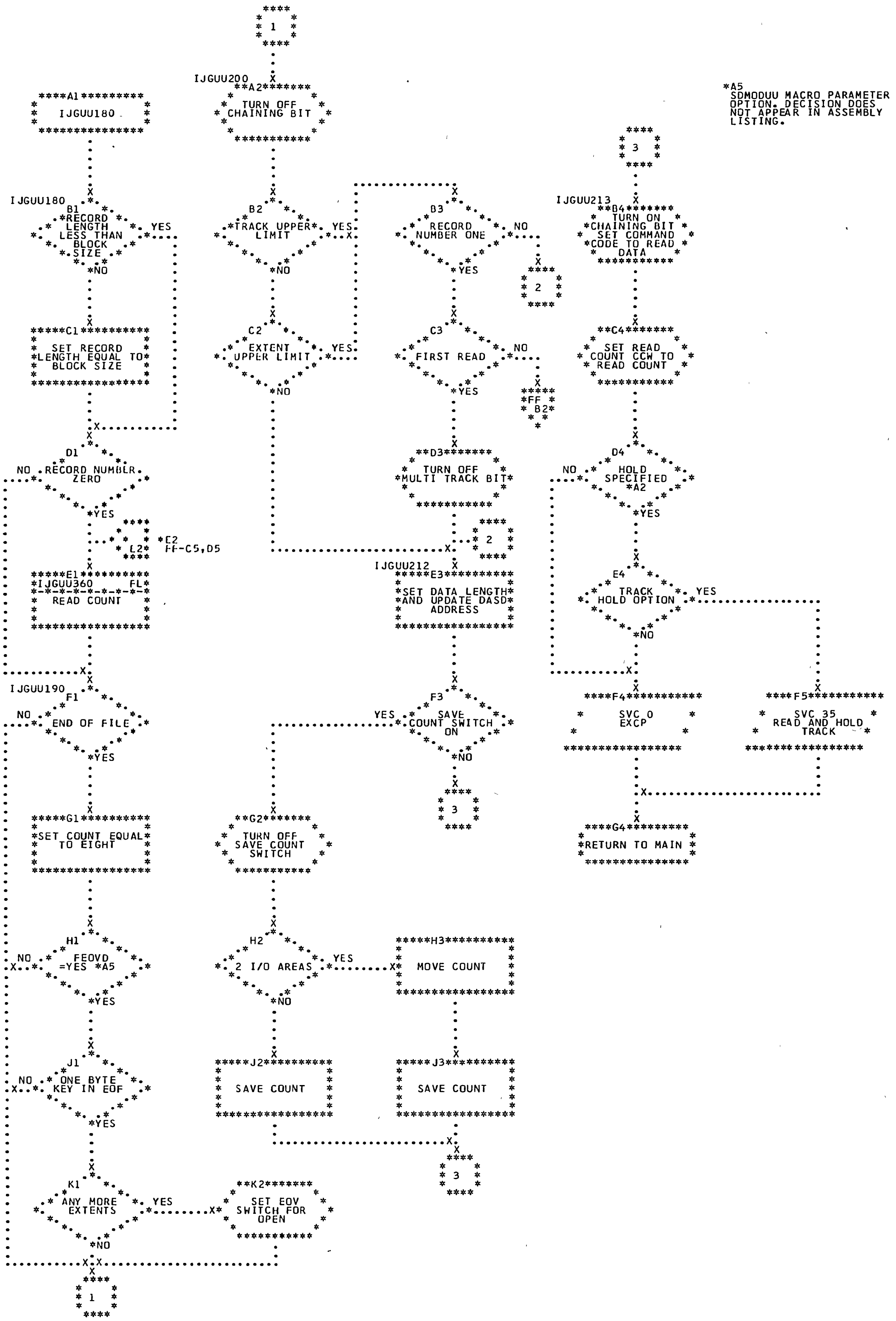
Chart FC. SDMODUU: GET Macro (Section 2 of 2)



## Chart FD. SDMODUU: PUT Macro



## Chart FE. SDMODUU: Subroutines (Section 1 of 8)



# Chart FF. SDMODUU: Subroutines (Section 2 of 8)

*A1
PARAMETER OPTION
DECISION DOES NOT
APPEAR IN AN
ASSEMBLY LISTING.

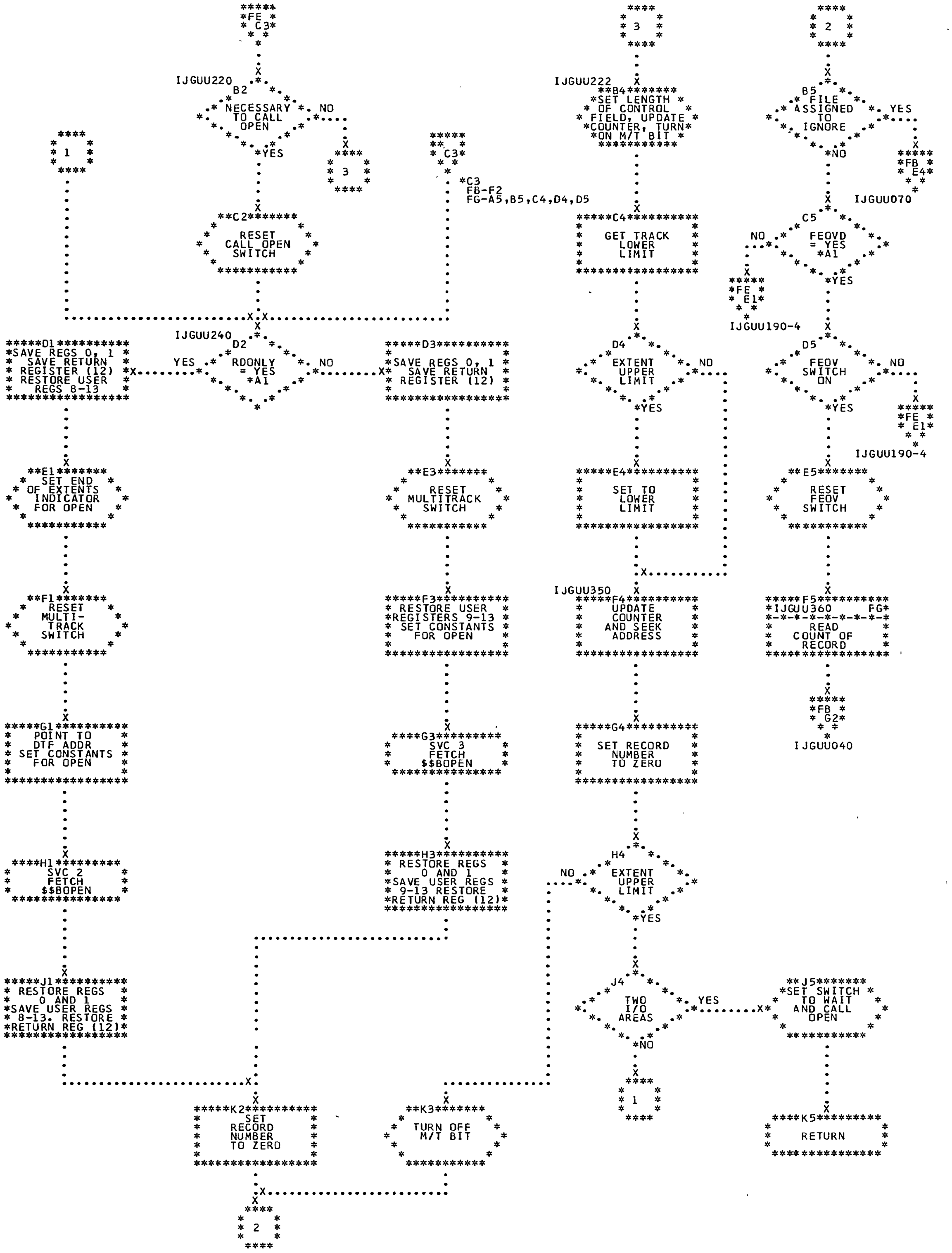

FE C3
IJGUU220
B2 NECESSARY TO CALL OPEN
NO
YES
3
1
C3
*C3 FB-F2 FG-A5,B5,C4,D4,D5
C2 RESET CALL OPEN SWITCH
3
IJGUU222
B4 SET LENGTH OF CONTROL FIELD, UPDATE COUNTER, TURN ON M/T BIT
C4 GET TRACK LOWER LIMIT
2
B5 FILE ASSIGNED TO IGNORE
YES
NO
FB E4
IJGUU070
C5 FEOVD = YES *A1
NO
YES
FE E1
IJGUU190-4
D1 SAVE REGS 0, 1 SAVE RETURN REGISTER (12) RESTORE USER REGS 8-13
IJGUU240
D2 RDONLY = YES *A1
YES
NO
D3 SAVE REGS 0, 1 SAVE RETURN REGISTER (12)
D4 EXTENT UPPER LIMIT
NO
YES
D5 FEOV SWITCH ON
NO
YES
FE E1
IJGUU190-4
E1 SET END OF EXTENTS INDICATOR FOR OPEN
E3 RESET MULTITRACK SWITCH
E4 SET TO LOWER LIMIT
E5 RESET FEOV SWITCH
F1 RESET MULTI- TRACK SWITCH
F3 RESTORE USER REGISTERS 9-13 SET CONSTANTS FOR OPEN
IJGUU350
F4 UPDATE COUNTER AND SEEK ADDRESS
F5 IJGUU360 FG READ COUNT OF RECORD
FB G2
IJGUU040
G1 POINT TO DTF ADDR SET CONSTANTS FOR OPEN
G3 SVC 3 FETCH $$BOPEN
G4 SET RECORD NUMBER TO ZERO
H1 SVC 2 FETCH $$BOPEN
H3 RESTORE REGS 0 AND 1 SAVE USER REGS 9-13 RESTORE RETURN REG (12)
H4 EXTENT UPPER LIMIT
NO
YES
J1 RESTORE REGS 0 AND 1 SAVE USER REGS 8-13. RESTORE RETURN REG (12)
J4 TWO I/O AREAS
YES
NO
J5 SET SWITCH TO WAIT AND CALL OPEN
1
K2 SET RECORD NUMBER TO ZERO
K3 TURN OFF M/T BIT
K5 RETURN
2

Chart FG. SDMODUU: Subroutines (Section 3 of 8)

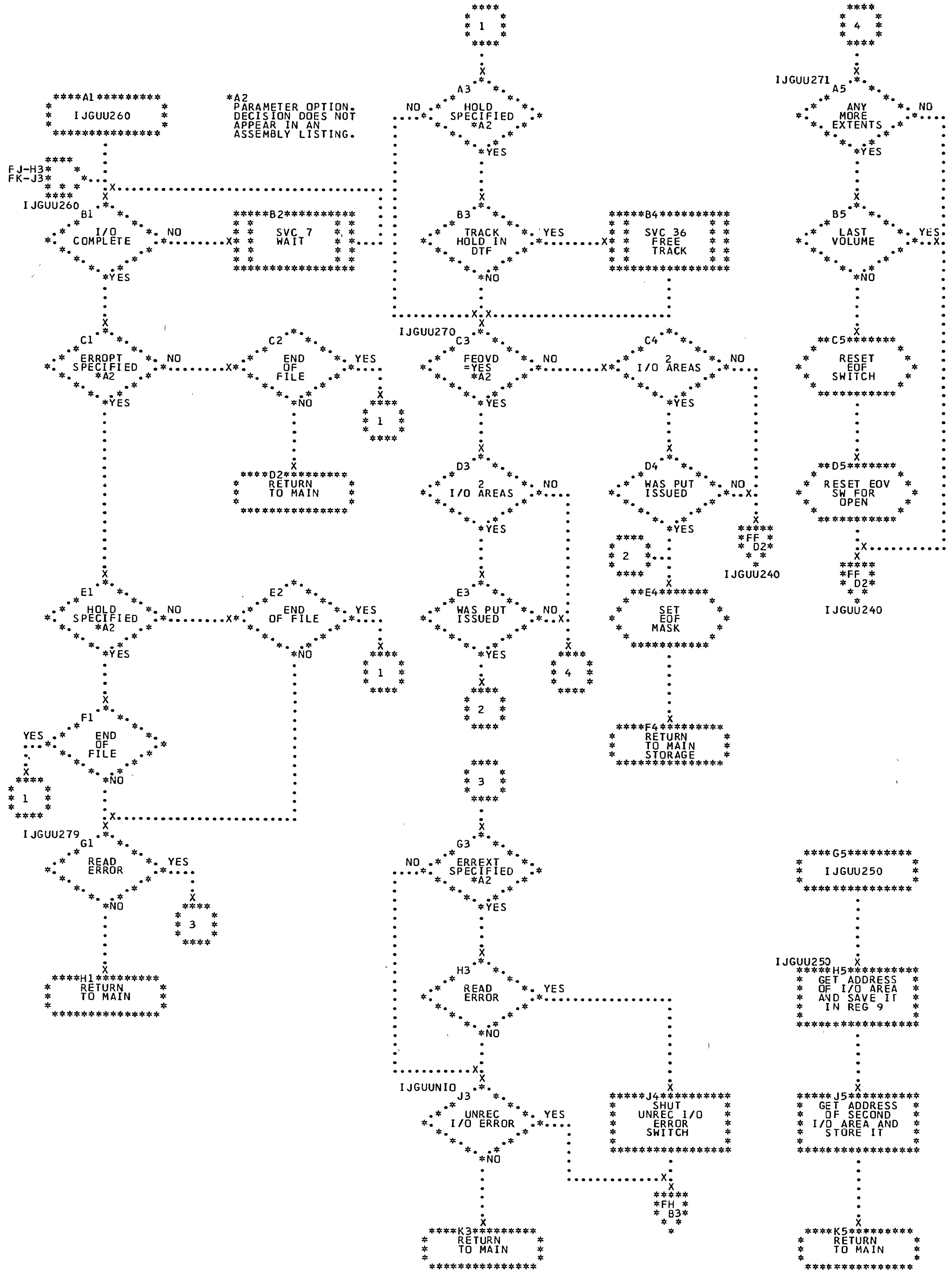


Chart FH. SDMODUU: Subroutines (Section 4 of 8)

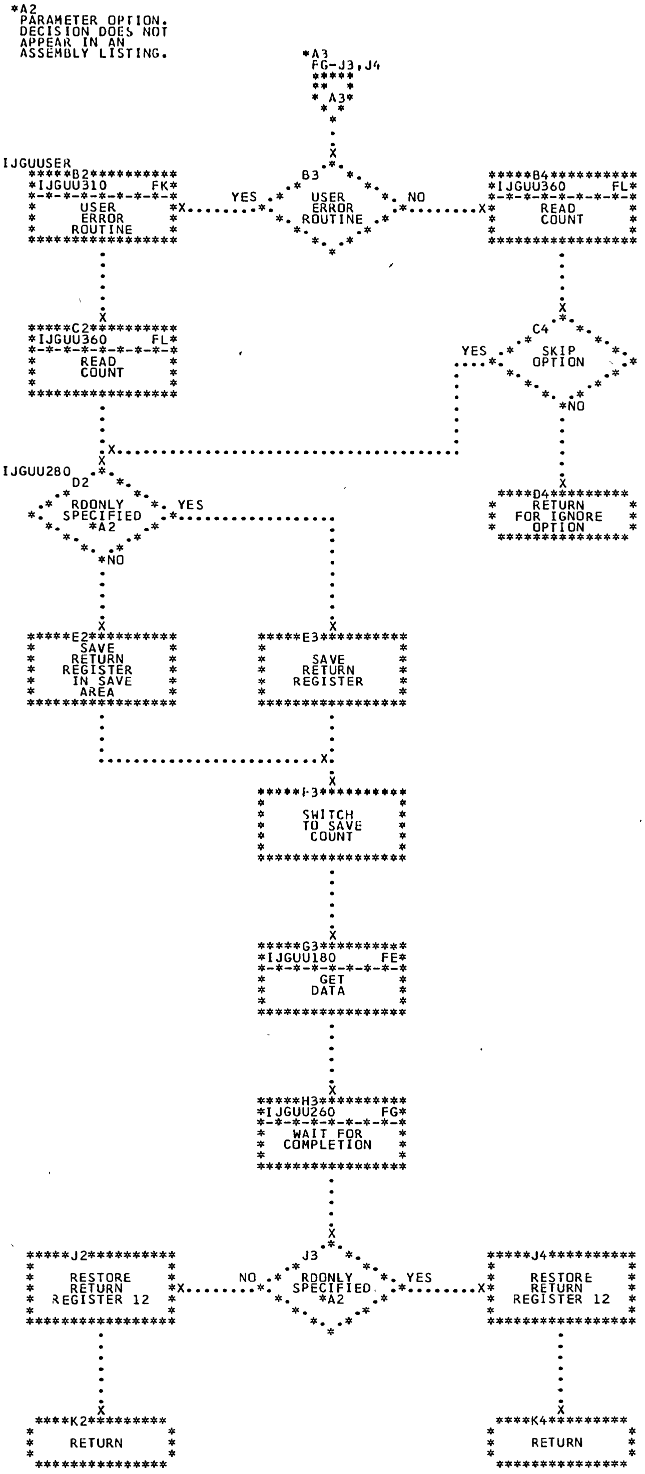
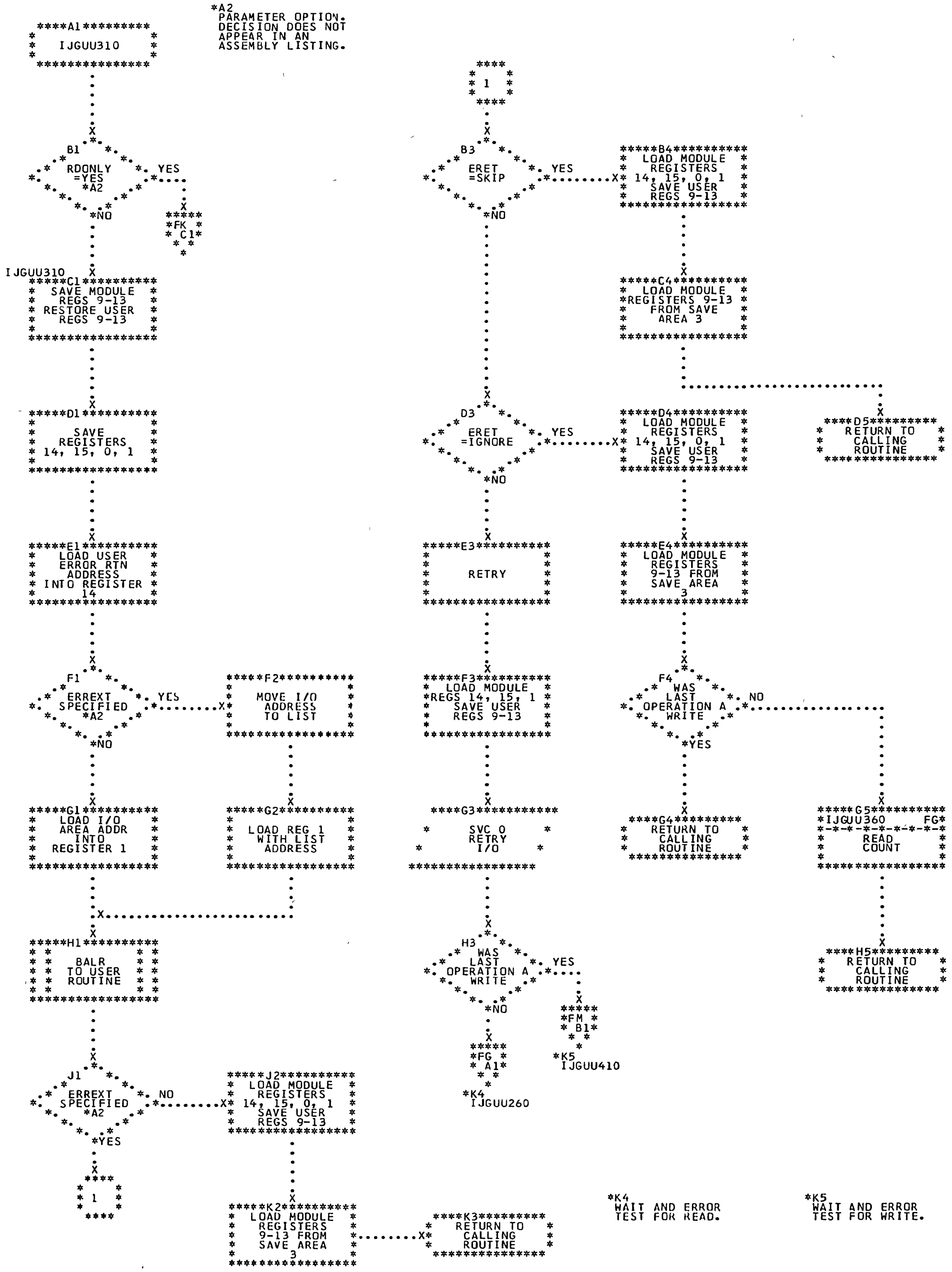


Chart FJ. SDMODUU: Subroutines (Section 5 of 8)

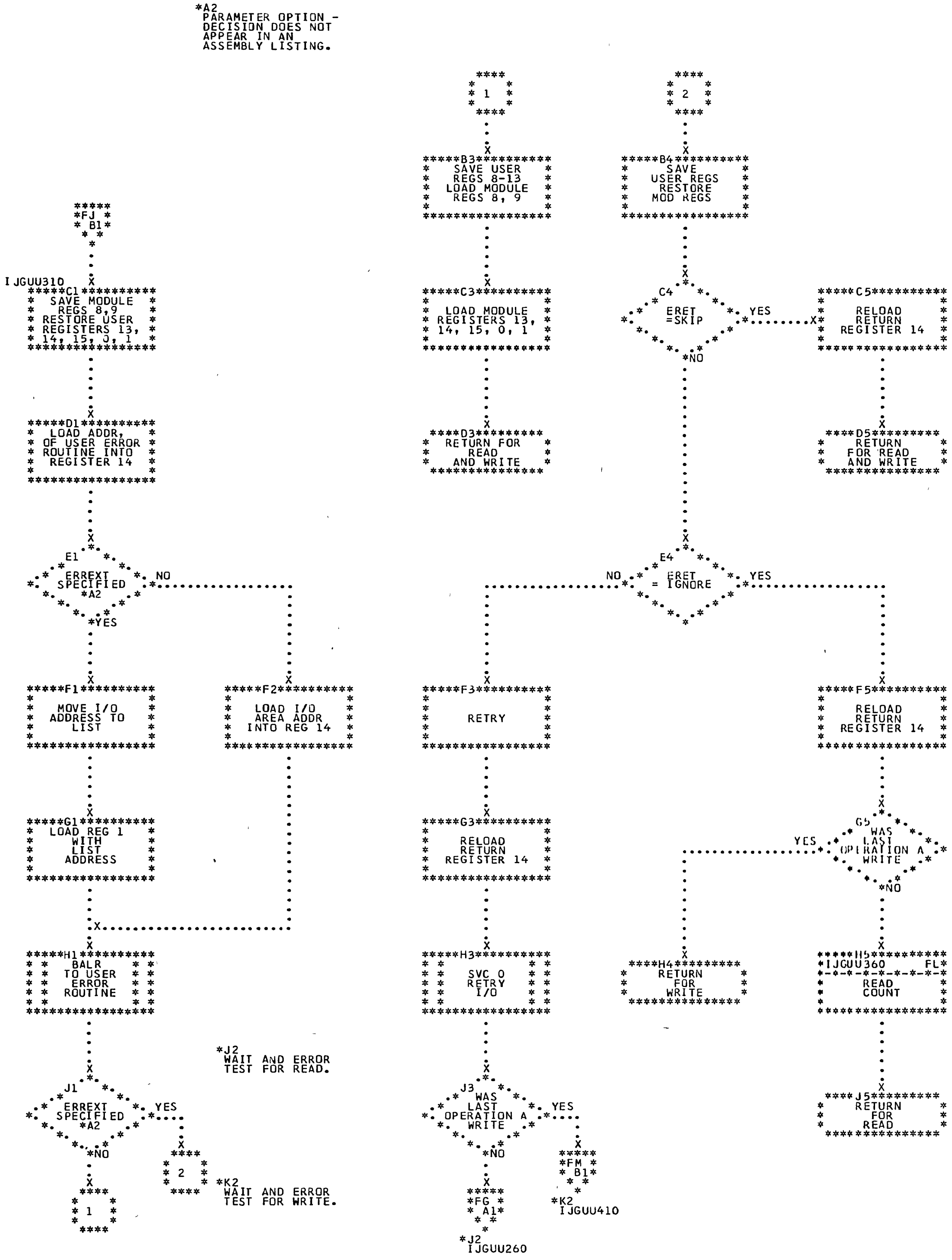

A1
IJGUU310
B1
RDONLY =YES *A2
YES
NO
FK C1
IJGUU310
C1
SAVE MODULE REGS 9-13 RESTORE USER REGS 9-13
D1
SAVE REGISTERS 14, 15, 0, 1
E1
LOAD USER ERROR RTN ADDRESS INTO REGISTER 14
F1
ERREXT SPECIFIED *A2
YES
NO
F2
MOVE I/O ADDRESS TO LIST
G1
LOAD I/O AREA ADDR INTO REGISTER 1
G2
LOAD REG 1 WITH LIST ADDRESS
H1
BALR TO USER ROUTINE
J1
ERREXT SPECIFIED *A2
NO
YES
J2
LOAD MODULE REGISTERS 14, 15, 0, 1 SAVE USER REGS 9-13
1
K2
LOAD MODULE REGISTERS 9-13 FROM SAVE AREA 3
K3
RETURN TO CALLING ROUTINE
1
B3
ERET =SKIP
YES
NO
B4
LOAD MODULE REGISTERS 14, 15, 0, 1 SAVE USER REGS 9-13
C4
LOAD MODULE REGISTERS 9-13 FROM SAVE AREA 3
D3
ERET =IGNORE
YES
NO
D4
LOAD MODULE REGISTERS 14, 15, 0, 1 SAVE USER REGS 9-13
D5
RETURN TO CALLING ROUTINE
E3
RETRY
E4
LOAD MODULE REGISTERS 9-13 FROM SAVE AREA 3
F3
LOAD MODULE REGS 14, 15, 1 SAVE USER REGS 9-13
F4
WAS LAST OPERATION A WRITE
NO
YES
G3
SVC 0 RETRY I/O
G4
RETURN TO CALLING ROUTINE
G5
IJGUU360 FG
READ COUNT
H3
WAS LAST OPERATION A WRITE
YES
NO
FM B1
*K5 IJGUU410
FG A1
*K4 IJGUU260
H5
RETURN TO CALLING ROUTINE

*A2
PARAMETER OPTION.
DECISION DOES NOT
APPEAR IN AN
ASSEMBLY LISTING.

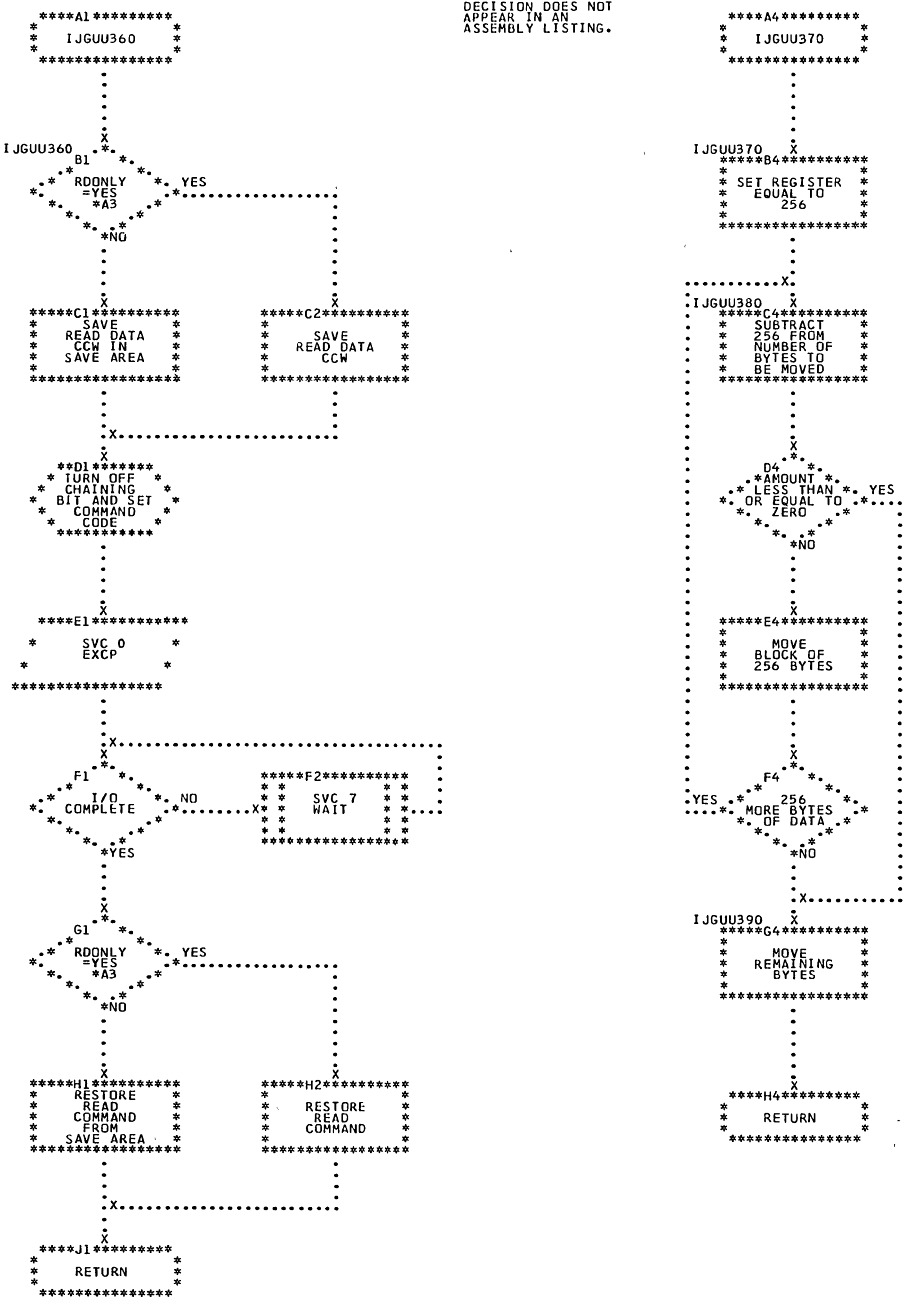
*K4
WAIT AND ERROR
TEST FOR READ.

*K5
WAIT AND ERROR
TEST FOR WRITE.

## Chart FK. SDMODUU: Subroutines (Section 6 of 8)

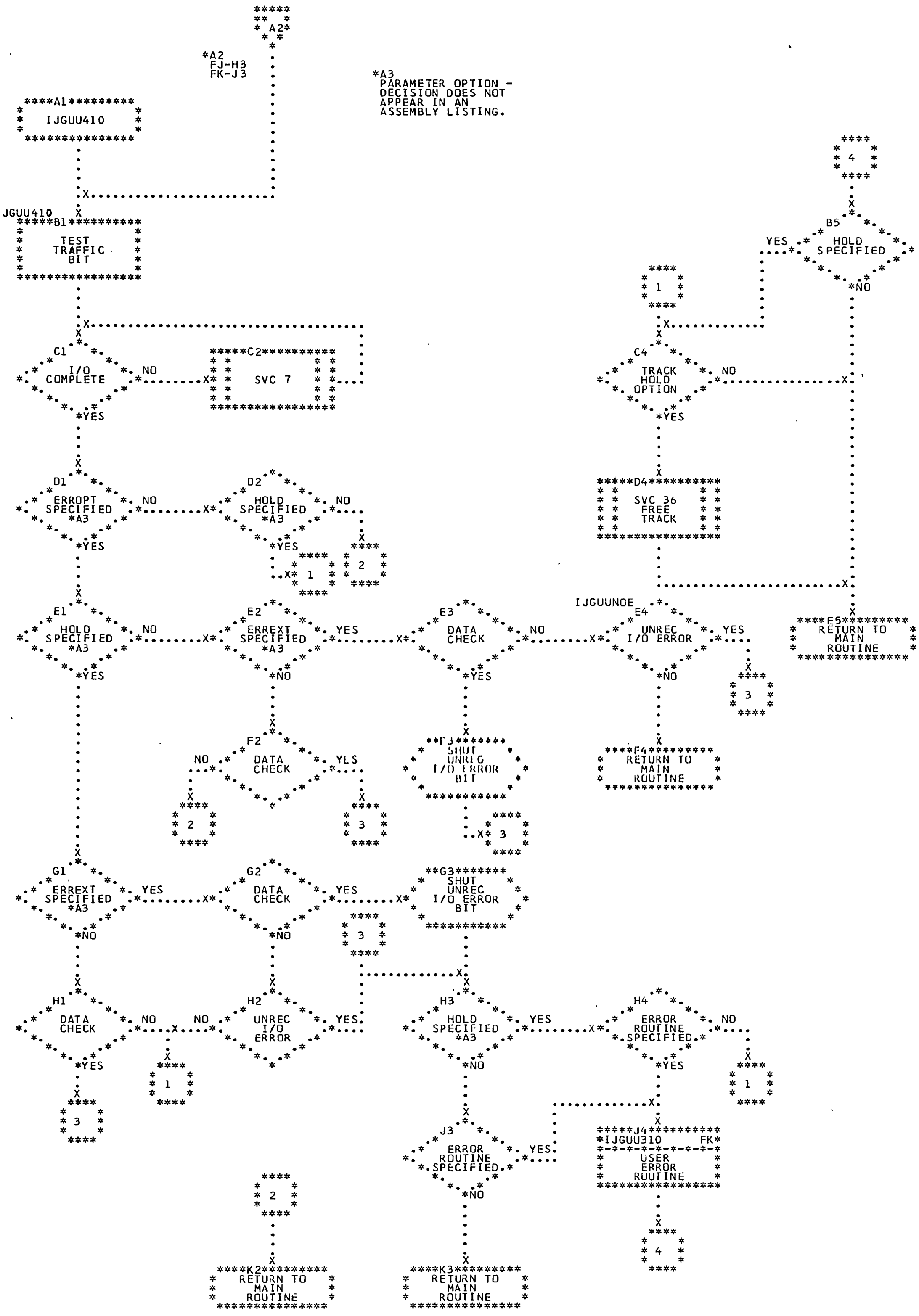

*A2 PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
1
2
FJ B1
IJGUU310
C1 SAVE MODULE REGS 8,9 RESTORE USER REGISTERS 13, 14, 15, 0, 1
D1 LOAD ADDR, OF USER ERROR ROUTINE INTO REGISTER 14
E1 ERREXT SPECIFIED *A2
NO
YES
F1 MOVE I/O ADDRESS TO LIST
F2 LOAD I/O AREA ADDR INTO REG 14
G1 LOAD REG 1 WITH LIST ADDRESS
H1 BALR TO USER ERROR ROUTINE
J1 ERREXT SPECIFIED *A2
YES
NO
*J2 WAIT AND ERROR TEST FOR READ.
*K2 WAIT AND ERROR TEST FOR WRITE.
B3 SAVE USER REGS 8-13 LOAD MODULE REGS 8, 9
C3 LOAD MODULE REGISTERS 13, 14, 15, 0, 1
D3 RETURN FOR READ AND WRITE
F3 RETRY
G3 RELOAD RETURN REGISTER 14
H3 SVC 0 RETRY I/O
J3 WAS LAST OPERATION A WRITE
YES
NO
FG A1
*J2 IJGUU260
FM B1
*K2 IJGUU410
B4 SAVE USER REGS RESTORE MOD REGS
C4 ERET =SKIP
YES
NO
C5 RELOAD RETURN REGISTER 14
D5 RETURN FOR READ AND WRITE
E4 ERET = IGNORE
NO
YES
F5 RELOAD RETURN REGISTER 14
G5 WAS LAST OPERATION A WRITE
YES
NO
H4 RETURN FOR WRITE
H5 IJGUU360 FL READ COUNT
J5 RETURN FOR READ

# Chart FL. SDMODUU: Subroutines (Section 7 of 8)

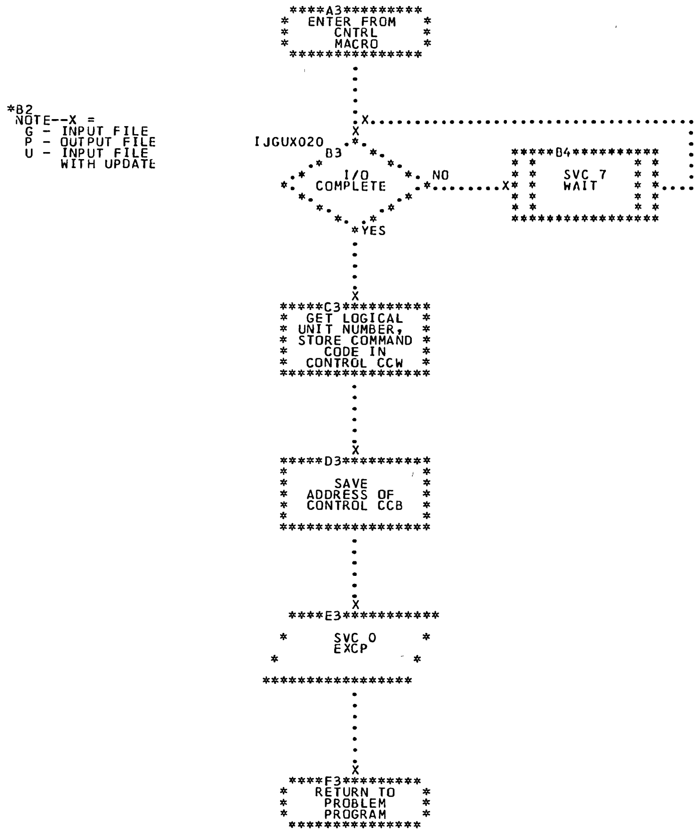


*A3
PARAMETER OPTION -
DECISION DOES NOT
APPEAR IN AN
ASSEMBLY LISTING.

## Chart FM. SDMODUU: Subroutines (Section 8 of 8)

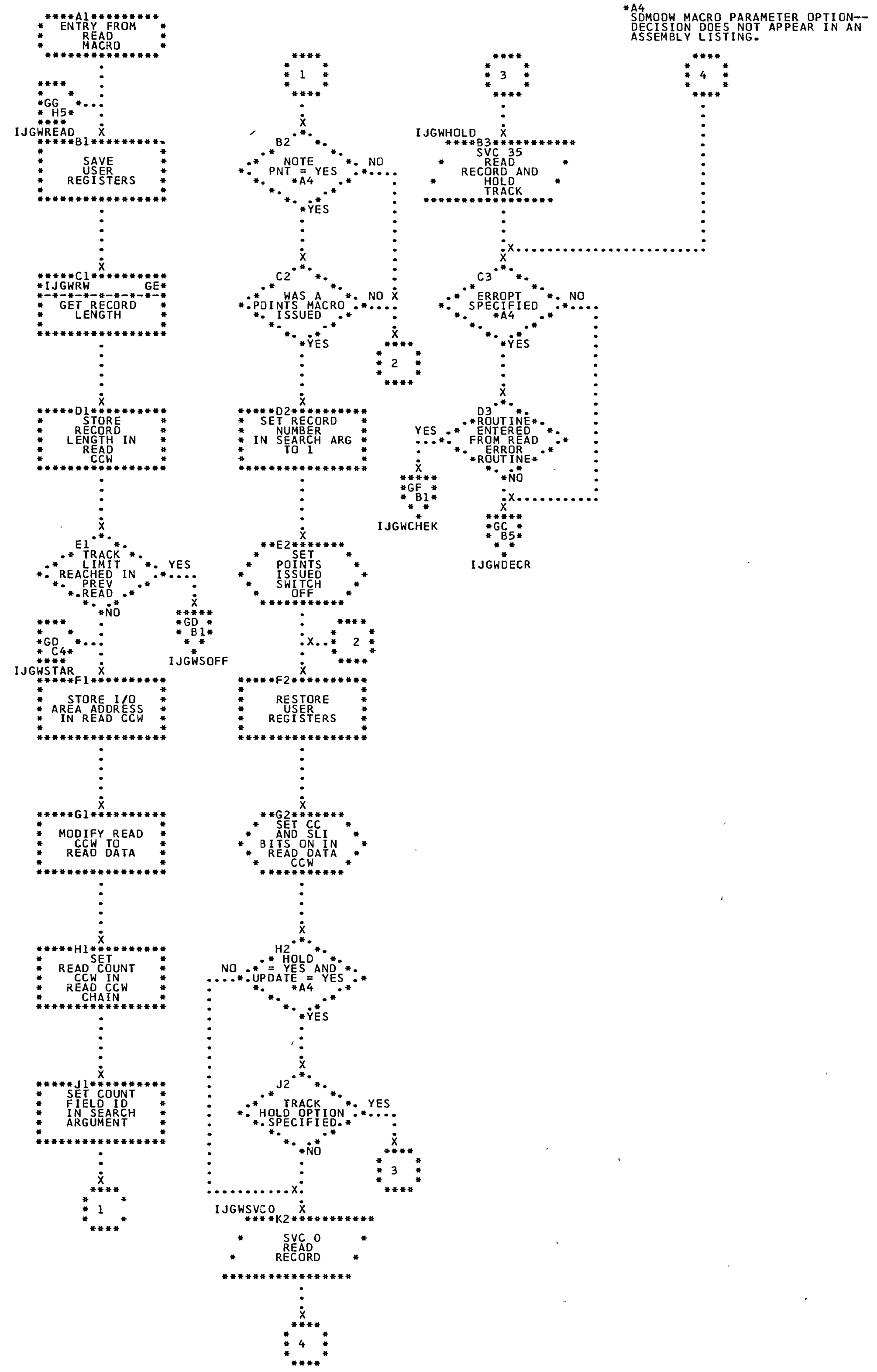


## Chart GA. CNTRL Macro, Undefined Length Records

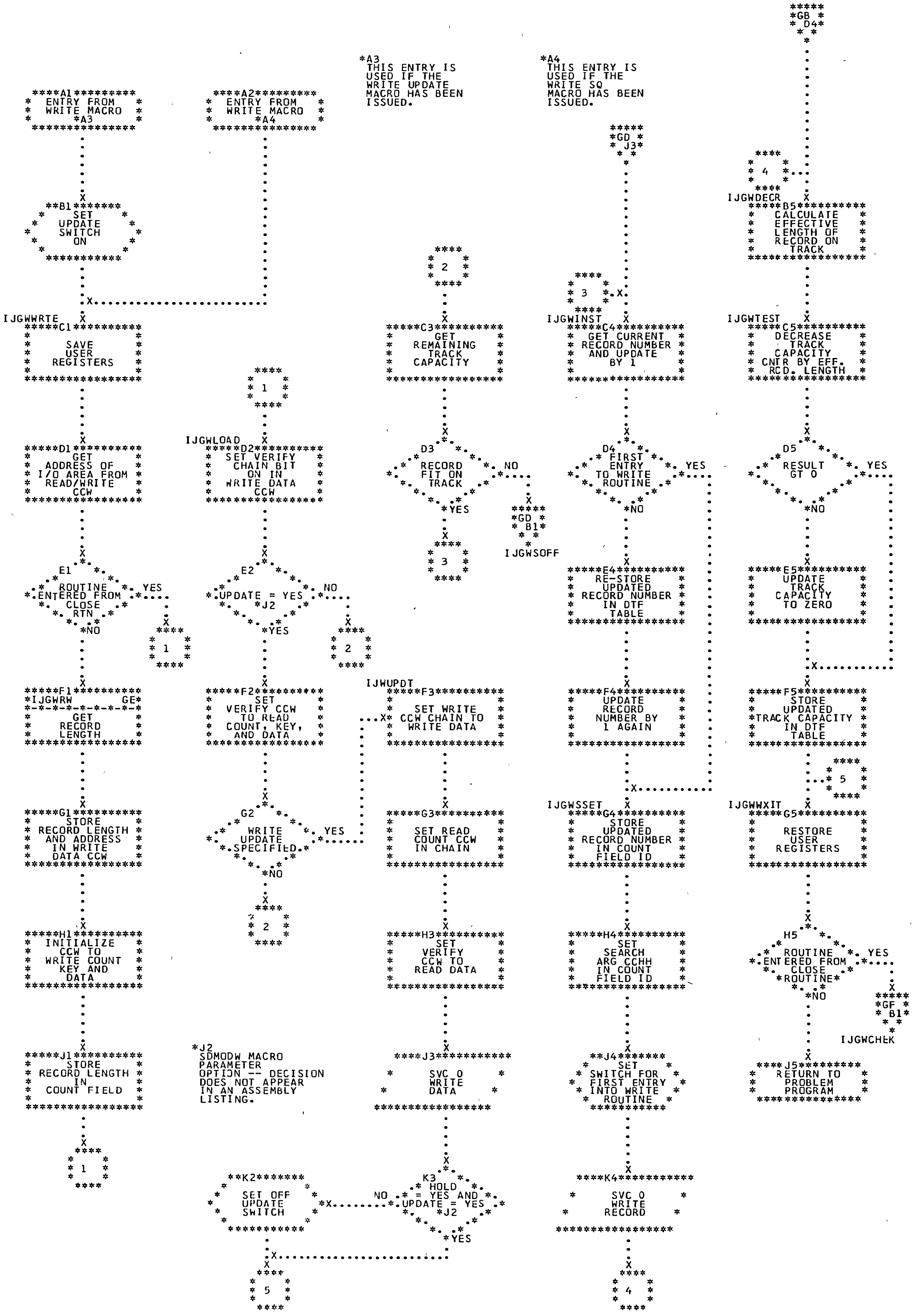


*B2
NOTE--X =
G - INPUT FILE
P - OUTPUT FILE
U - INPUT FILE WITH UPDATE
A3
ENTER FROM CNTRL MACRO
IJGUX020
D3
I/O COMPLETE
NO
YES
D4
SVC 7 WAIT
C3
GET LOGICAL UNIT NUMBER, STORE COMMAND CODE IN CONTROL CCW
D3
SAVE ADDRESS OF CONTROL CCB
E3
SVC 0 EXCP
F3
RETURN TO PROBLEM PROGRAM

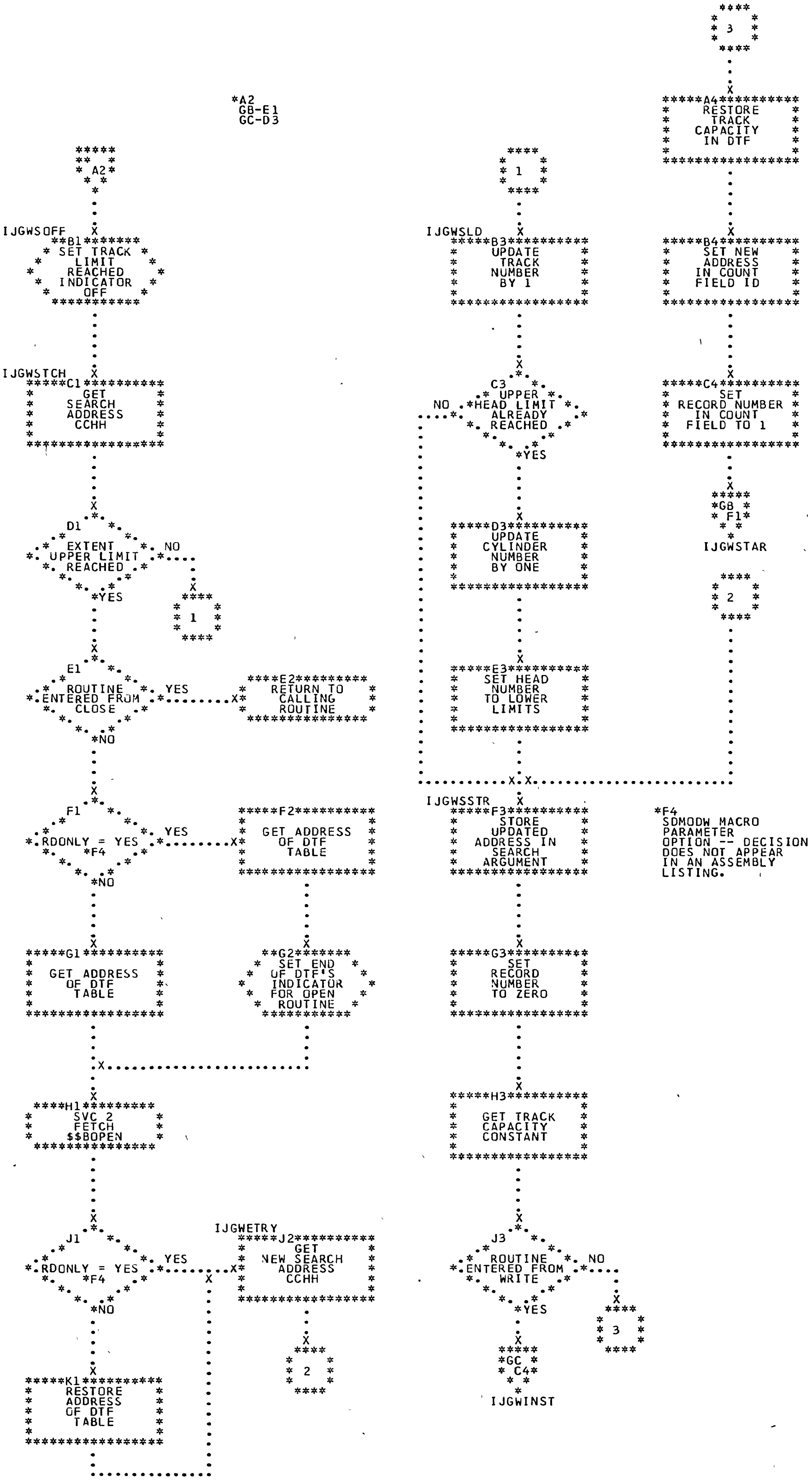
## Chart GB. SDMODW: READ Macro


*A4
SDMODW MACRO PARAMETER OPTION--
DECISION DOES NOT APPEAR IN AN
ASSEMBLY LISTING.

A1
ENTRY FROM READ MACRO
GG H5
IJGWREAD
B1
SAVE USER REGISTERS
C1
IJGWRW GE
GET RECORD LENGTH
D1
STORE RECORD LENGTH IN READ CCW
E1
TRACK LIMIT REACHED IN PREV READ
YES
NO
GD B1
IJGWSOFF
GD C4
IJGWSTAR
F1
STORE I/O AREA ADDRESS IN READ CCW
G1
MODIFY READ CCW TO READ DATA
H1
SET READ COUNT CCW IN READ CCW CHAIN
J1
SET COUNT FIELD ID IN SEARCH ARGUMENT
1
1
B2
NOTE PNT = YES *A4
NO
YES
C2
WAS A POINTS MACRO ISSUED
NO
YES
2
D2
SET RECORD NUMBER IN SEARCH ARG TO 1
E2
SET POINTS ISSUED SWITCH OFF
2
F2
RESTORE USER REGISTERS
G2
SET CC AND SLI BITS ON IN READ DATA CCW
H2
HOLD = YES AND UPDATE = YES *A4
NO
YES
J2
TRACK HOLD OPTION SPECIFIED
YES
NO
3
IJGWSVC0
K2
SVC 0 READ RECORD
4
3
IJGWHOLD
B3
SVC 35 READ RECORD AND HOLD TRACK
C3
ERROPT SPECIFIED *A4
NO
YES
D3
ROUTINE ENTERED FROM READ ERROR ROUTINE
YES
NO
GF B1
IJGWCHEK
GC B5
IJGWDECR
4


GB D4
A1 ENTRY FROM WRITE MACRO *A3
A2 ENTRY FROM WRITE MACRO *A4
*A3 THIS ENTRY IS USED IF THE WRITE UPDATE MACRO HAS BEEN ISSUED.
*A4 THIS ENTRY IS USED IF THE WRITE SQ MACRO HAS BEEN ISSUED.
GD J3
4
IJGWDECR
B5 CALCULATE EFFECTIVE LENGTH OF RECORD ON TRACK
B1 SET UPDATE SWITCH ON
2
3
IJGWRTE
C1 SAVE USER REGISTERS
C3 GET REMAINING TRACK CAPACITY
IJGWINST
C4 GET CURRENT RECORD NUMBER AND UPDATE BY 1
IJGWTEST
C5 DECREASE TRACK CAPACITY CNTR BY EFF. RCD. LENGTH
1
IJGWLOAD
D1 GET ADDRESS OF I/O AREA FROM READ/WRITE CCW
D2 SET VERIFY CHAIN BIT ON IN WRITE DATA CCW
D3 RECORD FIT ON TRACK
NO
YES
GD B1
IJGWSOFF
D4 FIRST ENTRY TO WRITE ROUTINE
YES
NO
D5 RESULT GT 0
YES
NO
E1 ROUTINE ENTERED FROM CLOSE RTN.
YES
NO
1
E2 UPDATE = YES *J2
NO
YES
2
3
E4 RE-STORE UPDATED RECORD NUMBER IN DTF TABLE
E5 UPDATE TRACK CAPACITY TO ZERO
IJWUPDT
F1 IJGWRW GE
GET RECORD LENGTH
F2 SET VERIFY CCW TO READ COUNT, KEY, AND DATA
F3 SET WRITE CCW CHAIN TO WRITE DATA
F4 UPDATE RECORD NUMBER BY 1 AGAIN
F5 STORE UPDATED TRACK CAPACITY IN DTF TABLE
5
IJGWSSET
IJGWWXIT
G1 STORE RECORD LENGTH AND ADDRESS IN WRITE DATA CCW
G2 WRITE UPDATE SPECIFIED
YES
NO
G3 SET READ COUNT CCW IN CHAIN
G4 STORE UPDATED RECORD NUMBER IN COUNT FIELD ID
G5 RESTORE USER REGISTERS
2
H1 INITIALIZE CCW TO WRITE COUNT KEY AND DATA
H3 SET VERIFY CCW TO READ DATA
H4 SET SEARCH ARG CCHH IN COUNT FIELD ID
H5 ROUTINE ENTERED FROM CLOSE ROUTINE
YES
NO
GF B1
IJGWCHEK
J1 STORE RECORD LENGTH IN COUNT FIELD
*J2 SDMODW MACRO PARAMETER OPTIDN -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
J3 SVC 0 WRITE DATA
J4 SET SWITCH FOR FIRST ENTRY INTO WRITE ROUTINE
J5 RETURN TO PROBLEM PROGRAM
1
K2 SET OFF UPDATE SWITCH
K3 HOLD = YES AND UPDATE = YES *J2
NO
YES
K4 SVC 0 WRITE RECORD
5
4

# Chart GD. SDMODW: READ, WRITE Macros, Common Routine


*A2
GB-E1
GC-D3
A2
IJGWSOFF
B1 SET TRACK LIMIT REACHED INDICATOR OFF
IJGWSTCH
C1 GET SEARCH ADDRESS CCHH
D1 EXTENT UPPER LIMIT REACHED
NO
YES
1
E1 ROUTINE ENTERED FROM CLOSE
YES
NO
E2 RETURN TO CALLING ROUTINE
F1 RDONLY = YES *F4
YES
NO
F2 GET ADDRESS OF DTF TABLE
G1 GET ADDRESS OF DTF TABLE
G2 SET END OF DTF'S INDICATOR FOR OPEN ROUTINE
H1 SVC 2 FETCH $$BOPEN
J1 RDONLY = YES *F4
YES
NO
IJGWETRY
J2 GET NEW SEARCH ADDRESS CCHH
2
K1 RESTORE ADDRESS OF DTF TABLE
1
IJGWSLD
B3 UPDATE TRACK NUMBER BY 1
C3 UPPER HEAD LIMIT ALREADY REACHED
NO
YES
D3 UPDATE CYLINDER NUMBER BY ONE
E3 SET HEAD NUMBER TO LOWER LIMITS
IJGWSSTR
F3 STORE UPDATED ADDRESS IN SEARCH ARGUMENT
G3 SET RECORD NUMBER TO ZERO
H3 GET TRACK CAPACITY CONSTANT
J3 ROUTINE ENTERED FROM WRITE
NO
YES
3
GC C4
IJGWINST
3
A4 RESTORE TRACK CAPACITY IN DTF
B4 SET NEW ADDRESS IN COUNT FIELD ID
C4 SET RECORD NUMBER IN COUNT FIELD TO 1
GB F1
IJGWSTAR
2
*F4
SDMODW MACRO PARAMETER OPTION -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.

## Chart GE. SDMODW: Read/Write Subroutine, Close Routine

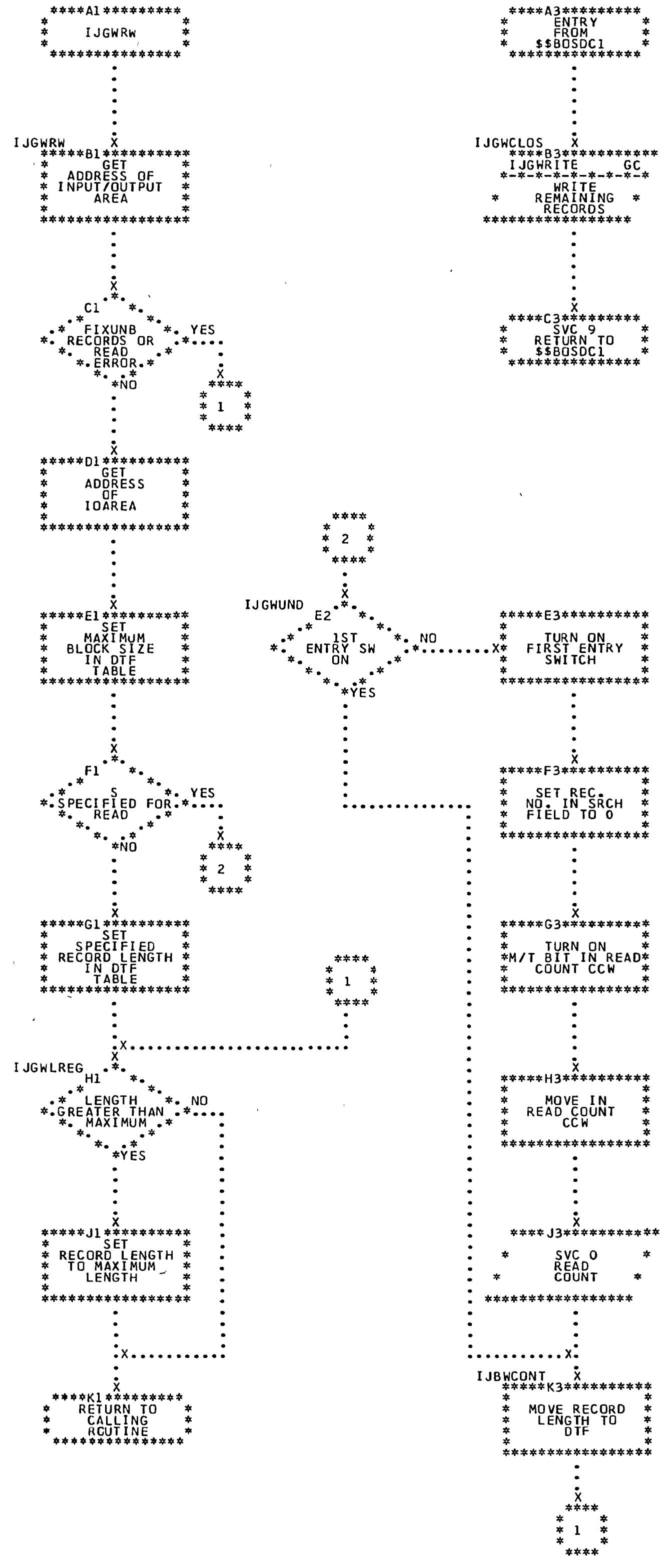

A1
IJGWRW
IJGWRW
B1
GET ADDRESS OF INPUT/OUTPUT AREA
C1
FIXUNB RECORDS OR READ ERROR
YES
NO
1
D1
GET ADDRESS OF IOAREA
E1
SET MAXIMUM BLOCK SIZE IN DTF TABLE
F1
S SPECIFIED FOR READ
YES
NO
2
G1
SET SPECIFIED RECORD LENGTH IN DTF TABLE
1
IJGWLREG
H1
LENGTH GREATER THAN MAXIMUM
NO
YES
J1
SET RECORD LENGTH TO MAXIMUM LENGTH
K1
RETURN TO CALLING RCUTINE
2
IJGWUND
E2
1ST ENTRY SW ON
NO
YES
A3
ENTRY FROM $$BOSDC1
IJGWCLOS
B3
IJGWRITE GC
WRITE REMAINING RECORDS
C3
SVC 9 RETURN TO $$BOSDC1
E3
TURN ON FIRST ENTRY SWITCH
F3
SET REC. NO. IN SRCH FIELD TO 0
G3
TURN ON M/T BIT IN READ COUNT CCW
H3
MOVE IN READ COUNT CCW
J3
SVC 0 READ COUNT
IJBWCONT
K3
MOVE RECORD LENGTH TO DTF
1

Chart GF. SDMODW: CHECK Macro (Section 1 of 5)

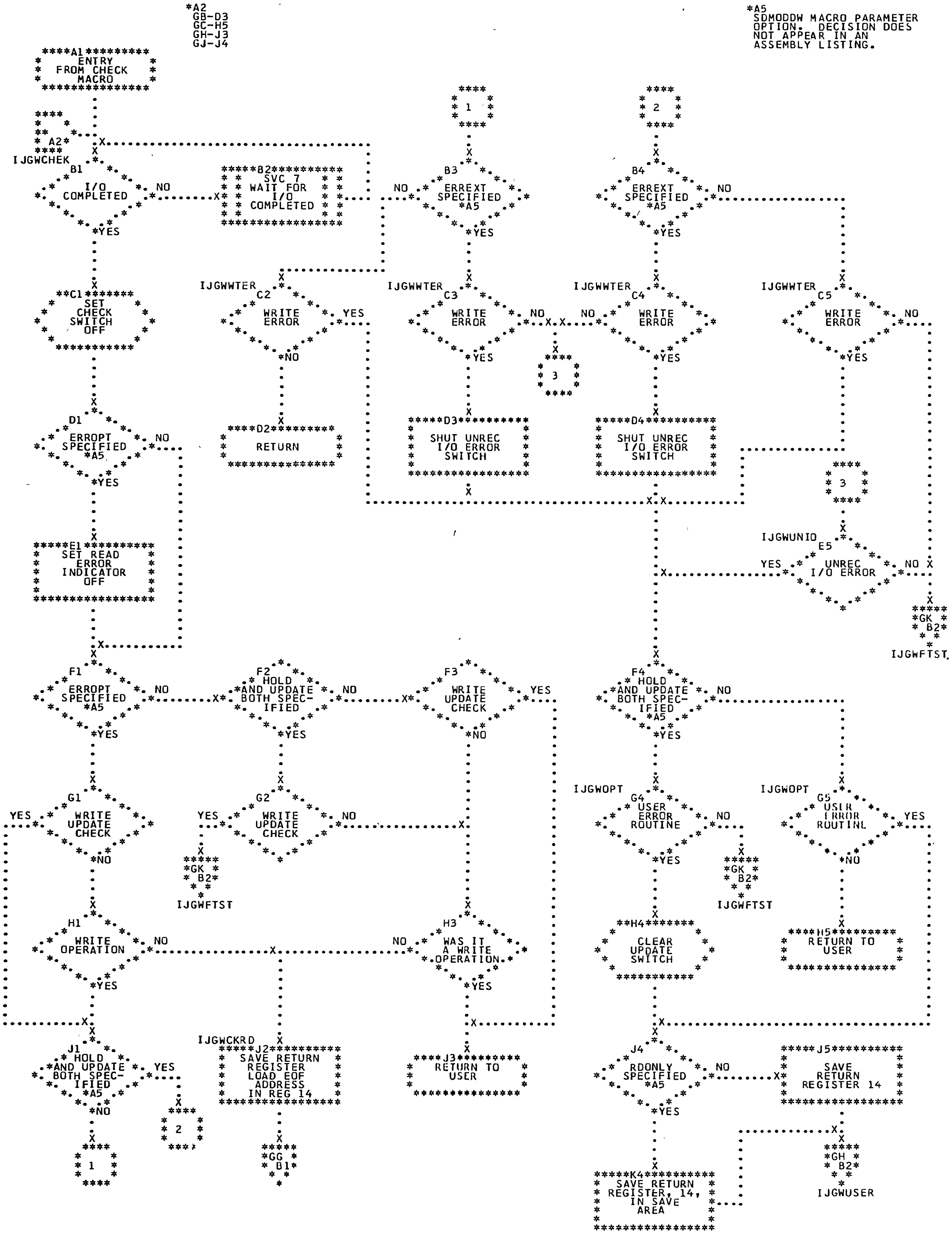
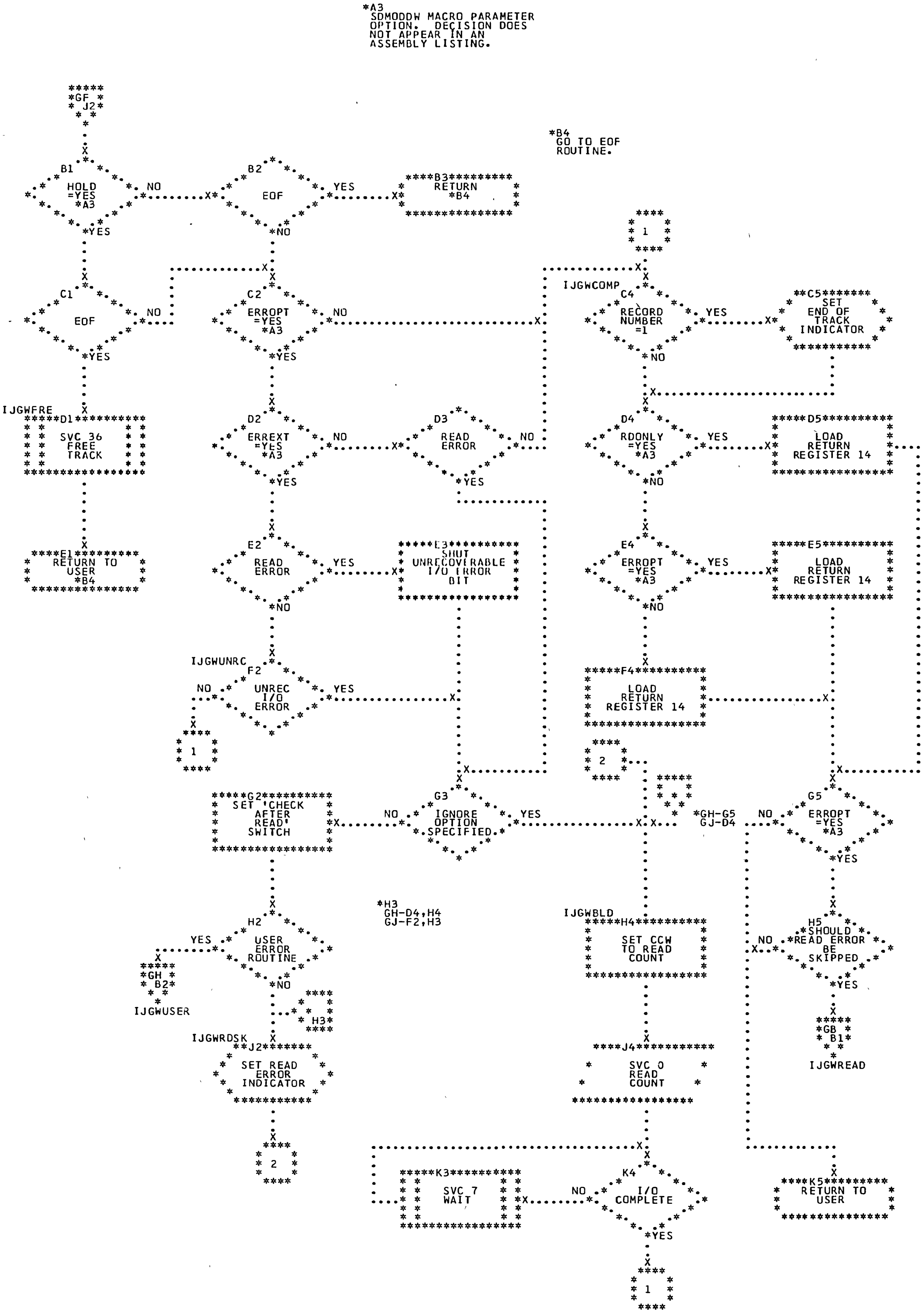


Chart GG. SDMODW: CHECK Macro (Section 2 of 5)


*A3 SDMODW MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
*B4 GO TO EOF ROUTINE.
GF J2
B1 HOLD =YES *A3
B2 EOF
B3 RETURN *B4
C1 EOF
C2 ERROPT =YES *A3
1
IJGWCOMP C4 RECORD NUMBER =1
C5 SET END OF TRACK INDICATOR
IJGWFRE D1 SVC 36 FREE TRACK
D2 ERREXT =YES *A3
D3 READ ERROR
D4 RDONLY =YES *A3
D5 LOAD RETURN REGISTER 14
E1 RETURN TO USER *B4
E2 READ ERROR
E3 SHUT UNRECOVERABLE I/O ERROR BIT
E4 ERROPT =YES *A3
E5 LOAD RETURN REGISTER 14
IJGWUNRC F2 UNREC I/O ERROR
F4 LOAD RETURN REGISTER 14
1
2
G2 SET 'CHECK AFTER READ' SWITCH
G3 IGNORE OPTION SPECIFIED
*GH-G5 GJ-D4
G5 ERROPT =YES *A3
*H3 GH-D4,H4 GJ-F2,H3
H2 USER ERROR ROUTINE
GH B2
IJGWUSER
H3
IJGWBLD H4 SET CCW TO READ COUNT
H5 SHOULD READ ERROR BE SKIPPED
GB B1
IJGWREAD
IJGWRDSK J2 SET READ ERROR INDICATOR
J4 SVC 0 READ COUNT
2
K3 SVC 7 WAIT
K4 I/O COMPLETE
K5 RETURN TO USER
1
YES
NO

## Chart GH. SDMODW: CHECK Macro (Section 3 of 5)

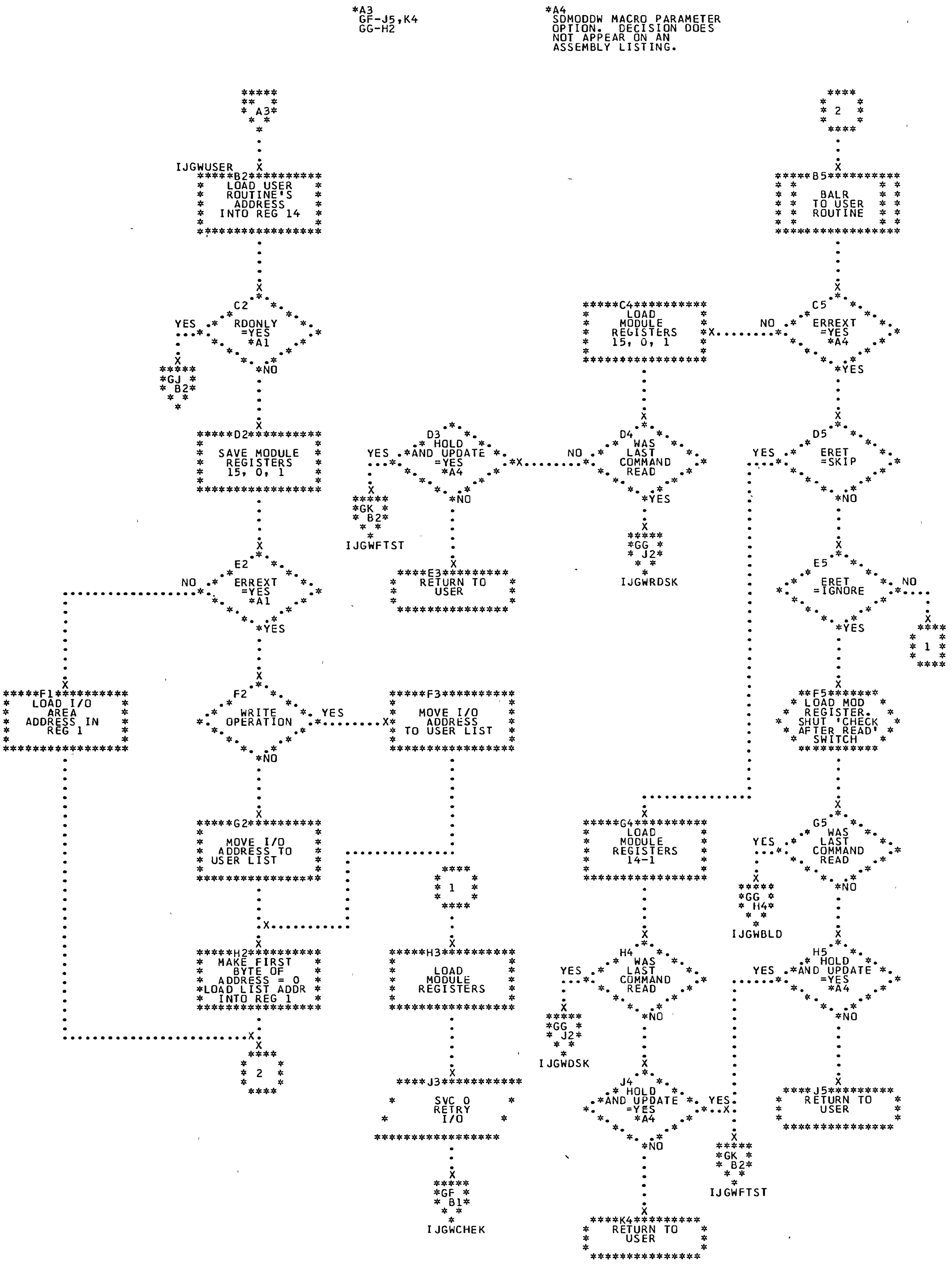
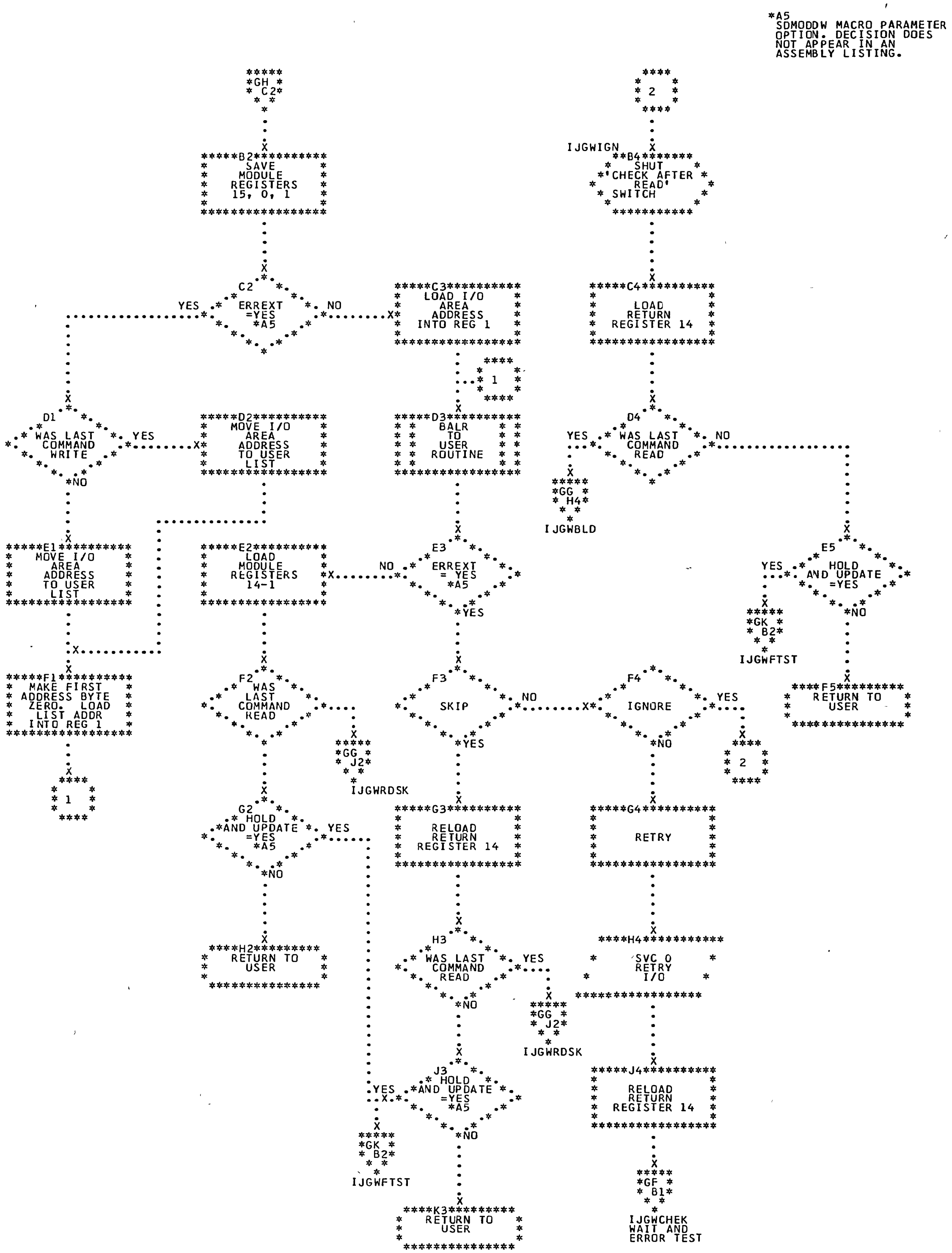


Chart GJ. SDMODW: CHECK Macro (Section 4 of 5)


*A5 SDMODDW MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
GH C2
B2 SAVE MODULE REGISTERS 15, 0, 1
C2 ERREXT =YES *A5
YES
NO
C3 LOAD I/O AREA ADDRESS INTO REG 1
1
D1 WAS LAST COMMAND WRITE
YES
NO
D2 MOVE I/O AREA ADDRESS TO USER LIST
D3 BALR TO USER ROUTINE
E1 MOVE I/O AREA ADDRESS TO USER LIST
E2 LOAD MODULE REGISTERS 14-1
E3 ERREXT = YES *A5
NO
YES
F1 MAKE FIRST ADDRESS BYTE ZERO. LOAD LIST ADDR INTO REG 1
1
F2 WAS LAST COMMAND READ
GG J2
IJGWRDSK
F3 SKIP
NO
YES
G2 HOLD AND UPDATE =YES *A5
YES
NO
H2 RETURN TO USER
G3 RELOAD RETURN REGISTER 14
H3 WAS LAST COMMAND READ
YES
NO
GG J2
IJGWRDSK
J3 HOLD AND UPDATE =YES *A5
YES
NO
GK B2
IJGWFTST
K3 RETURN TO USER
2
IJGWIGN
B4 SHUT 'CHECK AFTER READ' SWITCH
C4 LOAD RETURN REGISTER 14
D4 WAS LAST COMMAND READ
YES
NO
GG H4
IJGWBLD
F4 IGNORE
YES
NO
2
G4 RETRY
H4 SVC 0 RETRY I/O
J4 RELOAD RETURN REGISTER 14
GF B1
IJGWCHEK WAIT AND ERROR TEST
E5 HOLD AND UPDATE =YES
YES
NO
GK B2
IJGWFTST
F5 RETURN TO USER

## Chart GK. SDMODW: CHECK Macro (Section 5 of 5)

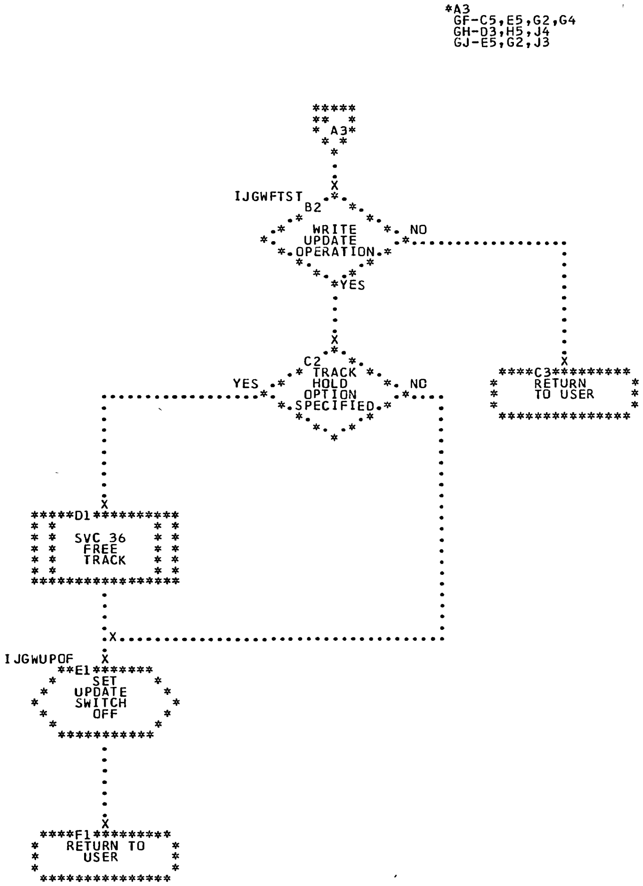
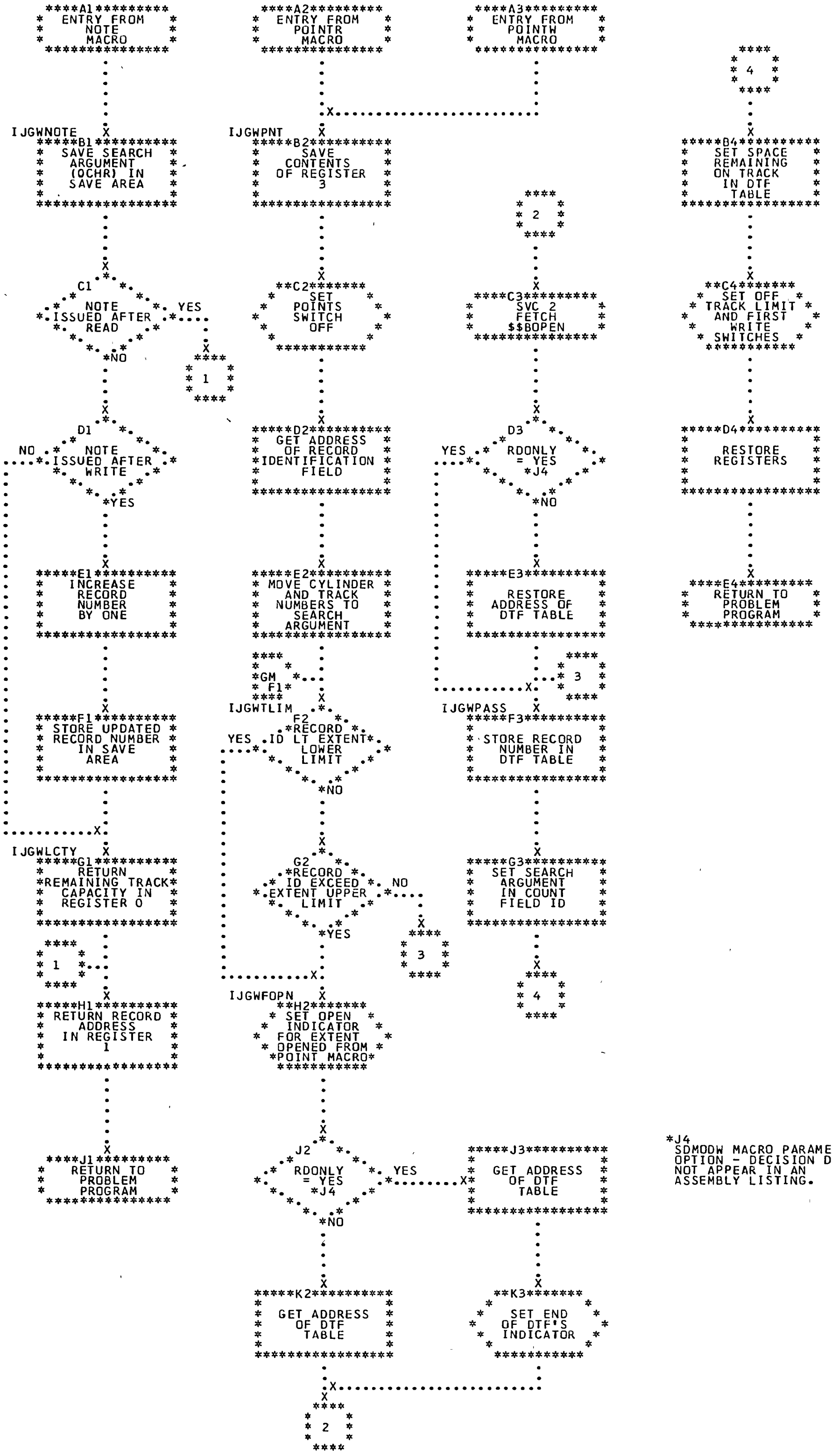
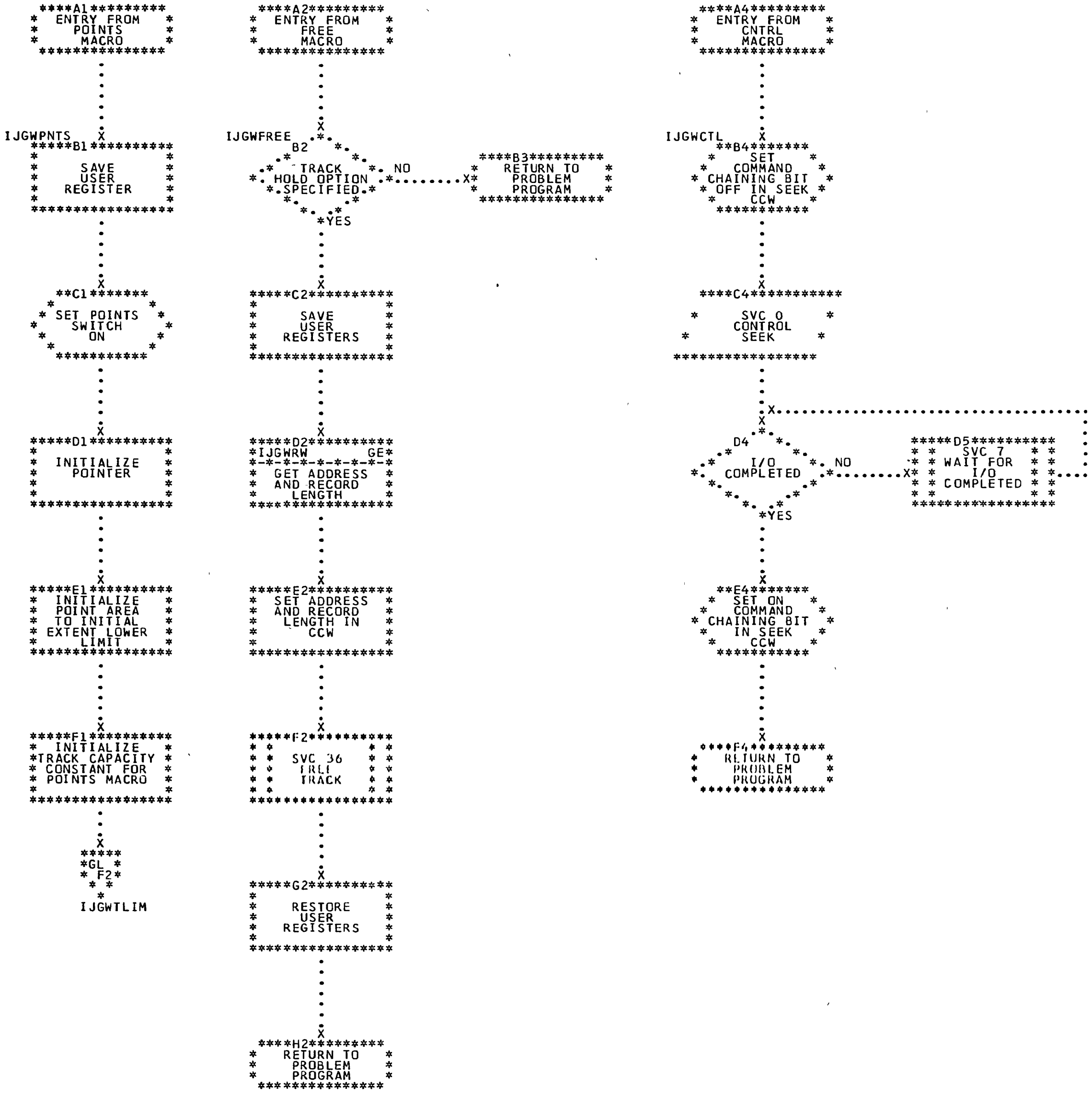


Chart GL. SDMODW: NOTE, POINTR, POINTW Macros


A1 ENTRY FROM NOTE MACRO
A2 ENTRY FROM POINTR MACRO
A3 ENTRY FROM POINTW MACRO
4
IJGWNOTE
B1 SAVE SEARCH ARGUMENT (QCHR) IN SAVE AREA
IJGWPNT
B2 SAVE CONTENTS OF REGISTER 3
B4 SET SPACE REMAINING ON TRACK IN DTF TABLE
2
C1 NOTE ISSUED AFTER READ
YES
NO
1
C2 SET POINTS SWITCH OFF
C3 SVC 2 FETCH $$BOPEN
C4 SET OFF TRACK LIMIT AND FIRST WRITE SWITCHES
D1 NOTE ISSUED AFTER WRITE
NO
YES
D2 GET ADDRESS OF RECORD IDENTIFICATION FIELD
D3 RDONLY = YES *J4
YES
NO
D4 RESTORE REGISTERS
E1 INCREASE RECORD NUMBER BY ONE
E2 MOVE CYLINDER AND TRACK NUMBERS TO SEARCH ARGUMENT
E3 RESTORE ADDRESS OF DTF TABLE
E4 RETURN TO PROBLEM PROGRAM
GM F1
3
IJGWTLIM
F2 RECORD ID LT EXTENT LOWER LIMIT
YES
NO
IJGWPASS
F1 STORE UPDATED RECORD NUMBER IN SAVE AREA
F3 STORE RECORD NUMBER IN DTF TABLE
IJGWLCTY
G1 RETURN REMAINING TRACK CAPACITY IN REGISTER 0
G2 RECORD ID EXCEED EXTENT UPPER LIMIT
NO
YES
3
G3 SET SEARCH ARGUMENT IN COUNT FIELD ID
1
4
H1 RETURN RECORD ADDRESS IN REGISTER 1
IJGWFOPN
H2 SET OPEN INDICATOR FOR EXTENT OPENED FROM POINT MACRO
J1 RETURN TO PROBLEM PROGRAM
J2 RDONLY = YES *J4
YES
NO
J3 GET ADDRESS OF DTF TABLE
K2 GET ADDRESS OF DTF TABLE
K3 SET END OF DTF'S INDICATOR
2

*J4
SDMODW MACRO PARAMETER
OPTION - DECISION DOES
NOT APPEAR IN AN
ASSEMBLY LISTING.

Chart GM. SDMODW: POINTS, FREE, CNTRL Macros


A1
ENTRY FROM POINTS MACRO
IJGWPNTS
B1
SAVE USER REGISTER
C1
SET POINTS SWITCH ON
D1
INITIALIZE POINTER
E1
INITIALIZE POINT AREA TO INITIAL EXTENT LOWER LIMIT
F1
INITIALIZE TRACK CAPACITY CONSTANT FOR POINTS MACRO
GL F2
IJGWTLIM
A2
ENTRY FROM FREE MACRO
IJGWFREE
B2
TRACK HOLD OPTION SPECIFIED
NO
YES
B3
RETURN TO PROBLEM PROGRAM
C2
SAVE USER REGISTERS
D2
IJGWRW GE
GET ADDRESS AND RECORD LENGTH
E2
SET ADDRESS AND RECORD LENGTH IN CCW
F2
SVC 36 FREE TRACK
G2
RESTORE USER REGISTERS
H2
RETURN TO PROBLEM PROGRAM
A4
ENTRY FROM CNTRL MACRO
IJGWCTL
B4
SET COMMAND CHAINING BIT OFF IN SEEK CCW
C4
SVC 0 CONTROL SEEK
D4
I/O COMPLETED
NO
YES
D5
SVC 7 WAIT FOR I/O COMPLETED
E4
SET ON COMMAND CHAINING BIT IN SEEK CCW
F4
RETURN TO PROBLEM PROGRAM

## Chart GN. SDMOD: FEOVD Macro

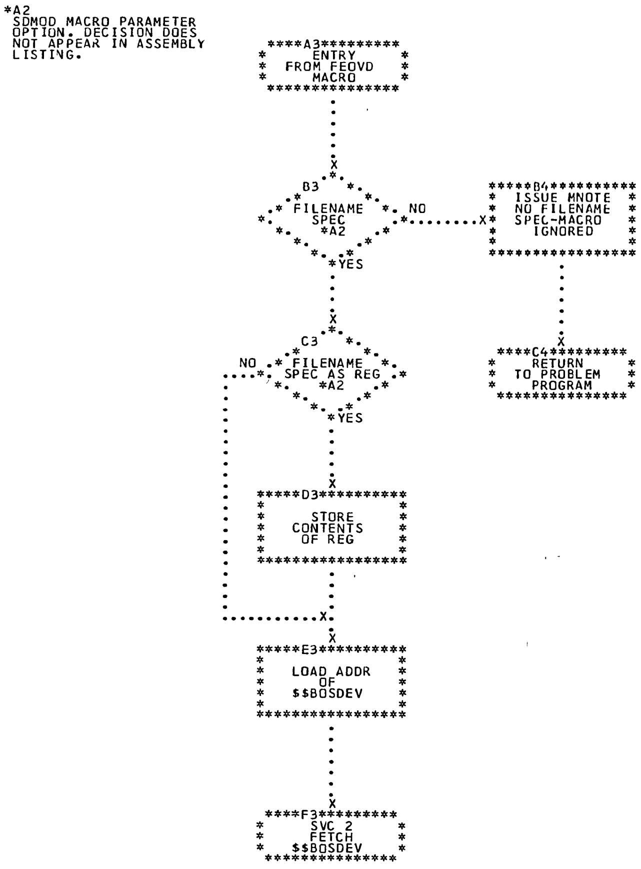
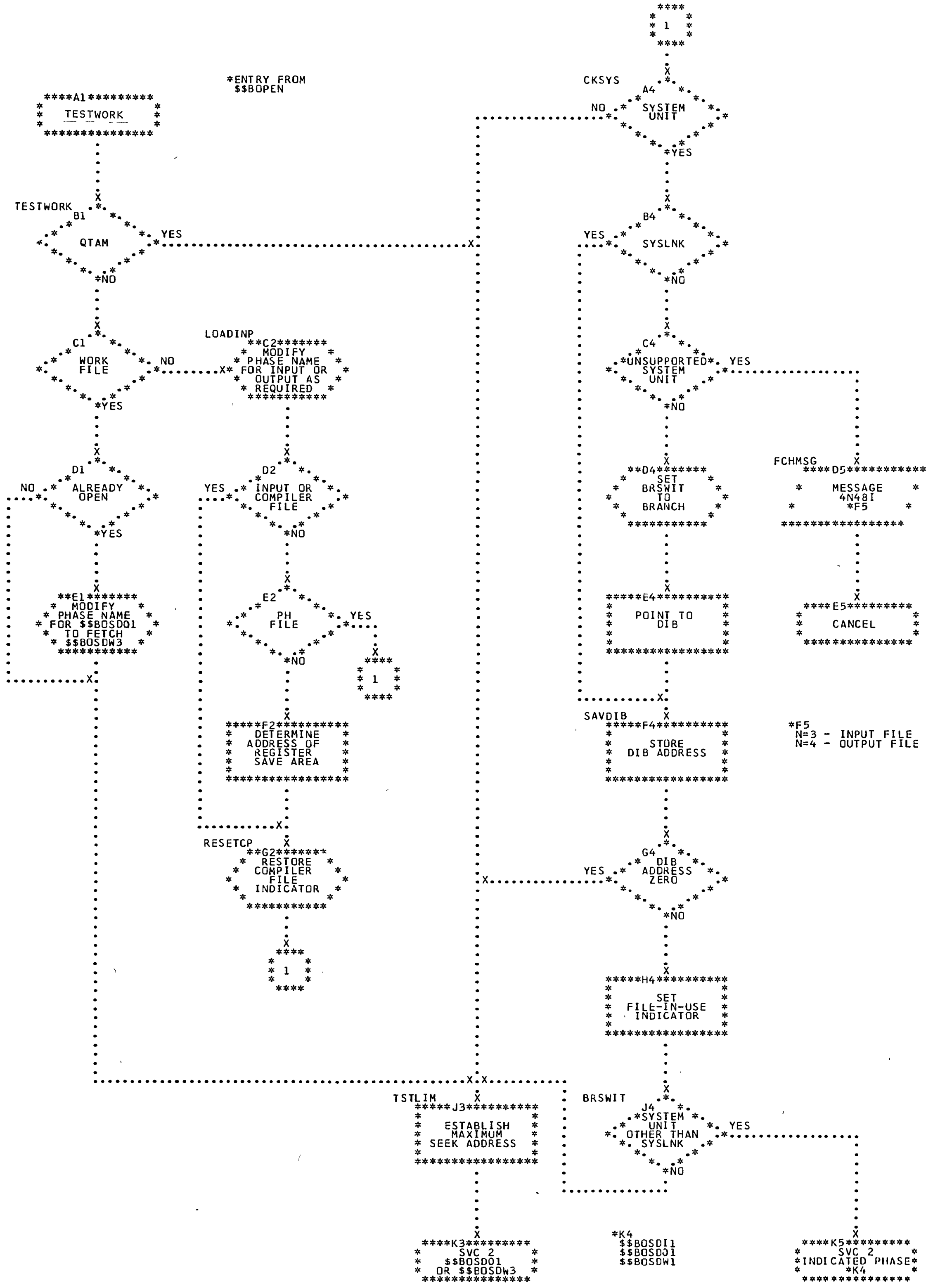


Chart HA. $$BOSD00: SD Open, Initialization



*ENTRY FROM $$BOPEN
A1 TESTWORK
TESTWORK B1 QTAM
YES
NO
C1 WORK FILE
NO
YES
LOADINP C2 MODIFY PHASE NAME FOR INPUT OR OUTPUT AS REQUIRED
D1 ALREADY OPEN
NO
YES
D2 INPUT OR COMPILER FILE
YES
NO
E1 MODIFY PHASE NAME FOR $$BOSDO1 TO FETCH $$BOSDW3
E2 PH FILE
YES
NO
1
F2 DETERMINE ADDRESS OF REGISTER SAVE AREA
RESETCP G2 RESTORE COMPILER FILE INDICATOR
1
1
CKSYS A4 SYSTEM UNIT
NO
YES
B4 SYSLNK
YES
NO
C4 UNSUPPORTED SYSTEM UNIT
YES
NO
D4 SET BRSWIT TO BRANCH
FCHMSG D5 MESSAGE 4N48I *F5
E4 POINT TO DIB
E5 CANCEL
SAVDIB F4 STORE DIB ADDRESS
*F5 N=3 - INPUT FILE N=4 - OUTPUT FILE
G4 DIB ADDRESS ZERO
YES
NO
H4 SET FILE-IN-USE INDICATOR
TSTLIM J3 ESTABLISH MAXIMUM SEEK ADDRESS
BRSWIT J4 SYSTEM UNIT OTHER THAN SYSLNK
YES
NO
K3 SVC 2 $$BOSDO1 OR $$BOSDW3
*K4 $$BOSDI1 $$BOSDO1 $$BOSDW1
K5 SVC 2 INDICATED PHASE *K4

## Chart HB. $$BOSD01: SD Open, DLBL Extents

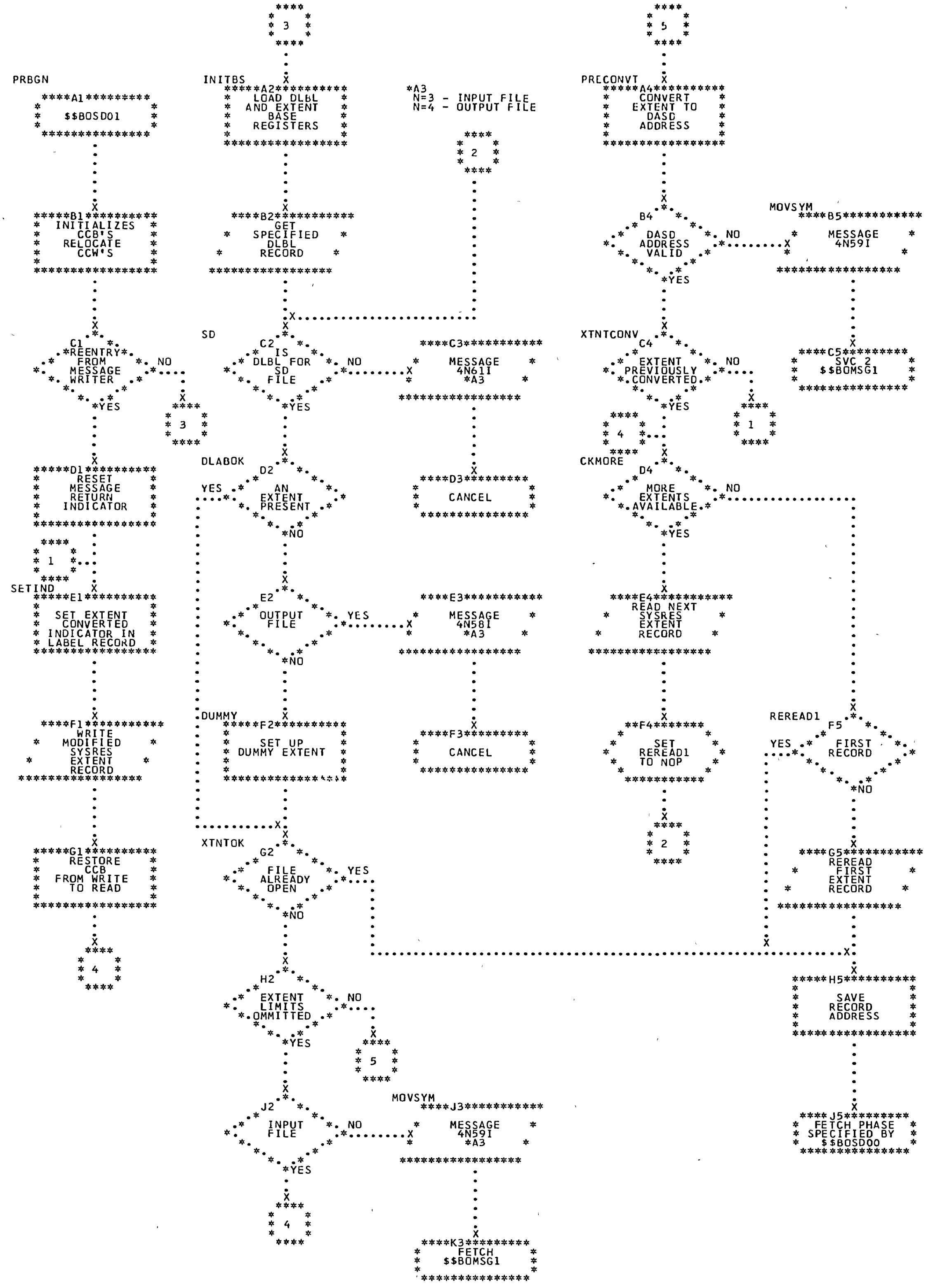


Chart HC. $$BOSDI1: SD Open Input, DLBL Extents (Section 1 of 3)

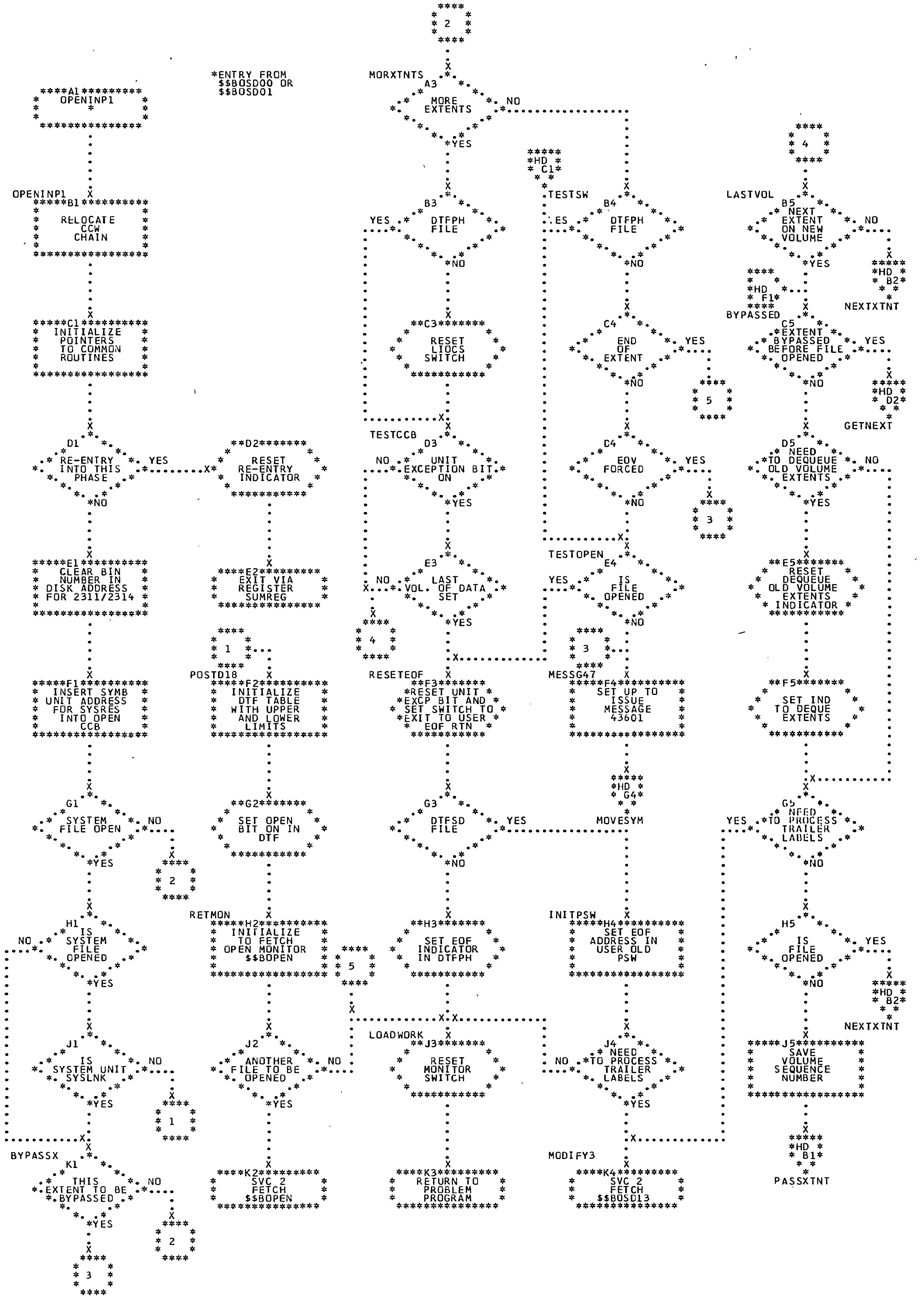

*ENTRY FROM $$BOSD00 OR $$BOSD01
A1 OPENINP1
OPENINP1 B1 RELOCATE CCW CHAIN
C1 INITIALIZE POINTERS TO COMMON ROUTINES
D1 RE-ENTRY INTO THIS PHASE
YES
NO
D2 RESET RE-ENTRY INDICATOR
E1 CLEAR BIN NUMBER IN DISK ADDRESS FOR 2311/2314
E2 EXIT VIA REGISTER SUMREG
1
F1 INSERT SYMB UNIT ADDRESS FOR SYSRES INTO OPEN CCB
POSTD18 F2 INITIALIZE DTF TABLE WITH UPPER AND LOWER LIMITS
G1 SYSTEM FILE OPEN
NO
YES
2
G2 SET OPEN BIT ON IN DTF
H1 IS SYSTEM FILE OPENED
NO
YES
RETMON H2 INITIALIZE TO FETCH OPEN MONITOR $$BOPEN
5
J1 IS SYSTEM UNIT SYSLNK
NO
YES
1
J2 ANOTHER FILE TO BE OPENED
NO
YES
BYPASSX K1 THIS EXTENT TO BE BYPASSED
NO
YES
2
3
K2 SVC 2 FETCH $$BOPEN
2
MORXTNTS A3 MORE EXTENTS
NO
YES
B3 DTFPH FILE
YES
NO
C3 RESET LIOCS SWITCH
TESTCCB D3 UNIT EXCEPTION BIT ON
NO
YES
E3 LAST VOL. OF DATA SET
NO
YES
4
RESETEOF F3 RESET UNIT EXCP BIT AND SET SWITCH TO EXIT TO USER EOF RTN
G3 DTFSD FILE
YES
NO
H3 SET EOF INDICATOR IN DTFPH
LOADWORK J3 RESET MONITOR SWITCH
K3 RETURN TO PROBLEM PROGRAM
HD C1
TESTSW B4 DTFPH FILE
YES
NO
C4 END OF EXTENT
YES
NO
5
D4 EOV FORCED
YES
NO
3
TESTOPEN E4 IS FILE OPENED
YES
NO
3
MESSG47 F4 SET UP TO ISSUE MESSAGE 4360I
HD G4
MOVESYM
INITPSW H4 SET EOF ADDRESS IN USER OLD PSW
J4 NEED TO PROCESS TRAILER LABELS
NO
YES
MODIFY3 K4 SVC 2 FETCH $$BOSDI3
4
LASTVOL B5 NEXT EXTENT ON NEW VOLUME
NO
YES
HD B2
NEXTXTNT
HD F1
BYPASSED C5 EXTENT BYPASSED BEFORE FILE OPENED
YES
NO
HD D2
GETNEXT
D5 NEED TO DEQUEUE OLD VOLUME EXTENTS
NO
YES
E5 RESET DEQUEUE OLD VOLUME EXTENTS INDICATOR
F5 SET IND TO DEQUE EXTENTS
G5 NEED TO PROCESS TRAILER LABELS
YES
NO
H5 IS FILE OPENED
YES
NO
HD B2
NEXTXTNT
J5 SAVE VOLUME SEQUENCE NUMBER
HD B1
PASSXTNT

Chart HD. $$BOSDI1: SD Open Input, DLBL Extents (Section 2 of 3)

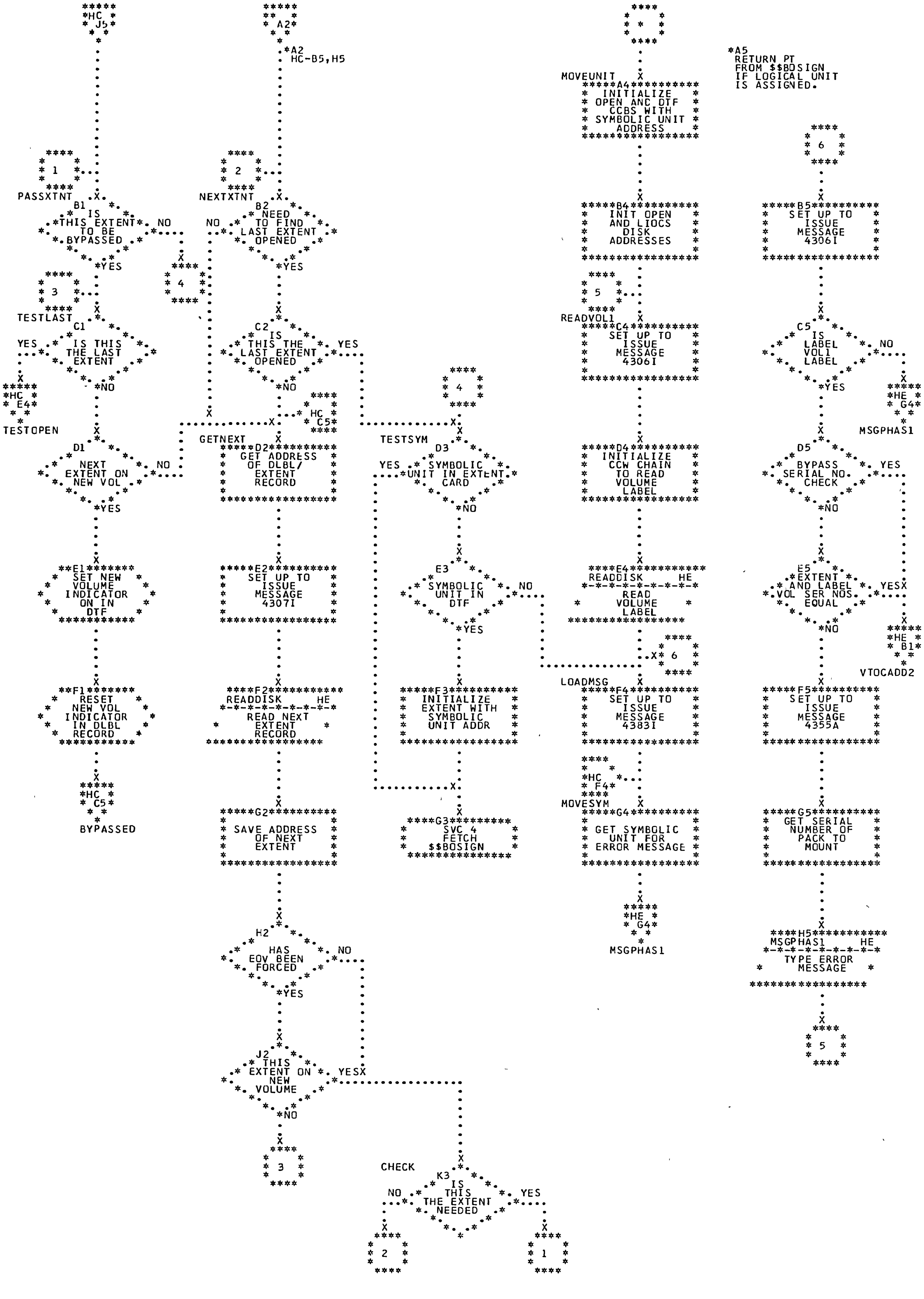

HC J5
A2
A2 HC-B5,H5
A5 RETURN PT FROM $$BOSIGN IF LOGICAL UNIT IS ASSIGNED.
MOVEUNIT
A4 INITIALIZE OPEN ANC DTF CCBS WITH SYMBOLIC UNIT ADDRESS
1
PASSXTNT
B1 IS THIS EXTENT TO BE BYPASSED
NO
YES
2
NEXTXTNT
B2 NEED TO FIND LAST EXTENT OPENED
NO
YES
B4 INIT OPEN AND LIOCS DISK ADDRESSES
6
B5 SET UP TO ISSUE MESSAGE 4306I
3
TESTLAST
C1 IS THIS THE LAST EXTENT
YES
NO
HC E4
4
C2 IS THIS THE LAST EXTENT OPENED
YES
NO
HC C5
5
READVOL1
C4 SET UP TO ISSUE MESSAGE 4306I
C5 IS LABEL VOL1 LABEL
NO
YES
HE G4
MSGPHAS1
4
TESTOPEN
D1 NEXT EXTENT ON NEW VOL
NO
YES
GETNEXT
D2 GET ADDRESS OF DLBL/ EXTENT RECORD
TESTSYM
D3 SYMBOLIC UNIT IN EXTENT CARD
YES
NO
D4 INITIALIZE CCW CHAIN TO READ VOLUME LABEL
D5 BYPASS SERIAL NO. CHECK
YES
NO
E1 SET NEW VOLUME INDICATOR ON IN DTF
E2 SET UP TO ISSUE MESSAGE 4307I
E3 SYMBOLIC UNIT IN DTF
NO
YES
E4 READDISK HE READ VOLUME LABEL
E5 EXTENT AND LABEL VOL SER NOS. EQUAL
YES
NO
HE B1
VTOCADD2
6
LOADMSG
F1 RESET NEW VOL INDICATOR IN DLBL RECORD
F2 READDISK HE READ NEXT EXTENT RECORD
F3 INITIALIZE EXTENT WITH SYMBOLIC UNIT ADDR
F4 SET UP TO ISSUE MESSAGE 4383I
F5 SET UP TO ISSUE MESSAGE 4355A
HC C5
BYPASSED
HC F4
MOVESYM
G2 SAVE ADDRESS OF NEXT EXTENT
G3 SVC 4 FETCH $$BOSIGN
G4 GET SYMBOLIC UNIT FOR ERROR MESSAGE
G5 GET SERIAL NUMBER OF PACK TO MOUNT
HE G4
MSGPHAS1
H2 HAS EOV BEEN FORCED
NO
YES
H5 MSGPHAS1 HE TYPE ERROR MESSAGE
5
J2 THIS EXTENT ON NEW VOLUME
YES
NO
3
CHECK
K3 IS THIS THE EXTENT NEEDED
NO
YES
2
1

Chart HE. $$BOSDI1: SD Open Input, DLBL Extents (Section 3 of 3)

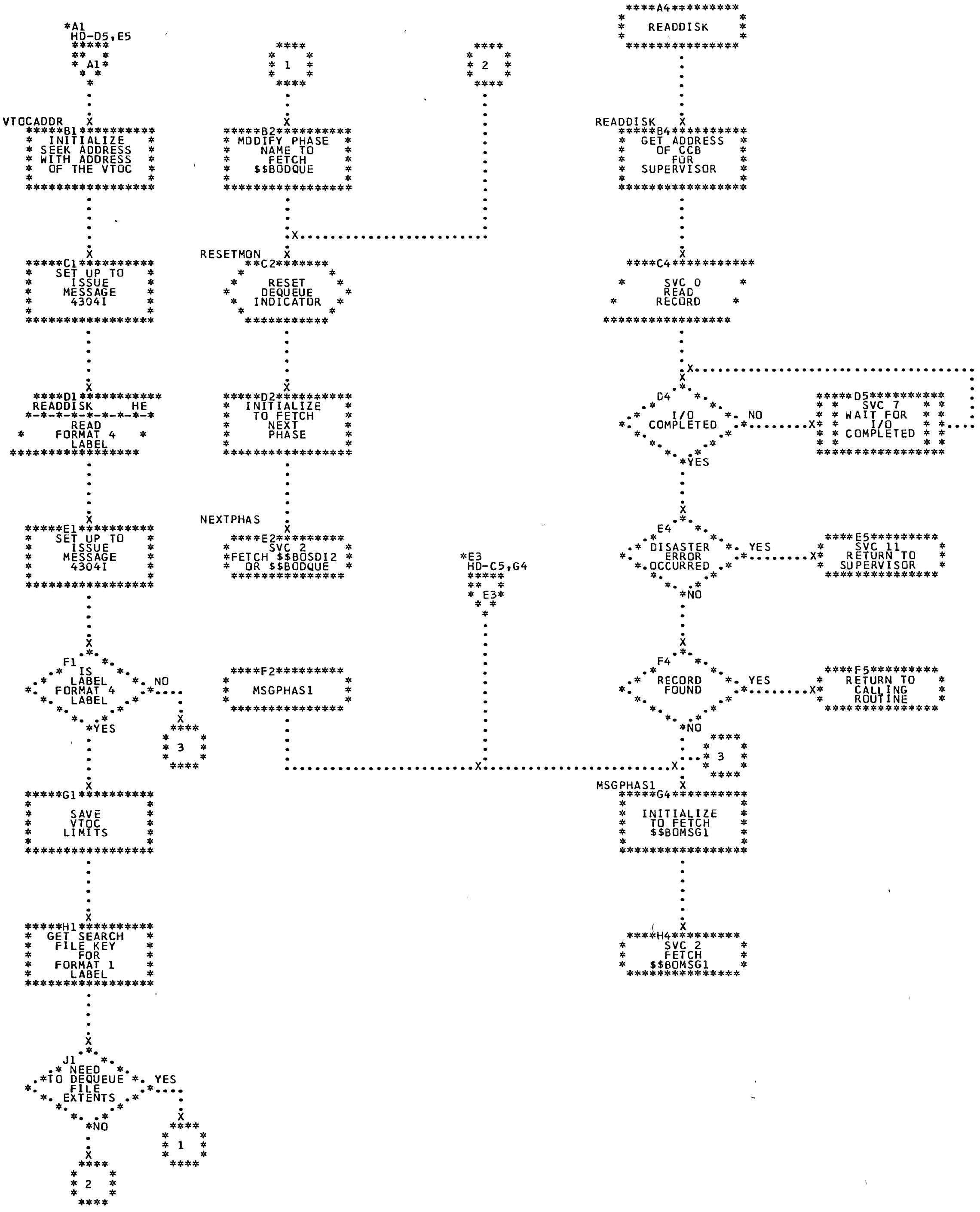
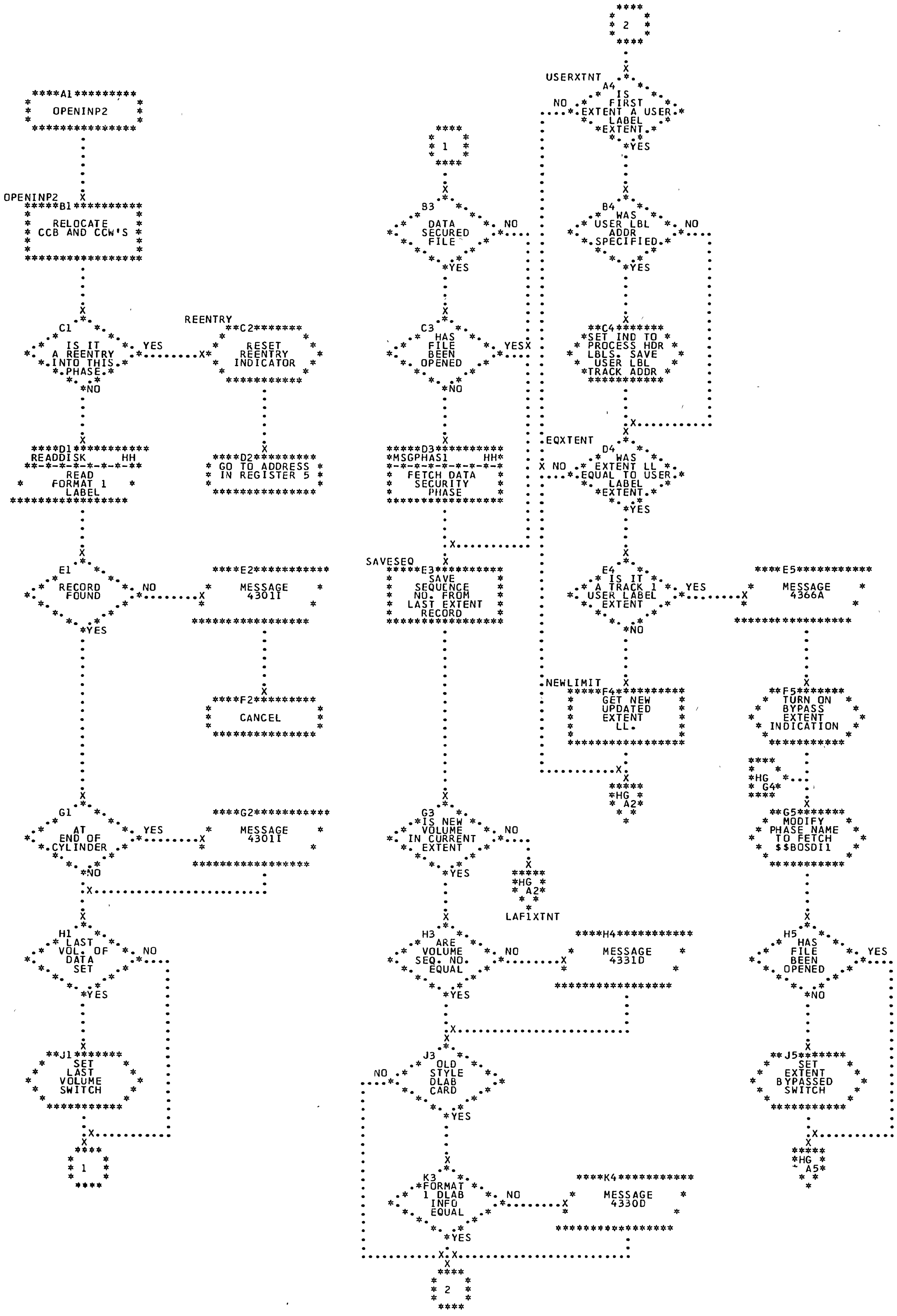


Chart HF. $$BOSDI2: SD Open Input, Extent to DTF (Section 1 of 3)


A1 OPENINP2
OPENINP2 B1 RELOCATE CCB AND CCW'S
C1 IS IT A REENTRY INTO THIS PHASE
YES
NO
REENTRY C2 RESET REENTRY INDICATOR
D1 READDISK HH READ FORMAT 1 LABEL
D2 GO TO ADDRESS IN REGISTER 5
E1 RECORD FOUND
NO
YES
E2 MESSAGE 4301I
F2 CANCEL
G1 AT END OF CYLINDER
YES
NO
G2 MESSAGE 4301I
H1 LAST VOL. OF DATA SET
NO
YES
J1 SET LAST VOLUME SWITCH
1
1
B3 DATA SECURED FILE
NO
YES
C3 HAS FILE BEEN OPENED
YES
NO
D3 MSGPHAS1 HH FETCH DATA SECURITY PHASE
SAVESEQ E3 SAVE SEQUENCE NO. FROM LAST EXTENT RECORD
G3 IS NEW VOLUME IN CURRENT EXTENT
NO
YES
HG A2
LAF1XTNT
H3 ARE VOLUME SEQ. NO. EQUAL
NO
YES
H4 MESSAGE 4331D
J3 OLD STYLE DLAB CARD
NO
YES
K3 FORMAT 1 DLAB INFO EQUAL
NO
YES
K4 MESSAGE 4330D
2
2
USERXTNT A4 IS FIRST EXTENT A USER LABEL EXTENT
NO
YES
B4 WAS USER LBL ADDR SPECIFIED
NO
YES
C4 SET IND TO PROCESS HDR LBLS. SAVE USER LBL TRACK ADDR
EQXTENT D4 WAS EXTENT LL EQUAL TO USER LABEL EXTENT
NO
YES
E4 IS IT A TRACK 1 USER LABEL EXTENT
YES
NO
E5 MESSAGE 4366A
NEWLIMIT F4 GET NEW UPDATED EXTENT LL.
HG A2
F5 TURN ON BYPASS EXTENT INDICATION
HG G4
G5 MODIFY PHASE NAME TO FETCH $$BOSDI1
H5 HAS FILE BEEN OPENED
YES
NO
J5 SET EXTENT BYPASSED SWITCH
HG A5

## Chart HG. $$BOSDI2: SD Open Input, Extent to DTF (Section 2 of 3)

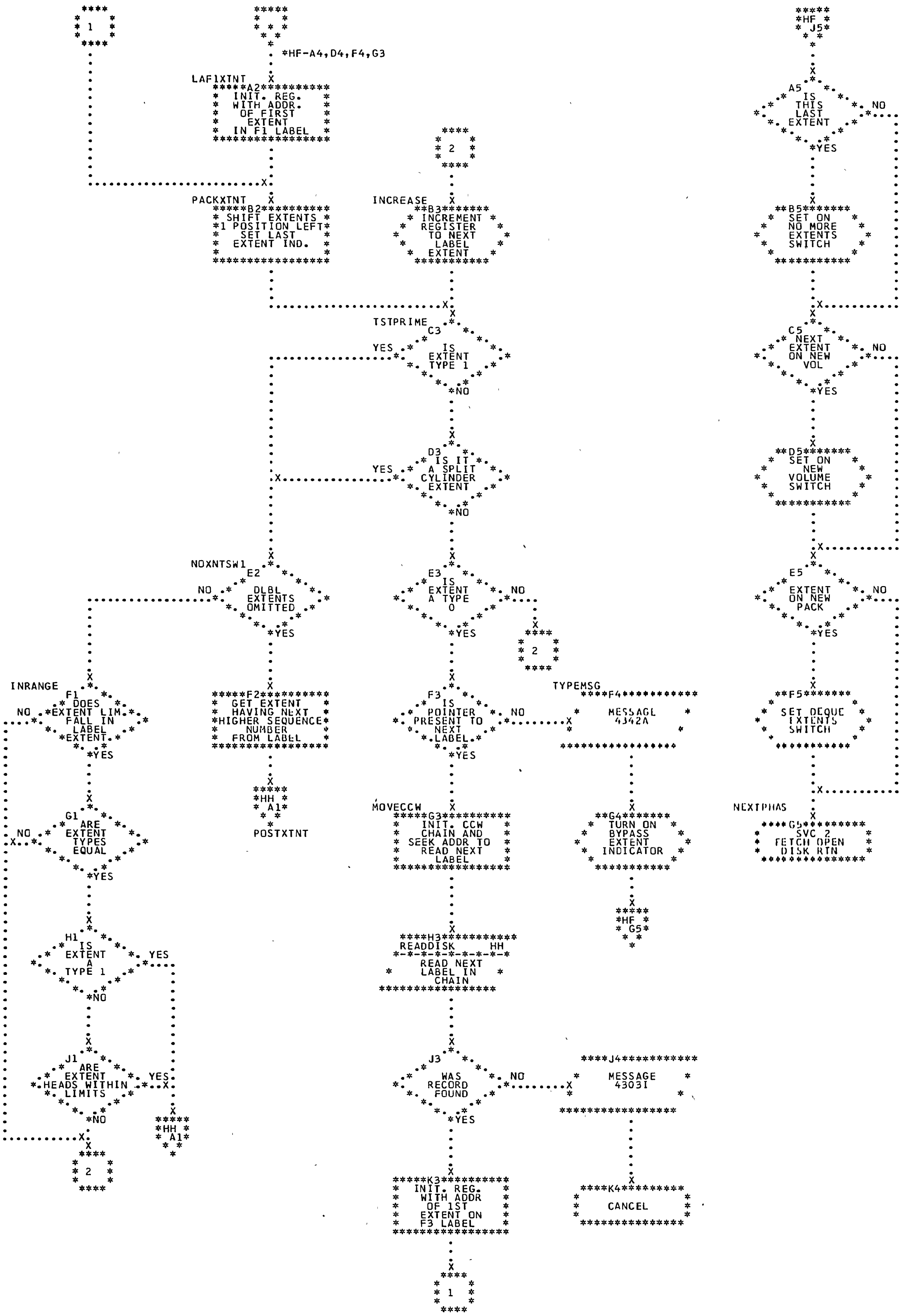


Chart HH. $$BOSDI2: SD Open Input, Extent to DTF (Section 3 of 3)

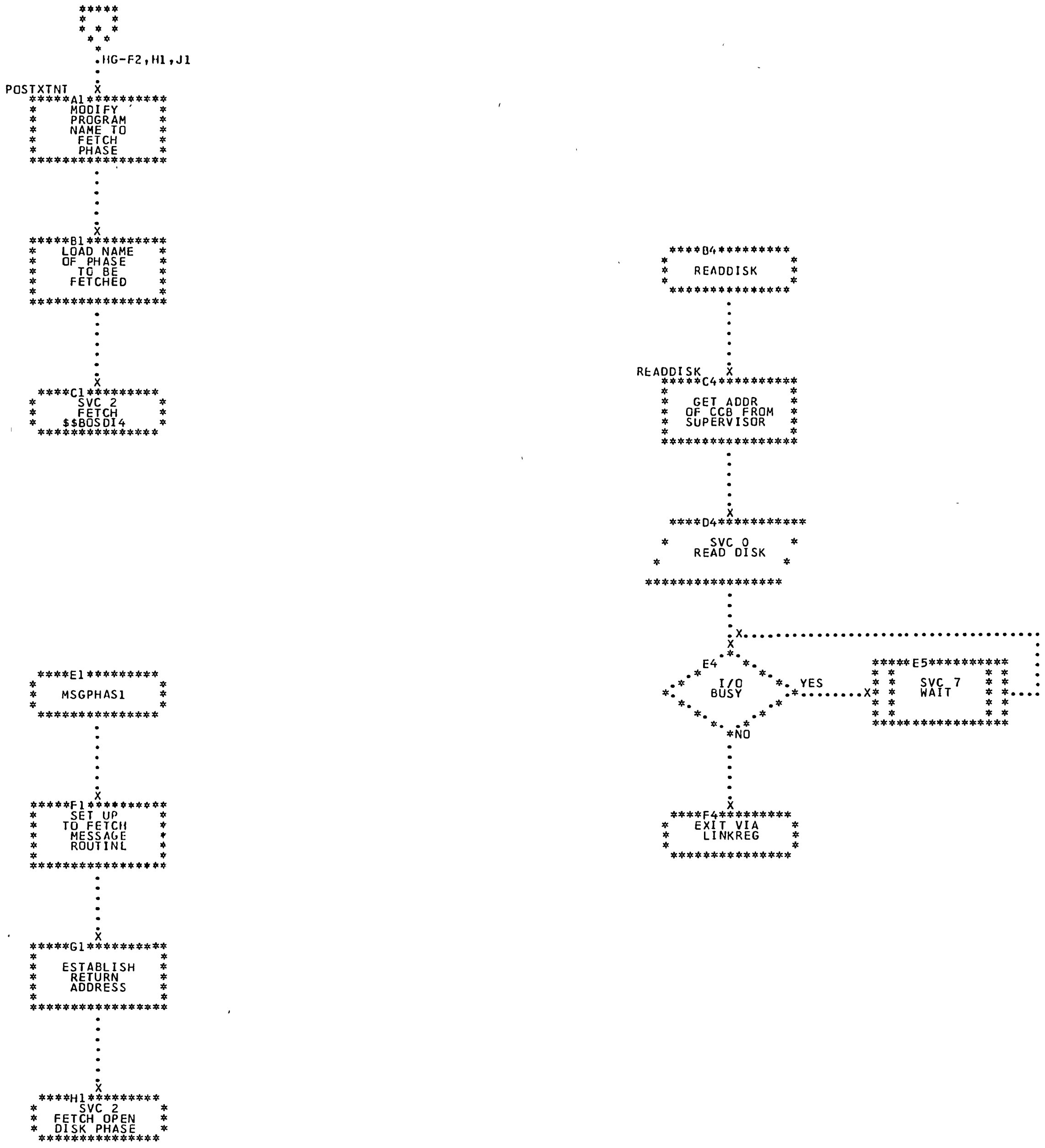
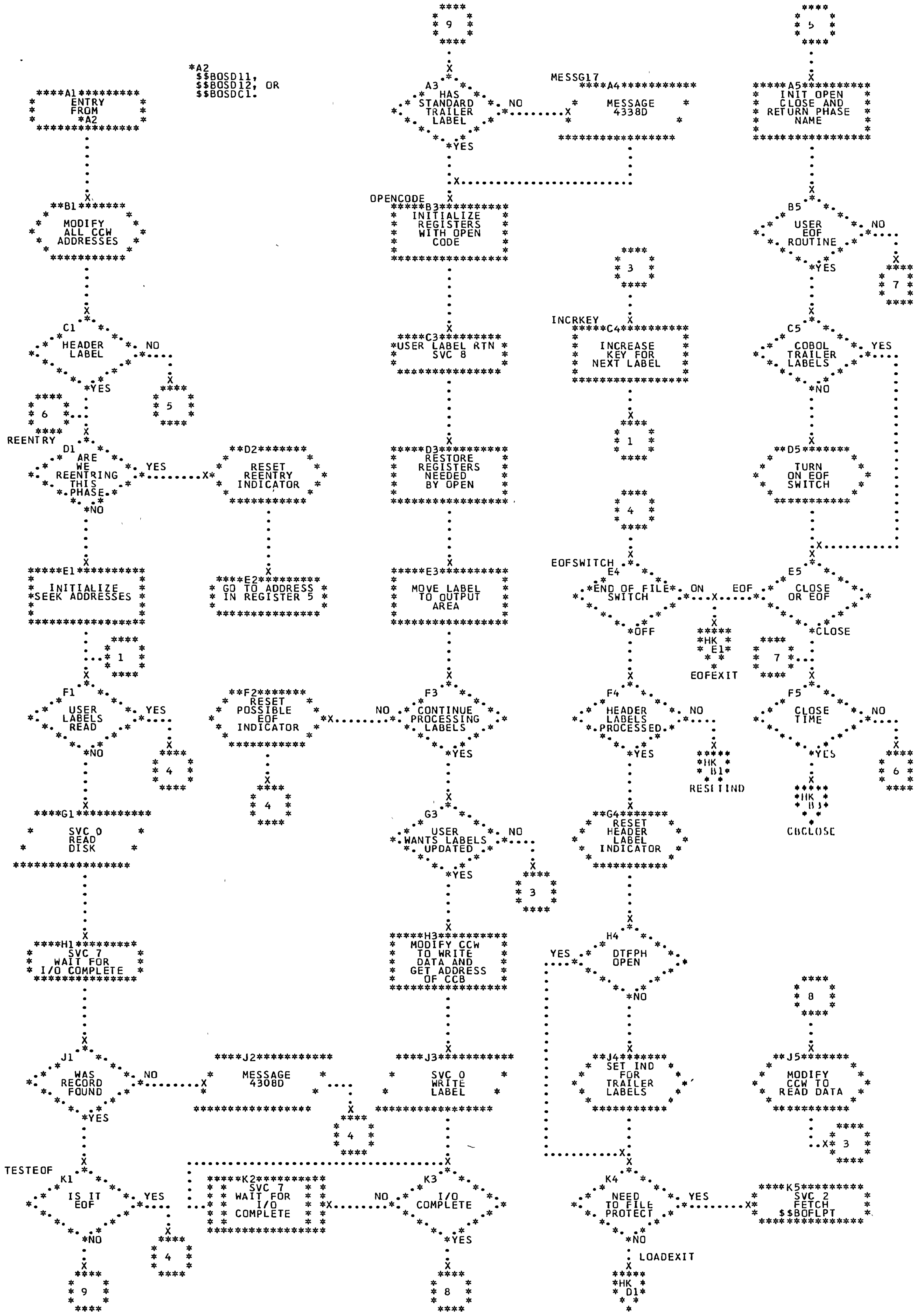


Chart HJ. $$BOSDI3: SD Open Input, User Labels (Section 1 of 2)


A1 ENTRY FROM *A2
*A2 $$BOSD11, $$BOSD12, OR $$BOSDC1.
B1 MODIFY ALL CCW ADDRESSES
C1 HEADER LABEL
NO
YES
5
6
REENTRY
D1 ARE WE REENTRING THIS PHASE.
YES
NO
D2 RESET REENTRY INDICATOR
E1 INITIALIZE SEEK ADDRESSES
1
F1 USER LABELS READ
YES
NO
4
G1 SVC 0 READ DISK
H1 SVC 7 WAIT FOR I/O COMPLETE
J1 WAS RECORD FOUND
NO
YES
J2 MESSAGE 4308D
4
TESTEOF
K1 IS IT EOF
YES
NO
4
9
E2 GO TO ADDRESS IN REGISTER 5
F2 RESET POSSIBLE EOF INDICATOR
4
K2 SVC 7 WAIT FOR I/O COMPLETE
9
A3 HAS STANDARD TRAILER LABEL
NO
YES
MESSG17
A4 MESSAGE 4338D
OPENCODE
B3 INITIALIZE REGISTERS WITH OPEN CODE
C3 USER LABEL RTN SVC 8
D3 RESTORE REGISTERS NEEDED BY OPEN
E3 MOVE LABEL TO OUTPUT AREA
F3 CONTINUE PROCESSING LABELS
NO
YES
G3 USER WANTS LABELS UPDATED
NO
YES
3
H3 MODIFY CCW TO WRITE DATA AND GET ADDRESS OF CCB
J3 SVC 0 WRITE LABEL
K3 I/O COMPLETE
NO
YES
8
3
INCRKEY
C4 INCREASE KEY FOR NEXT LABEL
1
4
EOFSWITCH
E4 END OF FILE SWITCH
ON
OFF
HK E1
EOFEXIT
F4 HEADER LABELS PROCESSED
NO
YES
HK B1
RESETIND
G4 RESET HEADER LABEL INDICATOR
H4 DTFPH OPEN
YES
NO
J4 SET IND FOR TRAILER LABELS
K4 NEED TO FILE PROTECT
YES
NO
LOADEXIT
HK D1
5
A5 INIT OPEN CLOSE AND RETURN PHASE NAME
B5 USER EOF ROUTINE
NO
YES
7
C5 COBOL TRAILER LABELS
YES
NO
D5 TURN ON EOF SWITCH
E5 CLOSE OR EOF
EOF
CLOSE
7
F5 CLOSE TIME
NO
YES
6
HK B3
CBCLOSE
8
J5 MODIFY CCW TO READ DATA
3
K5 SVC 2 FETCH $$BOFLPT

## Chart HK.  $$BOSDI3:  SD Open Input, User Labels (Section 2 of 2)

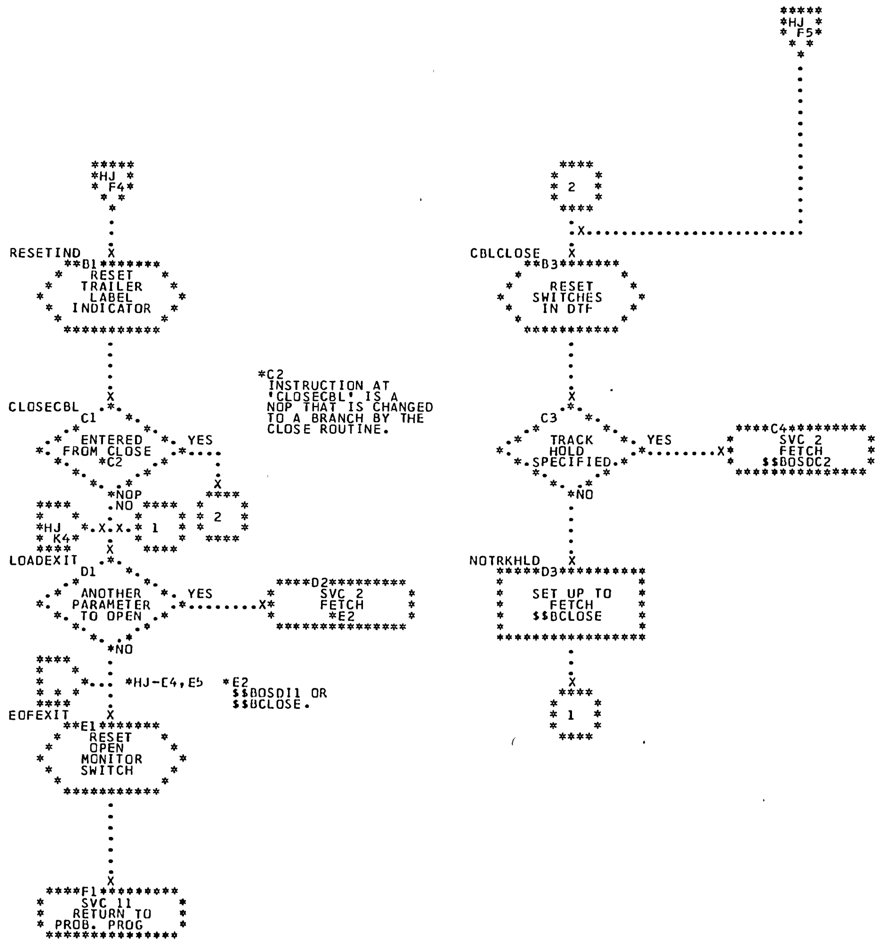


Chart HL. $$BOSDI4: SD Open Input, Initialization of DTF Table (Section 1 of 2)

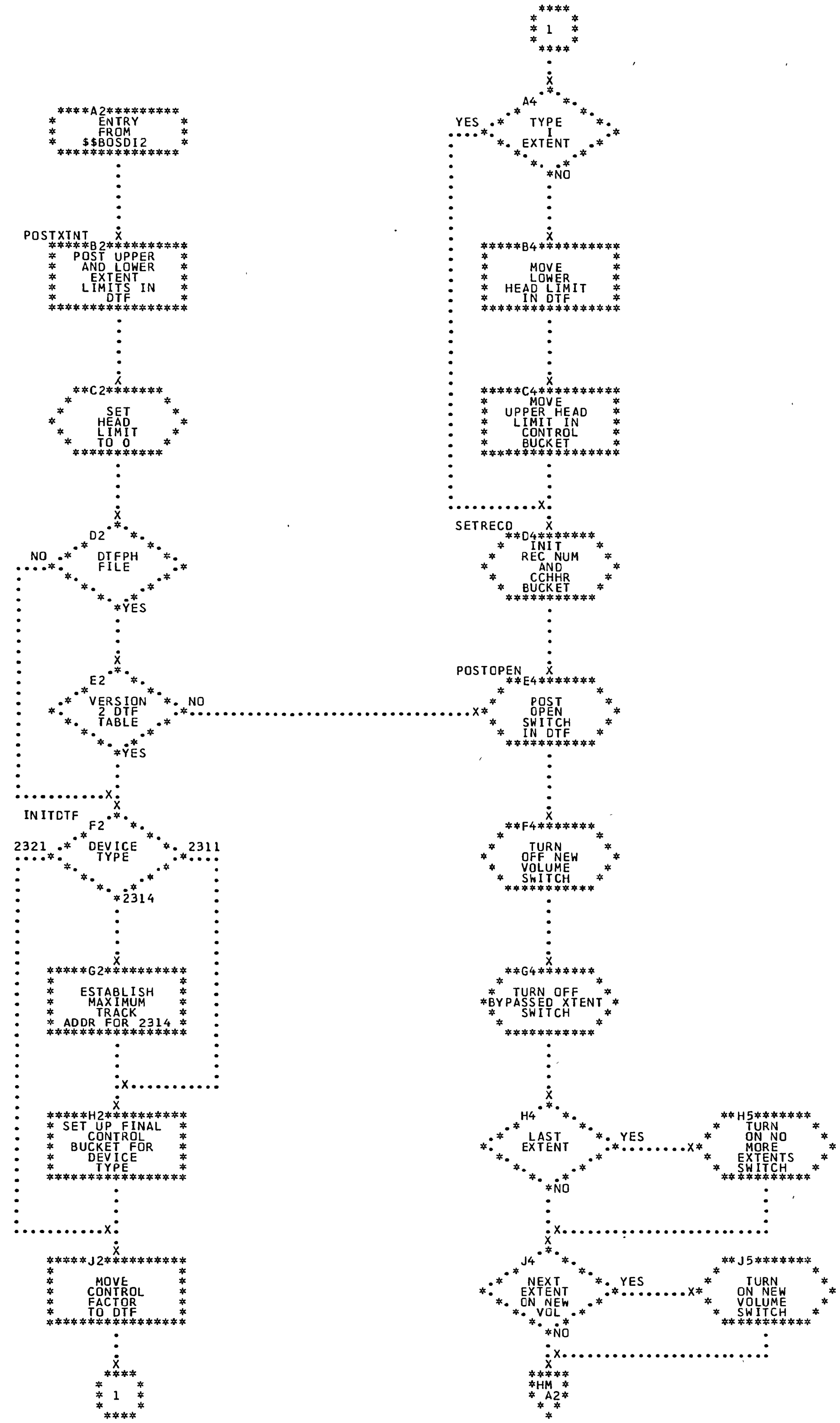

A2
ENTRY FROM $$BOSDI2
POSTXTNT
B2
POST UPPER AND LOWER EXTENT LIMITS IN DTF
C2
SET HEAD LIMIT TO 0
D2
DTFPH FILE
NO
YES
E2
VERSION 2 DTF TABLE
NO
YES
INITDTF
F2
DEVICE TYPE
2321
2311
2314
G2
ESTABLISH MAXIMUM TRACK ADDR FOR 2314
H2
SET UP FINAL CONTROL BUCKET FOR DEVICE TYPE
J2
MOVE CONTROL FACTOR TO DTF
1
1
A4
TYPE I EXTENT
YES
NO
B4
MOVE LOWER HEAD LIMIT IN DTF
C4
MOVE UPPER HEAD LIMIT IN CONTROL BUCKET
SETRECD
D4
INIT REC NUM AND CCHHR BUCKET
POSTOPEN
E4
POST OPEN SWITCH IN DTF
F4
TURN OFF NEW VOLUME SWITCH
G4
TURN OFF BYPASSED XTENT SWITCH
H4
LAST EXTENT
YES
NO
H5
TURN ON NO MORE EXTENTS SWITCH
J4
NEXT EXTENT ON NEW VOL
YES
NO
J5
TURN ON NEW VOLUME SWITCH
HM A2

Chart HM. $$BOSDI4: SD Open Input, Initialization of DTF Table (Section 2 of 2)

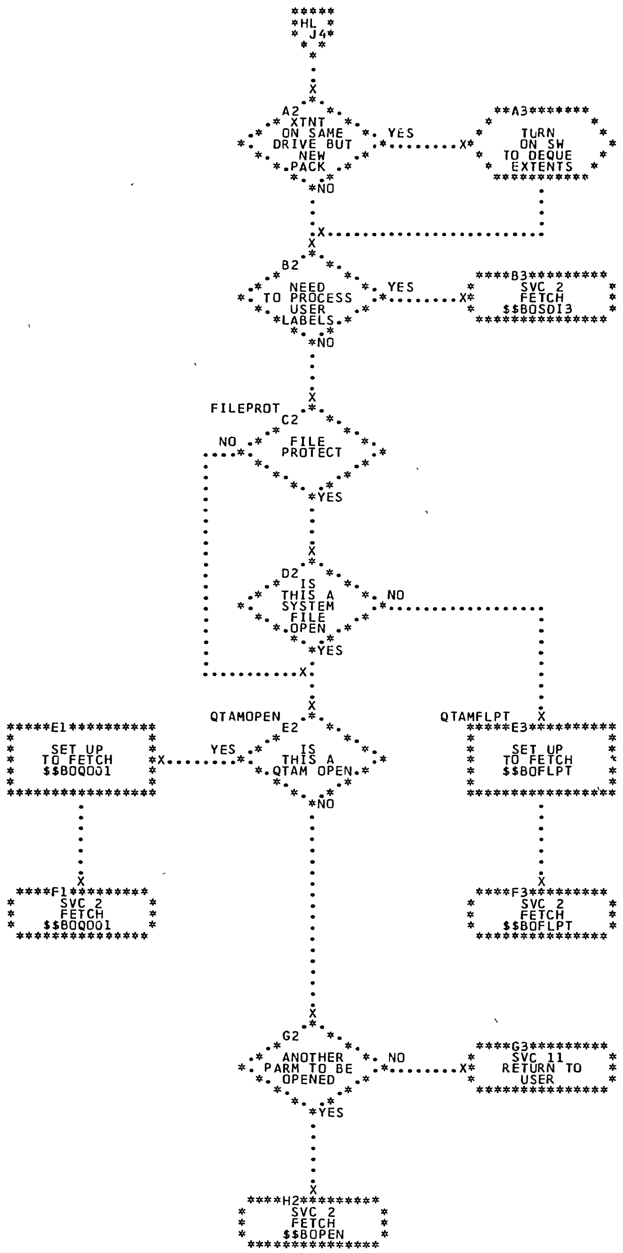


Chart JA. $$BOSD01: SD Open Output, Control (Section 1 of 4)

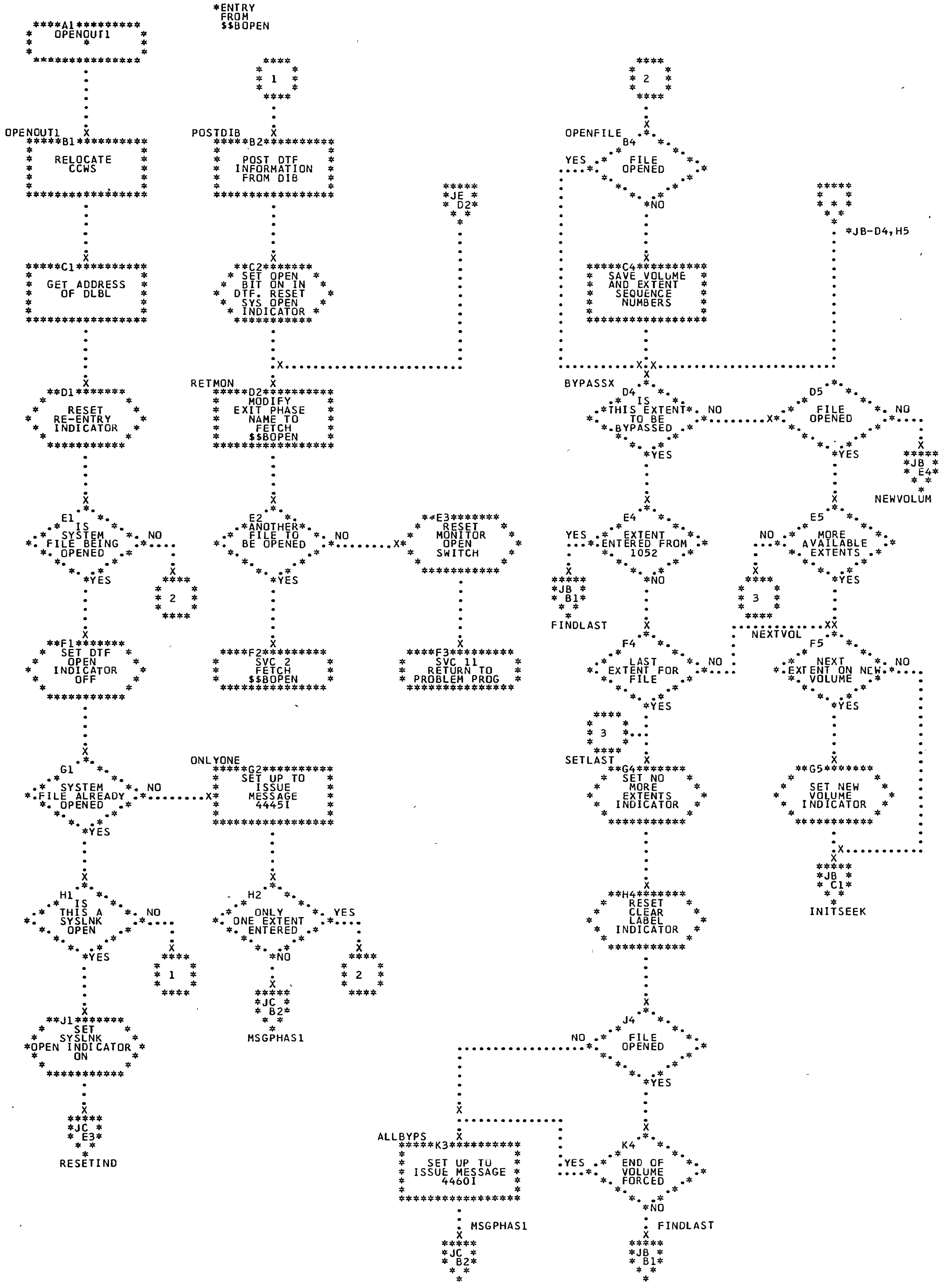

A1 OPENOUT1
*ENTRY FROM $$BOPEN
1
2
OPENOUT1 B1 RELOCATE CCWS
POSTDIB B2 POST DTF INFORMATION FROM DIB
JE D2
OPENFILE B4 FILE OPENED
YES
NO
JB-D4,H5
C1 GET ADDRESS OF DLBL
C2 SET OPEN BIT ON IN DTF. RESET SYS OPEN INDICATOR
C4 SAVE VOLUME AND EXTENT SEQUENCE NUMBERS
D1 RESET RE-ENTRY INDICATOR
RETMON D2 MODIFY EXIT PHASE NAME TO FETCH $$BOPEN
BYPASSX D4 IS THIS EXTENT TO BE BYPASSED
D5 FILE OPENED
JB E4 NEWVOLUM
E1 IS SYSTEM FILE BEING OPENED
E2 ANOTHER FILE TO BE OPENED
E3 RESET MONITOR OPEN SWITCH
E4 EXTENT ENTERED FROM 1052
E5 MORE AVAILABLE EXTENTS
JB B1 FINDLAST
3
NEXTVOL
F1 SET DTF OPEN INDICATOR OFF
F2 SVC 2 FETCH $$BOPEN
F3 SVC 11 RETURN TO PROBLEM PROG
F4 LAST EXTENT FOR FILE
F5 NEXT EXTENT ON NEW VOLUME
SETLAST G4 SET NO MORE EXTENTS INDICATOR
G1 SYSTEM FILE ALREADY OPENED
ONLYONE G2 SET UP TO ISSUE MESSAGE 4445I
G5 SET NEW VOLUME INDICATOR
JB C1 INITSEEK
H1 IS THIS A SYSLNK OPEN
H2 ONLY ONE EXTENT ENTERED
H4 RESET CLEAR LABEL INDICATOR
JC B2 MSGPHAS1
J1 SET SYSLNK OPEN INDICATOR ON
J4 FILE OPENED
JC E3 RESETIND
ALLBYPS K3 SET UP TO ISSUE MESSAGE 4460I
K4 END OF VOLUME FORCED
MSGPHAS1
JC B2
FINDLAST
JB B1

Chart JB. $$BOSD01: SD Open Output, Control (Section 2 of 4)

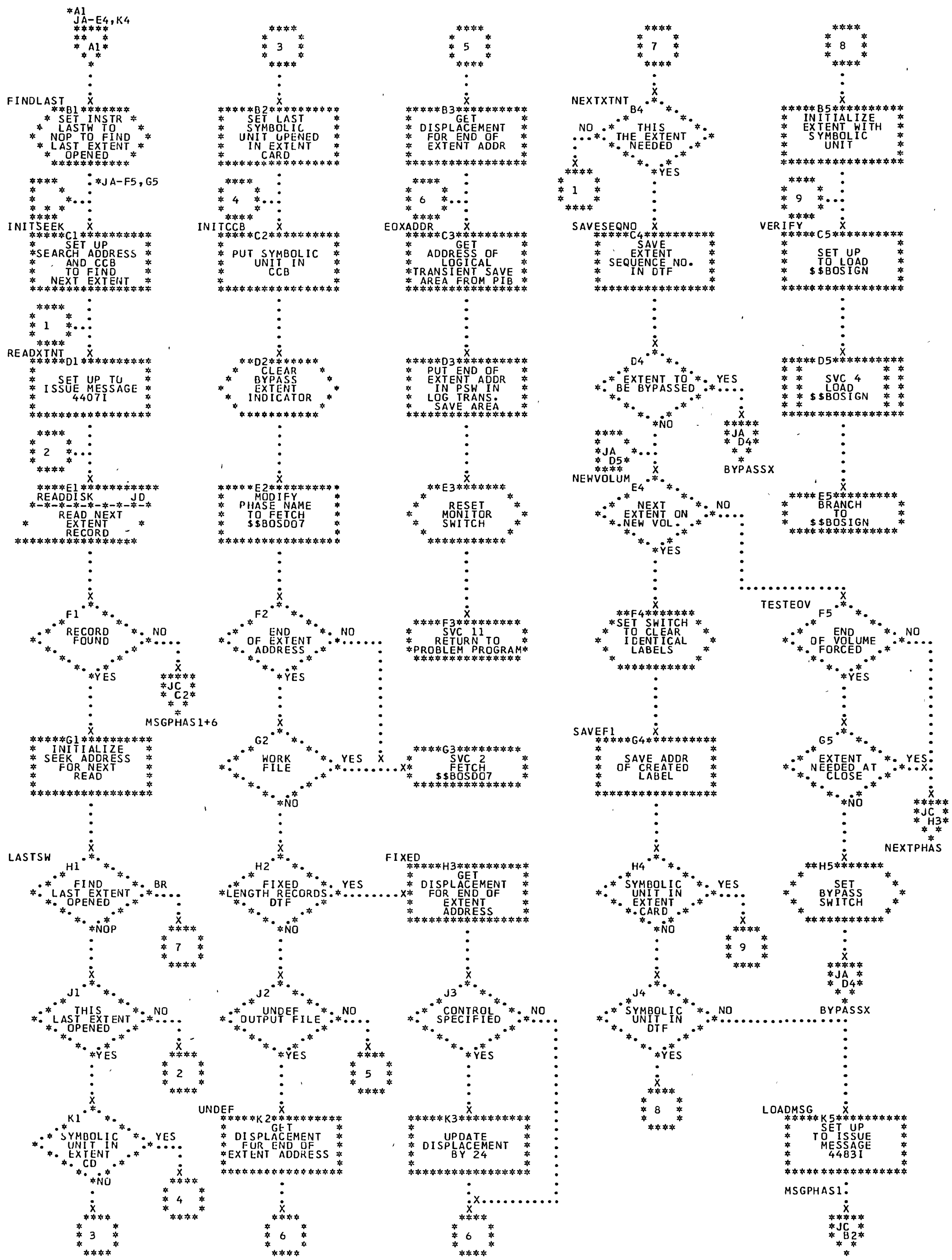
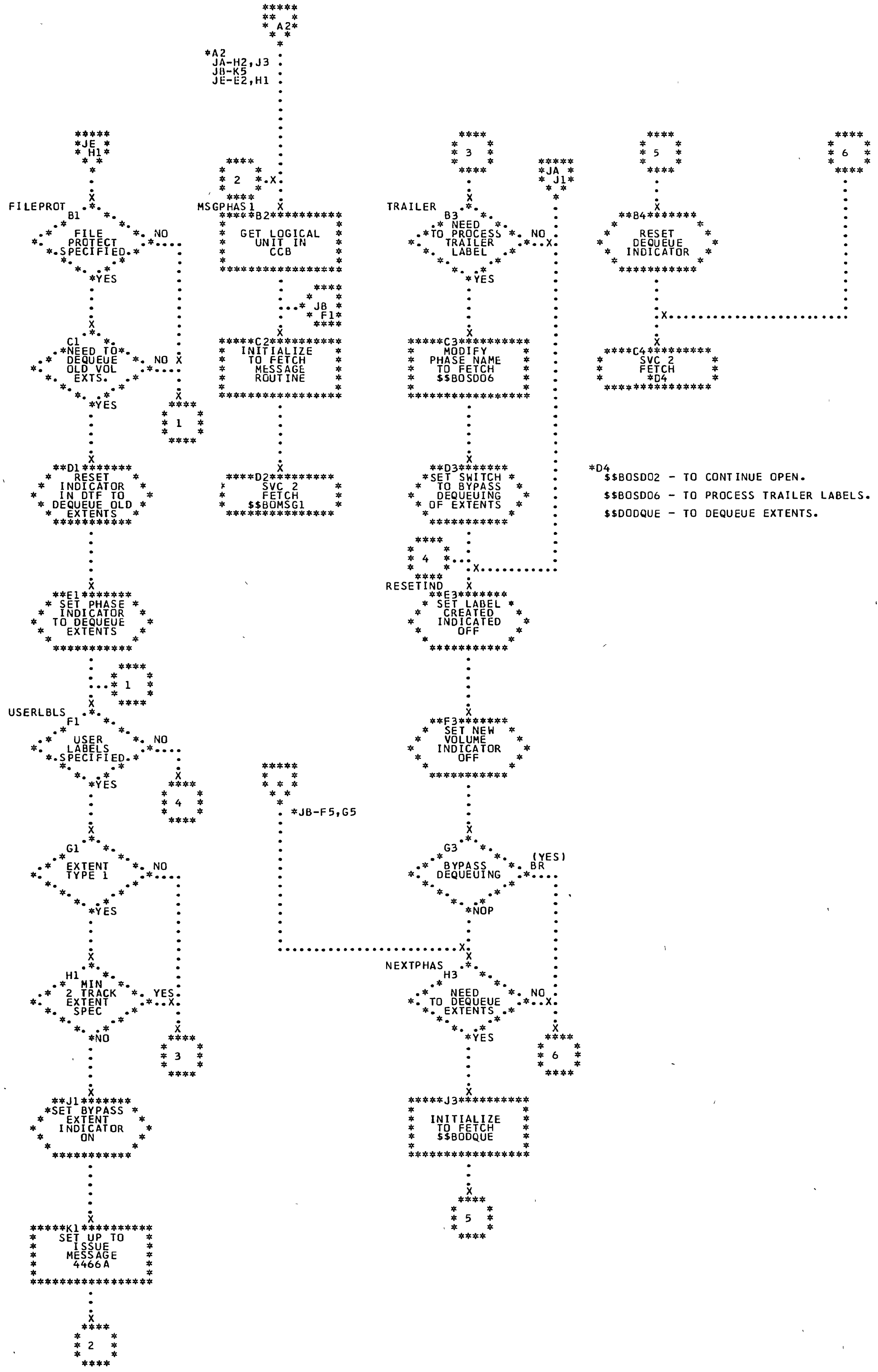
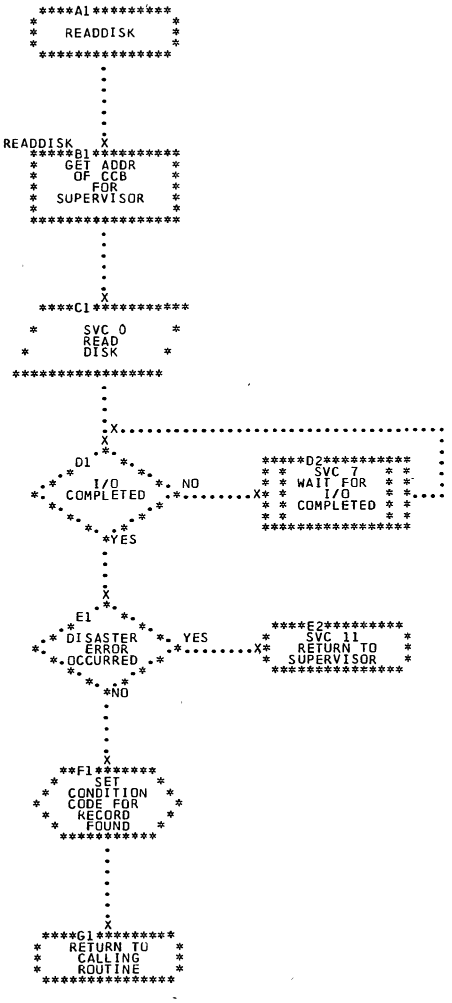


Chart JC. $$BOSD01: SD Open Output, Control (Section 3 of 4)


A2
*A2 JA-H2,J3 JB-K5 JE-E2,H1
JE H1
FILEPROT B1
FILE PROTECT SPECIFIED
NO
YES
C1
NEED TO DEQUEUE OLD VOL EXTS.
NO
YES
1
D1
RESET INDICATOR IN DTF TO DEQUEUE OLD EXTENTS
E1
SET PHASE INDICATOR TO DEQUEUE EXTENTS
1
USERLBLS F1
USER LABELS SPECIFIED
NO
YES
4
G1
EXTENT TYPE 1
NO
YES
H1
MIN 2 TRACK EXTENT SPEC
YES
NO
3
J1
SET BYPASS EXTENT INDICATOR ON
K1
SET UP TO ISSUE MESSAGE 4466A
2
2
MSGPHAS1 B2
GET LOGICAL UNIT IN CCB
JB F1
C2
INITIALIZE TO FETCH MESSAGE ROUTINE
D2
SVC 2 FETCH $$BOMSG1
*JB-F5,G5
3
TRAILER B3
NEED TO PROCESS TRAILER LABEL
NO
YES
JA J1
C3
MODIFY PHASE NAME TO FETCH $$BOSD06
D3
SET SWITCH TO BYPASS DEQUEUING OF EXTENTS
4
RESETIND E3
SET LABEL CREATED INDICATED OFF
F3
SET NEW VOLUME INDICATOR OFF
G3
BYPASS DEQUEUING
(YES) BR
NOP
NEXTPHAS H3
NEED TO DEQUEUE EXTENTS
NO
YES
6
J3
INITIALIZE TO FETCH $$BODQUE
5
5
B4
RESET DEQUEUE INDICATOR
6
C4
SVC 2 FETCH *D4

*D4
$$BOSD02 - TO CONTINUE OPEN.
$$BOSD06 - TO PROCESS TRAILER LABELS.
$$DODQUE - TO DEQUEUE EXTENTS.

Chart JD. $$BOSD01: SD Open Output, Control (Section 4 of 4)


A1
READDISK
READDISK
B1
GET ADDR OF CCB FOR SUPERVISOR
C1
SVC 0 READ DISK
D1
I/O COMPLETED
NO
YES
D2
SVC 7 WAIT FOR I/O COMPLETED
E1
DISASTER ERROR OCCURRED
YES
NO
E2
SVC 11 RETURN TO SUPERVISOR
F1
SET CONDITION CODE FOR RECORD FOUND
G1
RETURN TO CALLING ROUTINE

## Chart JE. $$BOSIGN: SD Open Ignore

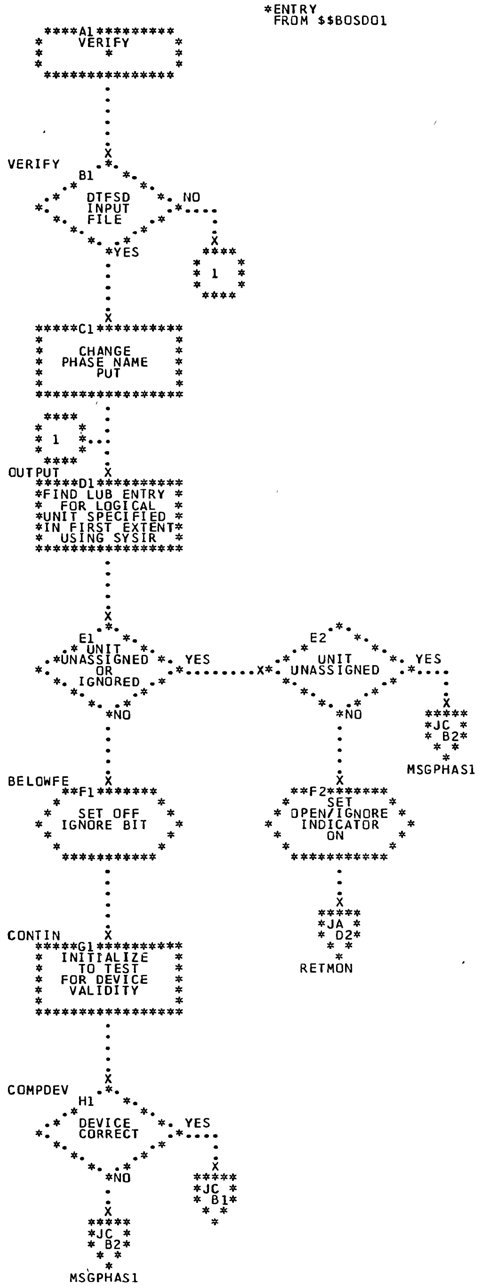


Chart JF. $$BOSDO2: SD Open Output, Volume Label

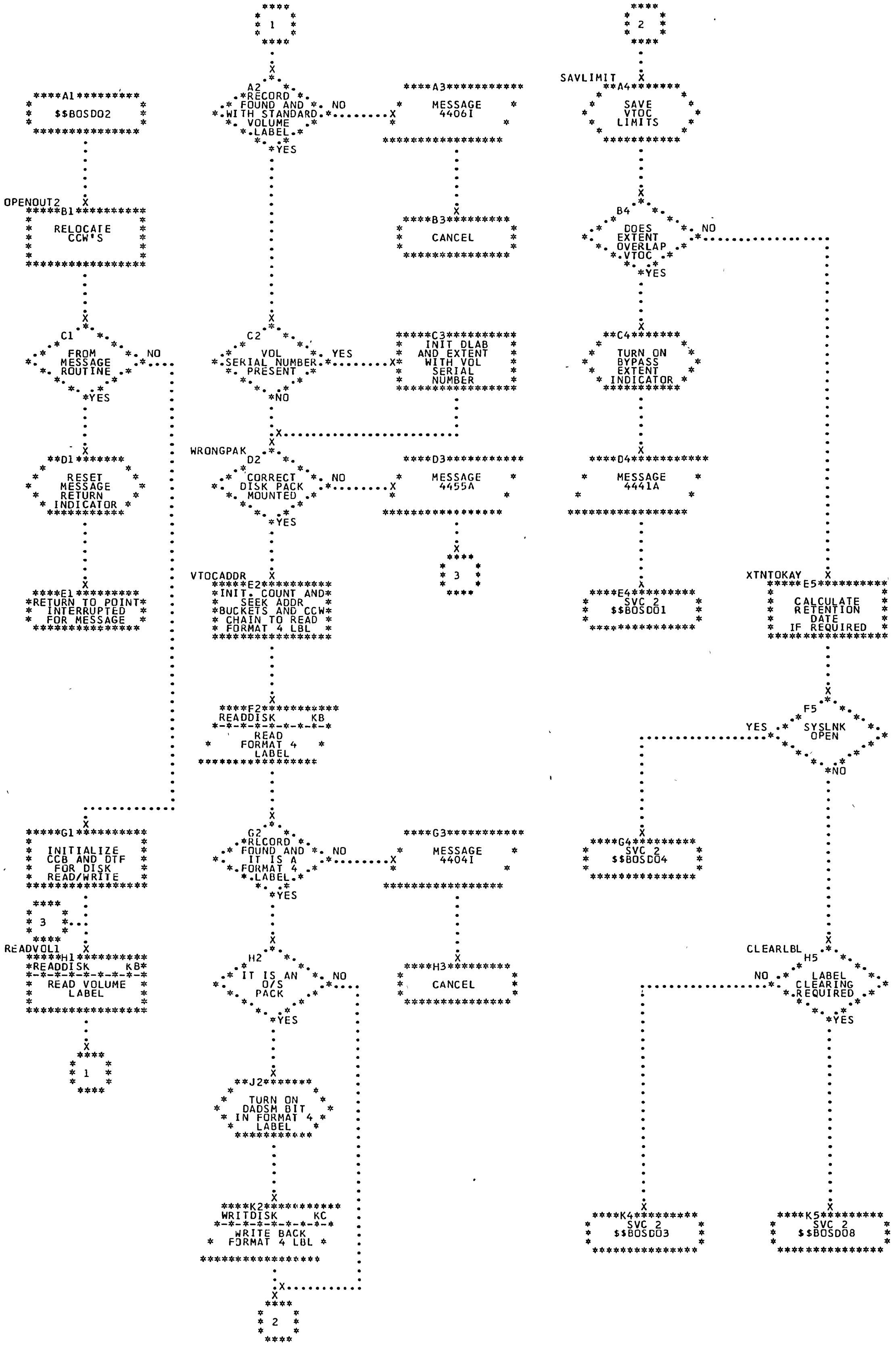


Chart JG. $$BOSDO3: SD Open Output, Extent Overlap (Section 1 of 4)

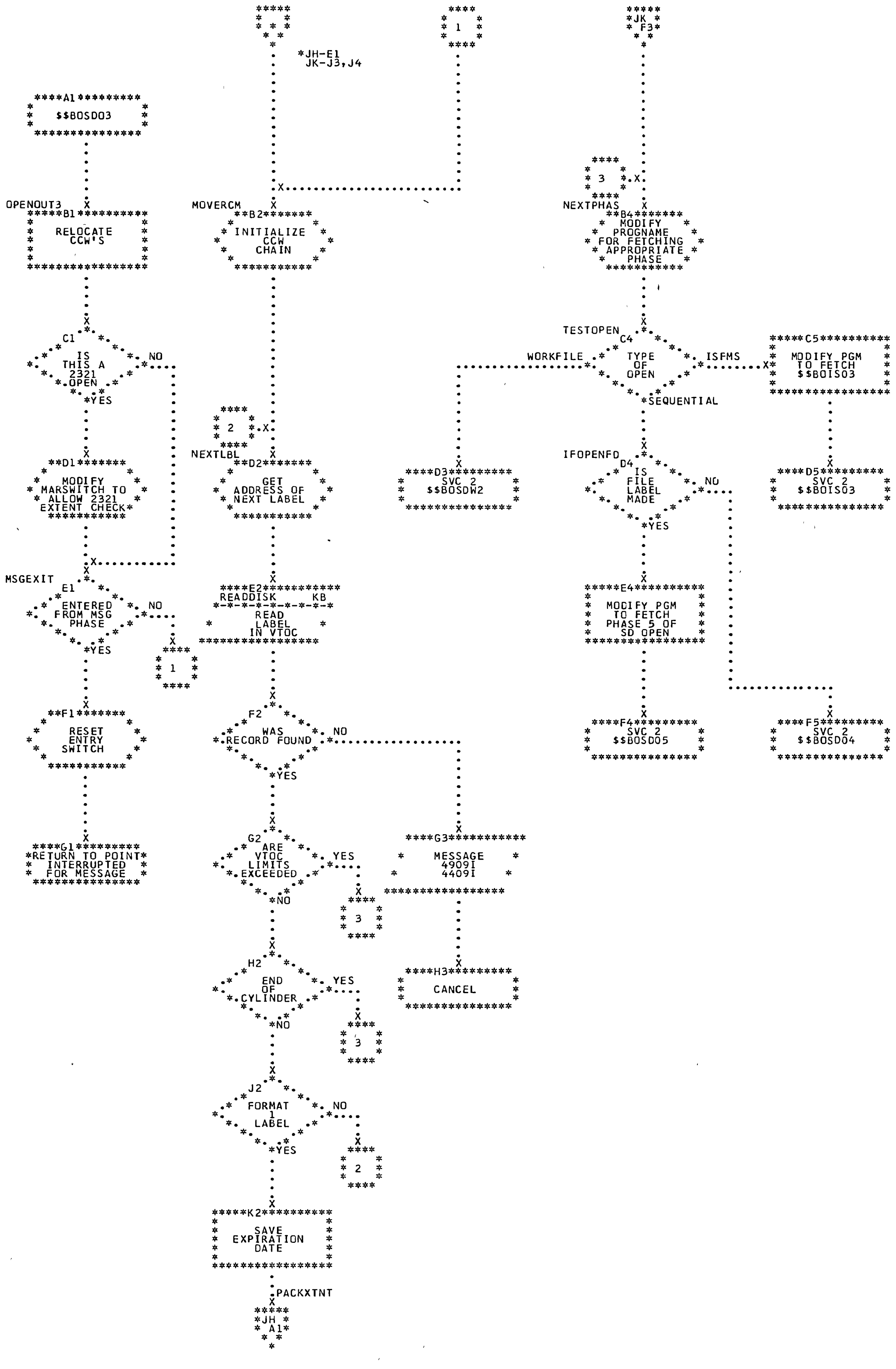


Chart JH. $$BOSD03: SD Open Output, Extent Overlap (Section 2 of 4)

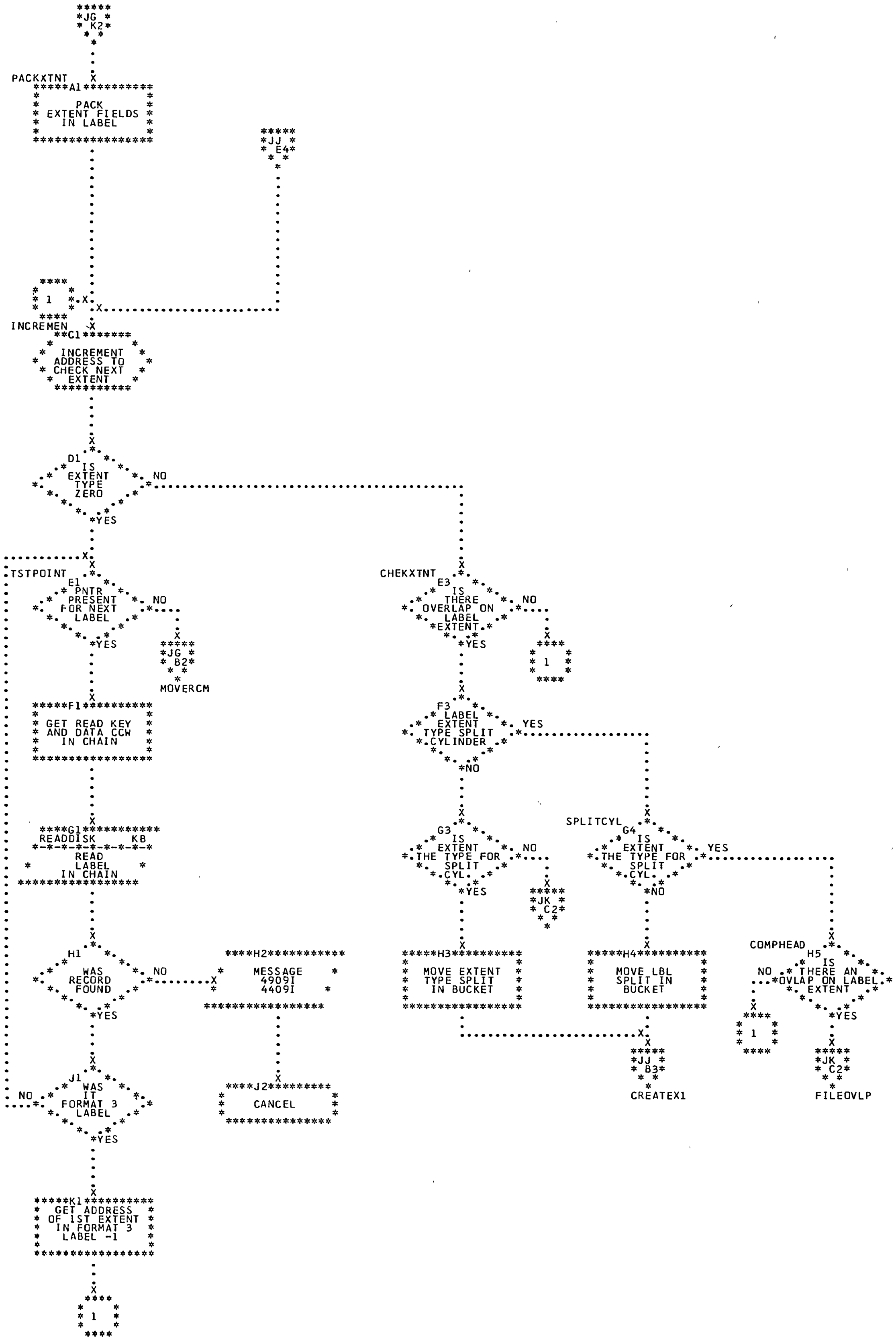

JG K2
PACKXTNT
A1
PACK EXTENT FIELDS IN LABEL
JJ E4
1
INCREMEN
C1
INCREMENT ADDRESS TO CHECK NEXT EXTENT
D1
IS EXTENT TYPE ZERO
NO
YES
TSTPOINT
E1
PNTR PRESENT FOR NEXT LABEL
NO
YES
JG B2
MOVERCM
F1
GET READ KEY AND DATA CCW IN CHAIN
G1
READDISK KB
READ LABEL IN CHAIN
H1
WAS RECORD FOUND
NO
YES
H2
MESSAGE 4909I 4409I
J1
WAS IT FORMAT 3 LABEL
NO
YES
J2
CANCEL
K1
GET ADDRESS OF 1ST EXTENT IN FORMAT 3 LABEL -1
1
CHEKXTNT
E3
IS THERE OVERLAP ON LABEL EXTENT
NO
YES
1
F3
LABEL EXTENT TYPE SPLIT CYLINDER
YES
NO
G3
IS EXTENT THE TYPE FOR SPLIT CYL.
NO
YES
JK C2
SPLITCYL
G4
IS EXTENT THE TYPE FOR SPLIT CYL.
YES
NO
H3
MOVE EXTENT TYPE SPLIT IN BUCKET
H4
MOVE LBL SPLIT IN BUCKET
COMPHEAD
H5
IS THERE AN OVLAP ON LABEL EXTENT
NO
YES
1
JJ B3
CREATEX1
JK C2
FILEOVLP

Chart JJ. $$BOSDO3: SD Open Output, Extent Overlap (Section 3 of 4)

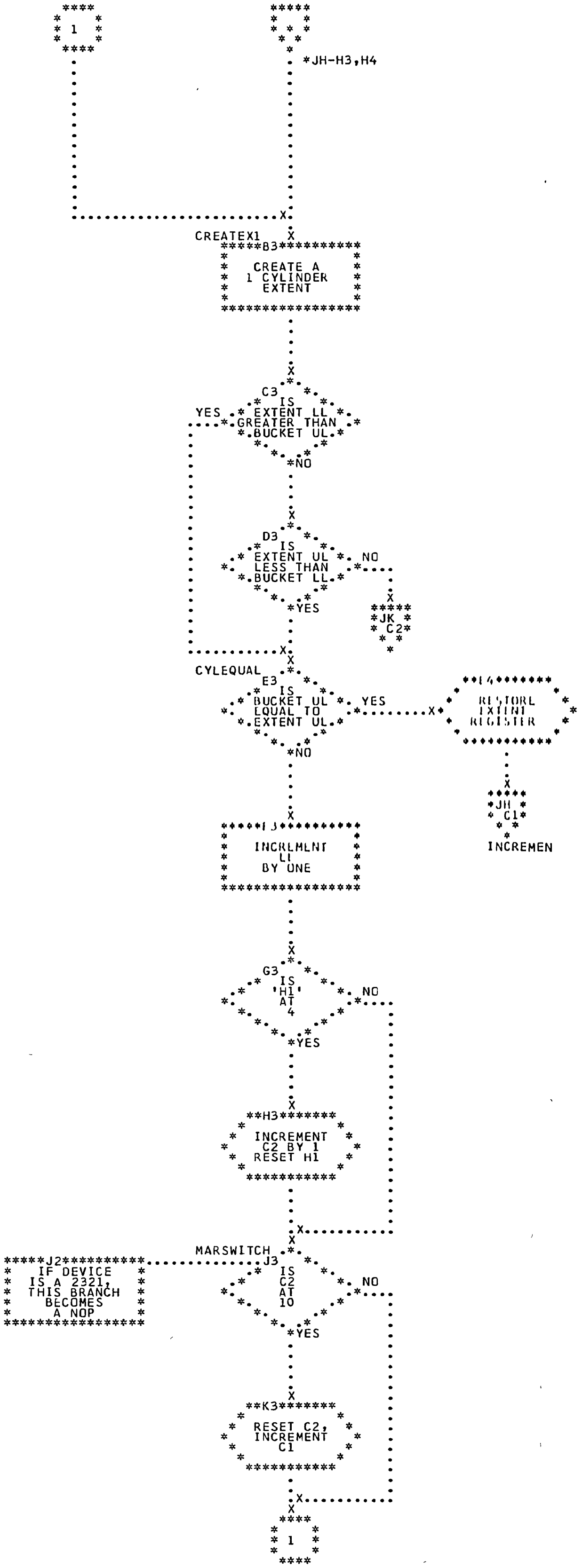

1
JH-H3,H4
CREATEX1
B3
CREATE A 1 CYLINDER EXTENT
C3
IS EXTENT LL GREATER THAN BUCKET UL
YES
NO
D3
IS EXTENT UL LESS THAN BUCKET LL
NO
YES
JK C2
CYLEQUAL
E3
IS BUCKET UL EQUAL TO EXTENT UL
YES
NO
E4
RESTORE EXTENT REGISTER
JH C1
INCREMEN
F3
INCREMENT LL BY ONE
G3
IS 'H1' AT 4
NO
YES
H3
INCREMENT C2 BY 1 RESET H1
MARSWITCH
J2
IF DEVICE IS A 2321, THIS BRANCH BECOMES A NOP
J3
IS C2 AT 10
NO
YES
K3
RESET C2, INCREMENT C1
1

Chart JK. $$BOSD03: SD Open Output, Extent Overlap (Section 4 of 4)

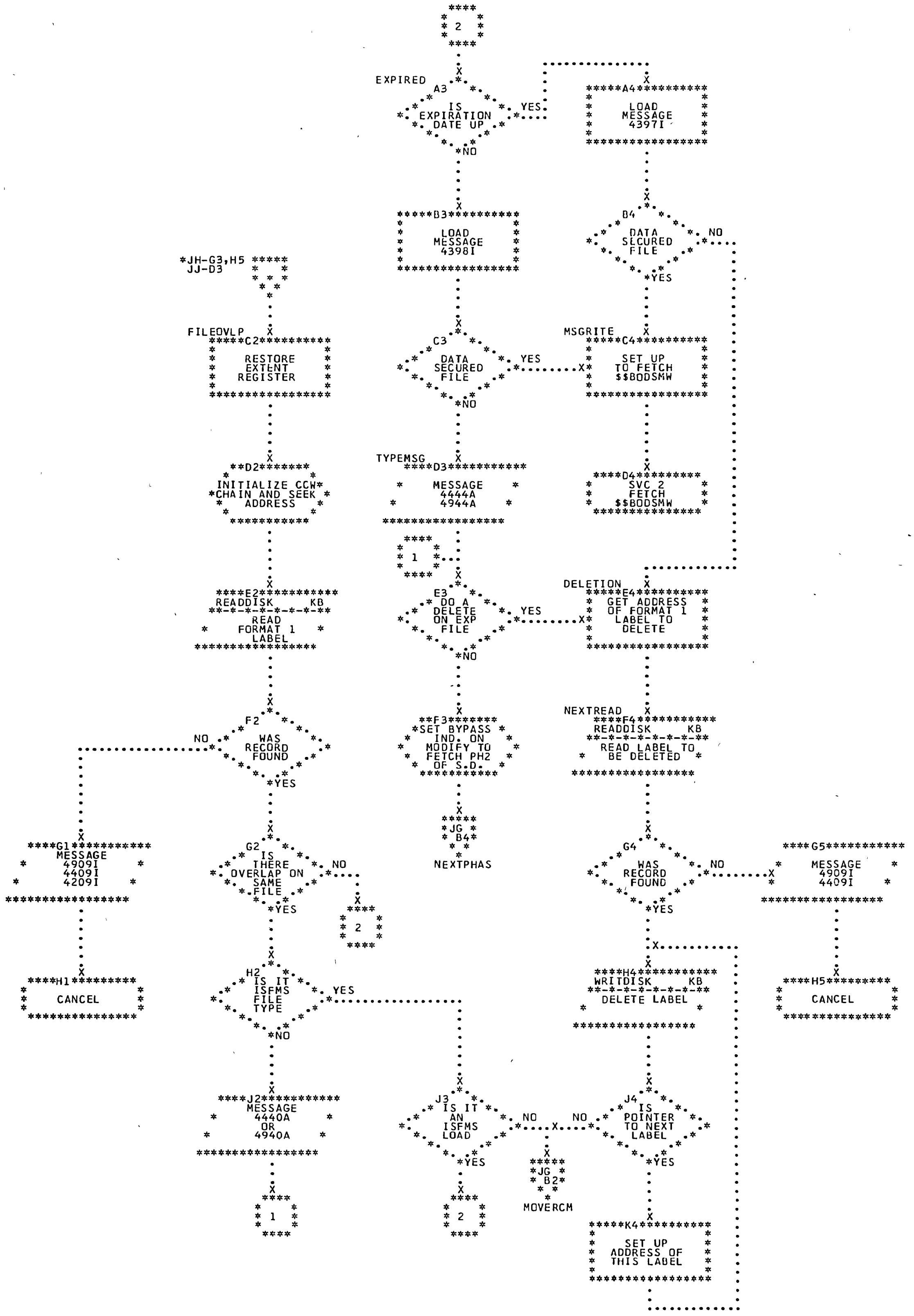


Chart JL. $$BOSDO4: SD Open Output, File Label (Section 1 of 2)

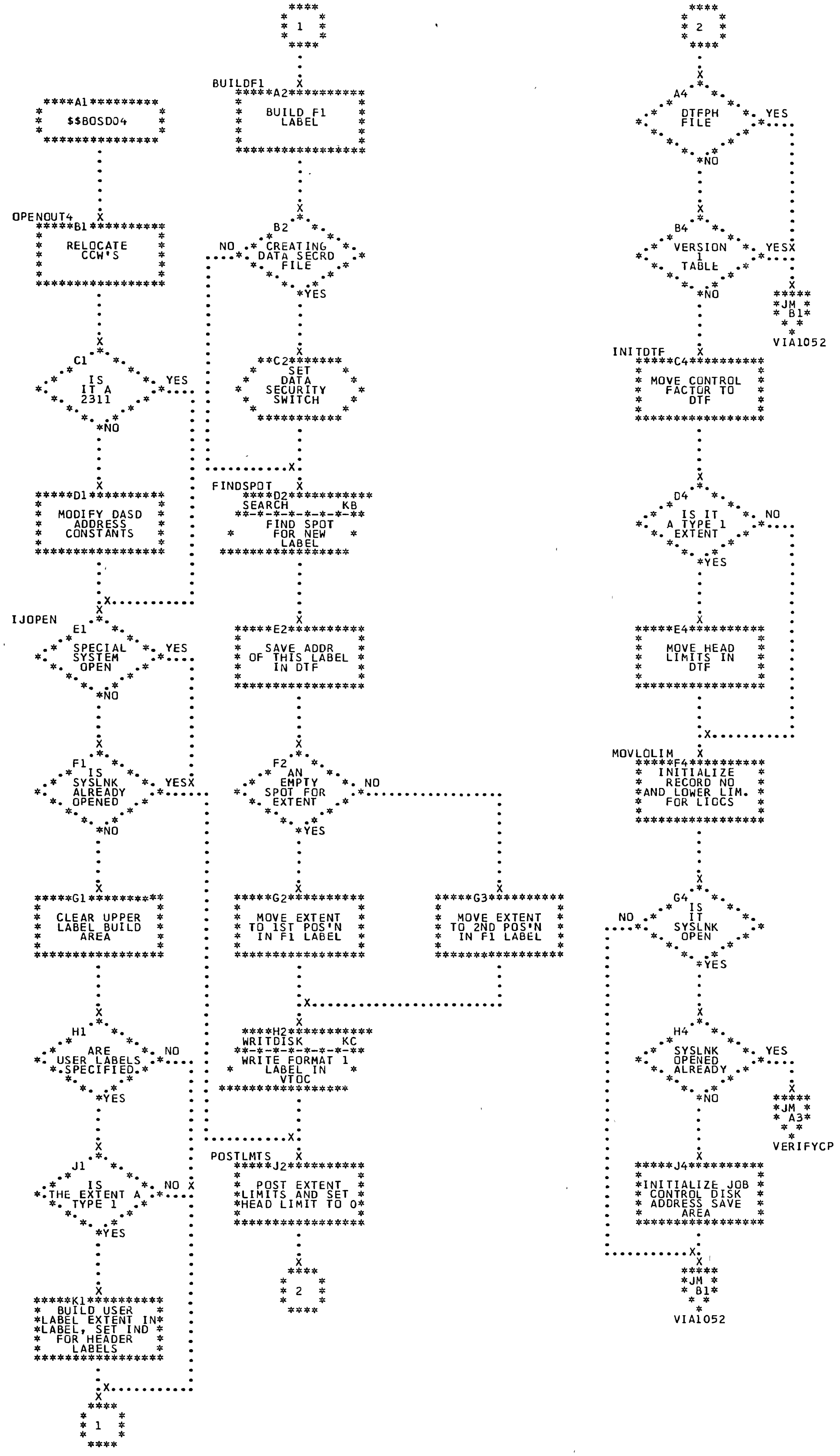

1
2
A1 $$BOSD04
BUILDF1 A2 BUILD F1 LABEL
A4 DTFPH FILE
YES
NO
OPENOUT4 B1 RELOCATE CCW'S
B2 CREATING DATA SECRD FILE
NO
YES
B4 VERSION 1 TABLE
YES
NO
JM B1
VIA1052
C1 IS IT A 2311
YES
NO
C2 SET DATA SECURITY SWITCH
INITDTF C4 MOVE CONTROL FACTOR TO DTF
D1 MODIFY DASD ADDRESS CONSTANTS
FINDSPOT D2 SEARCH KB FIND SPOT FOR NEW LABEL
D4 IS IT A TYPE 1 EXTENT
NO
YES
IJOPEN E1 SPECIAL SYSTEM OPEN
YES
NO
E2 SAVE ADDR OF THIS LABEL IN DTF
E4 MOVE HEAD LIMITS IN DTF
MOVLOLIM F4 INITIALIZE RECORD NO AND LOWER LIM. FOR LIOCS
F1 IS SYSLNK ALREADY OPENED
YES
NO
F2 AN EMPTY SPOT FOR EXTENT
NO
YES
G1 CLEAR UPPER LABEL BUILD AREA
G2 MOVE EXTENT TO 1ST POS'N IN F1 LABEL
G3 MOVE EXTENT TO 2ND POS'N IN F1 LABEL
G4 IS IT SYSLNK OPEN
NO
YES
H1 ARE USER LABELS SPECIFIED
NO
YES
H2 WRITDISK KC WRITE FORMAT 1 LABEL IN VTOC
H4 SYSLNK OPENED ALREADY
YES
NO
JM A3
VERIFYCP
POSTLMTS J2 POST EXTENT LIMITS AND SET HEAD LIMIT TO 0
J1 IS THE EXTENT A TYPE 1
NO
YES
J4 INITIALIZE JOB CONTROL DISK ADDRESS SAVE AREA
2
JM B1
VIA1052
K1 BUILD USER LABEL EXTENT IN LABEL, SET IND FOR HEADER LABELS
1

Chart JM. $$BOSD04: SD Open Output, File Label (Section 2 of 2)

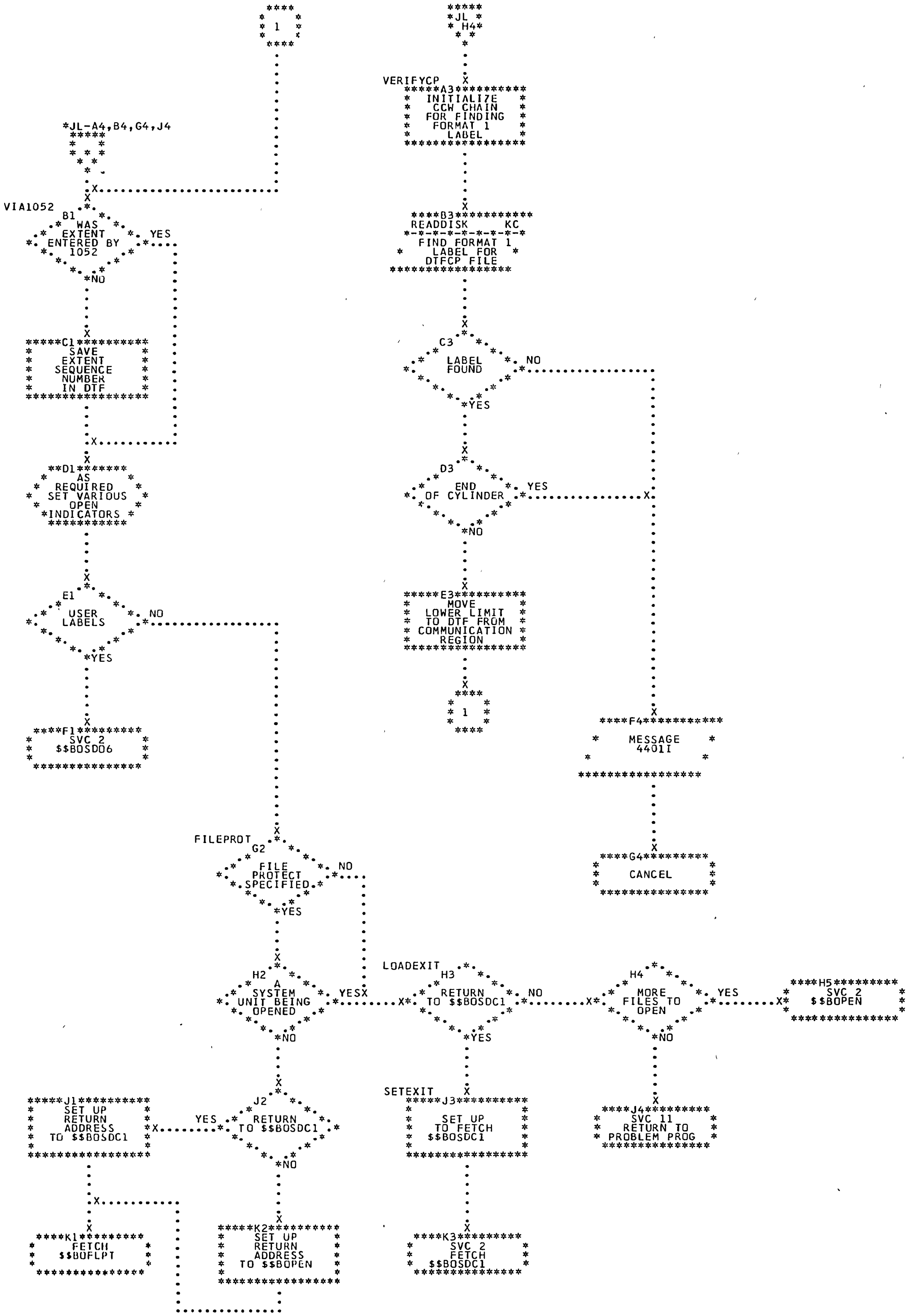


Chart JN. $$BOSDO5: SD Open Output, Format 3 Label (Section 1 of 2)

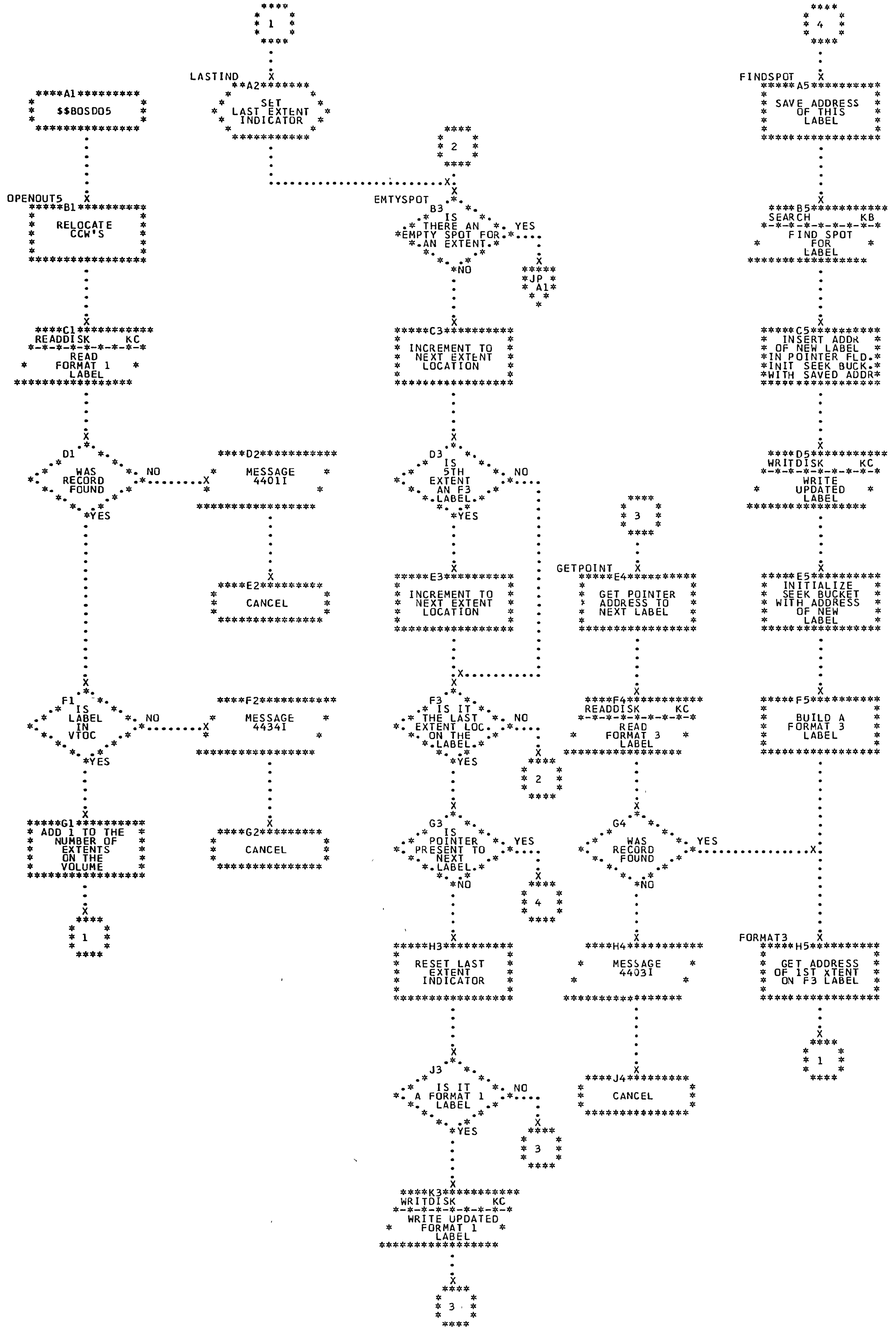
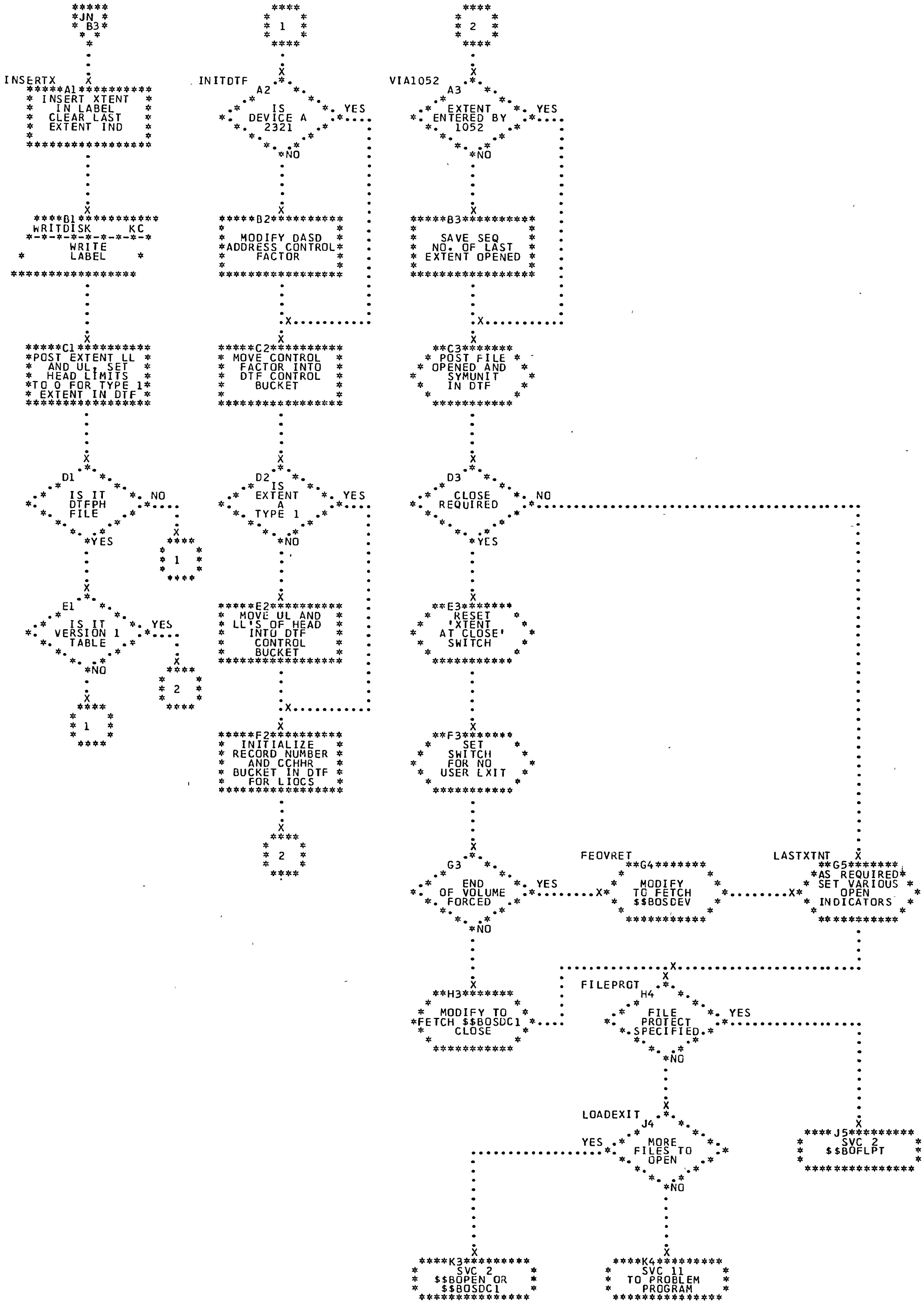
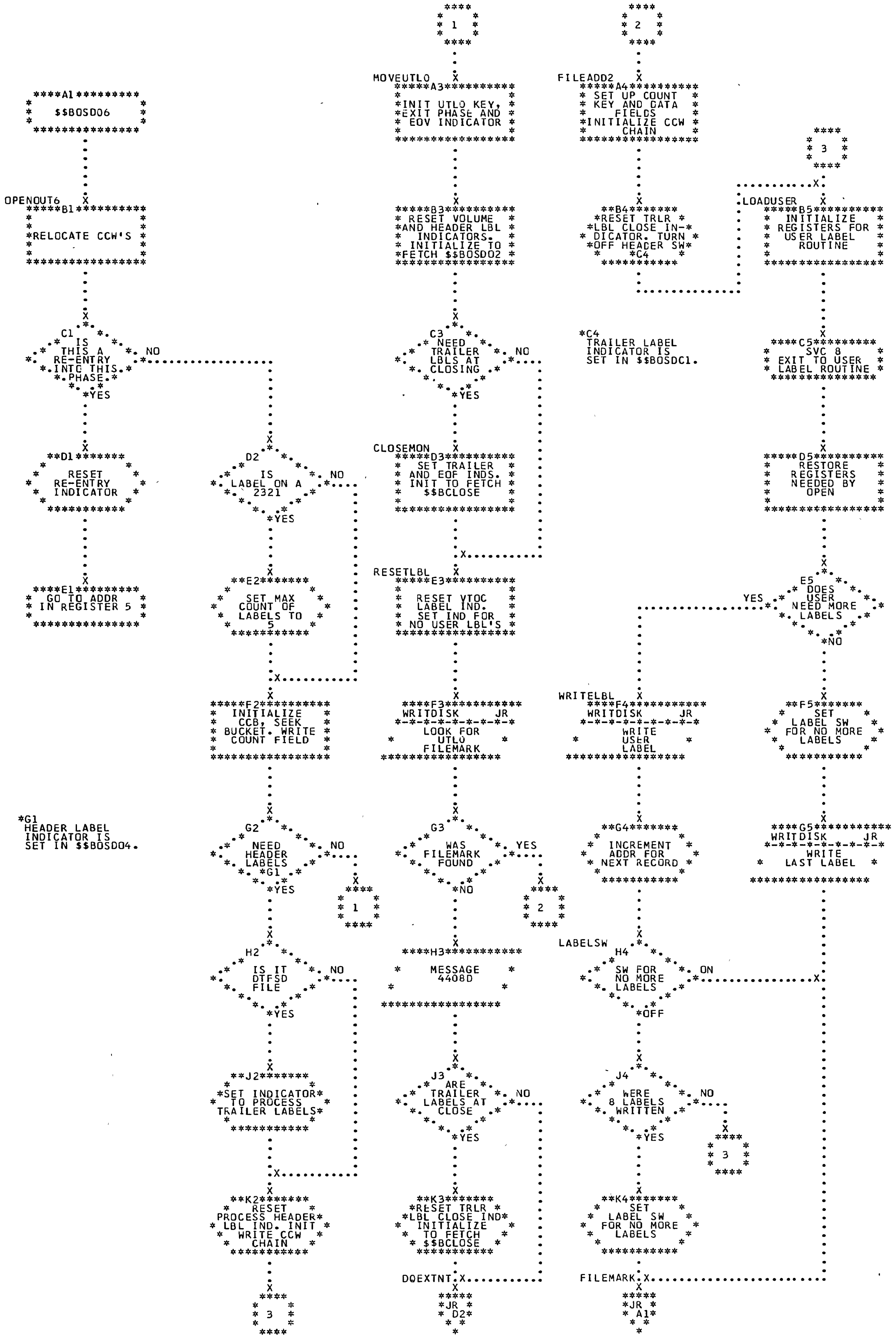

A1
$$BOSDO5
OPENOUT5
B1
RELOCATE CCW'S
C1
READDISK KC
READ FORMAT 1 LABEL
D1
WAS RECORD FOUND
NO
YES
D2
MESSAGE 4401I
E2
CANCEL
F1
IS LABEL IN VTOC
NO
YES
F2
MESSAGE 4434I
G2
CANCEL
G1
ADD 1 TO THE NUMBER OF EXTENTS ON THE VOLUME
1
1
LASTIND
A2
SET LAST EXTENT INDICATOR
2
EMTYSPOT
B3
IS THERE AN EMPTY SPOT FOR AN EXTENT
YES
NO
JP A1
C3
INCREMENT TO NEXT EXTENT LOCATION
D3
IS 5TH EXTENT AN F3 LABEL
NO
YES
E3
INCREMENT TO NEXT EXTENT LOCATION
F3
IS IT THE LAST EXTENT LOC. ON THE LABEL
NO
YES
2
G3
IS POINTER PRESENT TO NEXT LABEL
YES
NO
4
H3
RESET LAST EXTENT INDICATOR
J3
IS IT A FORMAT 1 LABEL
NO
YES
3
K3
WRITDISK KC
WRITE UPDATED FORMAT 1 LABEL
3
3
GETPOINT
E4
GET POINTER ADDRESS TO NEXT LABEL
F4
READDISK KC
READ FORMAT 3 LABEL
G4
WAS RECORD FOUND
YES
NO
H4
MESSAGE 4403I
J4
CANCEL
4
FINDSPOT
A5
SAVE ADDRESS OF THIS LABEL
B5
SEARCH KB
FIND SPOT FOR LABEL
C5
INSERT ADDR OF NEW LABEL IN POINTER FLD. INIT SEEK BUCK. WITH SAVED ADDR
D5
WRITDISK KC
WRITE UPDATED LABEL
E5
INITIALIZE SEEK BUCKET WITH ADDRESS OF NEW LABEL
F5
BUILD A FORMAT 3 LABEL
FORMAT3
H5
GET ADDRESS OF 1ST XTENT ON F3 LABEL
1

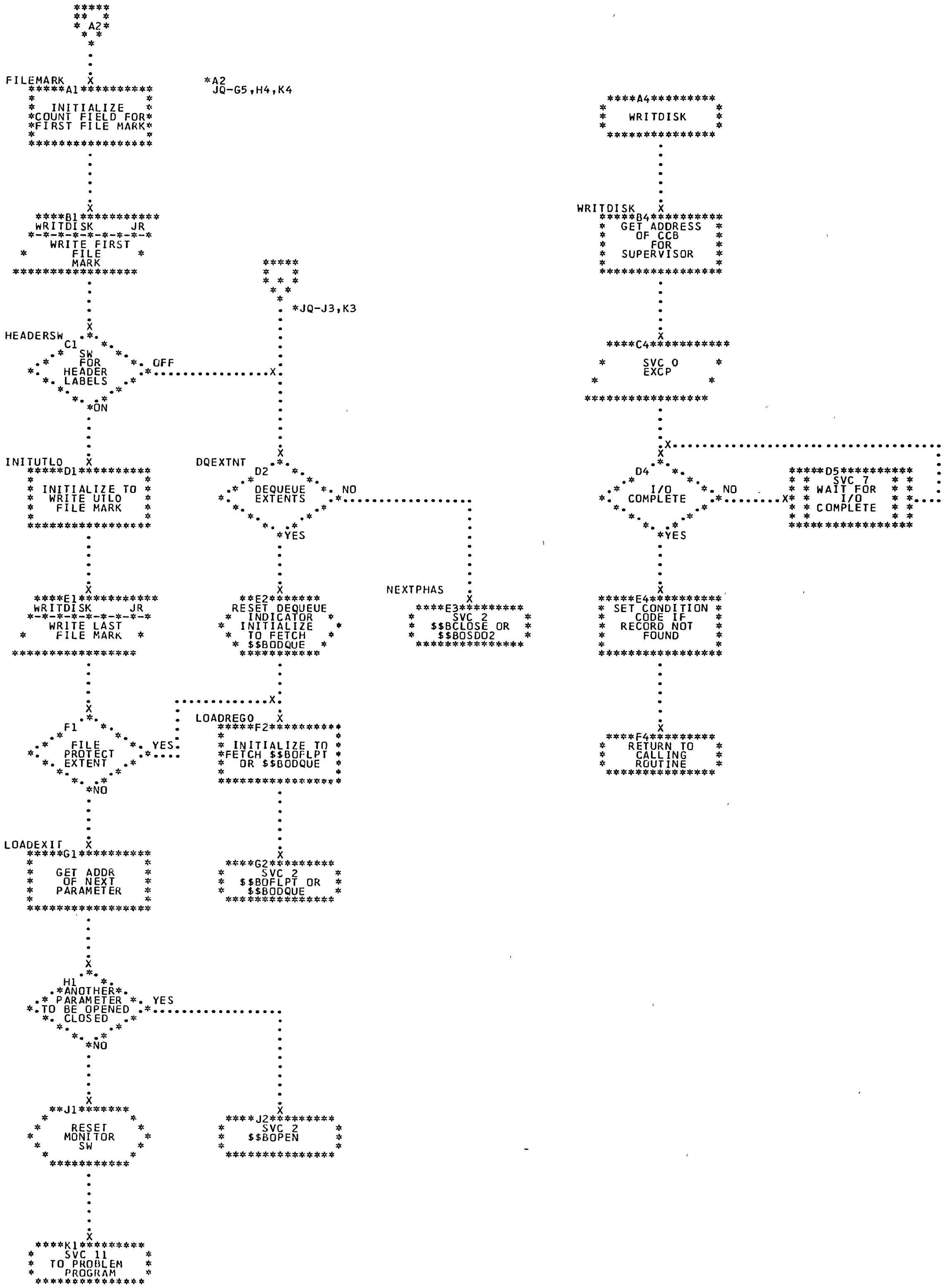
Chart JP. $$BOSD05: SD Open Output, Format 3 Label (Section 2 of 2)


JN B3
1
2
INSERTX A1
INSERT XTENT IN LABEL CLEAR LAST EXTENT IND
INITDTF A2
IS DEVICE A 2321
YES
NO
VIA1052 A3
EXTENT ENTERED BY 1052
YES
NO
B1 WRITDISK KC
WRITE LABEL
B2
MODIFY DASD ADDRESS CONTROL FACTOR
B3
SAVE SEQ NO. OF LAST EXTENT OPENED
C1
POST EXTENT LL AND UL, SET HEAD LIMITS TO 0 FOR TYPE 1 EXTENT IN DTF
C2
MOVE CONTROL FACTOR INTO DTF CONTROL BUCKET
C3
POST FILE OPENED AND SYMUNIT IN DTF
D1
IS IT DTFPH FILE
NO
YES
1
D2
IS EXTENT A TYPE 1
YES
NO
D3
CLOSE REQUIRED
NO
YES
E1
IS IT VERSION 1 TABLE
YES
NO
2
1
E2
MOVE UL AND LL'S OF HEAD INTO DTF CONTROL BUCKET
E3
RESET 'XTENT AT CLOSE' SWITCH
F2
INITIALIZE RECORD NUMBER AND CCHHR BUCKET IN DTF FOR LIOCS
2
F3
SET SWITCH FOR NO USER EXIT
G3
END OF VOLUME FORCED
YES
NO
FEOVRET G4
MODIFY TO FETCH $$BOSDEV
LASTXTNT G5
AS REQUIRED SET VARIOUS OPEN INDICATORS
H3
MODIFY TO FETCH $$BOSDC1 CLOSE
FILEPROT H4
FILE PROTECT SPECIFIED
YES
NO
LOADEXIT J4
MORE FILES TO OPEN
YES
NO
J5
SVC 2 $$BOFLPT
K3
SVC 2 $$BOPEN OR $$BOSDC1
K4
SVC 11 TO PROBLEM PROGRAM

## Chart JQ. $$BOSD06: SD Open Output, User Labels (Section 1 of 2)


1
2
3
A1
$$BOSD06
OPENOUT6
B1
RELOCATE CCW'S
C1
IS THIS A RE-ENTRY INTO THIS PHASE
NO
YES
D1
RESET RE-ENTRY INDICATOR
E1
GO TO ADDR IN REGISTER 5
*G1
HEADER LABEL INDICATOR IS SET IN $$BOSD04.
D2
IS LABEL ON A 2321
NO
YES
E2
SET MAX COUNT OF LABELS TO 5
F2
INITIALIZE CCB, SEEK BUCKET, WRITE COUNT FIELD
G2
NEED HEADER LABELS *G1
NO
YES
1
H2
IS IT DTFSD FILE
NO
YES
J2
SET INDICATOR TO PROCESS TRAILER LABELS
K2
RESET PROCESS HEADER LBL IND. INIT WRITE CCW CHAIN
3
MOVEUTLO
A3
INIT UTLO KEY, EXIT PHASE AND EOV INDICATOR
B3
RESET VOLUME AND HEADER LBL INDICATORS. INITIALIZE TO FETCH $$BOSD02
C3
NEED TRAILER LBLS AT CLOSING
NO
YES
CLOSEMON
D3
SET TRAILER AND EOF INDS. INIT TO FETCH $$BCLOSE
RESETLBL
E3
RESET VTOC LABEL IND. SET IND FOR NO USER LBL'S
F3
WRITDISK JR
LOOK FOR UTLO FILEMARK
G3
WAS FILEMARK FOUND
YES
NO
2
H3
MESSAGE 4408D
J3
ARE TRAILER LABELS AT CLOSE
NO
YES
K3
RESET TRLR LBL CLOSE IND INITIALIZE TO FETCH $$BCLOSE
DOEXTNT
JR D2
FILEADD2
A4
SET UP COUNT KEY AND DATA FIELDS INITIALIZE CCW CHAIN
B4
RESET TRLR LBL CLOSE INDICATOR. TURN OFF HEADER SW *C4
*C4
TRAILER LABEL INDICATOR IS SET IN $$BOSDC1.
WRITELBL
F4
WRITDISK JR
WRITE USER LABEL
G4
INCREMENT ADDR FOR NEXT RECORD
LABELSW
H4
SW FOR NO MORE LABELS
ON
OFF
J4
WERE 8 LABELS WRITTEN
NO
YES
3
K4
SET LABEL SW FOR NO MORE LABELS
FILEMARK
JR A1
LOADUSER
B5
INITIALIZE REGISTERS FOR USER LABEL ROUTINE
C5
SVC 8 EXIT TO USER LABEL ROUTINE
D5
RESTORE REGISTERS NEEDED BY OPEN
E5
DOES USER NEED MORE LABELS
YES
NO
F5
SET LABEL SW FOR NO MORE LABELS
G5
WRITDISK JR
WRITE LAST LABEL

## Chart JR. $$BOSD06: SD Open Output, User Labels (Section 2 of 2)


A2
FILEMARK
A1
INITIALIZE COUNT FIELD FOR FIRST FILE MARK
*A2 JQ-G5,H4,K4
B1
WRITDISK JR
WRITE FIRST FILE MARK
*JQ-J3,K3
HEADERSW
C1
SW FOR HEADER LABELS
OFF
ON
INITUTLO
D1
INITIALIZE TO WRITE UTLO FILE MARK
DQEXTNT
D2
DEQUEUE EXTENTS
NO
YES
E1
WRITDISK JR
WRITE LAST FILE MARK
E2
RESET DEQUEUE INDICATOR INITIALIZE TO FETCH $$BODQUE
NEXTPHAS
E3
SVC 2 $$BCLOSE OR $$BOSD02
F1
FILE PROTECT EXTENT
YES
NO
LOADREGO
F2
INITIALIZE TO FETCH $$BOFLPT OR $$BODQUE
LOADEXIT
G1
GET ADDR OF NEXT PARAMETER
G2
SVC 2 $$BOFLPT OR $$BODQUE
H1
ANOTHER PARAMETER TO BE OPENED CLOSED
YES
NO
J1
RESET MONITOR SW
J2
SVC 2 $$BOPEN
K1
SVC 11 TO PROBLEM PROGRAM
A4
WRITDISK
WRITDISK
B4
GET ADDRESS OF CCB FOR SUPERVISOR
C4
SVC 0 EXCP
D4
I/O COMPLETE
NO
YES
D5
SVC 7 WAIT FOR I/O COMPLETE
E4
SET CONDITION CODE IF RECORD NOT FOUND
F4
RETURN TO CALLING ROUTINE

## Chart JS. $$BOSD07: SD Open Output, Extents from Console

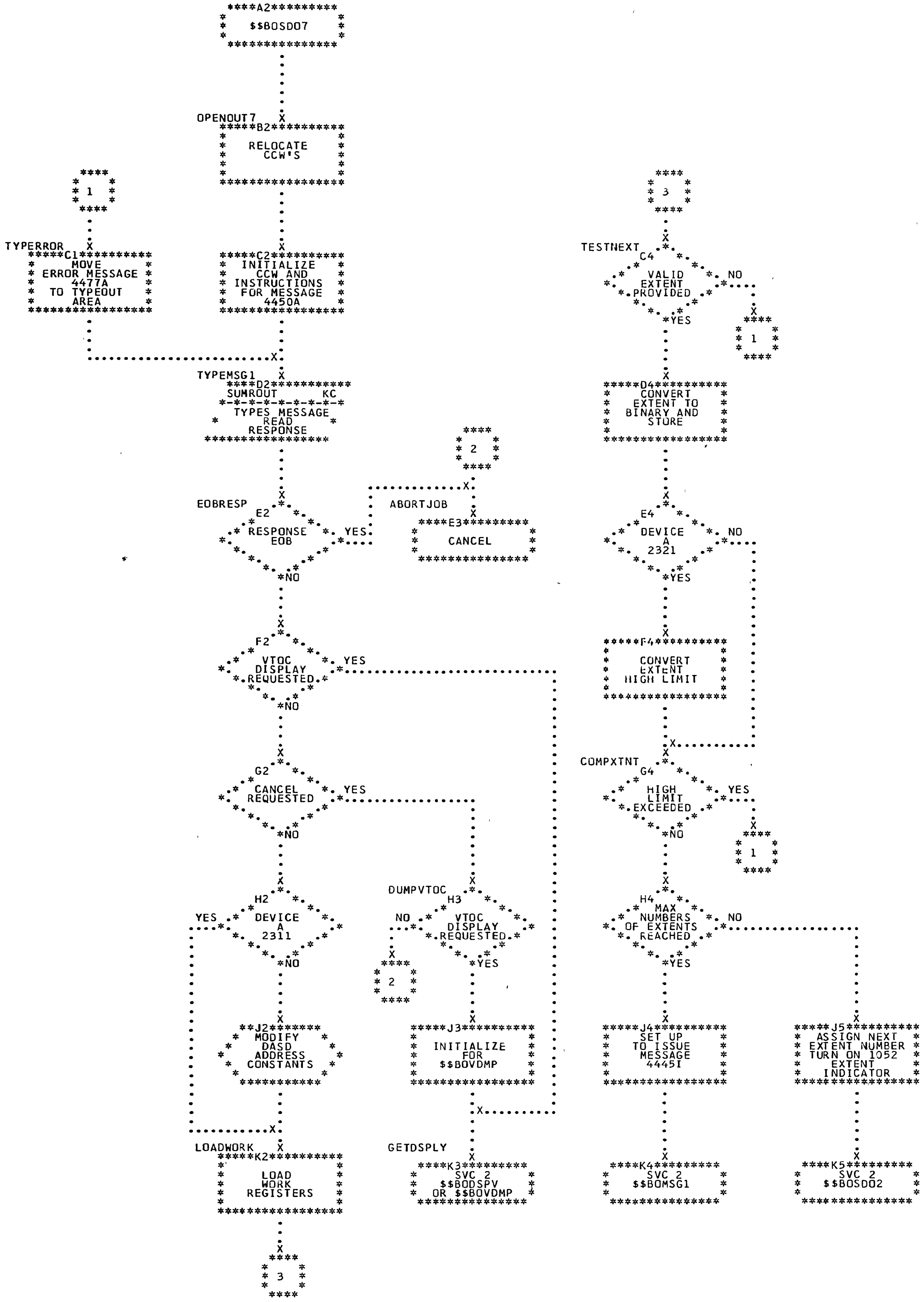
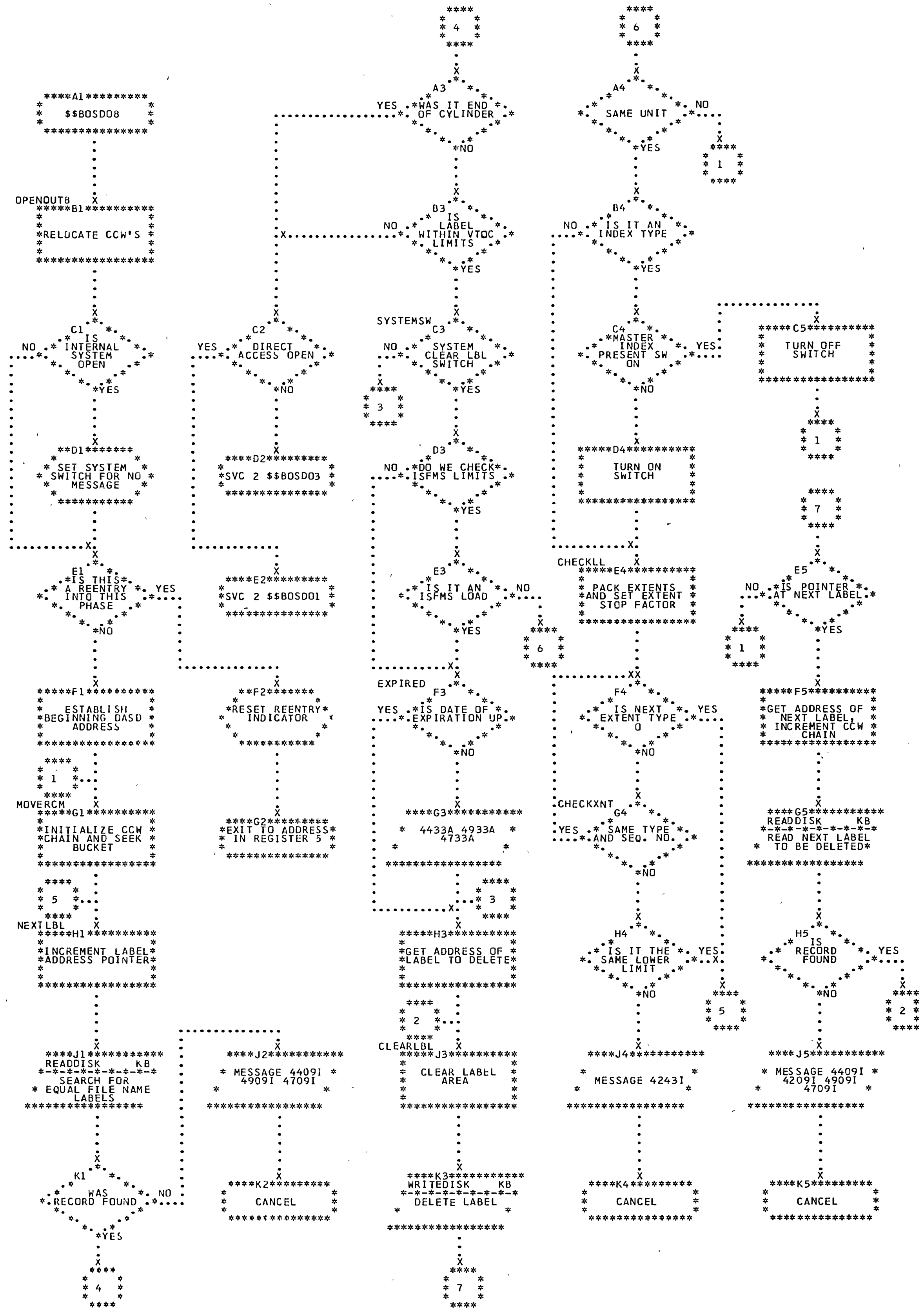


Chart KA. $$BOSD08: SD Open Output, Delete Label



A1 $$BOSD08
OPENOUT8 B1 RELOCATE CCW'S
C1 IS INTERNAL SYSTEM OPEN NO YES
D1 SET SYSTEM SWITCH FOR NO MESSAGE
E1 IS THIS A REENTRY INTO THIS PHASE YES NO
F1 ESTABLISH BEGINNING DASD ADDRESS
1
MOVERCM G1 INITIALIZE CCW CHAIN AND SEEK BUCKET
5
NEXTLBL H1 INCREMENT LABEL ADDRESS POINTER
J1 READDISK KB SEARCH FOR EQUAL FILE NAME LABELS
K1 WAS RECORD FOUND NO YES
4
C2 DIRECT ACCESS OPEN YES NO
D2 SVC 2 $$BOSD03
E2 SVC 2 $$BOSD01
F2 RESET REENTRY INDICATOR
G2 EXIT TO ADDRESS IN REGISTER 5
J2 MESSAGE 4409I 4909I 4709I
K2 CANCEL
4
A3 WAS IT END OF CYLINDER YES NO
B3 IS LABEL WITHIN VTOC LIMITS NO YES
SYSTEMSW C3 SYSTEM CLEAR LBL SWITCH NO YES
3
D3 DO WE CHECK ISFMS LIMITS NO YES
E3 IS IT AN ISFMS LOAD NO YES
6
EXPIRED F3 IS DATE OF EXPIRATION UP YES NO
G3 4433A 4933A 4733A
3
H3 GET ADDRESS OF LABEL TO DELETE
2
CLEARLBL J3 CLEAR LABEL AREA
K3 WRITEDISK KB DELETE LABEL
7
6
A4 SAME UNIT NO YES
1
B4 IS IT AN INDEX TYPE NO YES
C4 MASTER INDEX PRESENT SW ON YES NO
D4 TURN ON SWITCH
CHECKLL E4 PACK EXTENTS AND SET EXTENT STOP FACTOR
F4 IS NEXT EXTENT TYPE 0 YES NO
CHECKXNT G4 SAME TYPE AND SEQ. NO. YES NO
H4 IS IT THE SAME LOWER LIMIT YES NO
5
J4 MESSAGE 4243I
K4 CANCEL
C5 TURN OFF SWITCH
1
7
E5 IS POINTER AT NEXT LABEL NO YES
1
F5 GET ADDRESS OF NEXT LABEL, INCREMENT CCW CHAIN
G5 READDISK KB READ NEXT LABEL TO BE DELETED
H5 IS RECORD FOUND YES NO
2
J5 MESSAGE 4409I 4209I 4909I 4709I
K5 CANCEL

Chart KB. SD Open Output Subroutines (Section 1 of 2)

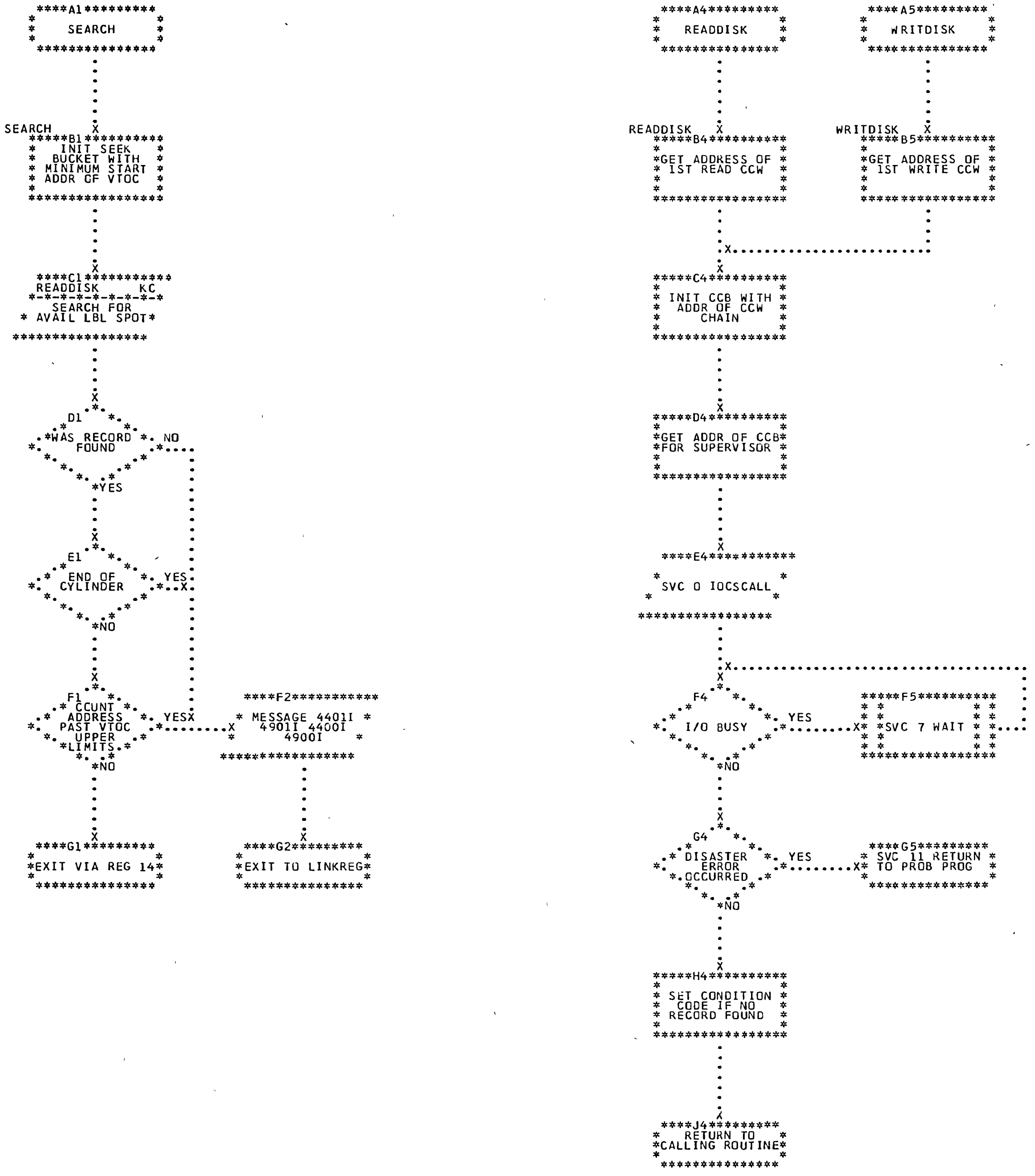
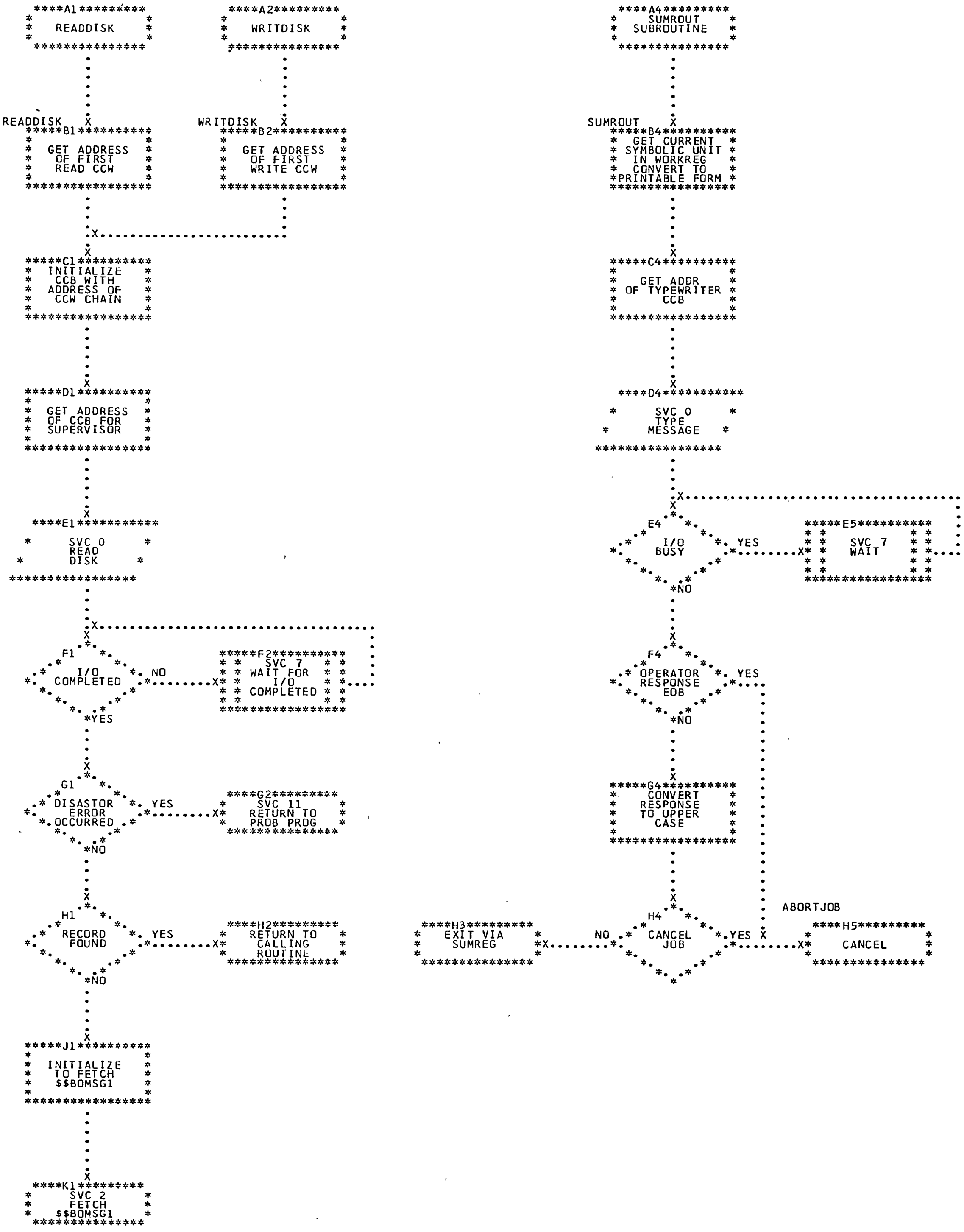

A1
SEARCH
SEARCH
B1
INIT SEEK BUCKET WITH MINIMUM START ADDR OF VTOC
C1
READDISK KC
SEARCH FOR AVAIL LBL SPOT
D1
WAS RECORD FOUND
NO
YES
E1
END OF CYLINDER
YES
NO
F1
COUNT ADDRESS PAST VTOC UPPER LIMITS
YES
NO
F2
MESSAGE 4401I 4901I 4400I 4900I
G1
EXIT VIA REG 14
G2
EXIT TO LINKREG
A4
READDISK
A5
WRITDISK
READDISK
B4
GET ADDRESS OF 1ST READ CCW
WRITDISK
B5
GET ADDRESS OF 1ST WRITE CCW
C4
INIT CCB WITH ADDR OF CCW CHAIN
D4
GET ADDR OF CCB FOR SUPERVISOR
E4
SVC 0 IOCSCALL
F4
I/O BUSY
YES
NO
F5
SVC 7 WAIT
G4
DISASTER ERROR OCCURRED
YES
NO
G5
SVC 11 RETURN TO PROB PROG
H4
SET CONDITION CODE IF NO RECORD FOUND
J4
RETURN TO CALLING ROUTINE

Chart KC. SD Open Output Subroutines (Section 2 of 2)



A1
READDISK
A2
WRITDISK
A4
SUMROUT SUBROUTINE
READDISK
B1
GET ADDRESS OF FIRST READ CCW
WRITDISK
B2
GET ADDRESS OF FIRST WRITE CCW
SUMROUT
B4
GET CURRENT SYMBOLIC UNIT IN WORKREG CONVERT TO PRINTABLE FORM
C1
INITIALIZE CCB WITH ADDRESS OF CCW CHAIN
C4
GET ADDR OF TYPEWRITER CCB
D1
GET ADDRESS OF CCB FOR SUPERVISOR
D4
SVC 0 TYPE MESSAGE
E1
SVC 0 READ DISK
E4
I/O BUSY
YES
NO
E5
SVC 7 WAIT
F1
I/O COMPLETED
NO
YES
F2
SVC 7 WAIT FOR I/O COMPLETED
F4
OPERATOR RESPONSE EOB
YES
NO
G1
DISASTOR ERROR OCCURRED
YES
NO
G2
SVC 11 RETURN TO PROB PROG
G4
CONVERT RESPONSE TO UPPER CASE
H1
RECORD FOUND
YES
NO
H2
RETURN TO CALLING ROUTINE
H3
EXIT VIA SUMREG
H4
CANCEL JOB
NO
YES
ABORTJOB
H5
CANCEL
J1
INITIALIZE TO FETCH $$BOMSG1
K1
SVC 2 FETCH $$BOMSG1

## Chart KD. $$BOSDW1: SD Open Work file, Volume Label (Section 1 of 3)

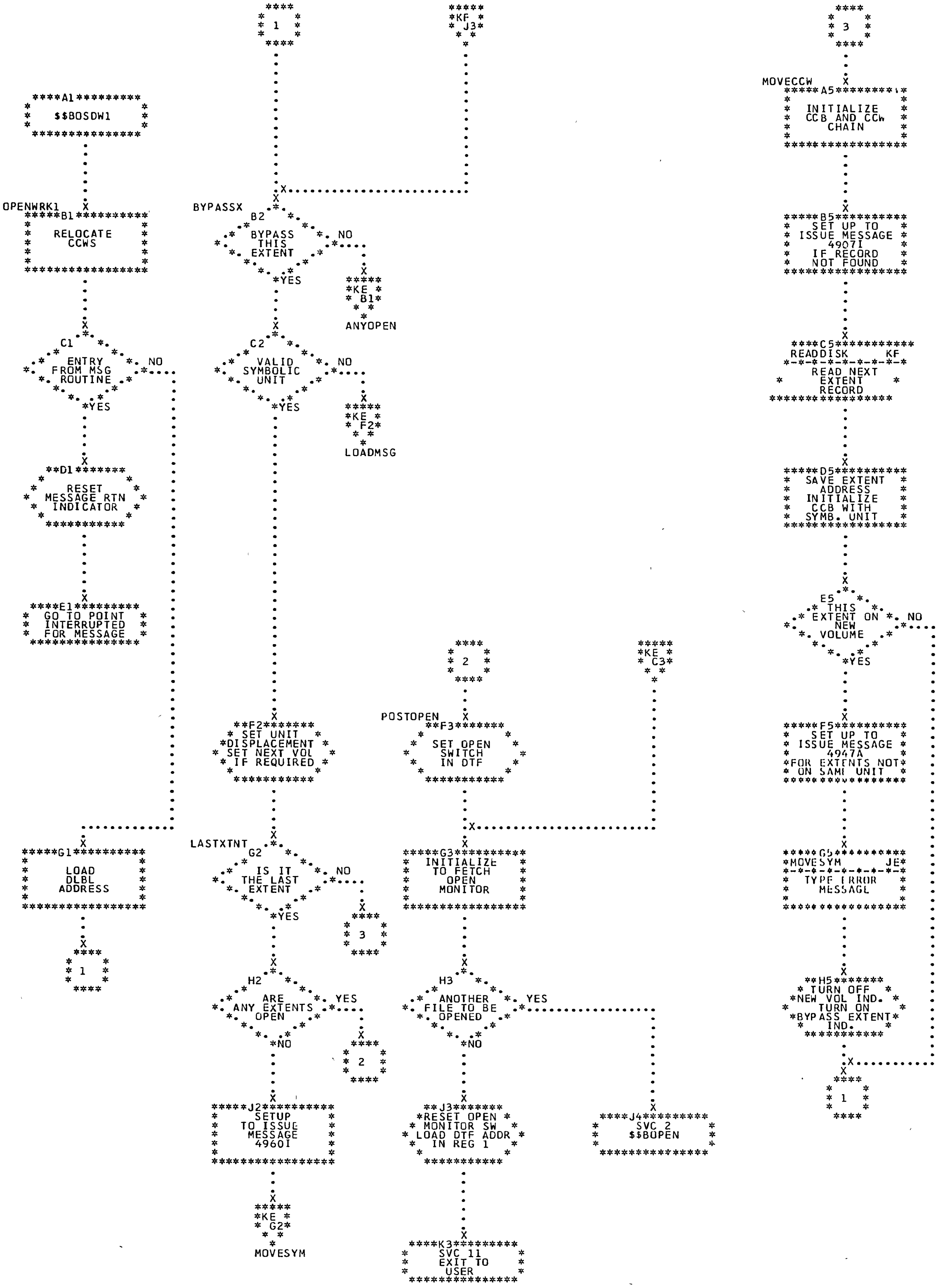

A1
$$BOSDW1
OPENWRK1
B1
RELOCATE CCWS
C1
ENTRY FROM MSG ROUTINE
NO
YES
D1
RESET MESSAGE RTN INDICATOR
E1
GO TO POINT INTERRUPTED FOR MESSAGE
G1
LOAD DLBL ADDRESS
1
1
KF J3
BYPASSX
B2
BYPASS THIS EXTENT
NO
YES
KE B1
ANYOPEN
C2
VALID SYMBOLIC UNIT
NO
YES
KE F2
LOADMSG
F2
SET UNIT DISPLACEMENT SET NEXT VOL IF REQUIRED
LASTXTNT
G2
IS IT THE LAST EXTENT
NO
YES
3
H2
ARE ANY EXTENTS OPEN
YES
NO
2
J2
SETUP TO ISSUE MESSAGE 4960I
KE G2
MOVESYM
2
POSTOPEN
F3
SET OPEN SWITCH IN DTF
KE C3
G3
INITIALIZE TO FETCH OPEN MONITOR
H3
ANOTHER FILE TO BE OPENED
YES
NO
J3
RESET OPEN MONITOR SW LOAD DTF ADDR IN REG 1
K3
SVC 11 EXIT TO USER
J4
SVC 2 $$BOPEN
3
MOVECCW
A5
INITIALIZE CCB AND CCW CHAIN
B5
SET UP TO ISSUE MESSAGE 4907I IF RECORD NOT FOUND
C5
READDISK KF
READ NEXT EXTENT RECORD
D5
SAVE EXTENT ADDRESS INITIALIZE CCB WITH SYMB. UNIT
E5
THIS EXTENT ON NEW VOLUME
NO
YES
F5
SET UP TO ISSUE MESSAGE 4947A FOR EXTENTS NOT ON SAME UNIT
G5
MOVESYM JE
TYPE ERROR MESSAGE
H5
TURN OFF NEW VOL IND. TURN ON BYPASS EXTENT IND.
1

Chart KE. $$BOSDW1: SD Open Work file, Volume Label (Section 2 of 3)

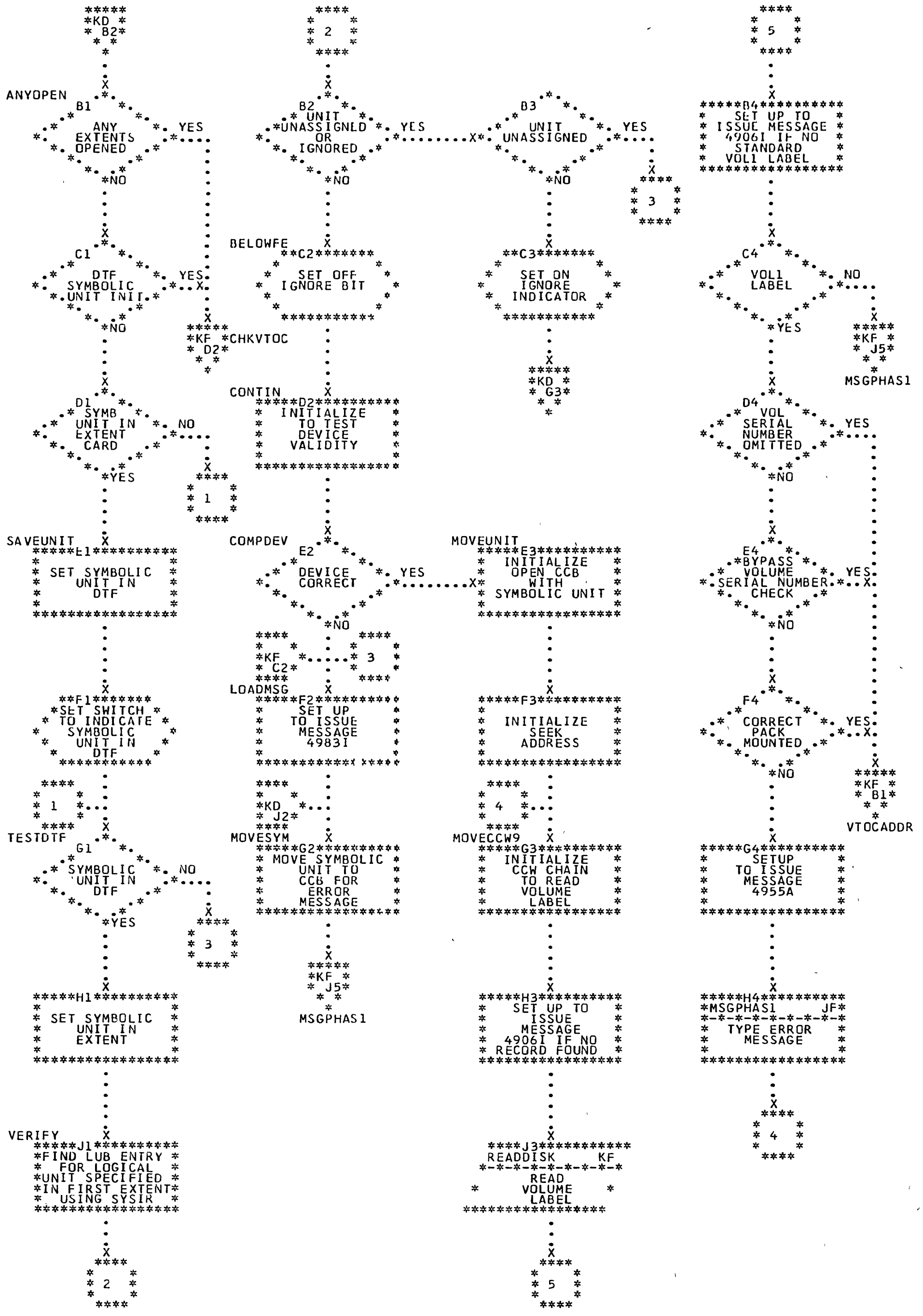


Chart KF. $$BOSDW1: SD Open Work file, Volume Label (Section 3 of 3)

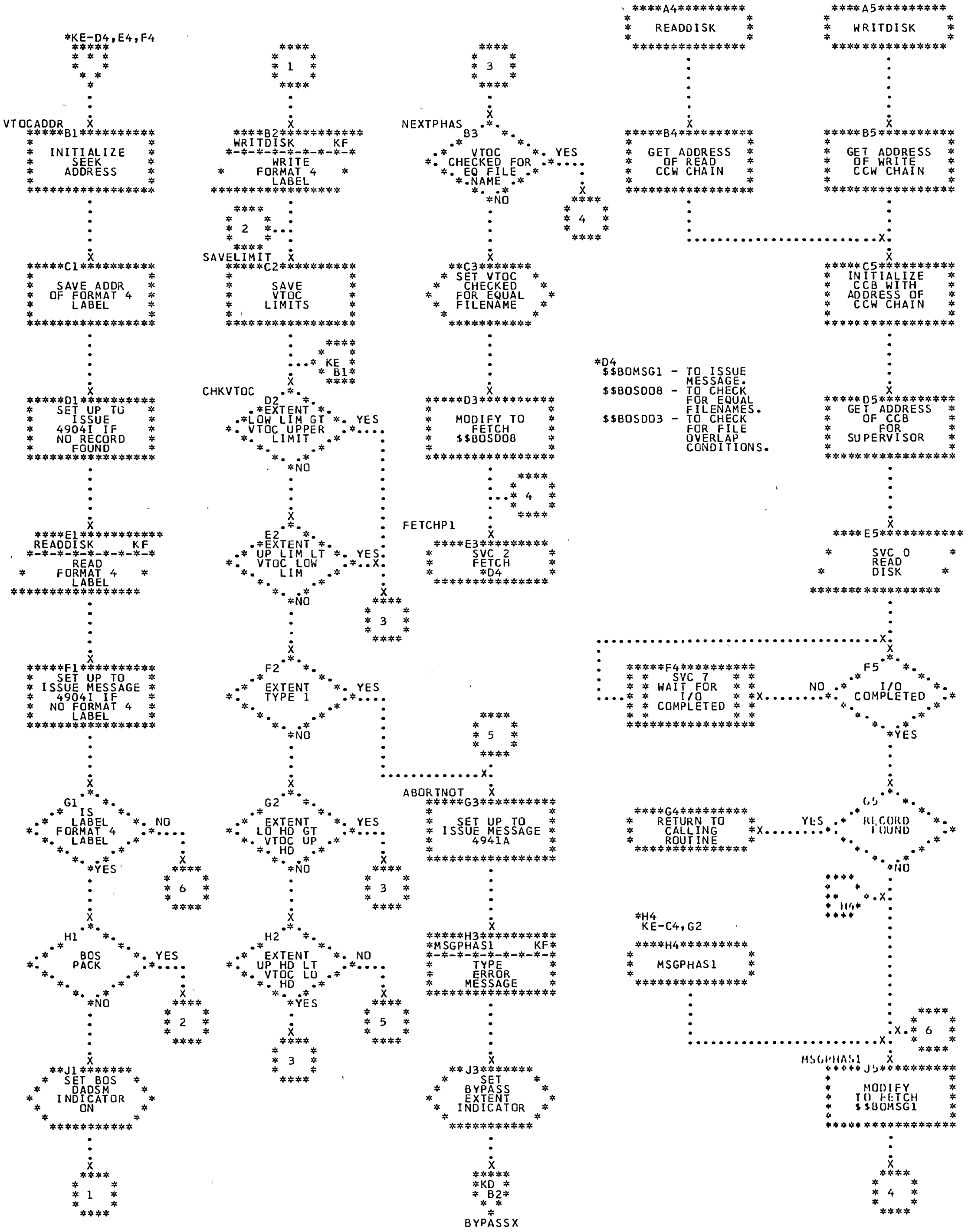

KE-D4,E4,F4
VTOCADDR
B1
INITIALIZE SEEK ADDRESS
C1
SAVE ADDR OF FORMAT 4 LABEL
D1
SET UP TO ISSUE 4904I IF NO RECORD FOUND
E1
READDISK KF
READ FORMAT 4 LABEL
F1
SET UP TO ISSUE MESSAGE 4904I IF NO FORMAT 4 LABEL
G1
IS LABEL FORMAT 4 LABEL
NO
6
YES
H1
BOS PACK
YES
2
NO
J1
SET BOS DADSM INDICATOR ON
1
1
B2
WRITDISK KF
WRITE FORMAT 4 LABEL
2
SAVELIMIT
C2
SAVE VTOC LIMITS
KE B1
CHKVTOC
D2
EXTENT LOW LIM GT VTOC UPPER LIMIT
YES
NO
E2
EXTENT UP LIM LT VTOC LOW LIM
YES
NO
3
F2
EXTENT TYPE 1
YES
NO
G2
EXTENT LO HD GT VTOC UP HD
YES
NO
3
H2
EXTENT UP HD LT VTOC LO HD
NO
YES
5
3
3
NEXTPHAS
B3
VTOC CHECKED FOR EQ FILE NAME
YES
4
NO
C3
SET VTOC CHECKED FOR EQUAL FILENAME
D3
MODIFY TO FETCH $$BOSDO8
4
FETCHP1
E3
SVC 2 FETCH
D4
5
ABORTNOT
G3
SET UP TO ISSUE MESSAGE 4941A
H3
MSGPHAS1 KF
TYPE ERROR MESSAGE
J3
SET BYPASS EXTENT INDICATOR
KD B2
BYPASSX
A4
READDISK
B4
GET ADDRESS OF READ CCW CHAIN
D4
$$BOMSG1 - TO ISSUE MESSAGE.
$$BOSDO8 - TO CHECK FOR EQUAL FILENAMES.
$$BOSDO3 - TO CHECK FOR FILE OVERLAP CONDITIONS.
F4
SVC 7 WAIT FOR I/O COMPLETED
G4
RETURN TO CALLING ROUTINE
H4
KE-C4,G2
H4
MSGPHAS1
A5
WRITDISK
B5
GET ADDRESS OF WRITE CCW CHAIN
C5
INITIALIZE CCB WITH ADDRESS OF CCW CHAIN
D5
GET ADDRESS OF CCB FOR SUPERVISOR
E5
SVC 0 READ DISK
F5
I/O COMPLETED
NO
YES
G5
RECORD FOUND
YES
NO
H4
6
MSGPHAS1
J5
MODIFY TO FETCH $$BOMSG1
4

Chart KG. $$BOSDW2: SD Open Work file, File Label (Section 1 of 2)

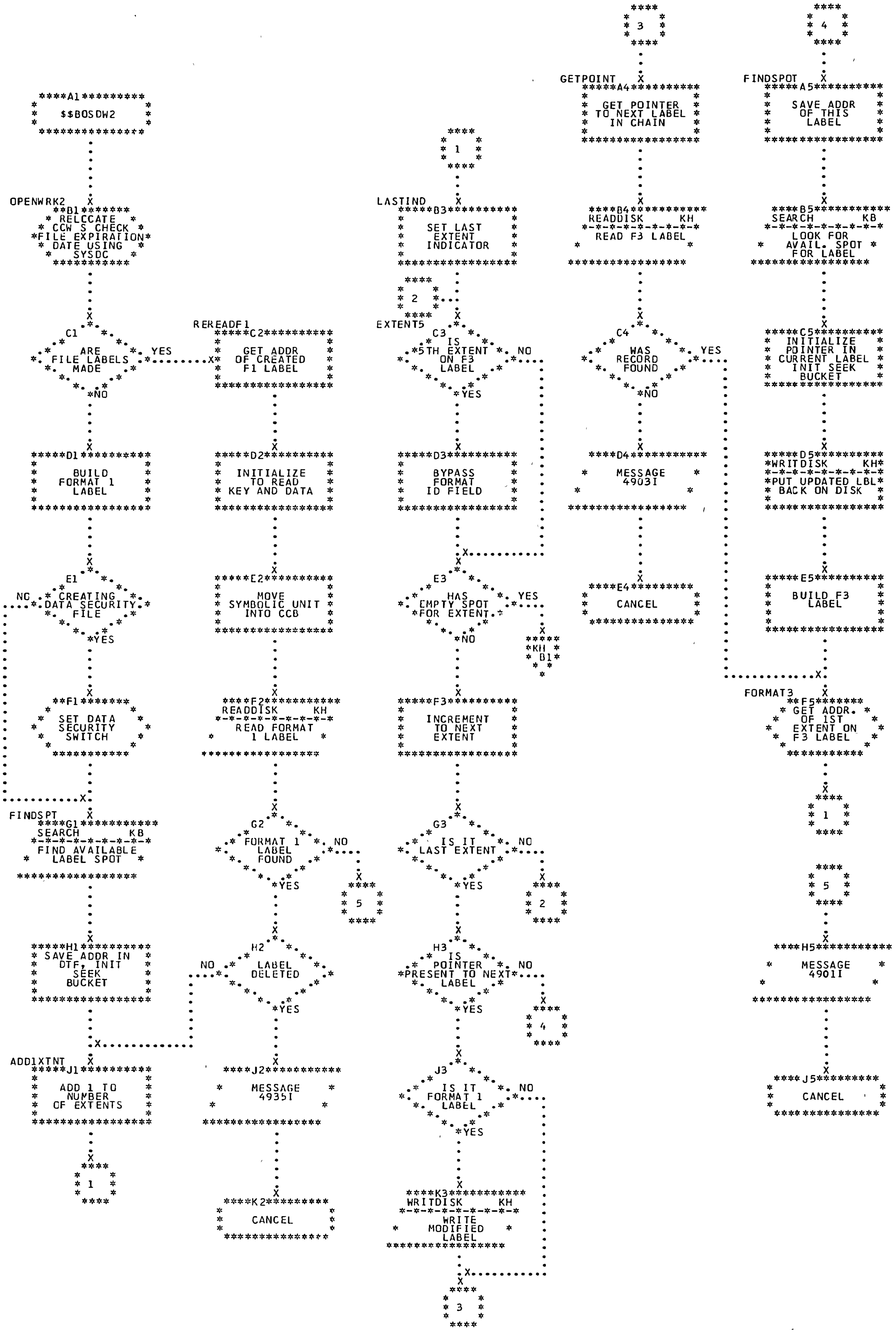

A1 $$BOSDW2
OPENWRK2
B1 RELCCATE CCW S CHECK FILE EXPIRATION DATE USING SYSDC
C1 ARE FILE LABELS MADE
YES
NO
REREADF1
C2 GET ADDR OF CREATED F1 LABEL
D1 BUILD FORMAT 1 LABEL
D2 INITIALIZE TO READ KEY AND DATA
E1 CREATING DATA SECURITY FILE
NO
YES
E2 MOVE SYMBOLIC UNIT INTO CCB
F1 SET DATA SECURITY SWITCH
F2 READDISK KH READ FORMAT 1 LABEL
FINDSPT
G1 SEARCH KB FIND AVAILABLE LABEL SPOT
G2 FORMAT 1 LABEL FOUND
NO
YES
5
H1 SAVE ADDR IN DTF, INIT SEEK BUCKET
H2 LABEL DELETED
NO
YES
ADD1XTNT
J1 ADD 1 TO NUMBER OF EXTENTS
J2 MESSAGE 4935I
1
K2 CANCEL
1
LASTIND
B3 SET LAST EXTENT INDICATOR
2
EXTENT5
C3 IS 5TH EXTENT ON F3 LABEL
NO
YES
D3 BYPASS FORMAT ID FIELD
E3 HAS EMPTY SPOT FOR EXTENT
YES
NO
KH B1
F3 INCREMENT TO NEXT EXTENT
G3 IS IT LAST EXTENT
NO
YES
2
H3 IS POINTER PRESENT TO NEXT LABEL
NO
YES
4
J3 IS IT FORMAT 1 LABEL
NO
YES
K3 WRITDISK KH WRITE MODIFIED LABEL
3
3
GETPOINT
A4 GET POINTER TO NEXT LABEL IN CHAIN
B4 READDISK KH READ F3 LABEL
C4 WAS RECORD FOUND
YES
NO
D4 MESSAGE 4903I
E4 CANCEL
4
FINDSPOT
A5 SAVE ADDR OF THIS LABEL
B5 SEARCH KB LOOK FOR AVAIL. SPOT FOR LABEL
C5 INITIALIZE POINTER IN CURRENT LABEL INIT SEEK BUCKET
D5 WRITDISK KH PUT UPDATED LBL BACK ON DISK
E5 BUILD F3 LABEL
FORMAT3
F5 GET ADDR. OF 1ST EXTENT ON F3 LABEL
1
5
H5 MESSAGE 4901I
J5 CANCEL

Chart KH. $$BOSDW2: SD Open Work file, File Label (Section 2 of 2)

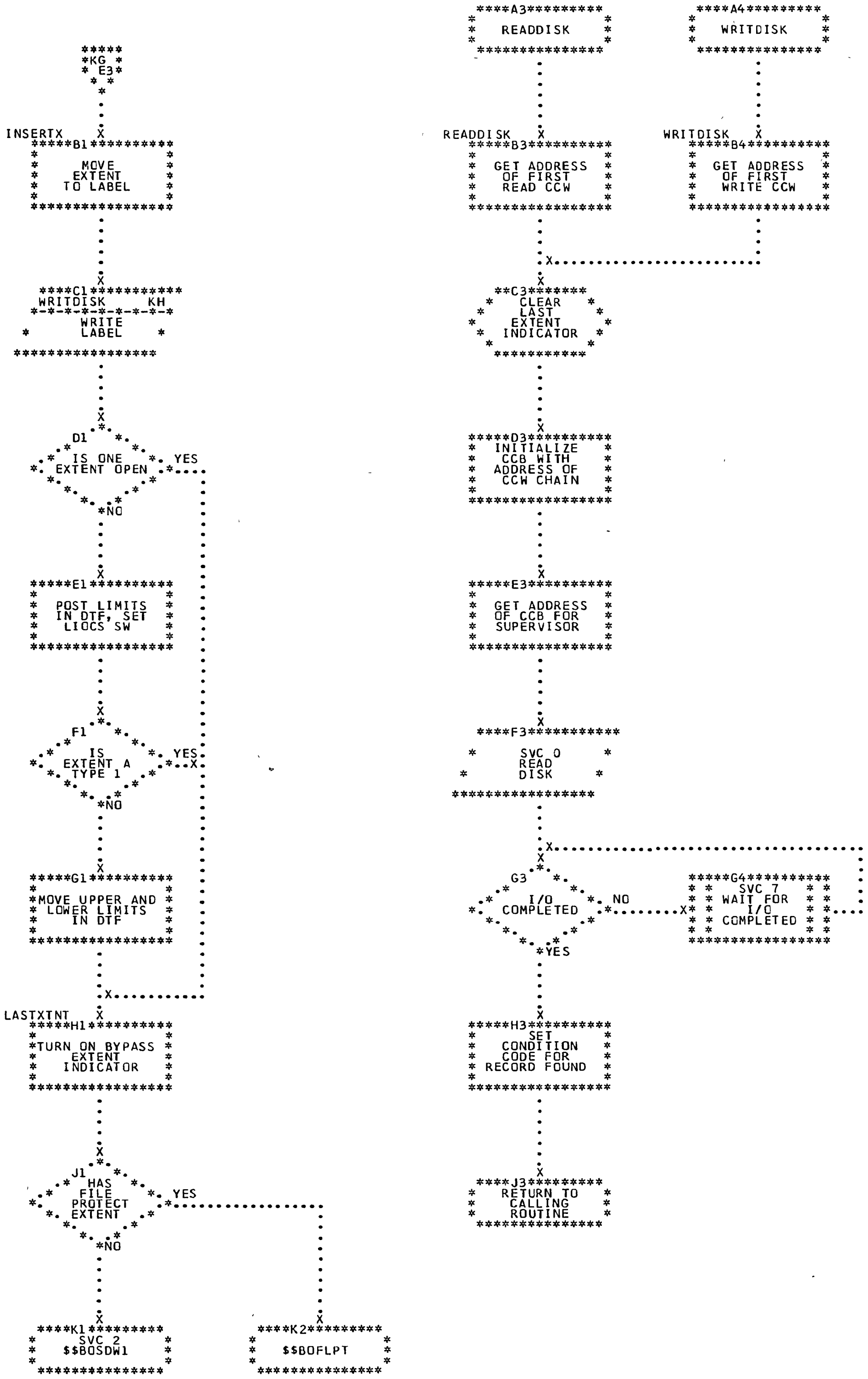

KG E3
INSERTX
B1
MOVE EXTENT TO LABEL
C1
WRITDISK KH
WRITE LABEL
D1
IS ONE EXTENT OPEN
YES
NO
E1
POST LIMITS IN DTF, SET LIOCS SW
F1
IS EXTENT A TYPE 1
YES
NO
G1
MOVE UPPER AND LOWER LIMITS IN DTF
LASTXTNT
H1
TURN ON BYPASS EXTENT INDICATOR
J1
HAS FILE PROTECT EXTENT
YES
NO
K1
SVC 2 $$BOSDW1
K2
$$BOFLPT
A3
READDISK
A4
WRITDISK
READDISK
B3
GET ADDRESS OF FIRST READ CCW
WRITDISK
B4
GET ADDRESS OF FIRST WRITE CCW
C3
CLEAR LAST EXTENT INDICATOR
D3
INITIALIZE CCB WITH ADDRESS OF CCW CHAIN
E3
GET ADDRESS OF CCB FOR SUPERVISOR
F3
SVC 0 READ DISK
G3
I/O COMPLETED
NO
YES
G4
SVC 7 WAIT FOR I/O COMPLETED
H3
SET CONDITION CODE FOR RECORD FOUND
J3
RETURN TO CALLING ROUTINE

Chart KJ. $$BOSDW3: SD Open Work file, Extent to DTF (Section 1 of 2)

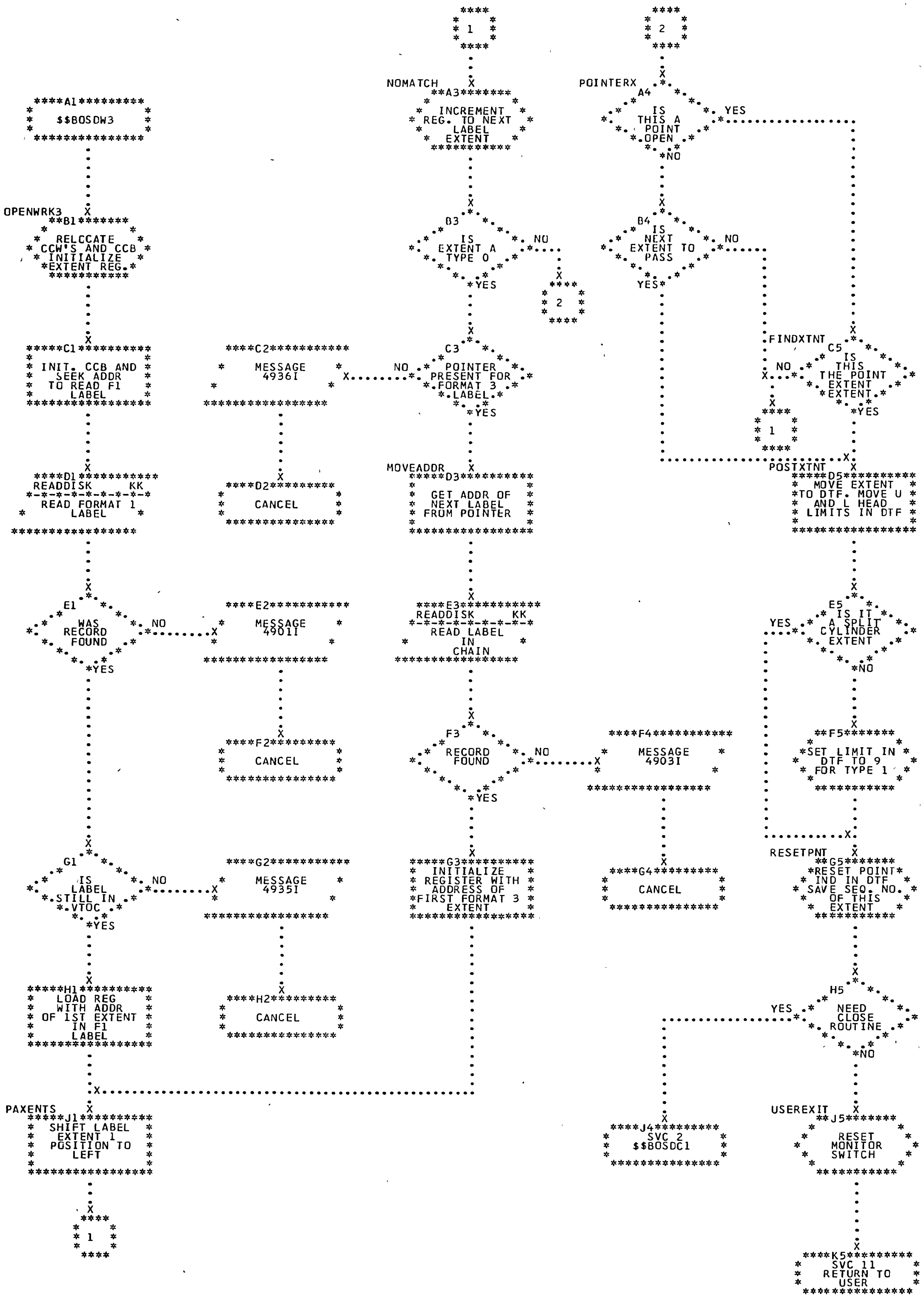

1
2
A1 $$BOSDW3
OPENWRK3
B1 RELCCATE CCW'S AND CCB INITIALIZE EXTENT REG.
C1 INIT. CCB AND SEEK ADDR TO READ F1 LABEL
D1 READDISK KK READ FORMAT 1 LABEL
E1 WAS RECORD FOUND
NO
YES
G1 IS LABEL STILL IN VTOC
NO
YES
H1 LOAD REG WITH ADDR OF 1ST EXTENT IN F1 LABEL
PAXENTS
J1 SHIFT LABEL EXTENT 1 POSITION TO LEFT
1
C2 MESSAGE 4936I
D2 CANCEL
E2 MESSAGE 4901I
F2 CANCEL
G2 MESSAGE 4935I
H2 CANCEL
NOMATCH
A3 INCREMENT REG. TO NEXT LABEL EXTENT
B3 IS EXTENT A TYPE 0
NO
2
YES
C3 POINTER PRESENT FOR FORMAT 3 LABEL
NO
YES
MOVEADDR
D3 GET ADDR OF NEXT LABEL FROM POINTER
E3 READDISK KK READ LABEL IN CHAIN
F3 RECORD FOUND
NO
YES
G3 INITIALIZE REGISTER WITH ADDRESS OF FIRST FORMAT 3 EXTENT
F4 MESSAGE 4903I
G4 CANCEL
J4 SVC 2 $$BOSDC1
POINTERX
A4 IS THIS A POINT OPEN
YES
NO
B4 IS NEXT EXTENT TO PASS
NO
YES
FINDXTNT
C5 IS THIS THE POINT EXTENT
NO
1
YES
POSTXTNT
D5 MOVE EXTENT TO DTF. MOVE U AND L HEAD LIMITS IN DTF
E5 IS IT A SPLIT CYLINDER EXTENT
YES
NO
F5 SET LIMIT IN DTF TO 9 FOR TYPE 1
RESETPNT
G5 RESET POINT IND IN DTF SAVE SEQ. NO. OF THIS EXTENT
H5 NEED CLOSE ROUTINE
YES
NO
USEREXIT
J5 RESET MONITOR SWITCH
K5 SVC 11 RETURN TO USER

Chart KK. $$BOSDW3: SD Open Work file, Extent to DTF (Section 2 of 2)

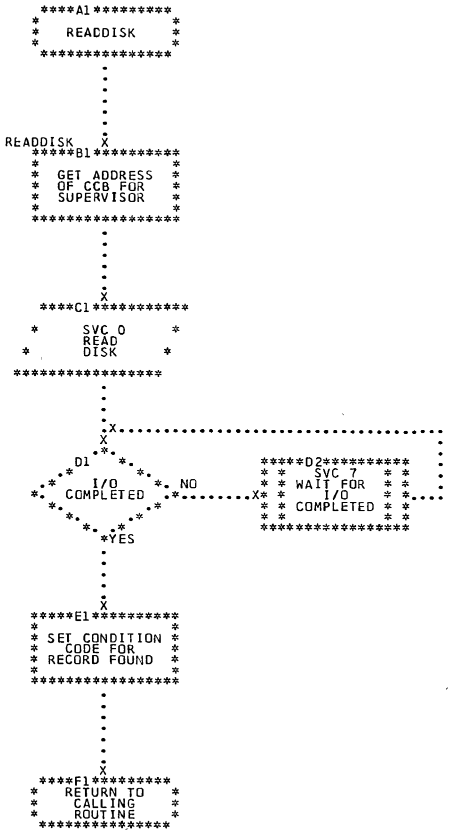
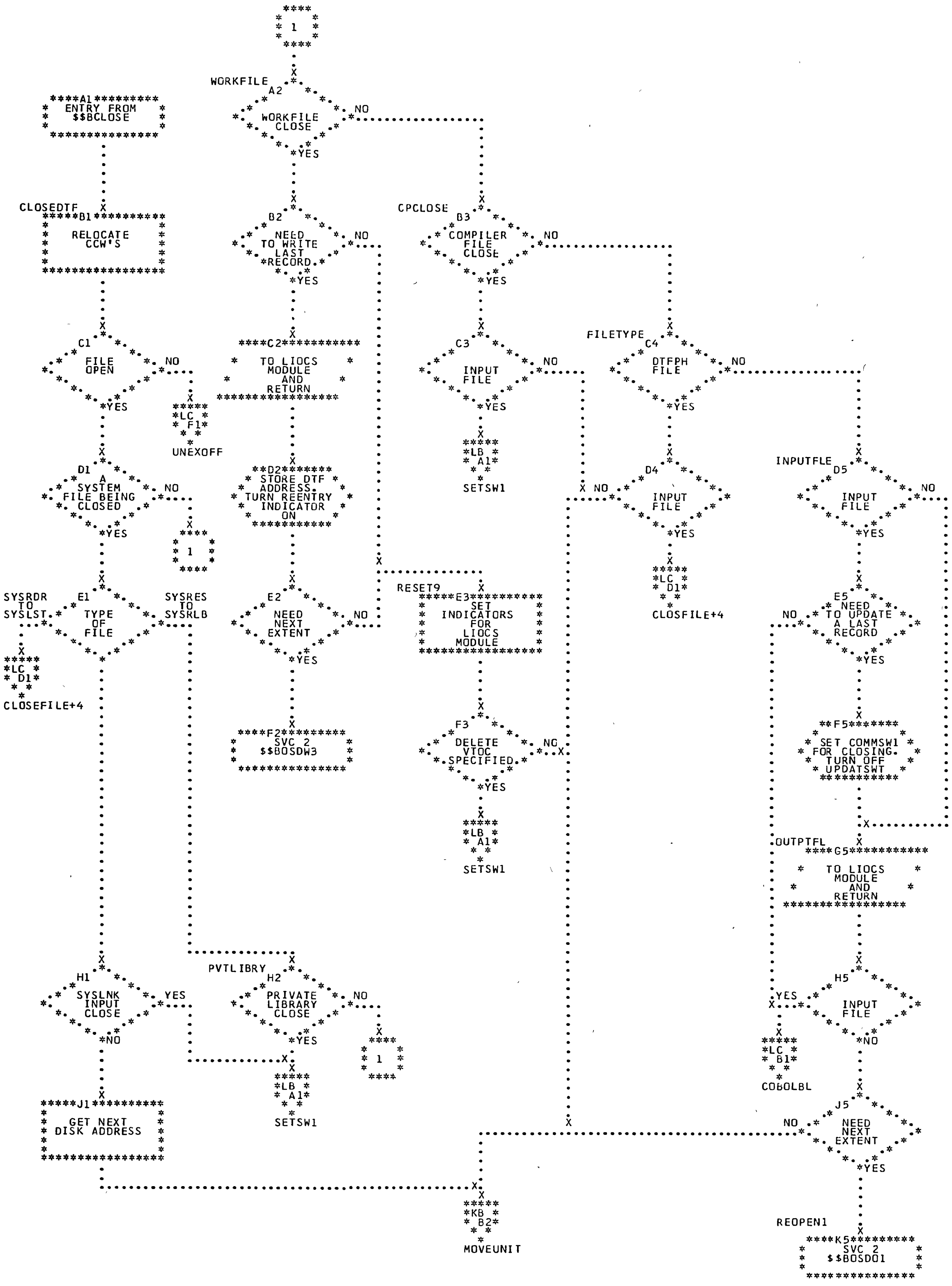

A1
READDISK
READDISK
B1
GET ADDRESS OF CCB FOR SUPERVISOR
C1
SVC 0 READ DISK
D1
I/O COMPLETED
NO
YES
D2
SVC 7 WAIT FOR I/O COMPLETED
E1
SET CONDITION CODE FOR RECORD FOUND
F1
RETURN TO CALLING ROUTINE

Chart LA. $$BOSDC1: SD Close Input and Output (Section 1 of 3)



## Chart LB. $$BOSDC1: SD Close Input and Output (Section 2 of 3)

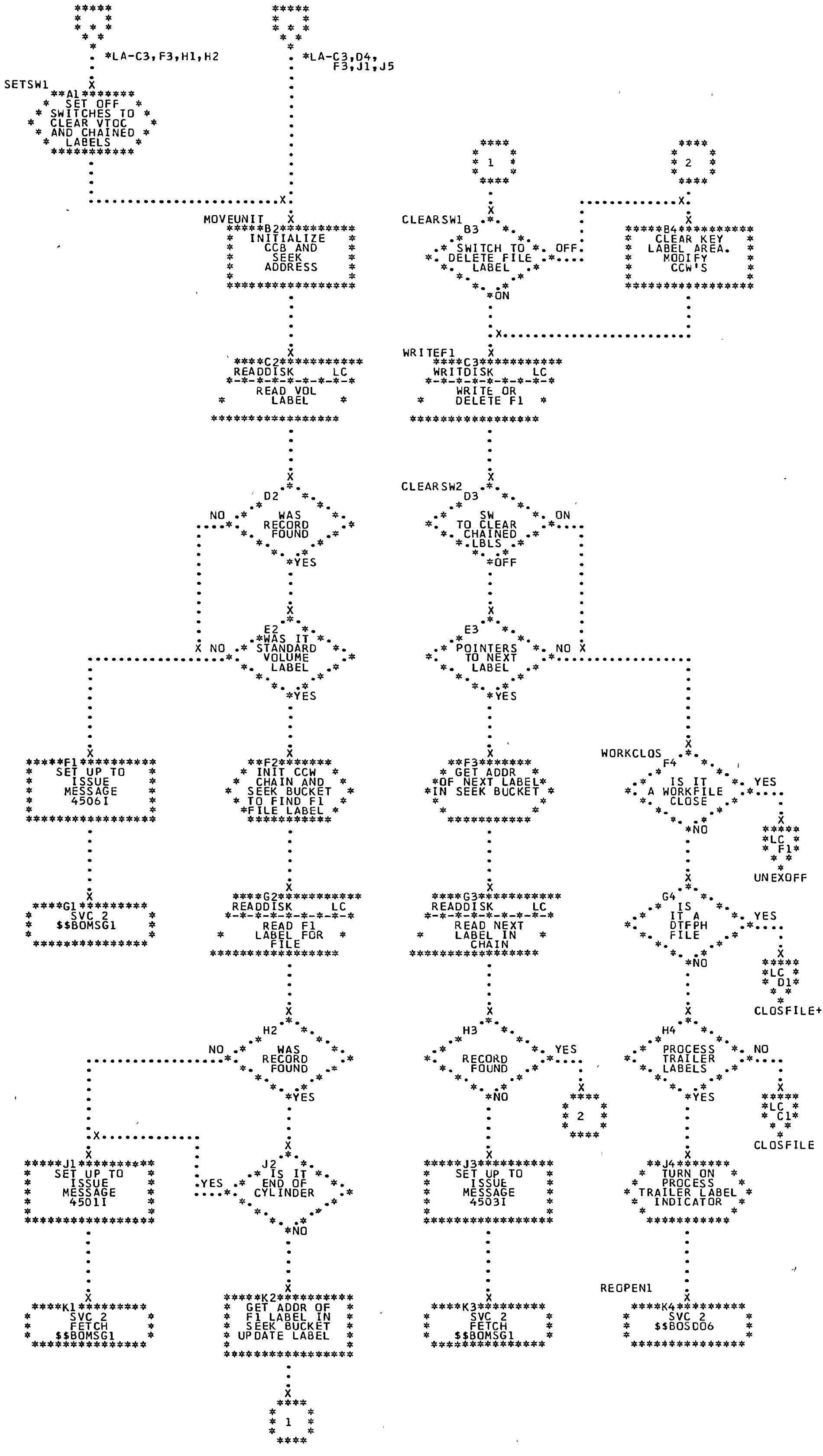
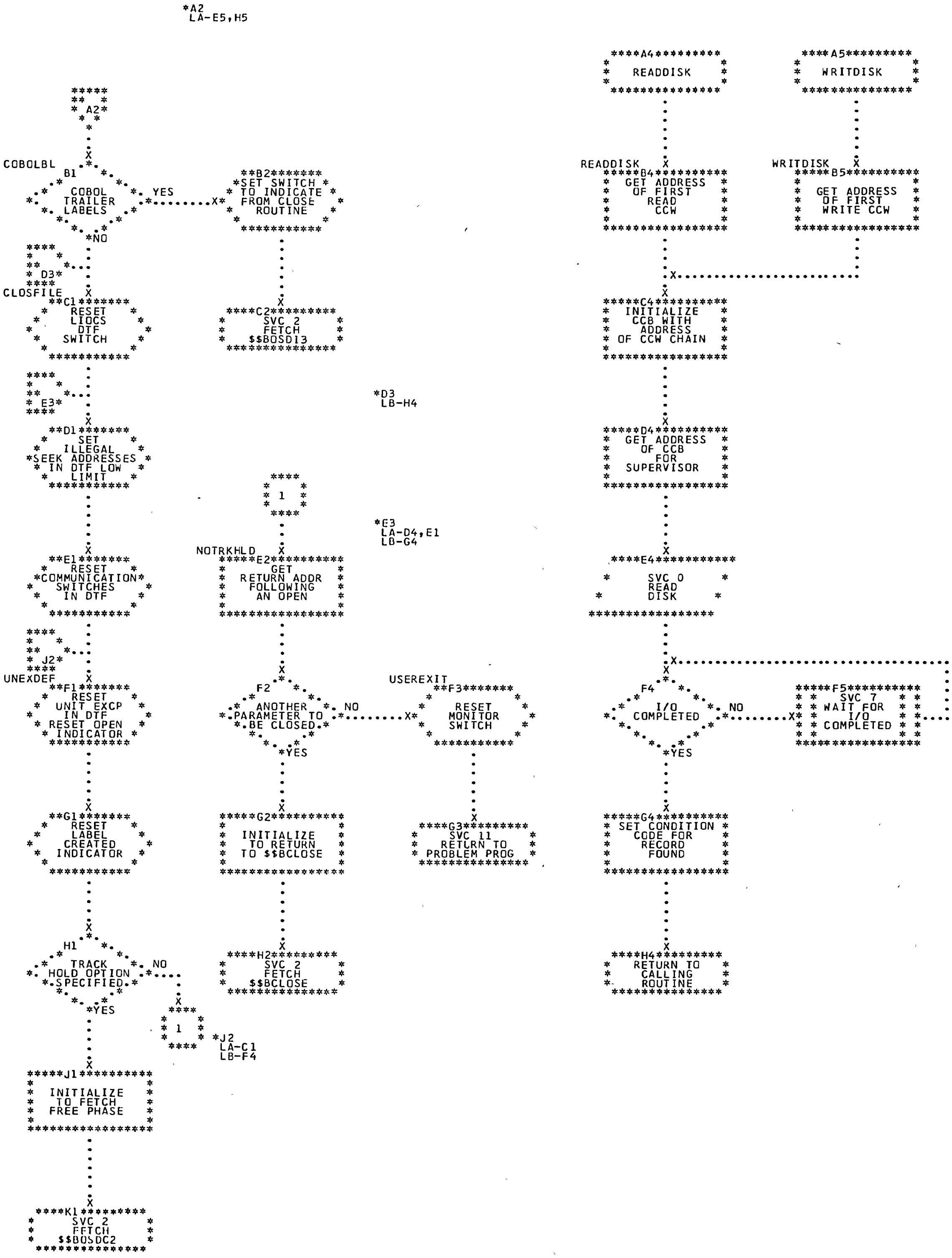
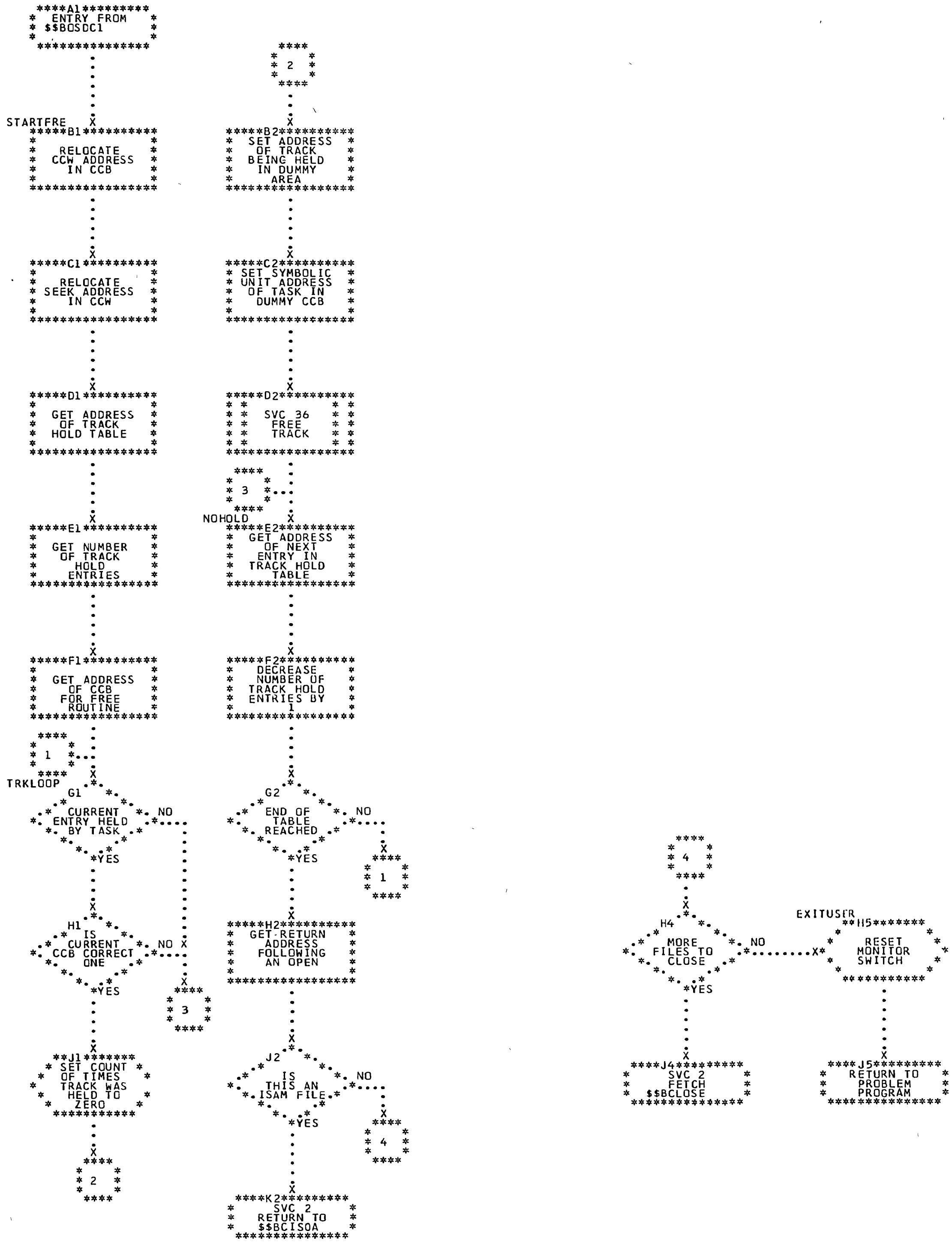

*LA-C3,F3,H1,H2
*LA-C3,D4, F3,J1,J5
SETSW1
A1 SET OFF SWITCHES TO CLEAR VTOC AND CHAINED LABELS
MOVEUNIT
B2 INITIALIZE CCB AND SEEK ADDRESS
C2 READDISK LC READ VOL LABEL
D2 WAS RECORD FOUND
NO
YES
E2 WAS IT STANDARD VOLUME LABEL
F1 SET UP TO ISSUE MESSAGE 4506I
G1 SVC 2 $$BOMSG1
F2 INIT CCW CHAIN AND SEEK BUCKET TO FIND F1 FILE LABEL
G2 READDISK LC READ F1 LABEL FOR FILE
H2 WAS RECORD FOUND
J1 SET UP TO ISSUE MESSAGE 4501I
K1 SVC 2 FETCH $$BOMSG1
J2 IS IT END OF CYLINDER
K2 GET ADDR OF F1 LABEL IN SEEK BUCKET UPDATE LABEL
1
2
CLEARSW1
B3 SWITCH TO DELETE FILE LABEL
OFF
ON
B4 CLEAR KEY LABEL AREA. MODIFY CCW'S
WRITEF1
C3 WRITDISK LC WRITE OR DELETE F1
CLEARSW2
D3 SW TO CLEAR CHAINED LBLS
E3 POINTERS TO NEXT LABEL
F3 GET ADDR OF NEXT LABEL IN SEEK BUCKET
G3 READDISK LC READ NEXT LABEL IN CHAIN
H3 RECORD FOUND
J3 SET UP TO ISSUE MESSAGE 4503I
K3 SVC 2 FETCH $$BOMSG1
WORKCLOS
F4 IS IT A WORKFILE CLOSE
LC F1
UNEXOFF
G4 IS IT A DTFPH FILE
LC D1
CLOSFILE+4
H4 PROCESS TRAILER LABELS
LC C1
CLOSFILE
J4 TURN ON PROCESS TRAILER LABEL INDICATOR
REOPEN1
K4 SVC 2 $$BOSD06

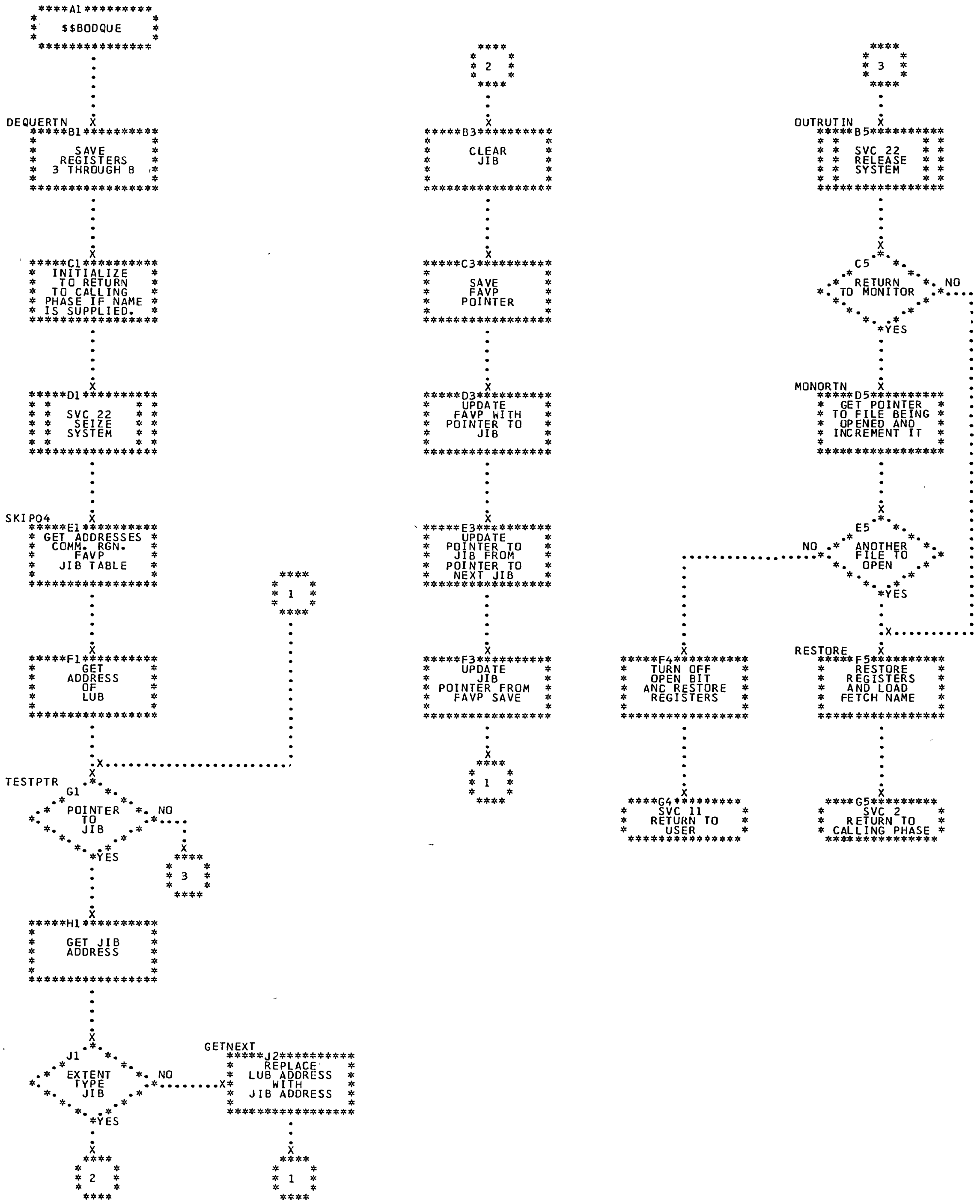
Chart LC. $$BOSDC1: SD Close Input and Output (Section 3 of 3)


*A2
LA-E5,H5
A2
COBOLBL
B1
COBOL TRAILER LABELS
YES
NO
B2
SET SWITCH TO INDICATE FROM CLOSE ROUTINE
D3
CLOSFILE
C1
RESET LIOCS DTF SWITCH
C2
SVC 2 FETCH $$BOSDI3
E3
D1
SET ILLEGAL SEEK ADDRESSES IN DTF LOW LIMIT
*D3
LB-H4
1
*E3
LA-D4,E1
LB-G4
NOTRKHLD
E1
RESET COMMUNICATION SWITCHES IN DTF
E2
GET RETURN ADDR FOLLOWING AN OPEN
J2
UNEXDEF
F1
RESET UNIT EXCP IN DTF RESET OPEN INDICATOR
F2
ANOTHER PARAMETER TO BE CLOSED
NO
YES
USEREXIT
F3
RESET MONITOR SWITCH
G1
RESET LABEL CREATED INDICATOR
G2
INITIALIZE TO RETURN TO $$BCLOSE
G3
SVC 11 RETURN TO PROBLEM PROG
H1
TRACK HOLD OPTION SPECIFIED
NO
YES
H2
SVC 2 FETCH $$BCLOSE
1
*J2
LA-C1
LB-F4
J1
INITIALIZE TO FETCH FREE PHASE
K1
SVC 2 FETCH $$BOSDC2
A4
READDISK
A5
WRITDISK
READDISK
B4
GET ADDRESS OF FIRST READ CCW
WRITDISK
B5
GET ADDRESS OF FIRST WRITE CCW
C4
INITIALIZE CCB WITH ADDRESS OF CCW CHAIN
D4
GET ADDRESS OF CCB FOR SUPERVISOR
E4
SVC 0 READ DISK
F4
I/O COMPLETED
NO
YES
F5
SVC 7 WAIT FOR I/O COMPLETED
G4
SET CONDITION CODE FOR RECORD FOUND
H4
RETURN TO CALLING ROUTINE

## Chart LD. $$BOSDC2: SD Close - Free Tracks Function


A1 ENTRY FROM $$BOSDC1
STARTFRE
B1 RELOCATE CCW ADDRESS IN CCB
C1 RELOCATE SEEK ADDRESS IN CCW
D1 GET ADDRESS OF TRACK HOLD TABLE
E1 GET NUMBER OF TRACK HOLD ENTRIES
F1 GET ADDRESS OF CCB FOR FREE ROUTINE
1
TRKLOOP
G1 CURRENT ENTRY HELD BY TASK
NO
YES
H1 IS CURRENT CCB CORRECT ONE
NO
YES
3
J1 SET COUNT OF TIMES TRACK WAS HELD TO ZERO
2
2
B2 SET ADDRESS OF TRACK BEING HELD IN DUMMY AREA
C2 SET SYMBOLIC UNIT ADDRESS OF TASK IN DUMMY CCB
D2 SVC 36 FREE TRACK
3
NOHOLD
E2 GET ADDRESS OF NEXT ENTRY IN TRACK HOLD TABLE
F2 DECREASE NUMBER OF TRACK HOLD ENTRIES BY 1
G2 END OF TABLE REACHED
NO
1
YES
H2 GET RETURN ADDRESS FOLLOWING AN OPEN
J2 IS THIS AN ISAM FILE
NO
4
YES
K2 SVC 2 RETURN TO $$BCISOA
4
H4 MORE FILES TO CLOSE
NO
YES
EXITUSER
H5 RESET MONITOR SWITCH
J4 SVC 2 FETCH $$BCLOSE
J5 RETURN TO PROBLEM PROGRAM

## Chart LE. $$BODQUE: Dequeue Extent JIBs


A1 $$BODQUE
DEQUERTN B1 SAVE REGISTERS 3 THROUGH 8
C1 INITIALIZE TO RETURN TO CALLING PHASE IF NAME IS SUPPLIED.
D1 SVC 22 SEIZE SYSTEM
SKIPQ4 E1 GET ADDRESSES COMM. RGN. FAVP JIB TABLE
F1 GET ADDRESS OF LUB
1
TESTPTR G1 POINTER TO JIB
NO
YES
3
H1 GET JIB ADDRESS
J1 EXTENT TYPE JIB
NO
YES
GETNEXT J2 REPLACE LUB ADDRESS WITH JIB ADDRESS
2
1
2
B3 CLEAR JIB
C3 SAVE FAVP POINTER
D3 UPDATE FAVP WITH POINTER TO JIB
E3 UPDATE POINTER TO JIB FROM POINTER TO NEXT JIB
F3 UPDATE JIB POINTER FROM FAVP SAVE
1
3
OUTRUTIN B5 SVC 22 RELEASE SYSTEM
C5 RETURN TO MONITOR
NO
YES
MONORTN D5 GET POINTER TO FILE BEING OPENED AND INCREMENT IT
E5 ANOTHER FILE TO OPEN
NO
YES
F4 TURN OFF OPEN BIT AND RESTORE REGISTERS
RESTORE F5 RESTORE REGISTERS AND LOAD FETCH NAME
G4 SVC 11 RETURN TO USER
G5 SVC 2 RETURN TO CALLING PHASE

## Chart LF. $$BOSDEV: Forced End of Volume for Disk (Section 1 of 2)

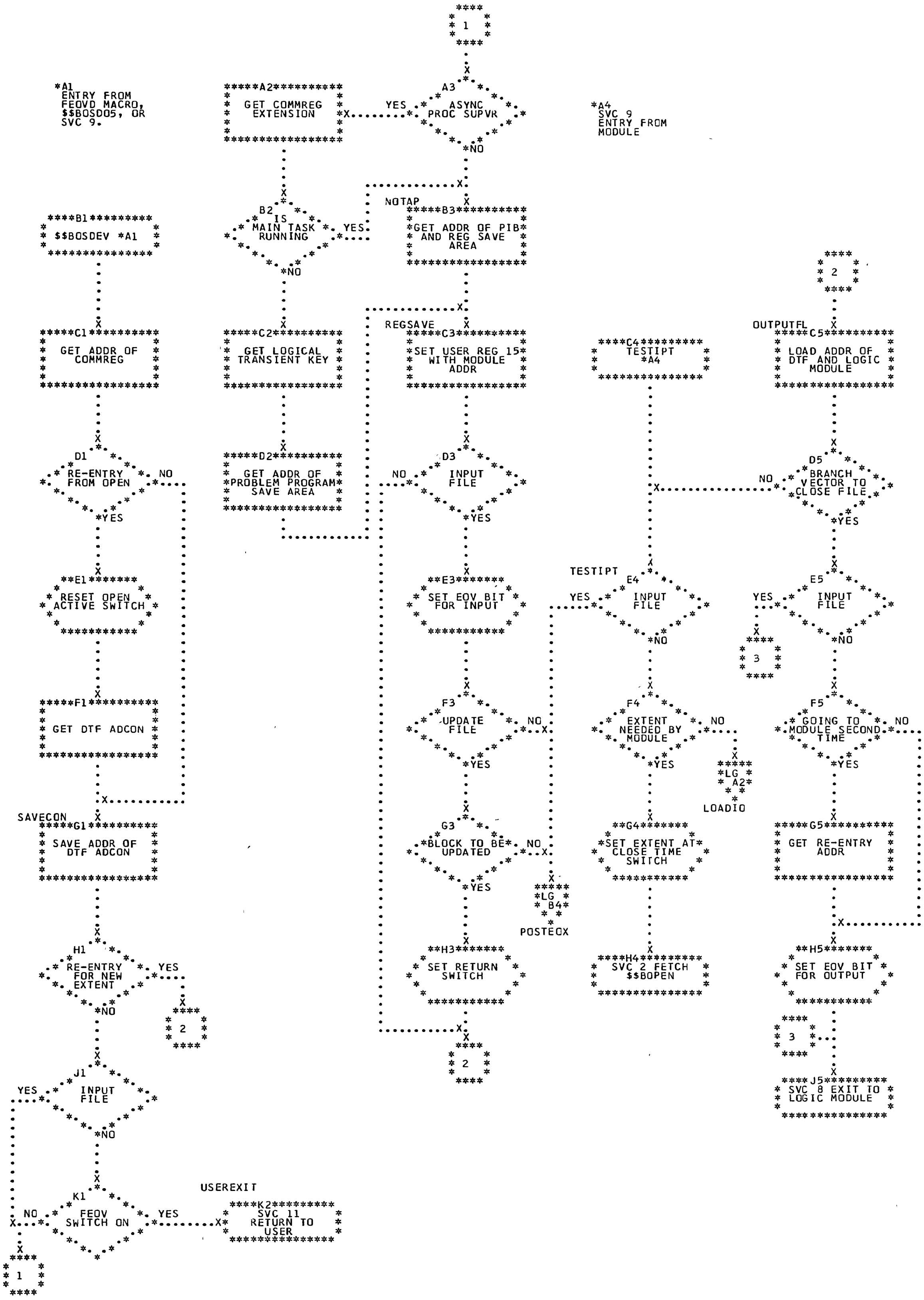


Chart LG. $$BOSDEV: Forced End of Volume for Disk (Section 2 of 2)

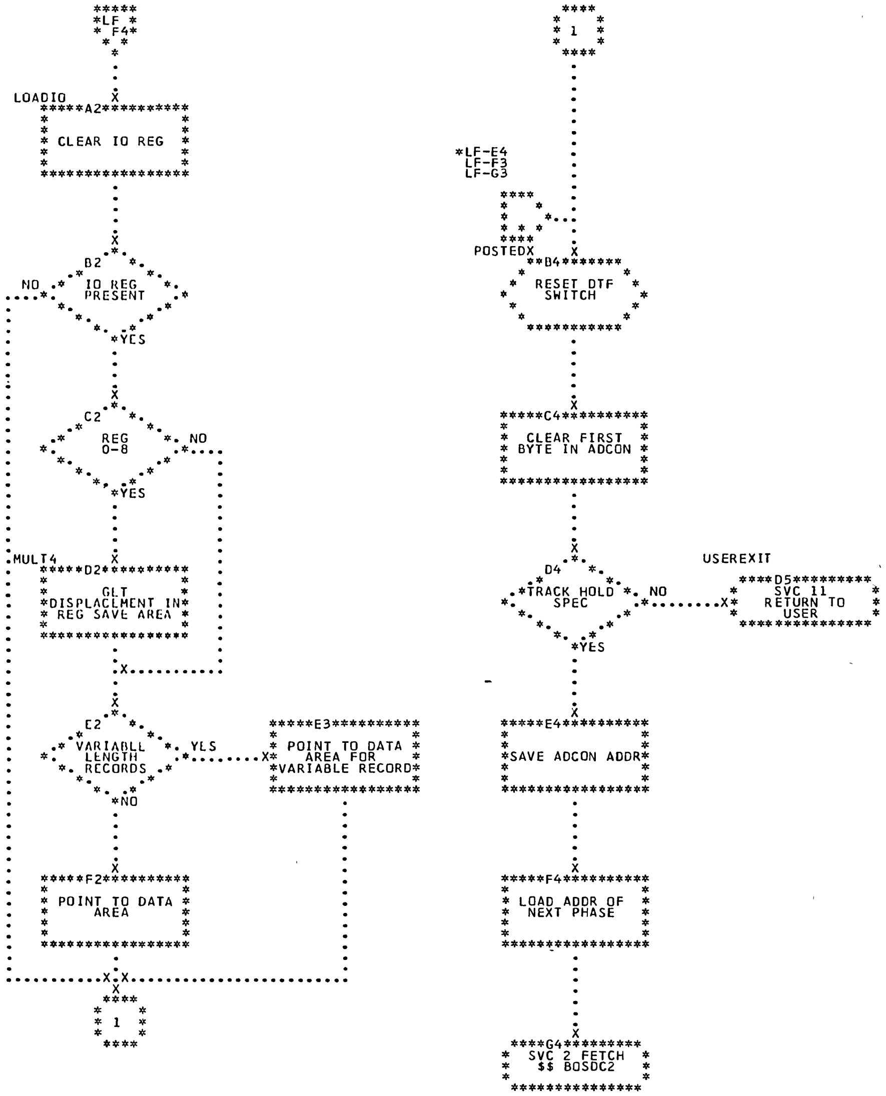
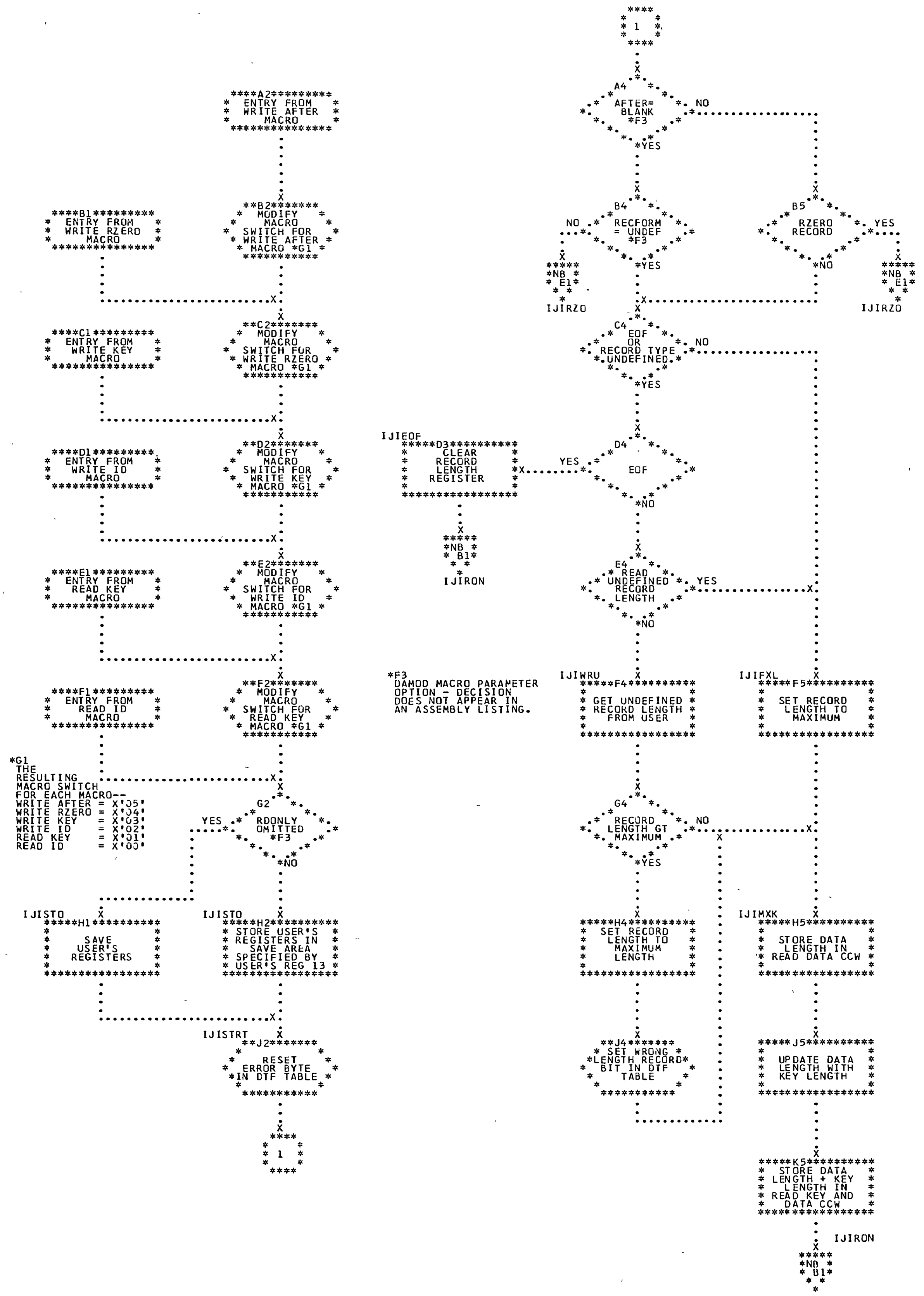


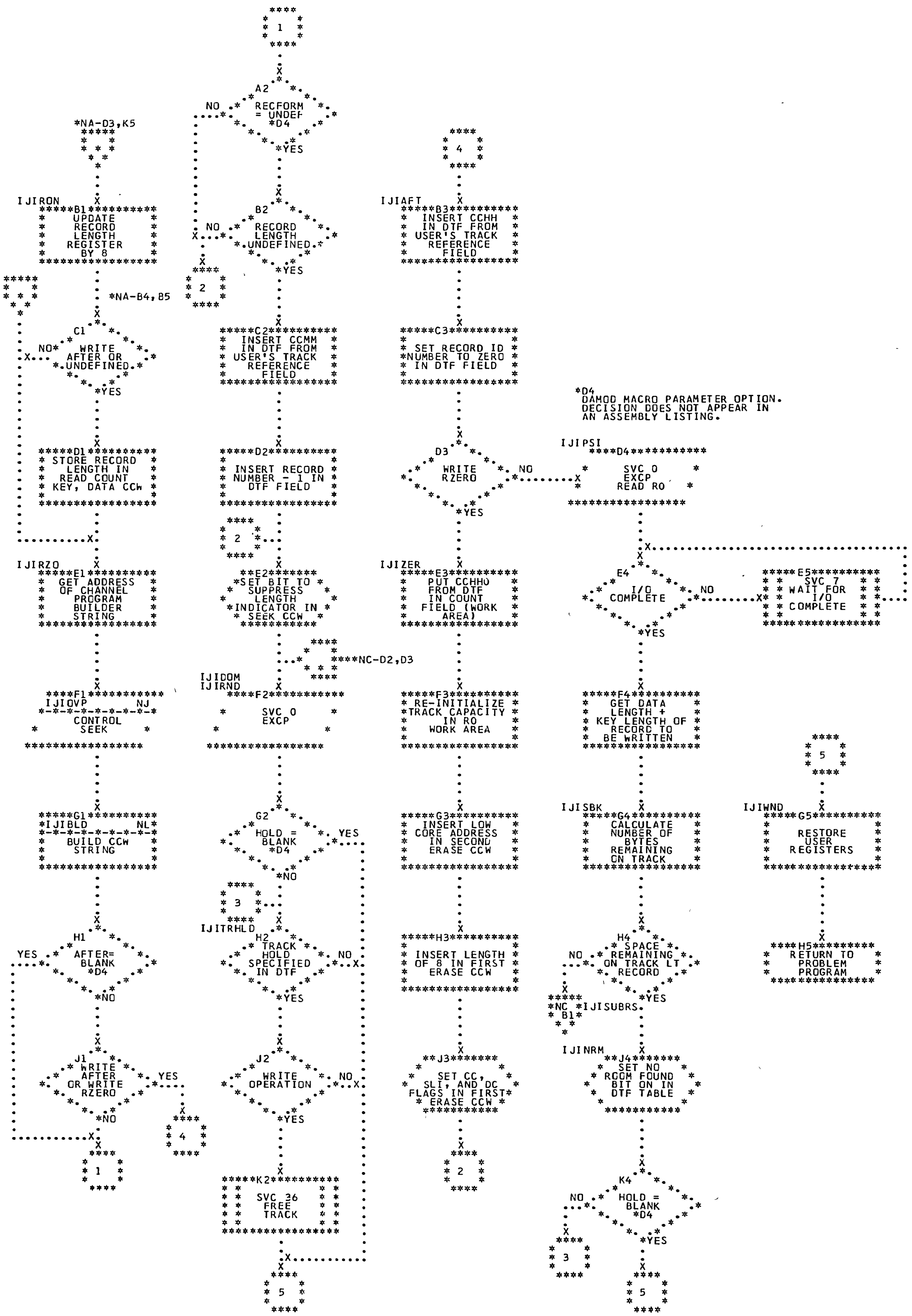
Chart NA. DAMOD: Input/Output Macros (Section 1 of 3)


A2
ENTRY FROM WRITE AFTER MACRO
B1
ENTRY FROM WRITE RZERO MACRO
B2
MODIFY MACRO SWITCH FOR WRITE AFTER MACRO *G1
C1
ENTRY FROM WRITE KEY MACRO
C2
MODIFY MACRO SWITCH FOR WRITE RZERO MACRO *G1
D1
ENTRY FROM WRITE ID MACRO
D2
MODIFY MACRO SWITCH FOR WRITE KEY MACRO *G1
E1
ENTRY FROM READ KEY MACRO
E2
MODIFY MACRO SWITCH FOR WRITE ID MACRO *G1
F1
ENTRY FROM READ ID MACRO
F2
MODIFY MACRO SWITCH FOR READ KEY MACRO *G1
G2
RDONLY OMITTED *F3
YES
NO
IJISTO
H1
SAVE USER'S REGISTERS
IJISTO
H2
STORE USER'S REGISTERS IN SAVE AREA SPECIFIED BY USER'S REG 13
IJISTRT
J2
RESET ERROR BYTE IN DTF TABLE
1
IJIEOF
D3
CLEAR RECORD LENGTH REGISTER
NB B1
IJIRON
1
A4
AFTER= BLANK *F3
NO
YES
B4
RECFORM = UNDEF *F3
NO
YES
NB E1
IJIRZO
B5
RZERO RECORD
YES
NO
NB E1
IJIRZO
C4
EOF OR RECORD TYPE UNDEFINED
NO
YES
D4
EOF
YES
NO
E4
READ UNDEFINED RECORD LENGTH
YES
NO
IJIWRU
F4
GET UNDEFINED RECORD LENGTH FROM USER
IJIFXL
F5
SET RECORD LENGTH TO MAXIMUM
G4
RECORD LENGTH GT MAXIMUM
NO
YES
H4
SET RECORD LENGTH TO MAXIMUM LENGTH
IJIMXK
H5
STORE DATA LENGTH IN READ DATA CCW
J4
SET WRONG LENGTH RECORD BIT IN DTF TABLE
J5
UPDATE DATA LENGTH WITH KEY LENGTH
K5
STORE DATA LENGTH + KEY LENGTH IN READ KEY AND DATA CCW
IJIRON
NB B1

*G1
THE RESULTING MACRO SWITCH FOR EACH MACRO--
WRITE AFTER = X'05'
WRITE RZERO = X'04'
WRITE KEY = X'03'
WRITE ID = X'02'
READ KEY = X'01'
READ ID = X'00'

*F3
DAMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.

## Chart NB. DAMOD: Input/Output Macros (Section 2 of 3)


1
A2
RECFORM = UNDEF *D4
NO
YES
*NA-D3,K5
IJIRON
B1
UPDATE RECORD LENGTH REGISTER BY 8
*NA-B4,B5
C1
WRITE AFTER OR UNDEFINED
NO
YES
D1
STORE RECORD LENGTH IN READ COUNT KEY, DATA CCW
IJIRZO
E1
GET ADDRESS OF CHANNEL PROGRAM BUILDER STRING
F1
IJIOVP NJ
CONTROL SEEK
G1
IJIBLD NL
BUILD CCW STRING
H1
AFTER= BLANK *D4
YES
NO
J1
WRITE AFTER OR WRITE RZERO
YES
NO
4
1
B2
RECORD LENGTH UNDEFINED
NO
YES
2
C2
INSERT CCMM IN DTF FROM USER'S TRACK REFERENCE FIELD
D2
INSERT RECORD NUMBER - 1 IN DTF FIELD
2
E2
SET BIT TO SUPPRESS LENGTH INDICATOR IN SEEK CCW
NC-D2,D3
IJIDOM
IJIRND
F2
SVC 0 EXCP
G2
HOLD = BLANK *D4
YES
NO
3
IJITRHLD
H2
TRACK HOLD SPECIFIED IN DTF
NO
YES
J2
WRITE OPERATION
NO
YES
K2
SVC 36 FREE TRACK
5
4
IJIAFT
B3
INSERT CCHH IN DTF FROM USER'S TRACK REFERENCE FIELD
C3
SET RECORD ID NUMBER TO ZERO IN DTF FIELD
D3
WRITE RZERO
NO
YES
IJIZER
E3
PUT CCHHO FROM DTF IN COUNT FIELD (WORK AREA)
F3
RE-INITIALIZE TRACK CAPACITY IN RO WORK AREA
G3
INSERT LOW CORE ADDRESS IN SECOND ERASE CCW
H3
INSERT LENGTH OF 8 IN FIRST ERASE CCW
J3
SET CC, SLI, AND DC FLAGS IN FIRST ERASE CCW
2
IJIPSI
D4
SVC 0 EXCP READ RO
E4
I/O COMPLETE
NO
YES
F4
GET DATA LENGTH + KEY LENGTH OF RECORD TO BE WRITTEN
IJISBK
G4
CALCULATE NUMBER OF BYTES REMAINING ON TRACK
H4
SPACE REMAINING ON TRACK LT RECORD
NO
YES
NC B1
IJISUBRS
IJINRM
J4
SET NO ROOM FOUND BIT ON IN DTF TABLE
K4
HOLD = BLANK *D4
NO
YES
3
5
E5
SVC 7 WAIT FOR I/O COMPLETE
5
IJIWND
G5
RESTORE USER REGISTERS
H5
RETURN TO PROBLEM PROGRAM

*D4
DAMOD MACRO PARAMETER OPTION. DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.

## Chart NC. DAMOD: Input/Output Macros (Section 3 of 3)

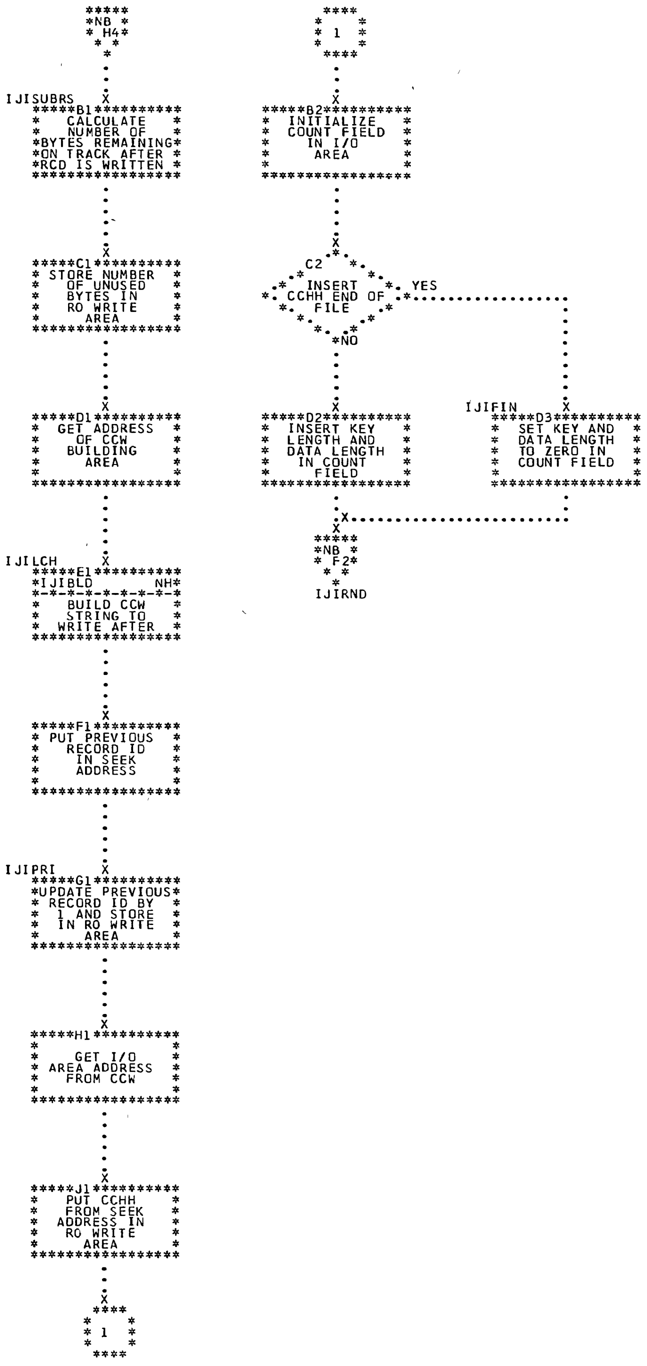
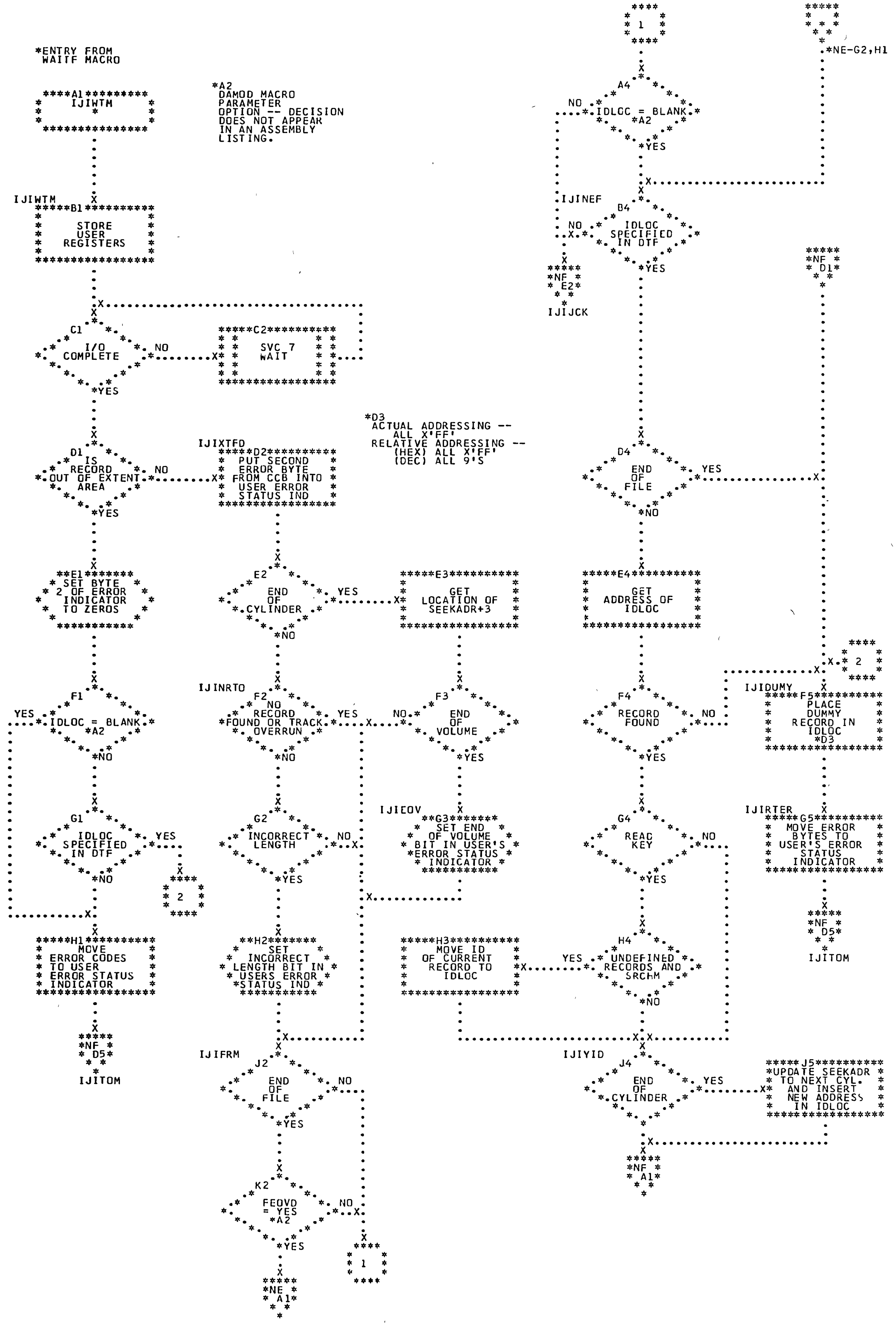
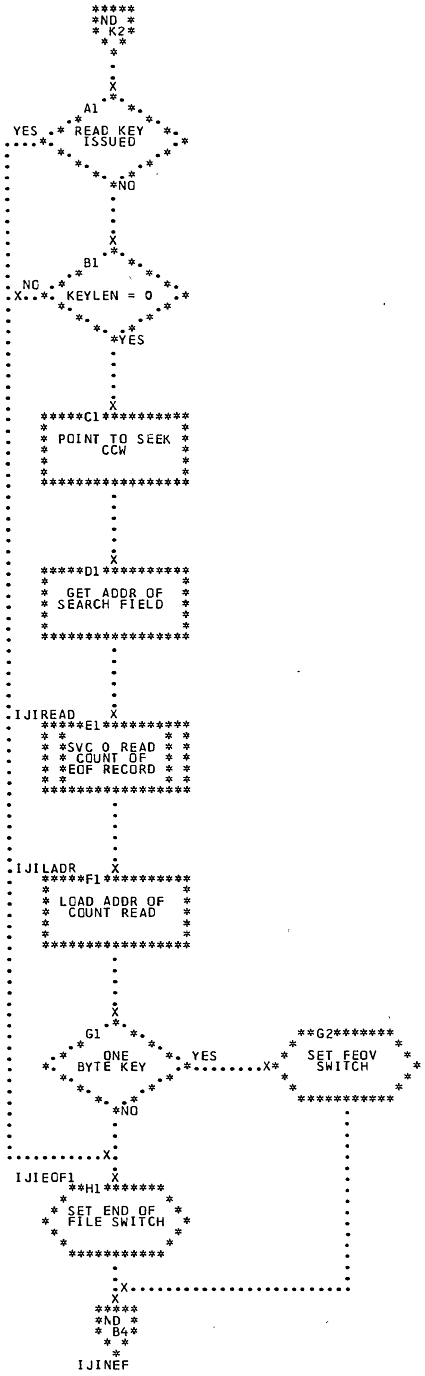

NB H4
IJISUBRS
B1
CALCULATE NUMBER OF BYTES REMAINING ON TRACK AFTER RCD IS WRITTEN
C1
STORE NUMBER OF UNUSED BYTES IN RO WRITE AREA
D1
GET ADDRESS OF CCW BUILDING AREA
IJILCH
E1
IJIBLD NH
BUILD CCW STRING TO WRITE AFTER
F1
PUT PREVIOUS RECORD ID IN SEEK ADDRESS
IJIPRI
G1
UPDATE PREVIOUS RECORD ID BY 1 AND STORE IN RO WRITE AREA
H1
GET I/O AREA ADDRESS FROM CCW
J1
PUT CCHH FROM SEEK ADDRESS IN RO WRITE AREA
1
1
B2
INITIALIZE COUNT FIELD IN I/O AREA
C2
INSERT CCHH END OF FILE
YES
NO
D2
INSERT KEY LENGTH AND DATA LENGTH IN COUNT FIELD
IJIFIN
D3
SET KEY AND DATA LENGTH TO ZERO IN COUNT FIELD
NB F2
IJIRND

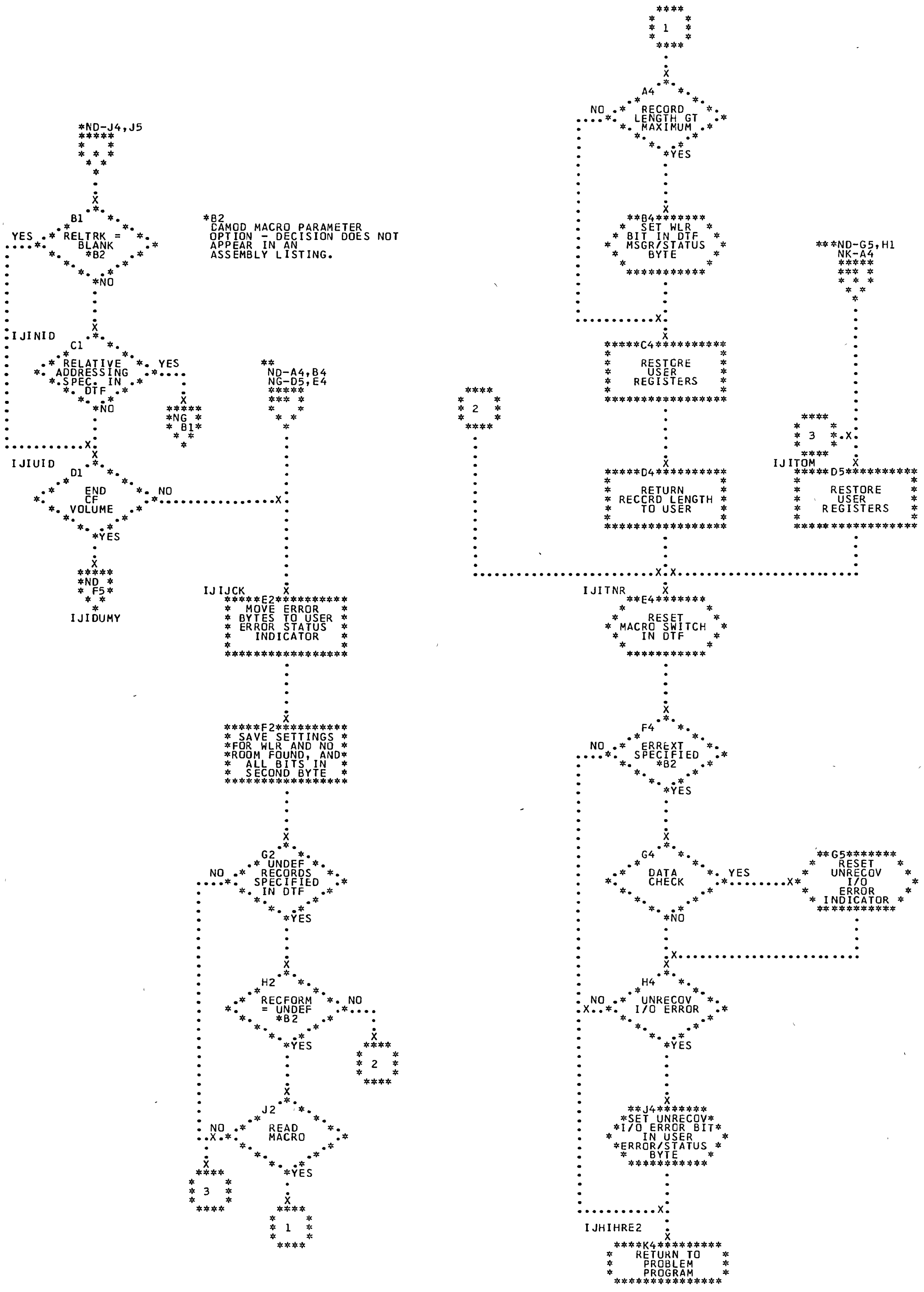
Chart ND. DAMOD: WAITF Macro (Section 1 of 4)


*ENTRY FROM WAITF MACRO
A1 IJIWTM
*A2 DAMOD MACRO PARAMETER OPTION -- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
IJIWTM B1 STORE USER REGISTERS
C1 I/O COMPLETE
NO
YES
C2 SVC 7 WAIT
D1 IS RECORD OUT OF EXTENT AREA
IJIXTFD D2 PUT SECOND ERROR BYTE FROM CCB INTO USER ERROR STATUS IND
*D3 ACTUAL ADDRESSING -- ALL X'FF' RELATIVE ADDRESSING -- (HEX) ALL X'FF' (DEC) ALL 9'S
E1 SET BYTE 2 OF ERROR INDICATOR TO ZEROS
E2 END OF CYLINDER
E3 GET LOCATION OF SEEKADR+3
F1 IDLOC = BLANK *A2
IJINRTO F2 NO RECORD FOUND OR TRACK OVERRUN
F3 END OF VOLUME
G1 IDLOC SPECIFIED IN DTF
2
G2 INCORRECT LENGTH
IJIEOV G3 SET END OF VOLUME BIT IN USER'S ERROR STATUS INDICATOR
H1 MOVE ERROR CODES TO USER ERROR STATUS INDICATOR
H2 SET INCORRECT LENGTH BIT IN USERS ERROR STATUS IND
H3 MOVE ID OF CURRENT RECORD TO IDLOC
NF D5
IJITOM
IJIFRM J2 END OF FILE
K2 FEOVD = YES *A2
NE A1
1
1
A4 IDLOC = BLANK *A2
NE-G2,H1
IJINEF B4 IDLOC SPECIFIED IN DTF
NF E2
IJIJCK
NF D1
D4 END OF FILE
E4 GET ADDRESS OF IDLOC
2
IJIDUMY F5 PLACE DUMMY RECORD IN IDLOC *D3
F4 RECORD FOUND
G4 READ KEY
IJIRTER G5 MOVE ERROR BYTES TO USER'S ERROR STATUS INDICATOR
NF D5
IJITOM
H4 UNDEFINED RECORDS AND SRCHM
IJIYID J4 END OF CYLINDER
J5 UPDATE SEEKADR TO NEXT CYL. AND INSERT NEW ADDRESS IN IDLOC
NF A1

## Chart NE. DAMOD: WAITF Macro (Section 2 of 4)

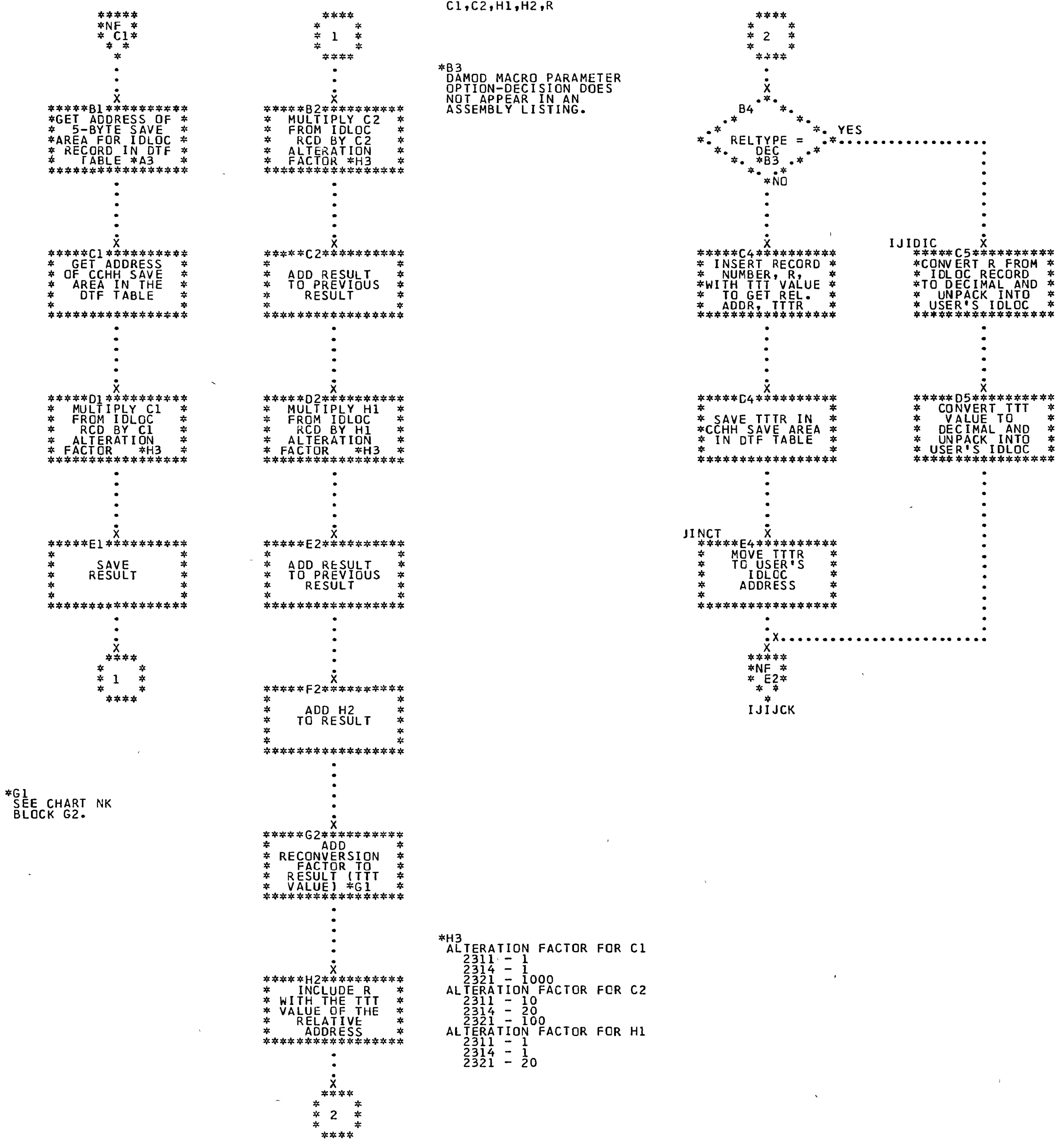


## Chart NF. DAMOD: WAITF Macro (Section 3 of 4)


*ND-J4,J5
B1
RELTRK = BLANK *B2
YES
NO
*B2
DAMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
IJINID
C1
RELATIVE ADDRESSING SPEC. IN DTF
YES
NO
NG B1
ND-A4,B4
NG-D5,E4
IJIUID
D1
END CF VOLUME
NO
YES
ND F5
IJIDUMY
IJIJCK
E2
MOVE ERROR BYTES TO USER ERROR STATUS INDICATOR
F2
SAVE SETTINGS FOR WLR AND NO ROOM FOUND, AND ALL BITS IN SECOND BYTE
G2
UNDEF RECORDS SPECIFIED IN DTF
NO
YES
H2
RECFORM = UNDEF *B2
NO
YES
2
J2
READ MACRO
NO
YES
3
1
1
A4
RECORD LENGTH GT MAXIMUM
NO
YES
B4
SET WLR BIT IN DTF MSGR/STATUS BYTE
C4
RESTCRE USER REGISTERS
2
D4
RETURN RECCRD LENGTH TO USER
*ND-G5,H1
NK-A4
3
IJITOM
D5
RESTORE USER REGISTERS
IJITNR
E4
RESET MACRO SWITCH IN DTF
F4
ERREXT SPECIFIED *B2
NO
YES
G4
DATA CHECK
YES
NO
G5
RESET UNRECOV I/O ERROR INDICATOR
H4
UNRECOV I/O ERROR
NO
YES
J4
SET UNRECOV I/O ERROR BIT IN USER ERROR/STATUS BYTE
IJHIHRE2
K4
RETURN TO PROBLEM PROGRAM

## Chart NG. DAMOD: WAITF Macro (Section 4 of 4)

NOTE - THIS ROUTINE CONVERTS AN ACTUAL DASD CCHHR ADDRESS TO A RELATIVE ADDRESS.



NF C1
GET ADDRESS OF 5-BYTE SAVE AREA FOR IDLOC RECORD IN DTF TABLE *A3
GET ADDRESS OF CCHH SAVE AREA IN THE DTF TABLE
MULTIPLY C1 FROM IDLOC RCD BY C1 ALTERATION FACTOR *H3
SAVE RESULT
1
1
MULTIPLY C2 FROM IDLOC RCD BY C2 ALTERATION FACTOR *H3
ADD RESULT TO PREVIOUS RESULT
MULTIPLY H1 FROM IDLOC RCD BY H1 ALTERATION FACTOR *H3
ADD RESULT TO PREVIOUS RESULT
ADD H2 TO RESULT
ADD RECONVERSION FACTOR TO RESULT (TTT VALUE) *G1
INCLUDE R WITH THE TTT VALUE OF THE RELATIVE ADDRESS
2
2
B4
RELTYPE = DEC *B3
YES
NO
C4
INSERT RECORD NUMBER, R, WITH TTT VALUE TO GET REL. ADDR, TTTR
D4
SAVE TTTR IN CCHH SAVE AREA IN DTF TABLE
JINCT
E4
MOVE TTTR TO USER'S IDLOC ADDRESS
IJIDIC
C5
CONVERT R FROM IDLOC RECORD TO DECIMAL AND UNPACK INTO USER'S IDLOC
D5
CONVERT TTT VALUE TO DECIMAL AND UNPACK INTO USER'S IDLOC
NF E2
IJIJCK
B1
C1
D1
E1
B2
C2
D2
E2
F2
G2
H2

*A3
IDLOC RECORD IN FORM-
C1,C2,H1,H2,R

*B3
DAMOD MACRO PARAMETER OPTION-DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.

*G1
SEE CHART NK BLOCK G2.

*H3
ALTERATION FACTOR FOR C1
2311 - 1
2314 - 1
2321 - 1000
ALTERATION FACTOR FOR C2
2311 - 10
2314 - 20
2321 - 100
ALTERATION FACTOR FOR H1
2311 - 1
2314 - 1
2321 - 20

## Chart NH. DAMOD: CNTRL and FREE Macros

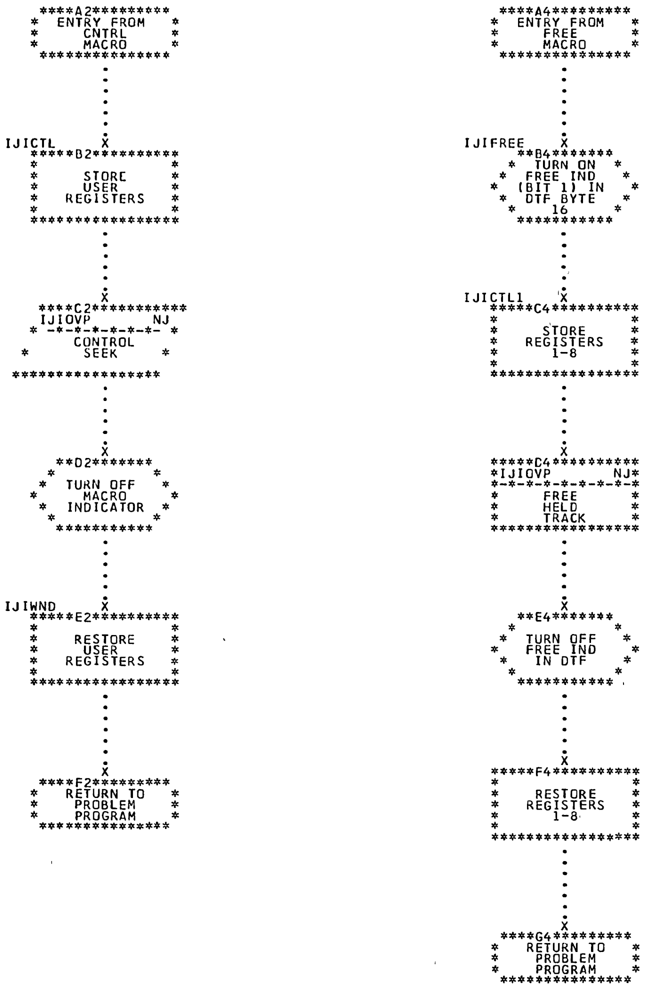


Chart NJ. DAMOD: Seek Overlap Subroutine (Section 1 of 2)

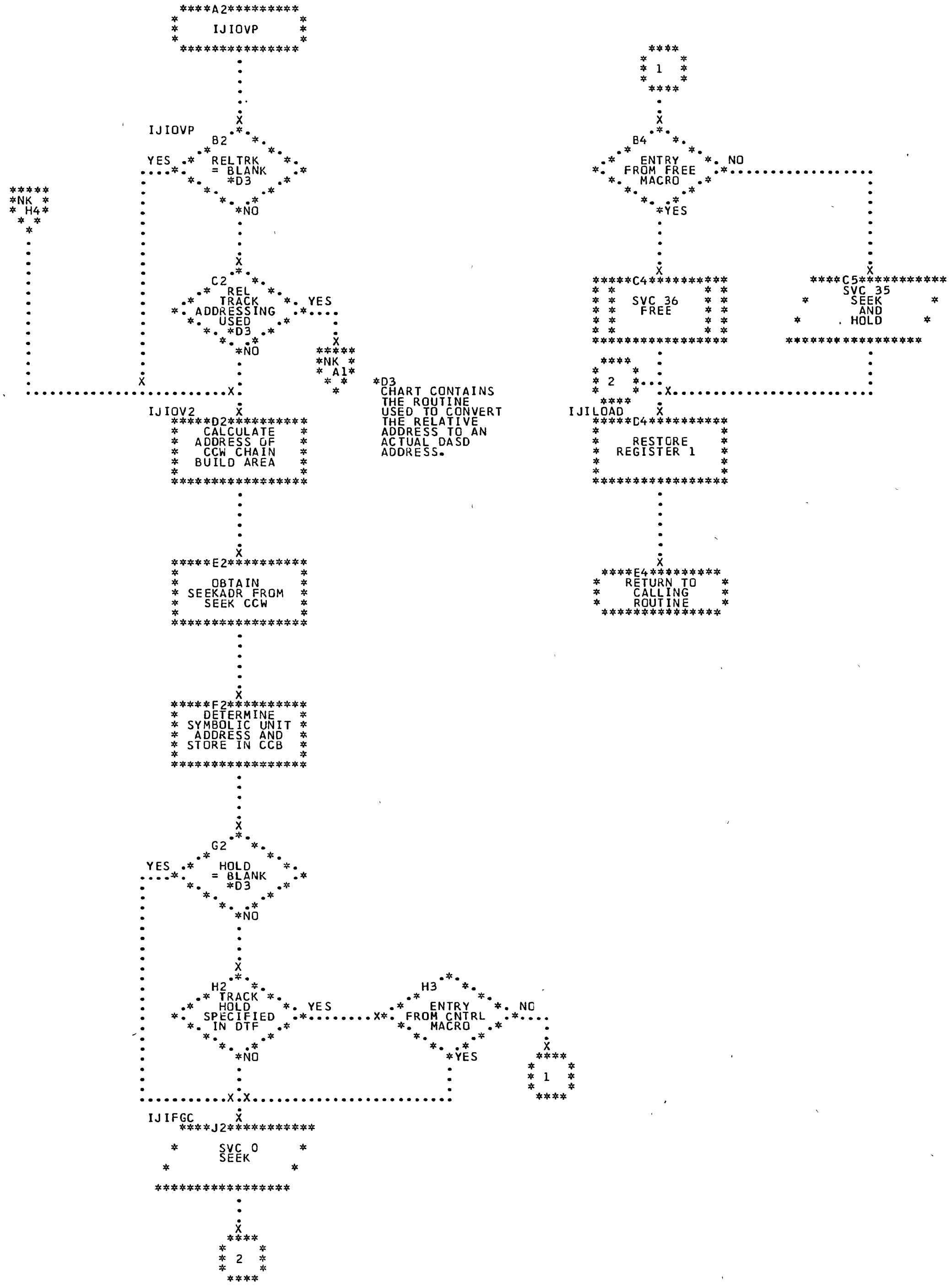
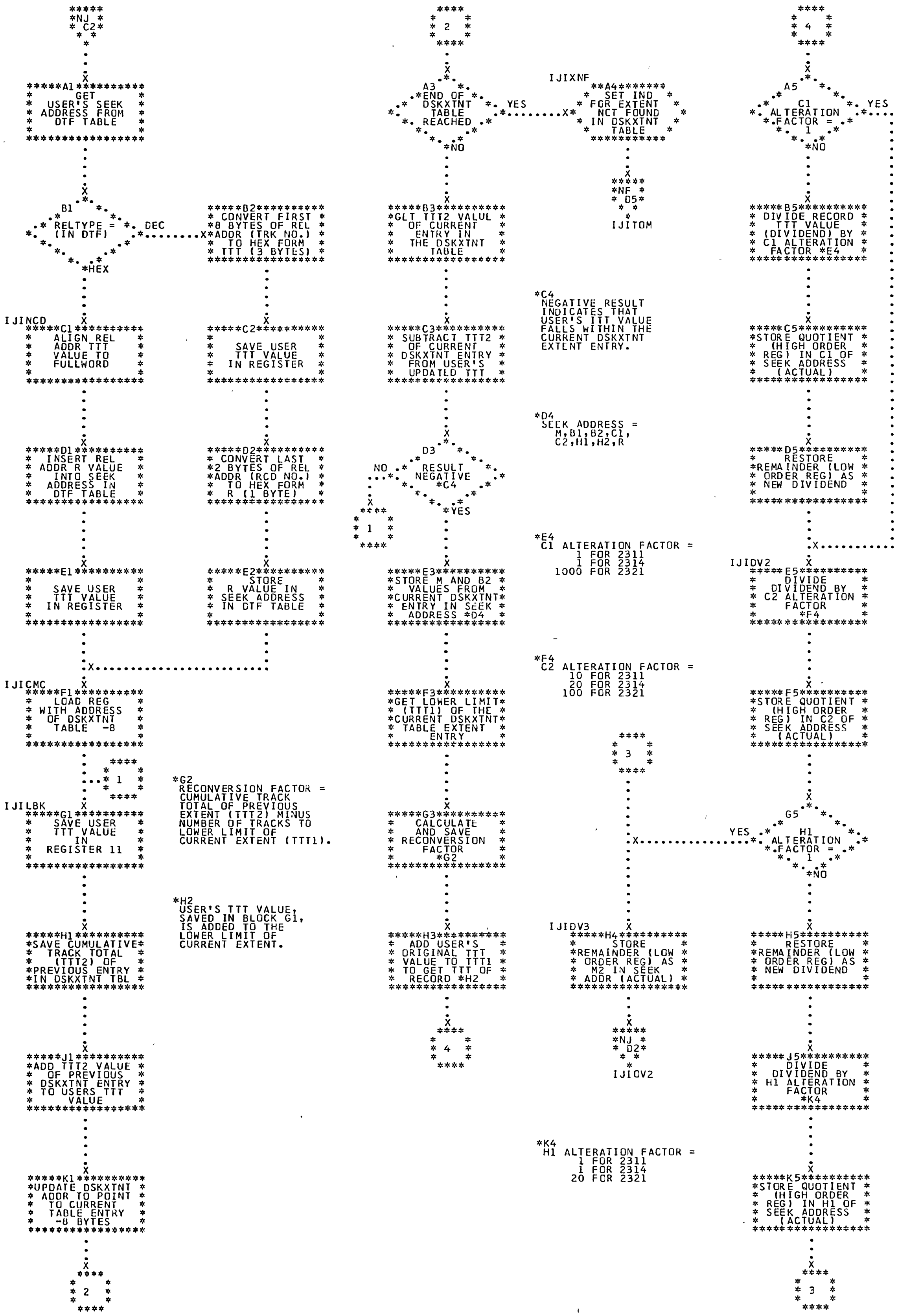
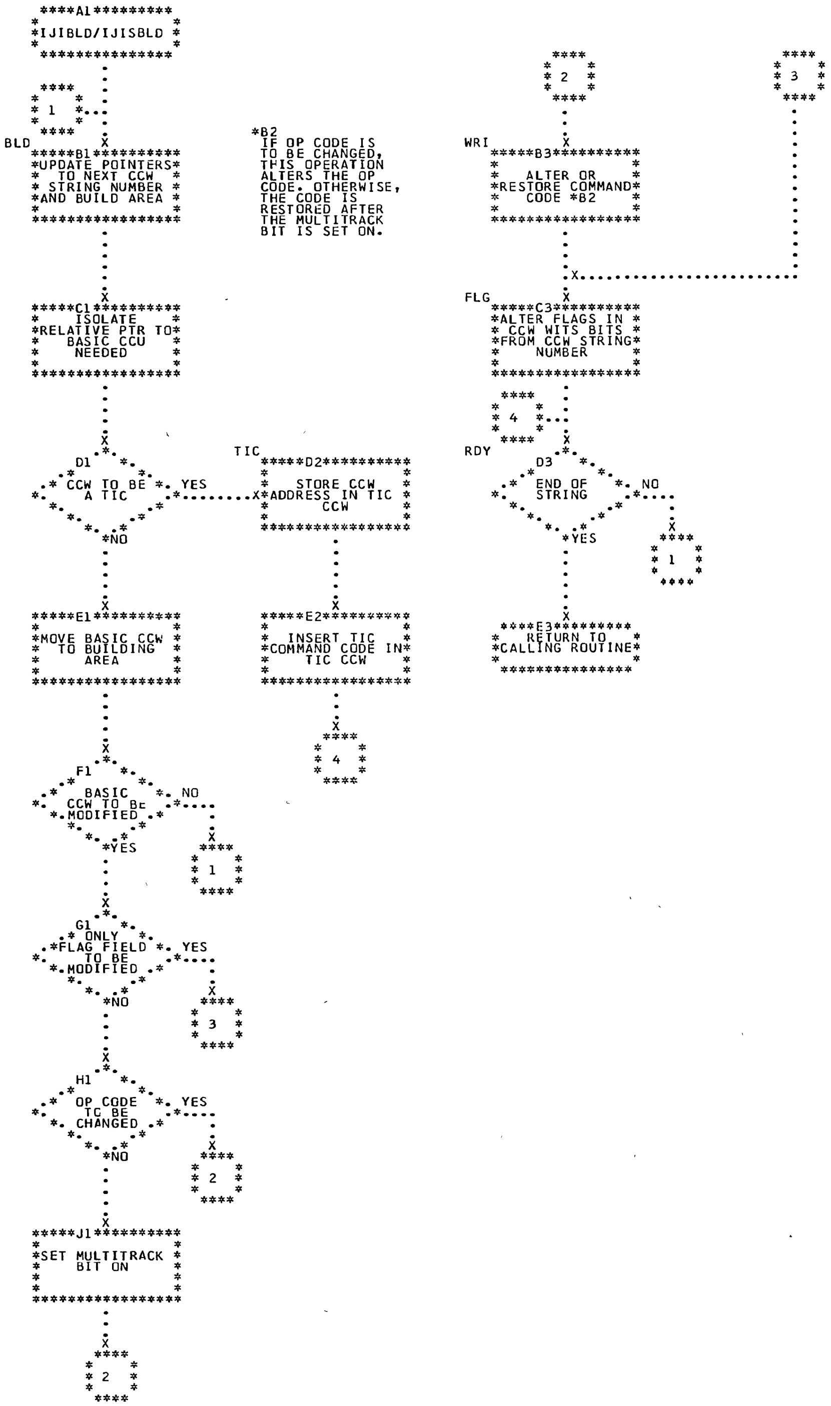

A2
IJIOVP
IJIOVP
B2
RELTRK = BLANK *D3
YES
NO
NK H4
C2
REL TRACK ADDRESSING USED *D3
YES
NO
NK A1
*D3 CHART CONTAINS THE ROUTINE USED TO CONVERT THE RELATIVE ADDRESS TO AN ACTUAL DASD ADDRESS.
IJIOV2
D2
CALCULATE ADDRESS OF CCW CHAIN BUILD AREA
E2
OBTAIN SEEKADR FROM SEEK CCW
F2
DETERMINE SYMBOLIC UNIT ADDRESS AND STORE IN CCB
G2
HOLD = BLANK *D3
YES
NO
H2
TRACK HOLD SPECIFIED IN DTF
YES
NO
H3
ENTRY FROM CNTRL MACRO
NO
YES
1
IJIFGC
J2
SVC 0 SEEK
2
1
B4
ENTRY FROM FREE MACRO
NO
YES
C4
SVC 36 FREE
C5
SVC 35 SEEK AND HOLD
2
IJILOAD
C4
RESTORE REGISTER 1
E4
RETURN TO CALLING ROUTINE

Chart NK. DAMOD: Seek Overlap Subroutine (Section 2 of 2)

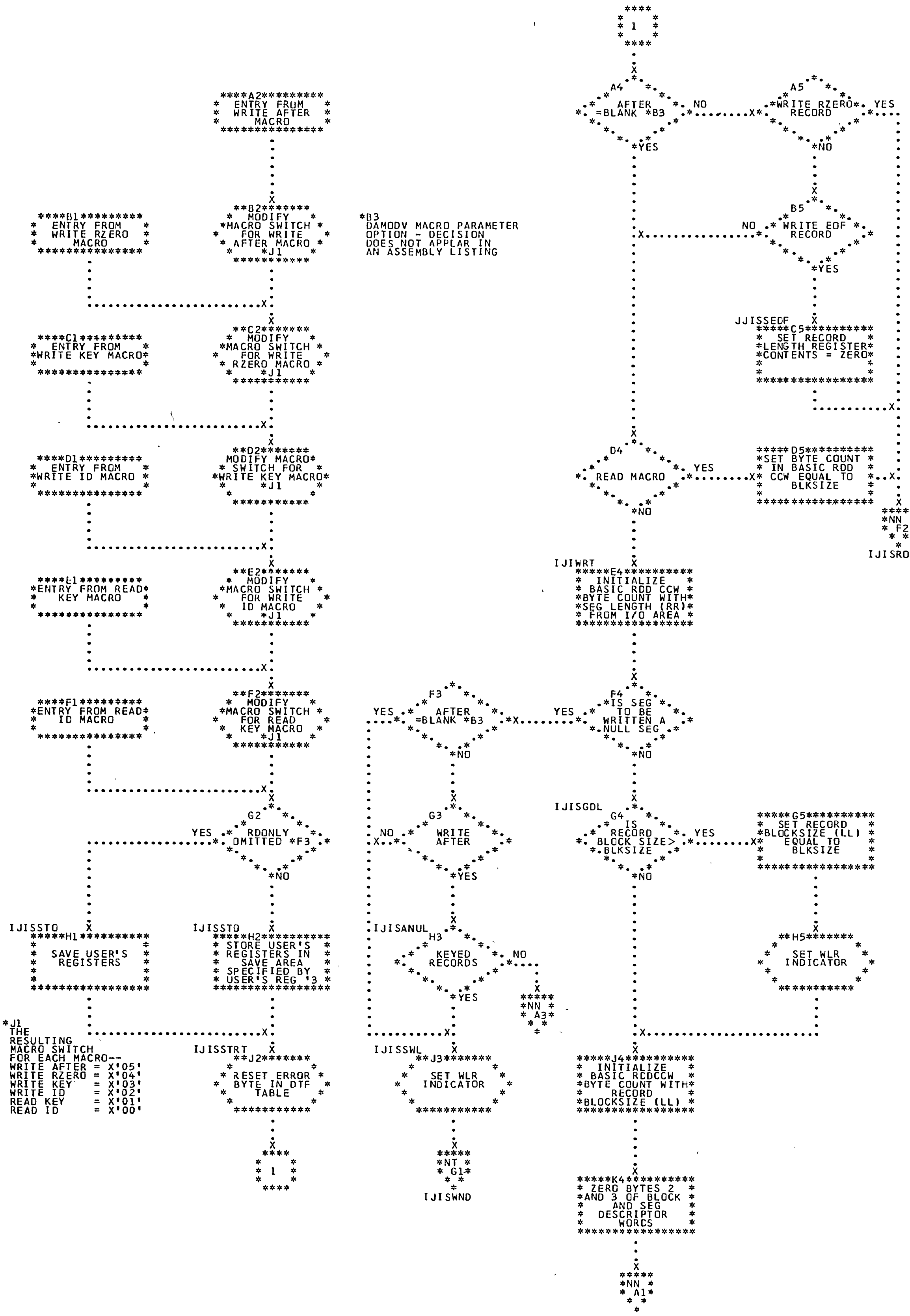

NJ C2
A1 GET USER'S SEEK ADDRESS FROM DTF TABLE
B1 RELTYPE = (IN DTF)
DEC
HEX
B2 CONVERT FIRST 8 BYTES OF REL ADDR (TRK NO.) TO HEX FORM TTT (3 BYTES)
IJINCD
C1 ALIGN REL ADDR TTT VALUE TO FULLWORD
C2 SAVE USER TTT VALUE IN REGISTER
D1 INSERT REL ADDR R VALUE INTO SEEK ADDRESS IN DTF TABLE
D2 CONVERT LAST 2 BYTES OF REL ADDR (RCD NO.) TO HEX FORM R (1 BYTE)
E1 SAVE USER TTT VALUE IN REGISTER
E2 STORE R VALUE IN SEEK ADDRESS IN DTF TABLE
IJICMC
F1 LOAD REG WITH ADDRESS OF DSKXTNT TABLE -8
1
IJILBK
G1 SAVE USER TTT VALUE IN REGISTER 11
*G2 RECONVERSION FACTOR = CUMULATIVE TRACK TOTAL OF PREVIOUS EXTENT (TTT2) MINUS NUMBER OF TRACKS TO LOWER LIMIT OF CURRENT EXTENT (TTT1).
H1 SAVE CUMULATIVE TRACK TOTAL (TTT2) OF PREVIOUS ENTRY IN DSKXTNT TBL
*H2 USER'S TTT VALUE, SAVED IN BLOCK G1, IS ADDED TO THE LOWER LIMIT OF CURRENT EXTENT.
J1 ADD TTT2 VALUE OF PREVIOUS DSKXTNT ENTRY TO USERS TTT VALUE
K1 UPDATE DSKXTNT ADDR TO POINT TO CURRENT TABLE ENTRY -8 BYTES
2
2
A3 END OF DSKXTNT TABLE REACHED
YES
NO
IJIXNF
A4 SET IND FOR EXTENT NCT FOUND IN DSKXTNT TABLE
NF D5
IJITOM
B3 GET TTT2 VALUE OF CURRENT ENTRY IN THE DSKXTNT TABLE
*C4 NEGATIVE RESULT INDICATES THAT USER'S TTT VALUE FALLS WITHIN THE CURRENT DSKXTNT EXTENT ENTRY.
C3 SUBTRACT TTT2 OF CURRENT DSKXTNT ENTRY FROM USER'S UPDATED TTT
*D4 SEEK ADDRESS = M,B1,B2,C1, C2,H1,H2,R
D3 RESULT NEGATIVE *C4
NO
YES
1
*E4 C1 ALTERATION FACTOR = 1 FOR 2311 1 FOR 2314 1000 FOR 2321
E3 STORE M AND B2 VALUES FROM CURRENT DSKXTNT ENTRY IN SEEK ADDRESS *D4
*F4 C2 ALTERATION FACTOR = 10 FOR 2311 20 FOR 2314 100 FOR 2321
F3 GET LOWER LIMIT (TTT1) OF THE CURRENT DSKXTNT TABLE EXTENT ENTRY
3
G3 CALCULATE AND SAVE RECONVERSION FACTOR *G2
IJIDV3
H3 ADD USER'S ORIGINAL TTT VALUE TO TTT1 TO GET TTT OF RECORD *H2
H4 STORE REMAINDER (LOW ORDER REG) AS M2 IN SEEK ADDR (ACTUAL)
4
NJ D2
IJIOV2
*K4 H1 ALTERATION FACTOR = 1 FOR 2311 1 FOR 2314 20 FOR 2321
4
A5 C1 ALTERATION FACTOR = 1
YES
NO
B5 DIVIDE RECORD TTT VALUE (DIVIDEND) BY C1 ALTERATION FACTOR *E4
C5 STORE QUOTIENT (HIGH ORDER REG) IN C1 OF SEEK ADDRESS (ACTUAL)
D5 RESTORE REMAINDER (LOW ORDER REG) AS NEW DIVIDEND
IJIDV2
E5 DIVIDE DIVIDEND BY C2 ALTERATION FACTOR *F4
F5 STORE QUOTIENT (HIGH ORDER REG) IN C2 OF SEEK ADDRESS (ACTUAL)
G5 H1 ALTERATION FACTOR = 1
YES
NO
H5 RESTORE REMAINDER (LOW ORDER REG) AS NEW DIVIDEND
J5 DIVIDE DIVIDEND BY H1 ALTERATION FACTOR *K4
K5 STORE QUOTIENT (HIGH ORDER REG) IN H1 OF SEEK ADDRESS (ACTUAL)
3

## Chart NL. DAMOD and DAMODV: Channel Program Builder Subroutine

NOTE--THIS SUBROUTINE IS USED
IN BOTH DAMOD AND DAMODV.
THE LABELS SHOWN ON THIS
FLOWCHART ARE PRECEDED BY
ONE OF THE PREFIXES--
DAMOD-IJI
DAMODV-IJIS


A1
IJIBLD/IJISBLD
1
BLD
B1
UPDATE POINTERS TO NEXT CCW STRING NUMBER AND BUILD AREA
*B2
IF OP CODE IS TO BE CHANGED, THIS OPERATION ALTERS THE OP CODE. OTHERWISE, THE CODE IS RESTORED AFTER THE MULTITRACK BIT IS SET ON.
C1
ISOLATE RELATIVE PTR TO BASIC CCU NEEDED
D1
CCW TO BE A TIC
YES
NO
TIC
D2
STORE CCW ADDRESS IN TIC CCW
E1
MOVE BASIC CCW TO BUILDING AREA
E2
INSERT TIC COMMAND CODE IN TIC CCW
4
F1
BASIC CCW TO BE MODIFIED
NO
YES
1
G1
ONLY FLAG FIELD TO BE MODIFIED
YES
NO
3
H1
OP CODE TC BE CHANGED
YES
NO
2
J1
SET MULTITRACK BIT ON
2
2
WRI
B3
ALTER OR RESTORE COMMAND CODE *B2
3
FLG
C3
ALTER FLAGS IN CCW WITS BITS FROM CCW STRING NUMBER
4
RDY
D3
END OF STRING
NO
YES
1
E3
RETURN TO CALLING ROUTINE

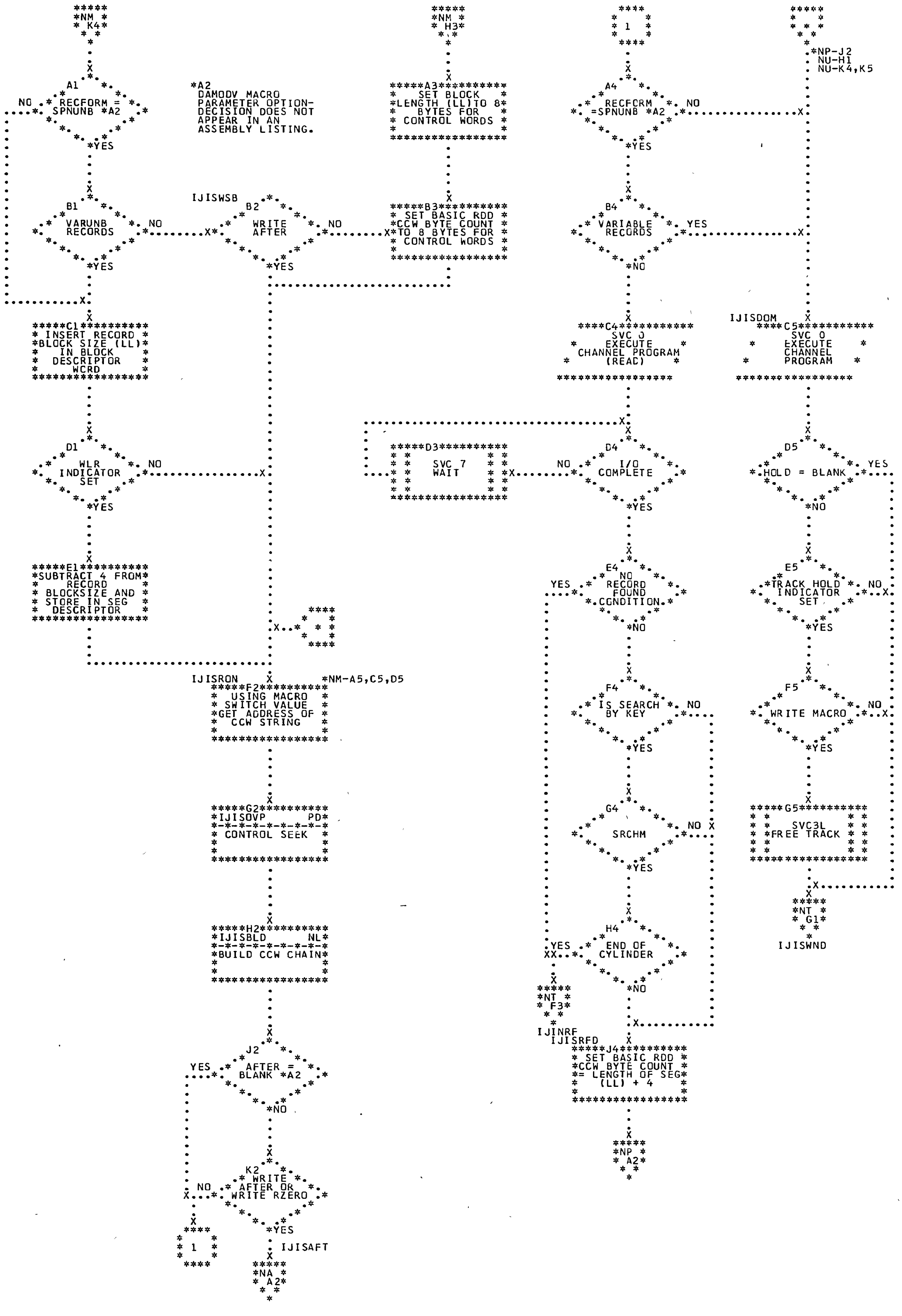
## Chart NM. DAMODV: Input/Output Macros (Section 1 of 11)


A2 ENTRY FROM WRITE AFTER MACRO
B1 ENTRY FROM WRITE RZERO MACRO
B2 MODIFY MACRO SWITCH FOR WRITE AFTER MACRO *J1
C1 ENTRY FROM WRITE KEY MACRO
C2 MODIFY MACRO SWITCH FOR WRITE RZERO MACRO *J1
D1 ENTRY FROM WRITE ID MACRO
D2 MODIFY MACRO SWITCH FOR WRITE KEY MACRO *J1
E1 ENTRY FROM READ KEY MACRO
E2 MODIFY MACRO SWITCH FOR WRITE ID MACRO *J1
F1 ENTRY FROM READ ID MACRO
F2 MODIFY MACRO SWITCH FOR READ KEY MACRO *J1
G2 RDONLY OMITTED *F3
YES
NO
IJISSTO
H1 SAVE USER'S REGISTERS
IJISSTO
H2 STORE USER'S REGISTERS IN SAVE AREA SPECIFIED BY USER'S REG '3
IJISSTRT
J2 RESET ERROR BYTE IN DTF TABLE
1
F3 AFTER =BLANK *B3
YES
NO
G3 WRITE AFTER
NO
YES
IJISANUL
H3 KEYED RECORDS
NO
NN A3
YES
IJISSWL
J3 SET WLR INDICATOR
NT G1
IJISWND
1
A4 AFTER =BLANK *B3
NO
YES
A5 WRITE RZERO RECORD
YES
NO
B5 WRITE EOF RECORD
NO
YES
JJISSEDF
C5 SET RECORD LENGTH REGISTER CONTENTS = ZERO
D4 READ MACRO
YES
NO
D5 SET BYTE COUNT IN BASIC RDD CCW EQUAL TO BLKSIZE
NN F2
IJISRON
IJIWRT
E4 INITIALIZE BASIC RDD CCW BYTE COUNT WITH SEG LENGTH (RR) FROM I/O AREA
F4 IS SEG TO BE WRITTEN A NULL SEG
YES
NO
IJISGDL
G4 IS RECORD BLOCK SIZE> BLKSIZE
YES
NO
G5 SET RECORD BLOCKSIZE (LL) EQUAL TO BLKSIZE
H5 SET WLR INDICATOR
J4 INITIALIZE BASIC RCDCCW BYTE COUNT WITH RECORD BLOCKSIZE (LL)
K4 ZERO BYTES 2 AND 3 OF BLOCK AND SEG DESCRIPTOR WORCS
NN A1

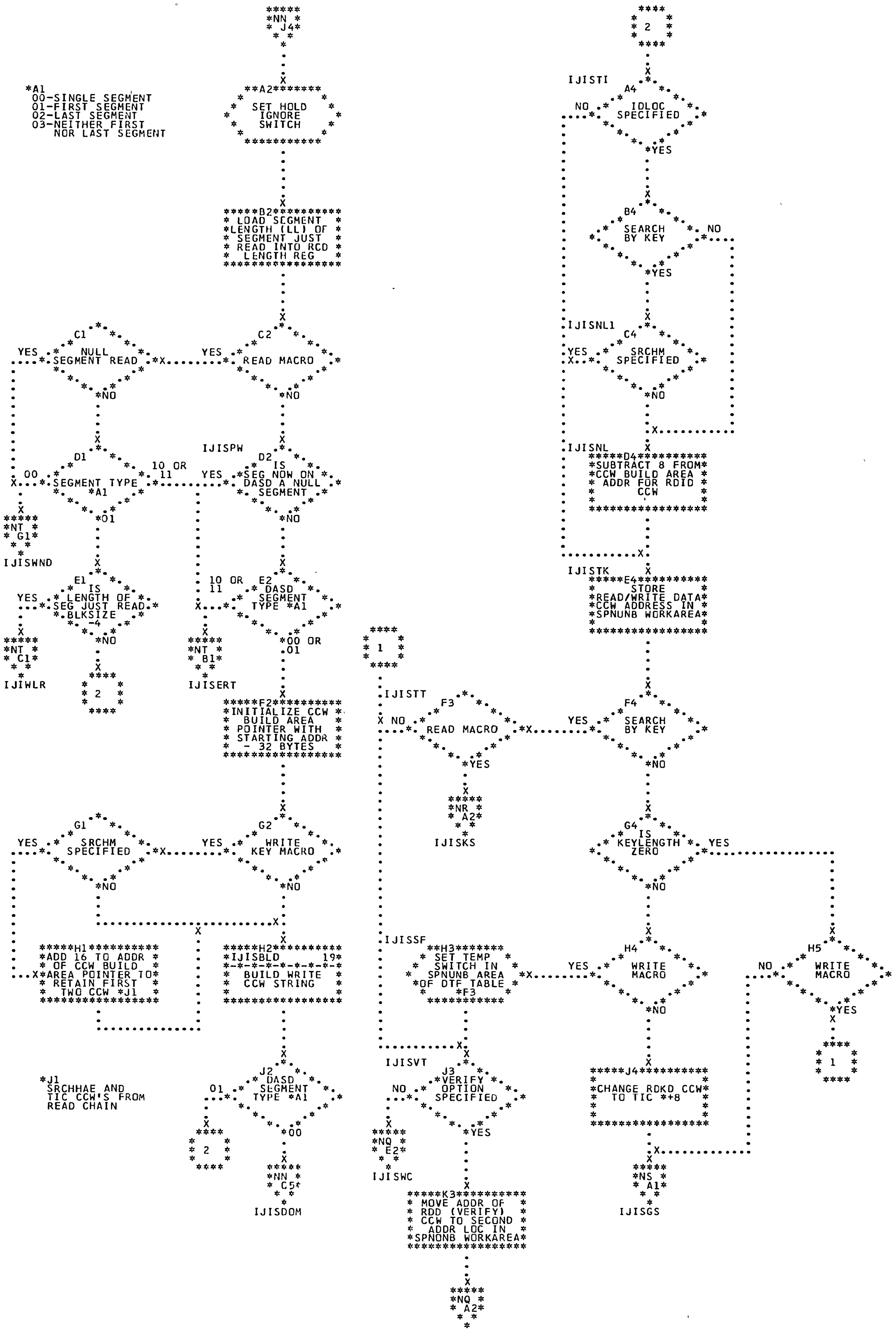
*B3
DAMODV MACRO PARAMETER
OPTION - DECISION
DOES NOT APPLAR IN
AN ASSEMBLY LISTING

*J1
THE
RESULTING
MACRO SWITCH
FOR EACH MACRO--
WRITE AFTER = X'05'
WRITE RZERO = X'04'
WRITE KEY = X'03'
WRITE ID = X'02'
READ KEY = X'01'
READ ID = X'00'

Chart NN. DAMODV: Input/Output Macros (Section 2 of 11)


NM K4
A1 RECFORM = SPNUNB *A2
NO
YES
*A2 DAMODV MACRO PARAMETER OPTION- DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
NM H3
A3 SET BLOCK LENGTH (LL) TO 8 BYTES FOR CONTROL WORDS
1
A4 RECFORM =SPNUNB *A2
NO
YES
NP-J2 NU-H1 NU-K4,K5
B1 VARUNB RECORDS
NO
YES
IJISWSB
B2 WRITE AFTER
NO
YES
B3 SET BASIC RDD CCW BYTE COUNT TO 8 BYTES FOR CONTROL WORDS
B4 VARIABLE RECORDS
YES
NO
C1 INSERT RECORD BLOCK SIZE (LL) IN BLOCK DESCRIPTOR WORD
C4 SVC 0 EXECUTE CHANNEL PROGRAM (READ)
IJISDOM
C5 SVC 0 EXECUTE CHANNEL PROGRAM
D1 WLR INDICATOR SET
NO
YES
D3 SVC 7 WAIT
D4 I/O COMPLETE
NO
YES
D5 HOLD = BLANK
YES
NO
E1 SUBTRACT 4 FROM RECORD BLOCKSIZE AND STORE IN SEG DESCRIPTOR
E4 NO RECORD FOUND CONDITION
YES
NO
E5 TRACK HOLD INDICATOR SET
NO
YES
IJISRON
F2 USING MACRO SWITCH VALUE GET ADDRESS OF CCW STRING
NM-A5,C5,D5
F4 IS SEARCH BY KEY
NO
YES
F5 WRITE MACRO
NO
YES
G2 IJISOVP PD CONTROL SEEK
G4 SRCHM
NO
YES
G5 SVC3L FREE TRACK
NT G1
IJISWND
H2 IJISBLD NL BUILD CCW CHAIN
H4 END OF CYLINDER
YES
NO
NT F3
IJINRF
IJISRFD
J4 SET BASIC RDD CCW BYTE COUNT = LENGTH OF SEG (LL) + 4
NP A2
J2 AFTER = BLANK *A2
YES
NO
K2 WRITE AFTER OR WRITE RZERO
NO
YES
1
IJISAFT
NA A2

## Chart NP. DAMODV: Input/Output Macros (Section 3 of 11)



NN J4
A2 SET HOLD IGNORE SWITCH
*A1
00-SINGLE SEGMENT
01-FIRST SEGMENT
02-LAST SEGMENT
03-NEITHER FIRST NOR LAST SEGMENT
B2 LOAD SEGMENT LENGTH (LL) OF SEGMENT JUST READ INTO RCD LENGTH REG
C1 NULL SEGMENT READ
YES
NO
C2 READ MACRO
YES
NO
D1 SEGMENT TYPE *A1
00
10 OR 11
01
NT G1
IJISWND
IJISPW
D2 IS SEG NOW ON DASD A NULL SEGMENT
YES
NO
E1 IS LENGTH OF SEG JUST READ BLKSIZE -4
YES
NO
NT C1
IJIWLR
2
E2 DASD SEGMENT TYPE *A1
10 OR 11
00 OR 01
NT B1
IJISERT
F2 INITIALIZE CCW BUILD AREA POINTER WITH STARTING ADDR - 32 BYTES
G1 SRCHM SPECIFIED
YES
NO
G2 WRITE KEY MACRO
YES
NO
H1 ADD 16 TO ADDR OF CCW BUILD AREA POINTER TO RETAIN FIRST TWO CCW *J1
H2 IJISBLD 19 BUILD WRITE CCW STRING
*J1
SRCHHAE AND TIC CCW'S FROM READ CHAIN
J2 DASD SEGMENT TYPE *A1
01
00
2
NN C5
IJISDOM
1
IJISTT
F3 READ MACRO
NO
YES
NR A2
IJISKS
IJISSF
H3 SET TEMP SWITCH IN SPNUNB AREA OF DTF TABLE *F3
IJISVT
J3 VERIFY OPTION SPECIFIED
NO
YES
NQ E2
IJISWC
K3 MOVE ADDR OF RDD (VERIFY) CCW TO SECOND ADDR LOC IN SPNUNB WORKAREA
NQ A2
2
IJISTI
A4 IDLOC SPECIFIED
NO
YES
B4 SEARCH BY KEY
NO
YES
IJISNL1
C4 SRCHM SPECIFIED
YES
NO
IJISNL
D4 SUBTRACT 8 FROM CCW BUILD AREA ADDR FOR RDID CCW
IJISTK
E4 STORE READ/WRITE DATA CCW ADDRESS IN SPNUNB WORKAREA
F4 SEARCH BY KEY
YES
NO
G4 IS KEYLENGTH ZERO
YES
NO
H4 WRITE MACRO
YES
NO
H5 WRITE MACRO
NO
YES
1
J4 CHANGE RDKD CCW TO TIC *+8
NS A1
IJISGS

## Chart NQ. DAMODV: Input/Output Macros (Section 4 of 11)

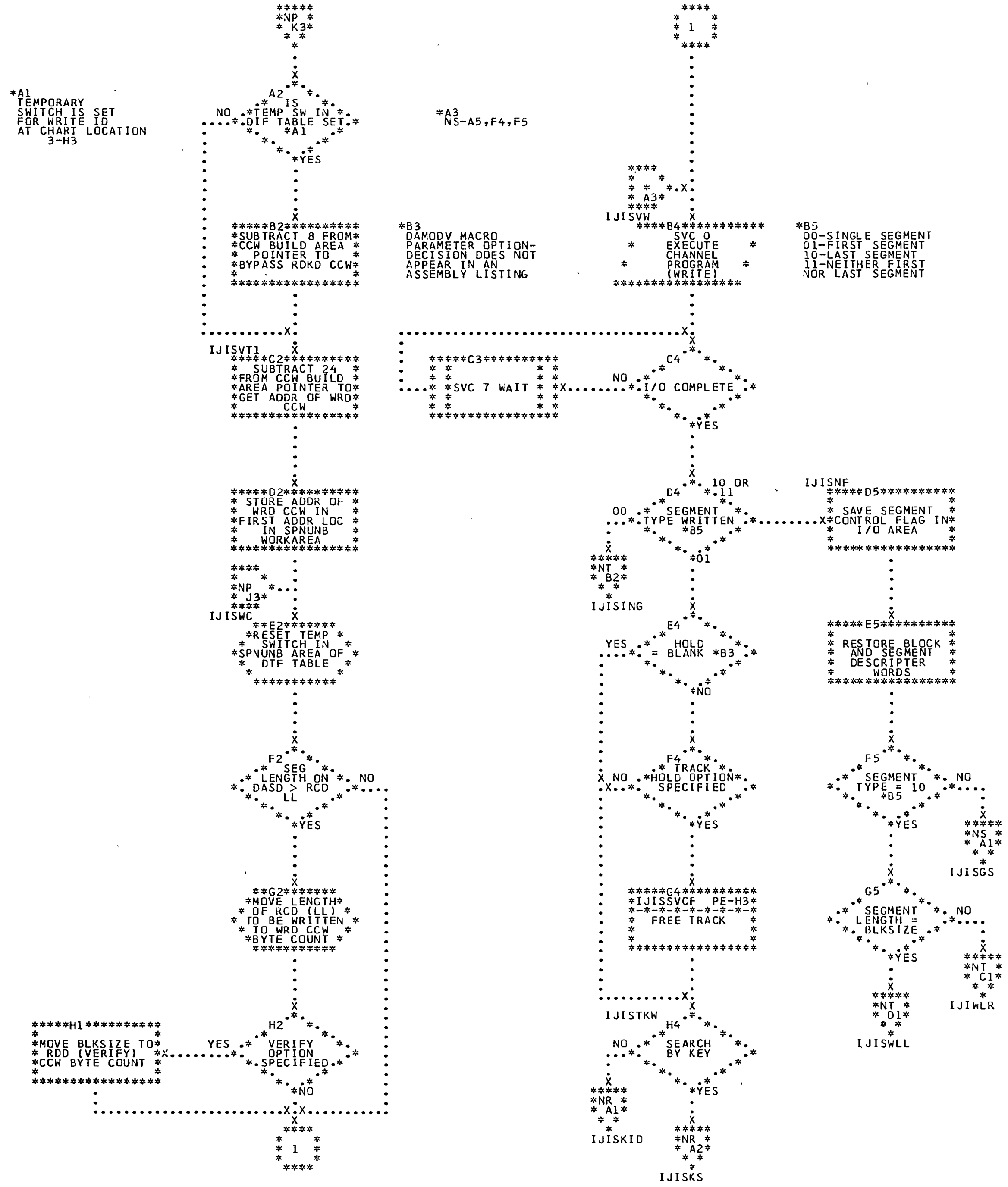
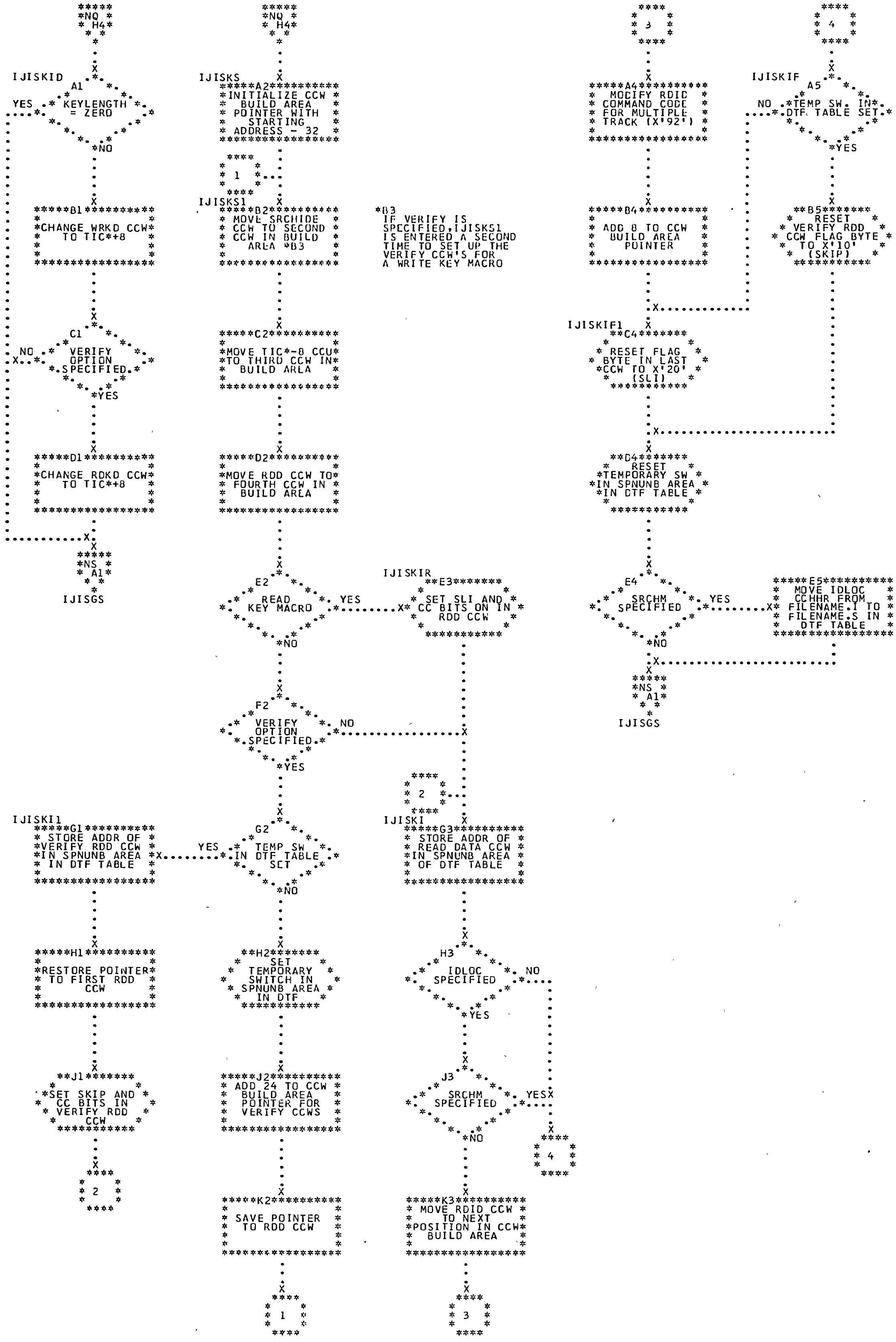
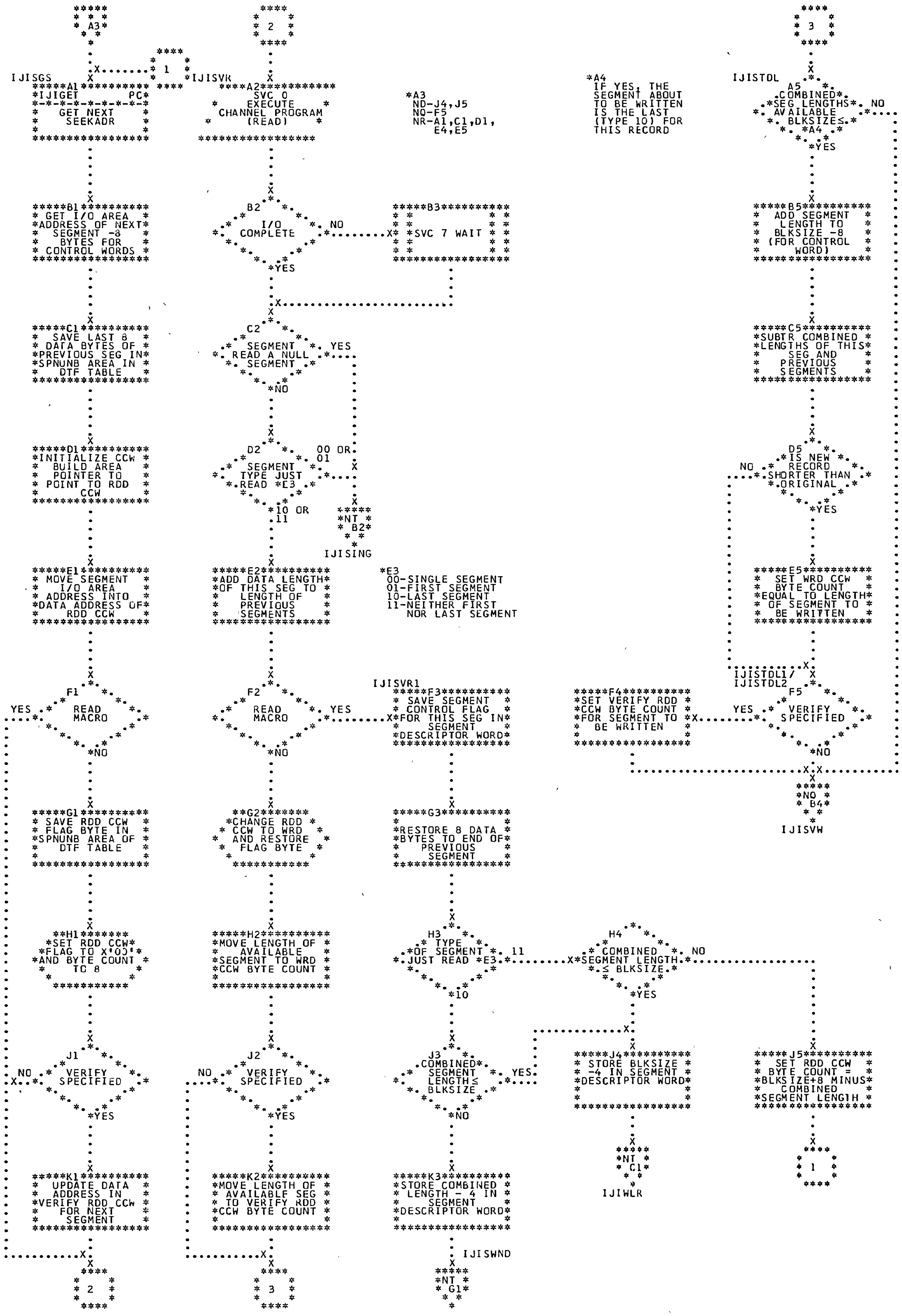

*A1
TEMPORARY SWITCH IS SET FOR WRITE ID AT CHART LOCATION 3-H3
NP K3
A2
IS TEMP SW IN DIF TABLE SET *A1
NO
YES
*A3
NS-A5,F4,F5
1
A3
IJISVW
B2
SUBTRACT 8 FROM CCW BUILD AREA POINTER TO BYPASS RDKD CCW
*B3
DAMODV MACRO PARAMETER OPTION-DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING
B4
SVC 0 EXECUTE CHANNEL PROGRAM (WRITE)
*B5
00-SINGLE SEGMENT
01-FIRST SEGMENT
10-LAST SEGMENT
11-NEITHER FIRST NOR LAST SEGMENT
IJISVT1
C2
SUBTRACT 24 FROM CCW BUILD AREA POINTER TO GET ADDR OF WRD CCW
C3
SVC 7 WAIT
C4
I/O COMPLETE
NO
YES
D2
STORE ADDR OF WRD CCW IN FIRST ADDR LOC IN SPNUNB WORKAREA
D4
SEGMENT TYPE WRITTEN *B5
00
10 OR 11
01
NT B2
IJISING
IJISNF
D5
SAVE SEGMENT CONTROL FLAG IN I/O AREA
NP J3
IJISWC
E2
RESET TEMP SWITCH IN SPNUNB AREA OF DTF TABLE
E4
HOLD = BLANK *B3
YES
NO
E5
RESTORE BLOCK AND SEGMENT DESCRIPTER WORDS
F2
SEG LENGTH ON DASD > RCD LL
NO
YES
F4
TRACK HOLD OPTION SPECIFIED
NO
YES
F5
SEGMENT TYPE = 10 *B5
NO
YES
NS A1
IJISGS
G2
MOVE LENGTH OF RCD (LL) TO BE WRITTEN TO WRD CCW BYTE COUNT
G4
IJISSVCF PE-H3
FREE TRACK
G5
SEGMENT LENGTH = BLKSIZE
NO
YES
NT C1
IJIWLR
NT D1
IJISWLL
H1
MOVE BLKSIZE TO RDD (VERIFY) CCW BYTE COUNT
H2
VERIFY OPTION SPECIFIED
YES
NO
IJISTKW
H4
SEARCH BY KEY
NO
YES
NR A1
IJISKID
NR A2
IJISKS
1

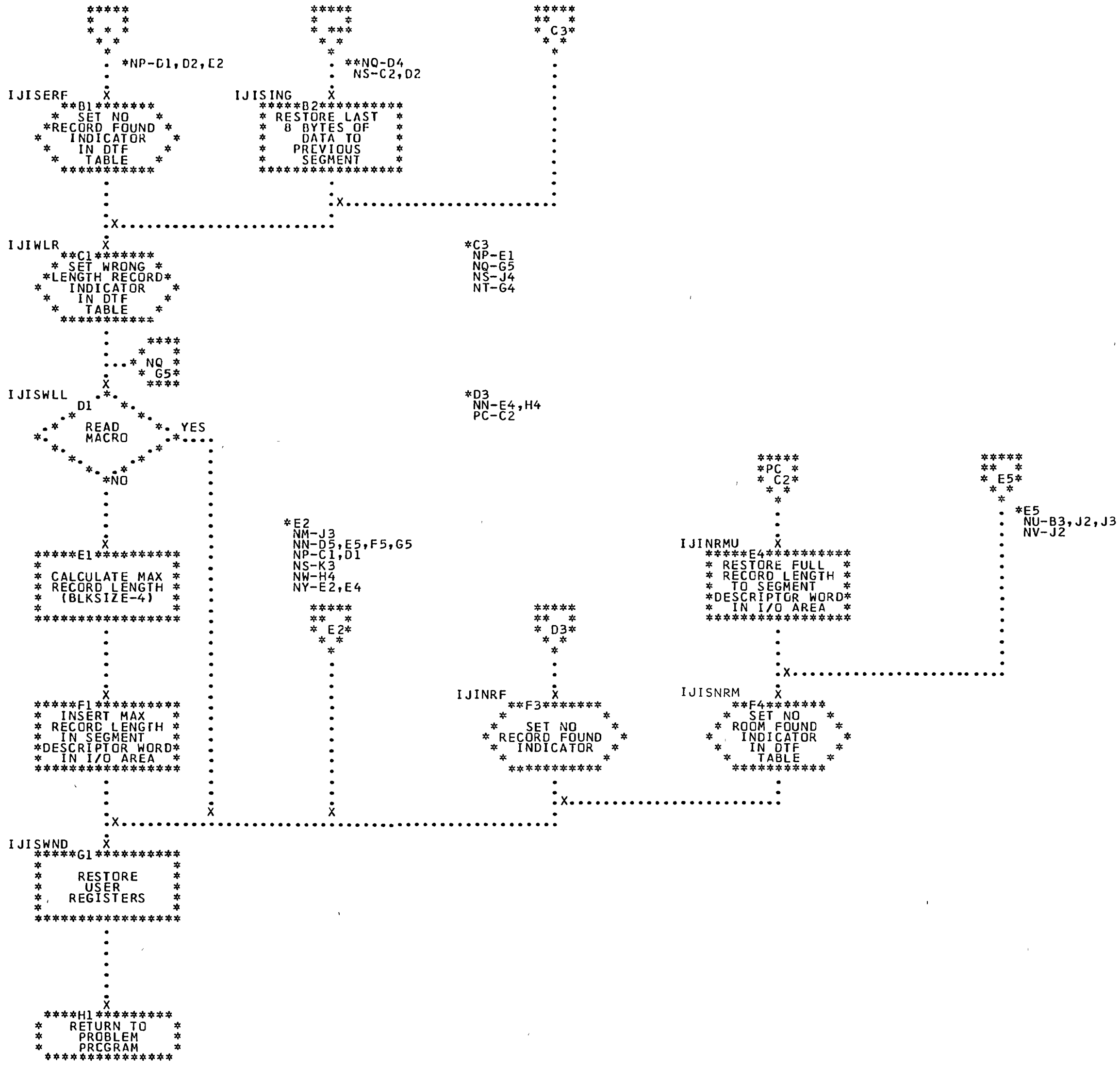
Chart NR. DAMODV: Input/Output Macros (Section 5 of 11)


NQ H4
IJISKID A1
KEYLENGTH = ZERO
YES
NO
B1 CHANGE WRKD CCW TO TIC*+8
C1 VERIFY OPTION SPECIFIED
NO
YES
D1 CHANGE RDKD CCW TO TIC*+8
NS A1
IJISGS
NQ H4
IJISKS A2 INITIALIZE CCW BUILD AREA POINTER WITH STARTING ADDRESS - 32
1
IJISKS1 B2 MOVE SRCHIDE CCW TO SECOND CCW IN BUILD AREA *B3
*B3 IF VERIFY IS SPECIFIED,IJISKS1 IS ENTERED A SECOND TIME TO SET UP THE VERIFY CCW'S FOR A WRITE KEY MACRO
C2 MOVE TIC*-8 CCW TO THIRD CCW IN BUILD AREA
D2 MOVE RDD CCW TO FOURTH CCW IN BUILD AREA
E2 READ KEY MACRO
YES
NO
IJISKIR E3 SET SLI AND CC BITS ON IN RDD CCW
F2 VERIFY OPTION SPECIFIED
NO
YES
2
IJISKI1 G1 STORE ADDR OF VERIFY RDD CCW IN SPNUNB AREA IN DTF TABLE
G2 TEMP SW IN DTF TABLE SET
YES
NO
IJISKI G3 STORE ADDR OF READ DATA CCW IN SPNUNB AREA OF DTF TABLE
H1 RESTORE POINTER TO FIRST RDD CCW
H2 SET TEMPORARY SWITCH IN SPNUNB AREA IN DTF
H3 IDLOC SPECIFIED
NO
YES
J1 SET SKIP AND CC BITS IN VERIFY RDD CCW
2
J2 ADD 24 TO CCW BUILD AREA POINTER FOR VERIFY CCWS
J3 SRCHM SPECIFIED
YES
NO
4
K2 SAVE POINTER TO RDD CCW
1
K3 MOVE RDID CCW TO NEXT POSITION IN CCW BUILD AREA
3
3
A4 MODIFY RDIC COMMAND CODE FOR MULTIPLE TRACK (X'92')
B4 ADD 8 TO CCW BUILD AREA POINTER
IJISKIF1 C4 RESET FLAG BYTE IN LAST CCW TO X'20' (SLI)
D4 RESET TEMPORARY SW IN SPNUNB AREA IN DTF TABLE
E4 SRCHM SPECIFIED
YES
NO
E5 MOVE IDLOC CCHHR FROM FILENAME.I TO FILENAME.S IN DTF TABLE
NS A1
IJISGS
4
IJISKIF A5 TEMP SW. IN DTF TABLE SET
NO
YES
B5 RESET VERIFY RDD CCW FLAG BYTE TO X'10' (SKIP)

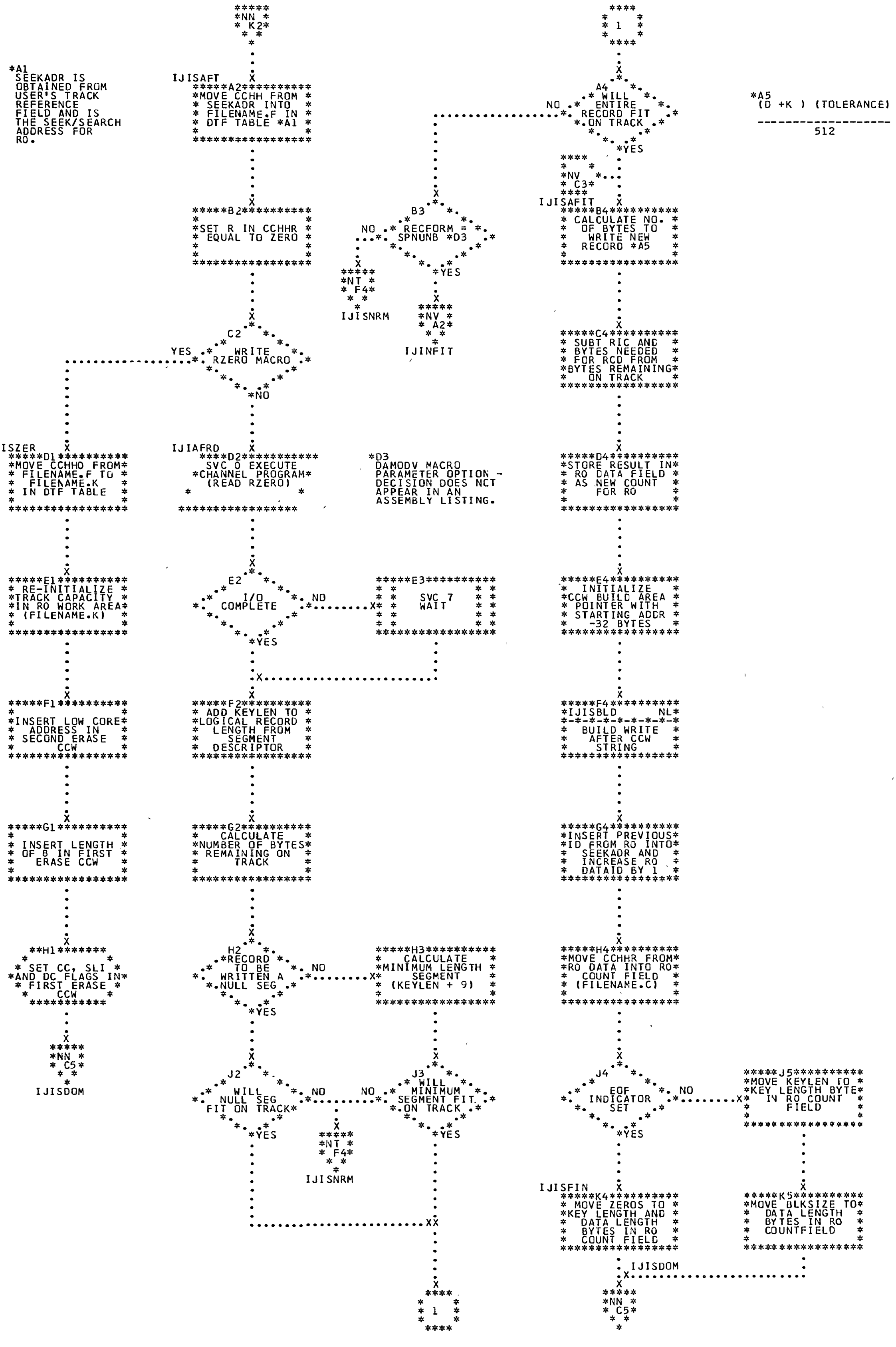
## Chart NS. DAMODV: Input/Output Macros (Section 6 of 11)


A3
1
IJISGS
A1
IJIGET PC
GET NEXT SEEKADR
B1
GET I/O AREA ADDRESS OF NEXT SEGMENT -8 BYTES FOR CONTROL WORDS
C1
SAVE LAST 8 DATA BYTES OF PREVIOUS SEG IN SPNUNB AREA IN DTF TABLE
D1
INITIALIZE CCW BUILD AREA POINTER TO POINT TO RDD CCW
E1
MOVE SEGMENT I/O AREA ADDRESS INTO DATA ADDRESS OF RDD CCW
F1
READ MACRO
YES
NO
G1
SAVE RDD CCW FLAG BYTE IN SPNUNB AREA OF DTF TABLE
H1
SET RDD CCW FLAG TO X'00' AND BYTE COUNT TO 8
J1
VERIFY SPECIFIED
NO
YES
K1
UPDATE DATA ADDRESS IN VERIFY RDD CCW FOR NEXT SEGMENT
2
2
IJISVR
A2
SVC 0 EXECUTE CHANNEL PROGRAM (READ)
B2
I/O COMPLETE
NO
YES
B3
SVC 7 WAIT
C2
SEGMENT READ A NULL SEGMENT
YES
NO
D2
SEGMENT TYPE JUST READ *E3
00 OR 01
10 OR 11
NT B2
IJISING
E2
ADD DATA LENGTH OF THIS SEG TO LENGTH OF PREVIOUS SEGMENTS
F2
READ MACRO
YES
NO
G2
CHANGE RDD CCW TO WRD AND RESTORE FLAG BYTE
H2
MOVE LENGTH OF AVAILABLE SEGMENT TO WRD CCW BYTE COUNT
J2
VERIFY SPECIFIED
NO
YES
K2
MOVE LENGTH OF AVAILABLE SEG TO VERIFY RDD CCW BYTE COUNT
3
*A3
ND-J4,J5
NO-F5
NR-A1,C1,D1, E4,E5
*E3
00-SINGLE SEGMENT
01-FIRST SEGMENT
10-LAST SEGMENT
11-NEITHER FIRST NOR LAST SEGMENT
IJISVR1
F3
SAVE SEGMENT CONTROL FLAG FOR THIS SEG IN SEGMENT DESCRIPTOR WORD
G3
RESTORE 8 DATA BYTES TO END OF PREVIOUS SEGMENT
H3
TYPE OF SEGMENT JUST READ *E3
11
10
J3
COMBINED SEGMENT LENGTH≤ BLKSIZE
YES
NO
K3
STORE COMBINED LENGTH - 4 IN SEGMENT DESCRIPTOR WORD
IJISWND
NT G1
*A4
IF YES, THE SEGMENT ABOUT TO BE WRITTEN IS THE LAST (TYPE 10) FOR THIS RECORD
F4
SET VERIFY RDD CCW BYTE COUNT FOR SEGMENT TO BE WRITTEN
H4
COMBINED SEGMENT LENGTH ≤ BLKSIZE
NO
YES
J4
STORE BLKSIZE -4 IN SEGMENT DESCRIPTOR WORD
NT C1
IJIWLR
3
IJISTDL
A5
COMBINED SEG LENGTHS AVAILABLE BLKSIZE≤ *A4
NO
YES
B5
ADD SEGMENT LENGTH TO BLKSIZE -8 (FOR CONTROL WORD)
C5
SUBTR COMBINED LENGTHS OF THIS SEG AND PREVIOUS SEGMENTS
D5
IS NEW RECORD SHORTER THAN ORIGINAL
NO
YES
E5
SET WRD CCW BYTE COUNT EQUAL TO LENGTH OF SEGMENT TO BE WRITTEN
IJISTDL1/ IJISTDL2
F5
VERIFY SPECIFIED
YES
NO
NO B4
IJISVW
J5
SET RDD CCW BYTE COUNT = BLKSIZE+8 MINUS COMBINED SEGMENT LENGTH
1

# Chart NT. DAMODV: Input/Output Macros (Section 7 of 11)


*NP-D1,D2,E2
**NQ-D4
NS-C2,D2
C3
IJISERF
B1
SET NO RECORD FOUND INDICATOR IN DTF TABLE
IJISING
B2
RESTORE LAST 8 BYTES OF DATA TO PREVIOUS SEGMENT
*C3
NP-E1
NQ-G5
NS-J4
NT-G4
IJIWLR
C1
SET WRONG LENGTH RECORD INDICATOR IN DTF TABLE
NQ G5
IJISWLL
D1
READ MACRO
YES
NO
*D3
NN-E4,H4
PC-C2
*E2
NM-J3
NN-D5,E5,F5,G5
NP-C1,D1
NS-K3
NW-H4
NY-E2,E4
PC C2
E5
*E5
NU-B3,J2,J3
NV-J2
E1
CALCULATE MAX RECORD LENGTH (BLKSIZE-4)
E2
D3
IJINRMU
E4
RESTORE FULL RECORD LENGTH TO SEGMENT DESCRIPTOR WORD IN I/O AREA
F1
INSERT MAX RECORD LENGTH IN SEGMENT DESCRIPTOR WORD IN I/O AREA
IJINRF
F3
SET NO RECORD FOUND INDICATOR
IJISNRM
F4
SET NO ROOM FOUND INDICATOR IN DTF TABLE
IJISWND
G1
RESTORE USER REGISTERS
H1
RETURN TO PROBLEM PROGRAM

## Chart NU. DAMODV: Input/Output Macros (Section 8 of 11)


NN K2
*A1 SEEKADR IS OBTAINED FROM USER'S TRACK REFERENCE FIELD AND IS THE SEEK/SEARCH ADDRESS FOR RO.
IJISAFT
A2 MOVE CCHH FROM SEEKADR INTO FILENAME.F IN DTF TABLE *A1
B2 SET R IN CCHHR EQUAL TO ZERO
C2 WRITE RZERO MACRO
YES
NO
IJISZER
D1 MOVE CCHHO FROM FILENAME.F TO FILENAME.K IN DTF TABLE
E1 RE-INITIALIZE TRACK CAPACITY IN RO WORK AREA (FILENAME.K)
F1 INSERT LOW CORE ADDRESS IN SECOND ERASE CCW
G1 INSERT LENGTH OF 8 IN FIRST ERASE CCW
H1 SET CC, SLI AND DC FLAGS IN FIRST ERASE CCW
NN C5
IJISDOM
IJIAFRD
D2 SVC 0 EXECUTE CHANNEL PROGRAM (READ RZERO)
E2 I/O COMPLETE
NO
YES
E3 SVC 7 WAIT
F2 ADD KEYLEN TO LOGICAL RECORD LENGTH FROM SEGMENT DESCRIPTOR
G2 CALCULATE NUMBER OF BYTES REMAINING ON TRACK
H2 RECORD TO BE WRITTEN A NULL SEG
NO
YES
H3 CALCULATE MINIMUM LENGTH SEGMENT (KEYLEN + 9)
J2 WILL NULL SEG FIT ON TRACK
NO
YES
NT F4
IJISNRM
J3 WILL MINIMUM SEGMENT FIT ON TRACK
NO
YES
1
B3 RECFORM = SPNUNB *D3
NO
YES
NT F4
IJISNRM
NV A2
IJINFIT
*D3 DAMODV MACRO PARAMETER OPTION - DECISION DOES NCT APPEAR IN AN ASSEMBLY LISTING.
1
A4 WILL ENTIRE RECORD FIT ON TRACK
NO
YES
NV C3
IJISAFIT
B4 CALCULATE NO. OF BYTES TO WRITE NEW RECORD *A5
*A5 (D +K ) (TOLERANCE)
512
C4 SUBT RIC AND BYTES NEEDED FOR RCD FROM BYTES REMAINING ON TRACK
D4 STORE RESULT IN RO DATA FIELD AS NEW COUNT FOR RO
E4 INITIALIZE CCW BUILD AREA POINTER WITH STARTING ADCR -32 BYTES
F4 IJISBLD NL
BUILD WRITE AFTER CCW STRING
G4 INSERT PREVIOUS ID FROM RO INTO SEEKADR AND INCREASE RO DATAID BY 1
H4 MOVE CCHHR FROM RO DATA INTO RO COUNT FIELD (FILENAME.C)
J4 EOF INDICATOR SET
NO
YES
J5 MOVE KEYLEN TO KEY LENGTH BYTE IN RO COUNT FIELD
IJISFIN
K4 MOVE ZEROS TO KEY LENGTH AND DATA LENGTH BYTES IN RO COUNT FIELD
K5 MOVE BLKSIZE TO DATA LENGTH BYTES IN RO COUNTFIELD
IJISDOM
NN C5

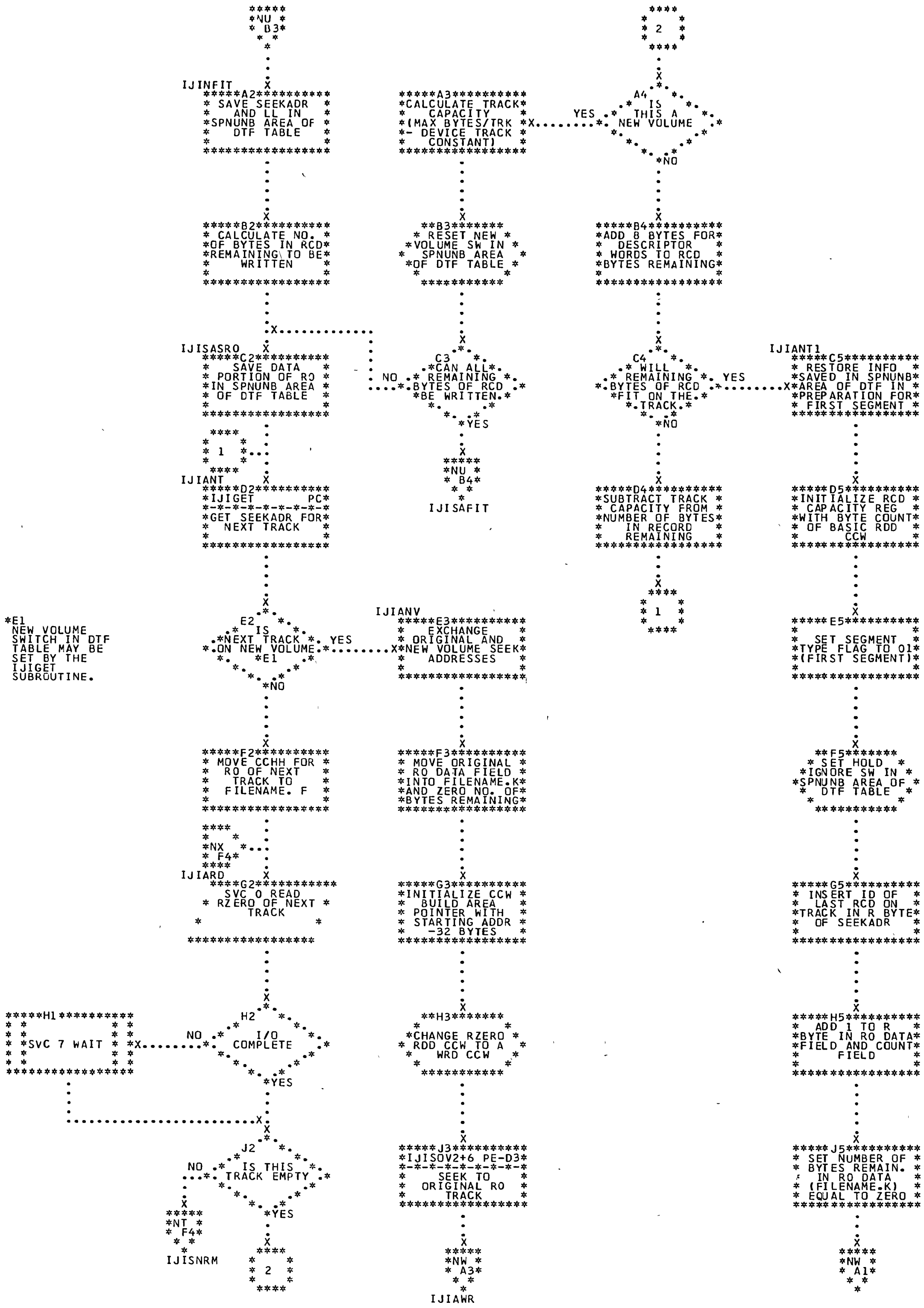

NU B3
2
IJINFIT
A2 SAVE SEEKADR AND LL IN SPNUNB AREA OF DTF TABLE
A3 CALCULATE TRACK CAPACITY (MAX BYTES/TRK - DEVICE TRACK CONSTANT)
A4 IS THIS A NEW VOLUME
YES
NO
B2 CALCULATE NO. OF BYTES IN RCD REMAINING TO BE WRITTEN
B3 RESET NEW VOLUME SW IN SPNUNB AREA OF DTF TABLE
B4 ADD 8 BYTES FOR DESCRIPTOR WORDS TO RCD BYTES REMAINING
IJISASRO
C2 SAVE DATA PORTION OF RO IN SPNUNB AREA OF DTF TABLE
C3 CAN ALL REMAINING BYTES OF RCD BE WRITTEN
NO
YES
C4 WILL REMAINING BYTES OF RCD FIT ON THE TRACK
YES
NO
IJIANT1
C5 RESTORE INFO SAVED IN SPNUNB AREA OF DTF IN PREPARATION FOR FIRST SEGMENT
1
IJIANT
D2 IJIGET PC GET SEEKADR FOR NEXT TRACK
NU B4
IJISAFIT
D4 SUBTRACT TRACK CAPACITY FROM NUMBER OF BYTES IN RECORD REMAINING
D5 INITIALIZE RCD CAPACITY REG WITH BYTE COUNT OF BASIC RDD CCW
1
E1 NEW VOLUME SWITCH IN DTF TABLE MAY BE SET BY THE IJIGET SUBROUTINE.
E2 IS NEXT TRACK ON NEW VOLUME E1
YES
NO
IJIANV
E3 EXCHANGE ORIGINAL AND NEW VOLUME SEEK ADDRESSES
E5 SET SEGMENT TYPE FLAG TO 01 (FIRST SEGMENT)
F2 MOVE CCHH FOR RO OF NEXT TRACK TO FILENAME. F
F3 MOVE ORIGINAL RO DATA FIELD INTO FILENAME.K AND ZERO NO. OF BYTES REMAINING
F5 SET HOLD IGNORE SW IN SPNUNB AREA OF DTF TABLE
NX F4
IJIARD
G2 SVC 0 READ RZERO OF NEXT TRACK
G3 INITIALIZE CCW BUILD AREA POINTER WITH STARTING ADDR -32 BYTES
G5 INSERT ID OF LAST RCD ON TRACK IN R BYTE OF SEEKADR
H1 SVC 7 WAIT
H2 I/O COMPLETE
NO
YES
H3 CHANGE RZERO RDD CCW TO A WRD CCW
H5 ADD 1 TO R BYTE IN RO DATA FIELD AND COUNT FIELD
J2 IS THIS TRACK EMPTY
NO
YES
J3 IJISOV2+6 PE-D3 SEEK TO ORIGINAL RO TRACK
J5 SET NUMBER OF BYTES REMAIN. IN RO DATA (FILENAME.K) EQUAL TO ZERO
NT F4
IJISNRM
2
NW A3
IJIAWR
NW A1

Chart NW. DAMODV: Input/Output Macros (Section 10 of 11)

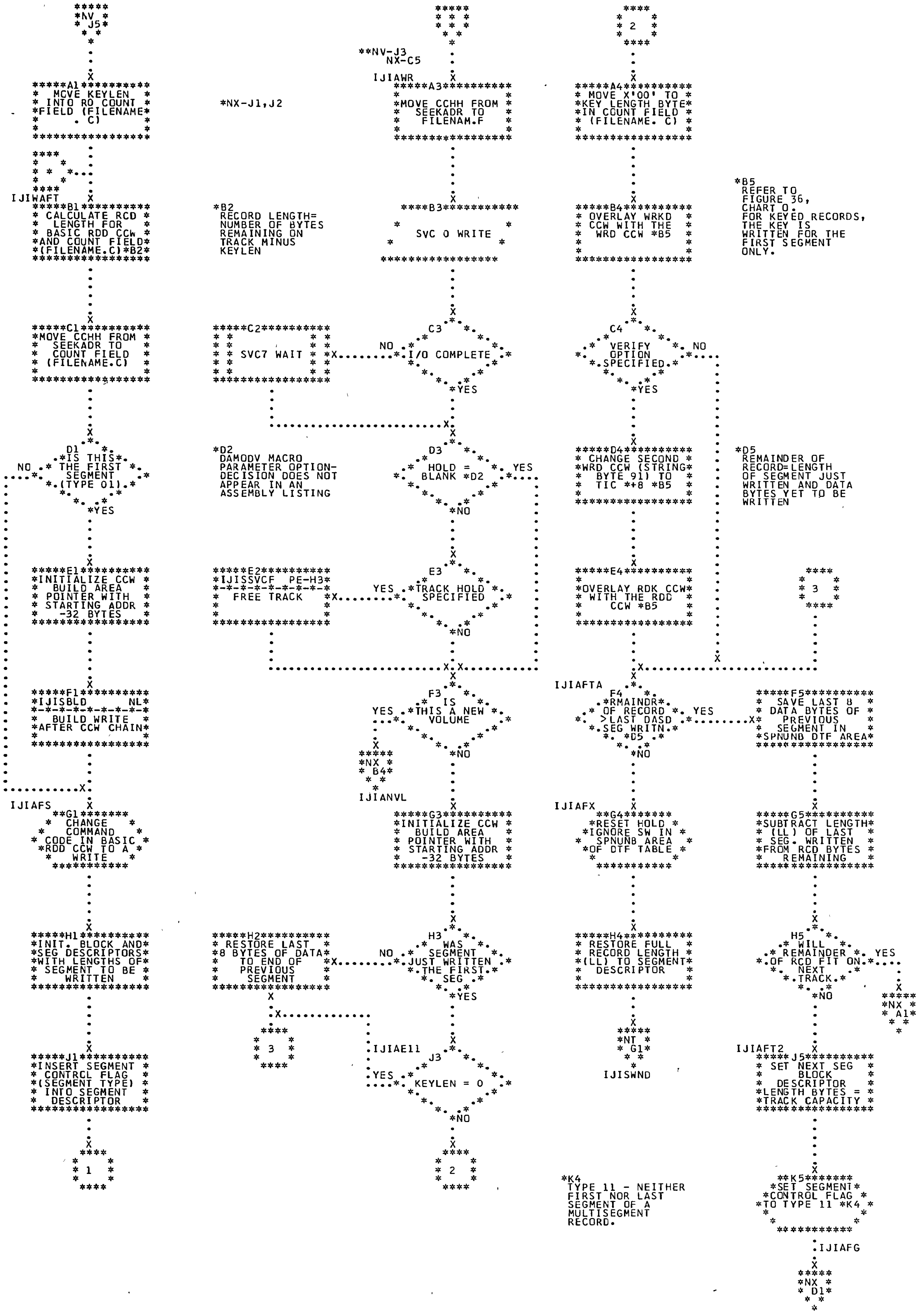
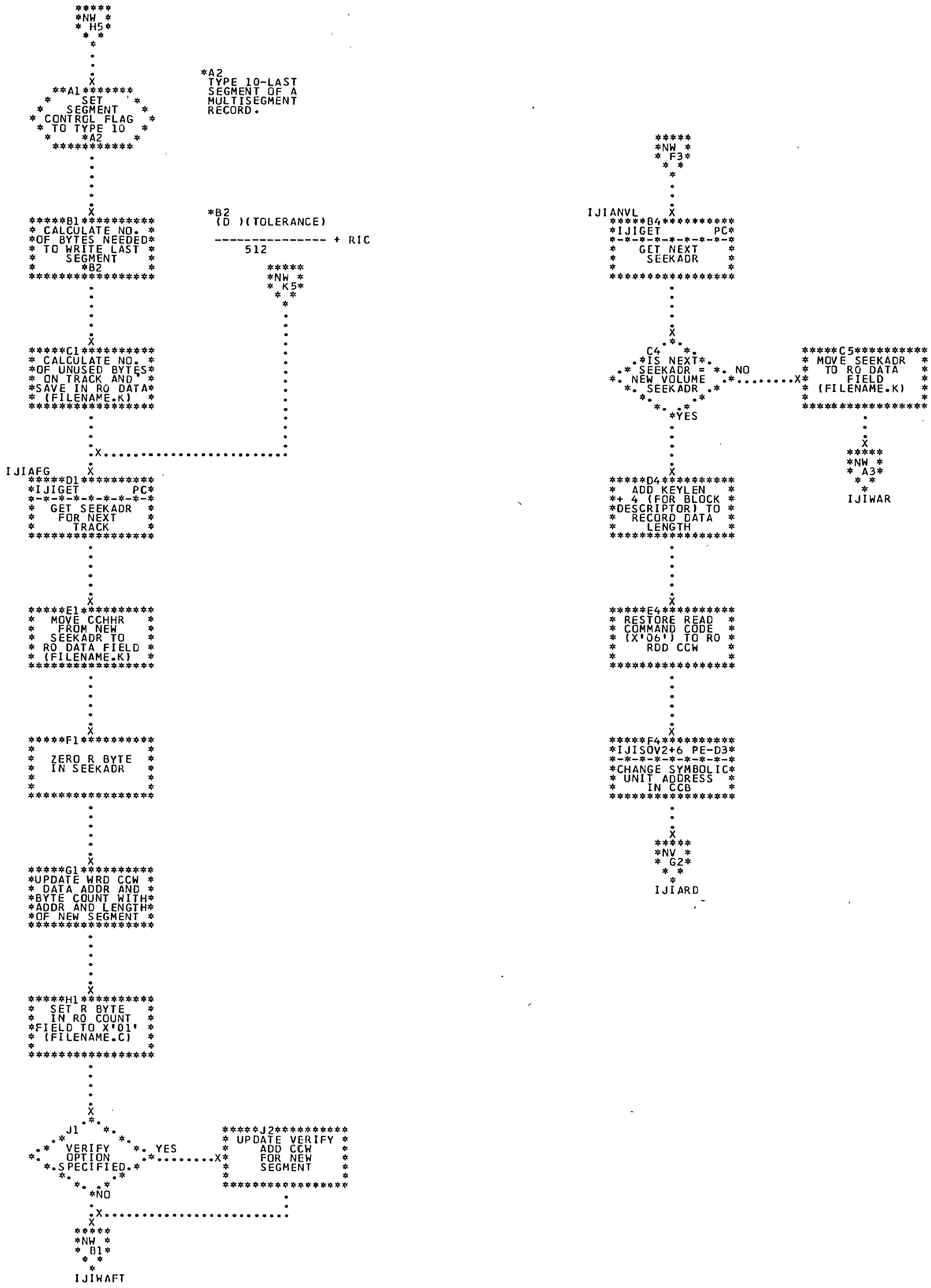

NV J5
A1 MOVE KEYLEN INTO RO COUNT FIELD (FILENAME. C)
IJIWAFT
B1 CALCULATE RCD LENGTH FOR BASIC RDD CCW AND COUNT FIELD (FILENAME.C) *B2
C1 MOVE CCHH FROM SEEKADR TO COUNT FIELD (FILENAME.C)
D1 IS THIS THE FIRST SEGMENT (TYPE 01)
NO
YES
E1 INITIALIZE CCW BUILD AREA POINTER WITH STARTING ADDR -32 BYTES
F1 IJISBLD NL BUILD WRITE AFTER CCW CHAIN
IJIAFS
G1 CHANGE COMMAND CODE IN BASIC RDD CCW TO A WRITE
H1 INIT. BLOCK AND SEG DESCRIPTORS WITH LENGTHS OF SEGMENT TO BE WRITTEN
J1 INSERT SEGMENT CONTROL FLAG (SEGMENT TYPE) INTO SEGMENT DESCRIPTOR
1
*NX-J1,J2
*B2 RECORD LENGTH= NUMBER OF BYTES REMAINING ON TRACK MINUS KEYLEN
C2 SVC7 WAIT
*D2 DAMODV MACRO PARAMETER OPTION-DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING
E2 IJISSVCF PE-H3 FREE TRACK
H2 RESTORE LAST 8 BYTES OF DATA TO END OF PREVIOUS SEGMENT
3
**NV-J3 NX-C5
IJIAWR
A3 MOVE CCHH FROM SEEKADR TO FILENAM.F
B3 SVC 0 WRITE
C3 I/O COMPLETE
NO
YES
D3 HOLD = BLANK *D2
YES
NO
E3 TRACK HOLD SPECIFIED
YES
NO
F3 IS THIS A NEW VOLUME
YES
NO
NX B4
IJIANVL
G3 INITIALIZE CCW BUILD AREA POINTER WITH STARTING ADDR -32 BYTES
H3 WAS SEGMENT JUST WRITTEN THE FIRST SEG
NO
YES
IJIAE11
J3 KEYLEN = 0
YES
NO
2
2
A4 MOVE X'00' TO KEY LENGTH BYTE IN COUNT FIELD (FILENAME. C)
B4 OVERLAY WRKD CCW WITH THE WRD CCW *B5
C4 VERIFY OPTION SPECIFIED
NO
YES
D4 CHANGE SECOND WRD CCW (STRING BYTE 91) TO TIC *+8 *B5
E4 OVERLAY RDK CCW WITH THE RDD CCW *B5
IJIAFTA
F4 RMAINDR OF RECORD >LAST DASD SEG WRITN. *D5
YES
NO
IJIAFX
G4 RESET HOLD IGNORE SW IN SPNUNB AREA OF DTF TABLE
H4 RESTORE FULL RECORD LENGTH (LL) TO SEGMENT DESCRIPTOR
NT G1
IJISWND
*K4 TYPE 11 - NEITHER FIRST NOR LAST SEGMENT OF A MULTISEGMENT RECORD.
*B5 REFER TO FIGURE 36, CHART 0. FOR KEYED RECORDS, THE KEY IS WRITTEN FOR THE FIRST SEGMENT ONLY.
*D5 REMAINDER OF RECORD= LENGTH OF SEGMENT JUST WRITTEN AND DATA BYTES YET TO BE WRITTEN
3
F5 SAVE LAST 8 DATA BYTES OF PREVIOUS SEGMENT IN SPNUNB DTF AREA
G5 SUBTRACT LENGTH (LL) OF LAST SEG. WRITTEN FROM RCD BYTES REMAINING
H5 WILL REMAINDER OF RCD FIT ON NEXT TRACK
YES
NO
NX A1
IJIAFT2
J5 SET NEXT SEG BLOCK DESCRIPTOR LENGTH BYTES = TRACK CAPACITY
K5 SET SEGMENT CONTROL FLAG TO TYPE 11 *K4
IJIAFG
NX D1

Chart NX. DAMODV: Input/Output Macros (Section 11 of 11)


NW H5
A1 SET SEGMENT CONTROL FLAG TO TYPE 10 *A2
*A2 TYPE 10-LAST SEGMENT OF A MULTISEGMENT RECORD.
B1 CALCULATE NO. OF BYTES NEEDED TO WRITE LAST SEGMENT *B2
*B2 (D )(TOLERANCE)
--------------- + RIC
512
NW K5
C1 CALCULATE NO. OF UNUSED BYTES ON TRACK AND SAVE IN RO DATA (FILENAME.K)
IJIAFG
D1 IJIGET PC GET SEEKADR FOR NEXT TRACK
E1 MOVE CCHHR FROM NEW SEEKADR TO RO DATA FIELD (FILENAME.K)
F1 ZERO R BYTE IN SEEKADR
G1 UPDATE WRD CCW DATA ADDR AND BYTE COUNT WITH ADDR AND LENGTH OF NEW SEGMENT
H1 SET R BYTE IN RO COUNT FIELD TO X'01' (FILENAME.C)
J1 VERIFY OPTION SPECIFIED
YES
NO
J2 UPDATE VERIFY ADD CCW FOR NEW SEGMENT
NW B1
IJIWAFT
NW F3
IJIANVL
B4 IJIGET PC GET NEXT SEEKADR
C4 IS NEXT SEEKADR = NEW VOLUME SEEKADR
NO
YES
C5 MOVE SEEKADR TO RO DATA FIELD (FILENAME.K)
NW A3
IJIWAR
D4 ADD KEYLEN + 4 (FOR BLOCK DESCRIPTOR) TO RECORD DATA LENGTH
E4 RESTORE READ COMMAND CODE (X'06') TO RO RDD CCW
F4 IJISOV2+6 PE-D3 CHANGE SYMBOLIC UNIT ADDRESS IN CCB
NV G2
IJIARD

## Chart NY. DAMODV: CNTRL and FREE Macros

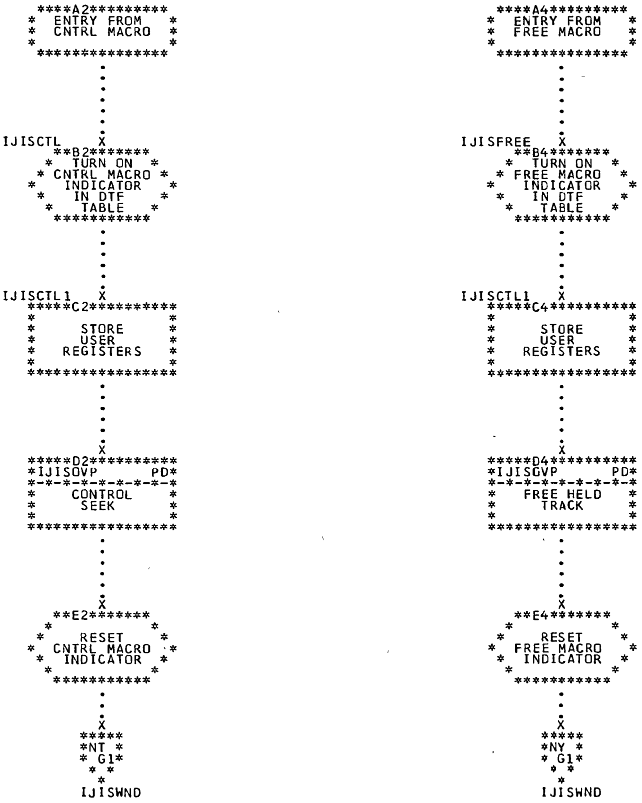
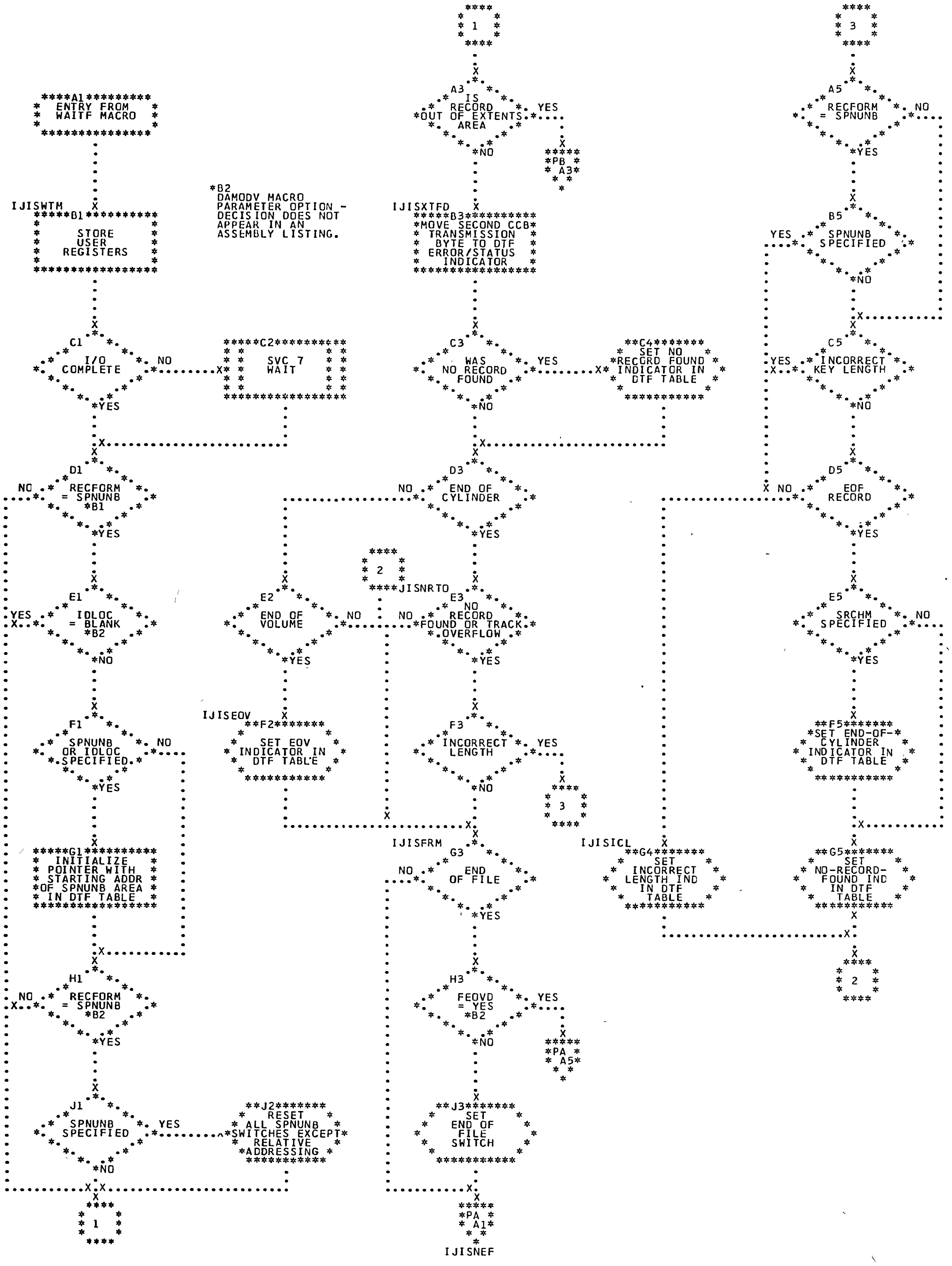
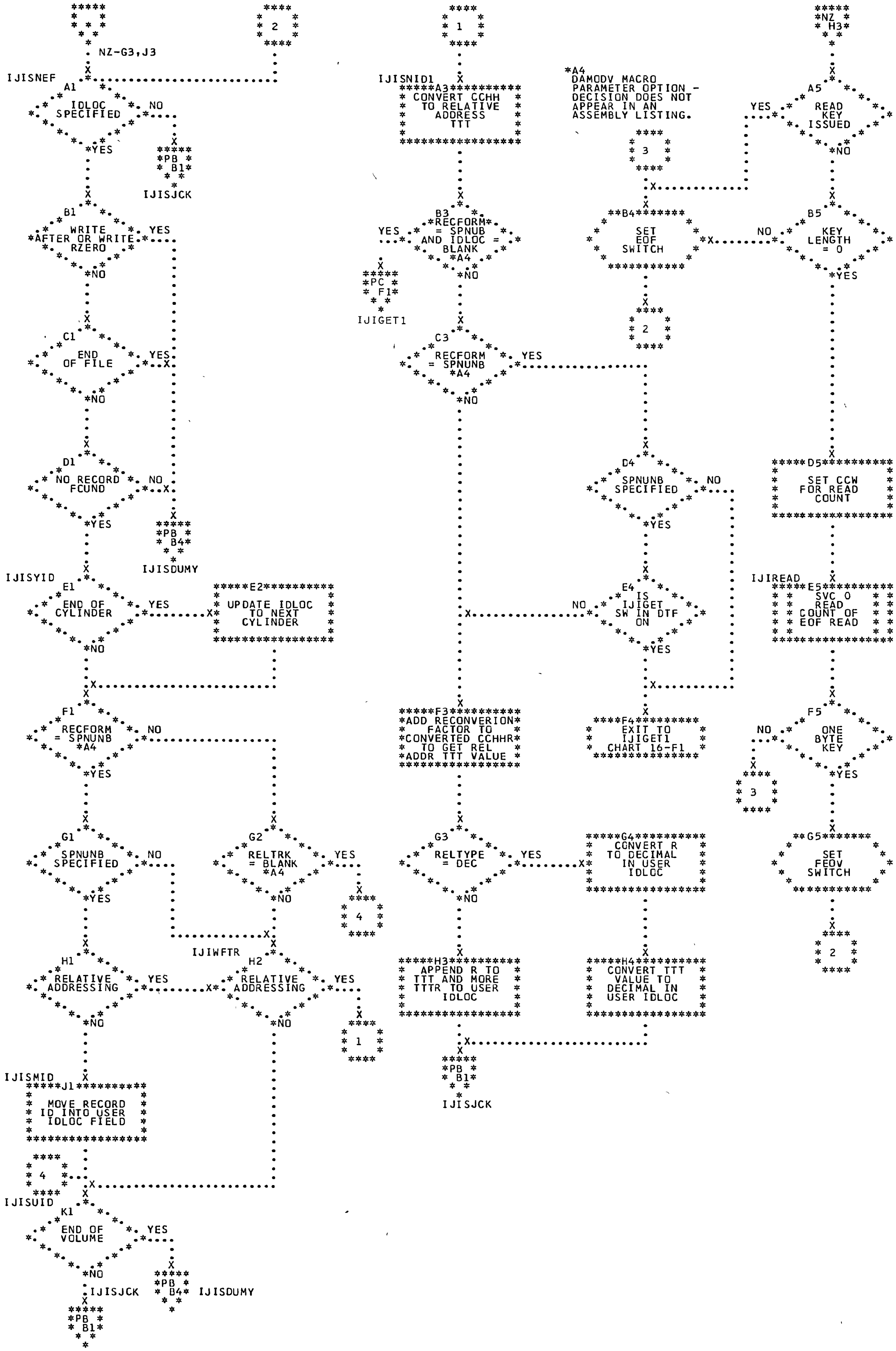


Chart NZ. DAMODV: WAITF Macro (Section 1 of 3)

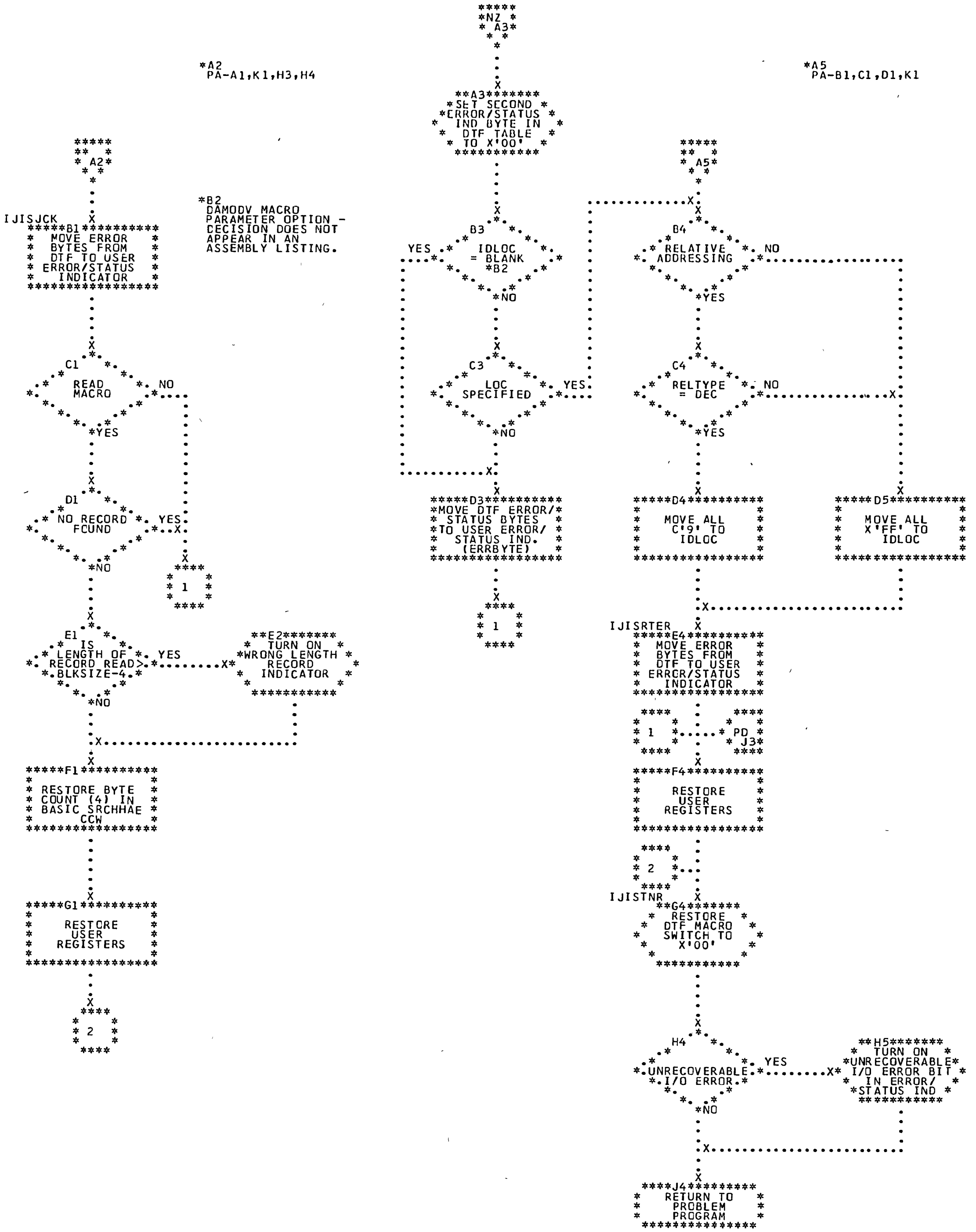


A1 ENTRY FROM WAITF MACRO
IJISWTM
B1 STORE USER REGISTERS
*B2 DAMODV MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
C1 I/O COMPLETE
NO
YES
C2 SVC 7 WAIT
D1 RECFORM = SPNUNB *B1
E1 IDLOC = BLANK *B2
F1 SPNUNB OR IDLOC SPECIFIED
G1 INITIALIZE POINTER WITH STARTING ADDR OF SPNUNB AREA IN DTF TABLE
H1 RECFORM = SPNUNB *B2
J1 SPNUNB SPECIFIED
J2 RESET ALL SPNUNB SWITCHES EXCEPT RELATIVE ADDRESSING
1
A3 IS RECORD OUT OF EXTENTS AREA
PB A3
IJISXTFD
B3 MOVE SECOND CCB TRANSMISSION BYTE TO DTF ERROR/STATUS INDICATOR
C3 WAS NO RECORD FOUND
C4 SET NO RECORD FOUND INDICATOR IN DTF TABLE
D3 END OF CYLINDER
E2 END OF VOLUME
2
IJISNRTO
E3 NO RECORD FOUND OR TRACK OVERFLOW
IJISEOV
F2 SET EOV INDICATOR IN DTF TABLE
F3 INCORRECT LENGTH
3
IJISFRM
G3 END OF FILE
H3 FEOVD = YES *B2
PA A5
J3 SET END OF FILE SWITCH
PA A1
IJISNEF
3
A5 RECFORM = SPNUNB
B5 SPNUNB SPECIFIED
C5 INCORRECT KEY LENGTH
D5 EOF RECORD
E5 SRCHM SPECIFIED
F5 SET END-OF-CYLINDER INDICATOR IN DTF TABLE
IJISICL
G4 SET INCORRECT LENGTH IND IN DTF TABLE
G5 SET NO-RECORD-FOUND IND IN DTF TABLE
2

Chart PA. DAMODV: WAITF Macro (Section 2 of 3)



## Chart PB. DAMODV: WAITF Macro (Section 3 of 3)



NZ A3
*A2 PA-A1,K1,H3,H4
*A5 PA-B1,C1,D1,K1
A3 SET SECOND ERROR/STATUS IND BYTE IN DTF TABLE TO X'00'
A2
A5
*B2 DAMODV MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
IJISJCK
B1 MOVE ERROR BYTES FROM DTF TO USER ERROR/STATUS INDICATOR
B3 IDLOC = BLANK *B2
YES
NO
B4 RELATIVE ADDRESSING
NO
YES
C1 READ MACRO
NO
YES
C3 LOC SPECIFIED
YES
NO
C4 RELTYPE = DEC
NO
YES
D1 NO RECORD FCUND
YES
NO
1
D3 MOVE DTF ERROR/ STATUS BYTES TO USER ERROR/ STATUS IND. (ERRBYTE)
1
D4 MOVE ALL C'9' TO IDLOC
D5 MOVE ALL X'FF' TO IDLOC
E1 IS LENGTH OF RECORD READ> BLKSIZE-4
YES
NO
E2 TURN ON WRONG LENGTH RECORD INDICATOR
IJISRTER
E4 MOVE ERROR BYTES FROM DTF TO USER ERRCR/STATUS INDICATOR
1
PD J3
F1 RESTORE BYTE COUNT (4) IN BASIC SRCHHAE CCW
F4 RESTORE USER REGISTERS
2
IJISTNR
G1 RESTCRE USER REGISTERS
G4 RESTORE DTF MACRO SWITCH TO X'00'
2
H4 UNRECOVERABLE I/O ERROR
YES
NO
H5 TURN ON UNRECOVERABLE I/O ERROR BIT IN ERROR/ STATUS IND
J4 RETURN TO PROBLEM PROGRAM

# Chart PC. DAMODV: IJIGET Subroutine

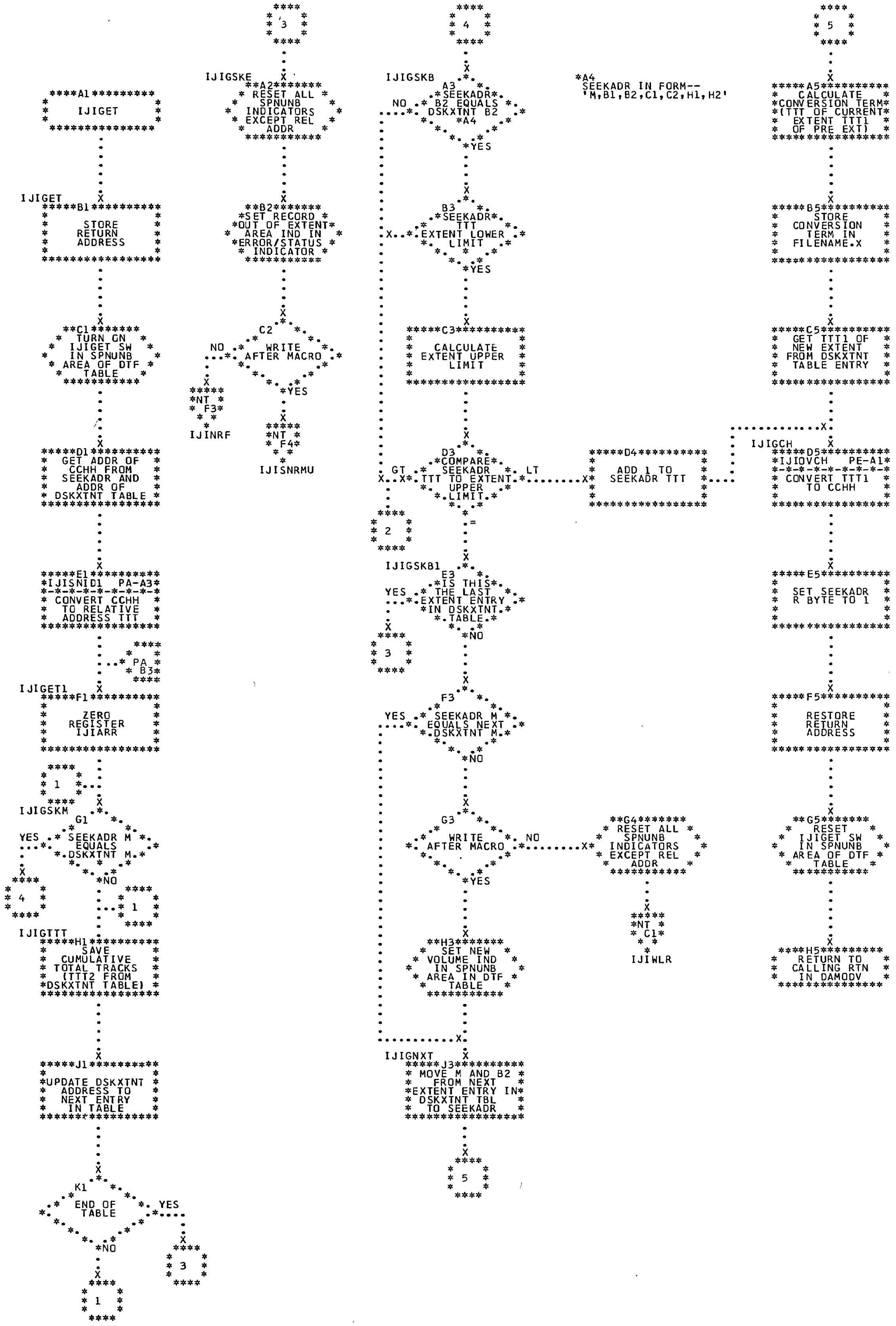
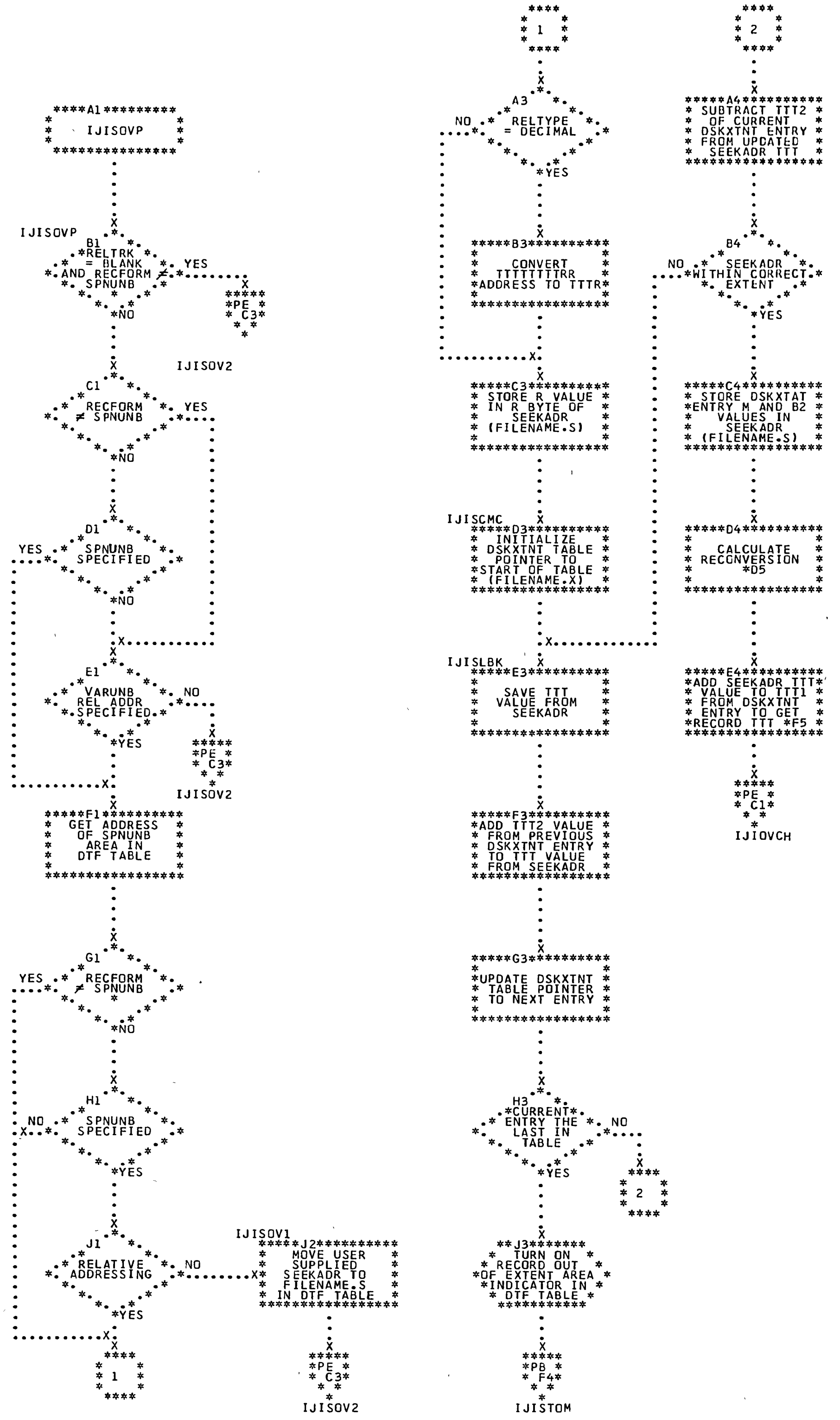

A1 IJIGET
IJIGET
B1 STORE RETURN ADDRESS
C1 TURN ON IJIGET SW IN SPNUNB AREA OF DTF TABLE
D1 GET ADDR OF CCHH FROM SEEKADR AND ADDR OF DSKXTNT TABLE
E1 IJISNID1 PA-A3 CONVERT CCHH TO RELATIVE ADDRESS TTT
PA B3
IJIGET1
F1 ZERO REGISTER IJIARR
1
IJIGSKM
G1 SEEKADR M EQUALS DSKXTNT M
YES
NO
4
1
IJIGTTT
H1 SAVE CUMULATIVE TOTAL TRACKS (TTT2 FROM DSKXTNT TABLE)
J1 UPDATE DSKXTNT ADDRESS TO NEXT ENTRY IN TABLE
K1 END OF TABLE
YES
NO
3
1
3
IJIGSKE
A2 RESET ALL SPNUNB INDICATORS EXCEPT REL ADDR
B2 SET RECORD OUT OF EXTENT AREA IND IN ERROR/STATUS INDICATOR
C2 WRITE AFTER MACRO
NO
YES
NT F3
IJINRF
NT F4
IJISNRMU
4
IJIGSKB
A3 SEEKADR B2 EQUALS DSKXTNT B2 *A4
NO
YES
B3 SEEKADR TTT EXTENT LOWER LIMIT
YES
C3 CALCULATE EXTENT UPPER LIMIT
D3 COMPARE SEEKADR TTT TO EXTENT UPPER LIMIT
GT
LT
=
2
IJIGSKB1
E3 IS THIS THE LAST EXTENT ENTRY IN DSKXTNT TABLE
YES
NO
3
F3 SEEKADR M EQUALS NEXT DSKXTNT M
YES
NO
G3 WRITE AFTER MACRO
NO
YES
H3 SET NEW VOLUME IND IN SPNUNB AREA IN DTF TABLE
IJIGNXT
J3 MOVE M AND B2 FROM NEXT EXTENT ENTRY IN DSKXTNT TBL TO SEEKADR
5
*A4
SEEKADR IN FORM--
'M,B1,B2,C1,C2,H1,H2'
D4 ADD 1 TO SEEKADR TTT
G4 RESET ALL SPNUNB INDICATORS EXCEPT REL ADDR
NT C1
IJIWLR
5
A5 CALCULATE CONVERSION TERM (TTT OF CURRENT EXTENT TTT1 OF PRE EXT)
B5 STORE CONVERSION TERM IN FILENAME.X
C5 GET TTT1 OF NEW EXTENT FROM DSKXTNT TABLE ENTRY
IJIGCH
D5 IJIOVCH PE-A1 CONVERT TTT1 TO CCHH
E5 SET SEEKADR R BYTE TO 1
F5 RESTORE RETURN ADDRESS
G5 RESET IJIGET SW IN SPNUNB AREA OF DTF TABLE
H5 RETURN TO CALLING RTN IN DAMODV

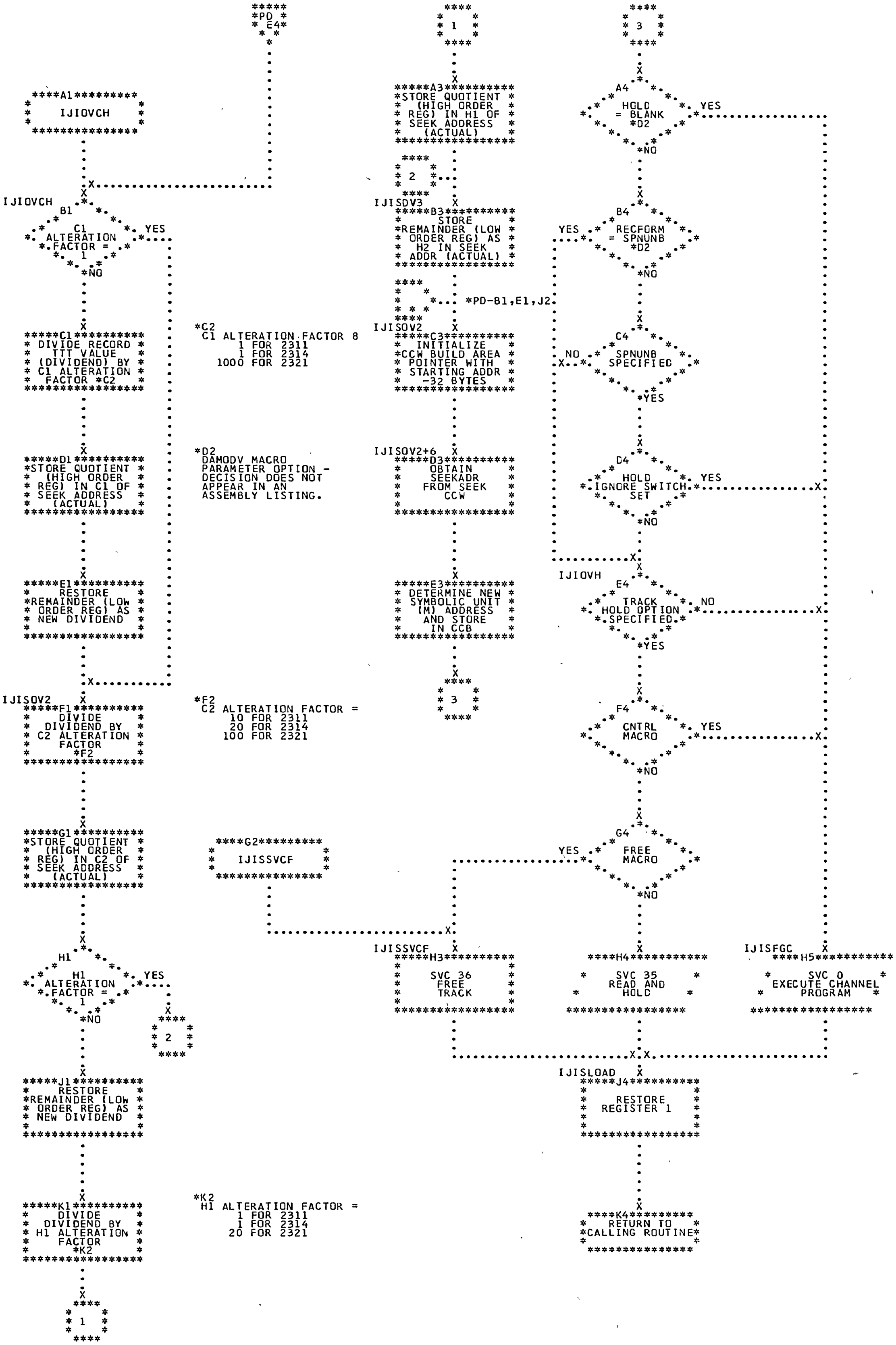
Chart PD. DAMODV: IJISOVP Seek Overlap Subroutine (Section 1 of 2)


A1 IJISOVP
IJISOVP
B1 RELTRK = BLANK AND RECFORM ≠ SPNUNB
YES
NO
PE C3
IJISOV2
C1 RECFORM ≠ SPNUNB
YES
NO
D1 SPNUNB SPECIFIED
YES
NO
E1 VARUNB REL ADDR SPECIFIED
NO
YES
PE C3
IJISOV2
F1 GET ADDRESS OF SPNUNB AREA IN DTF TABLE
G1 RECFORM ≠ SPNUNB
YES
NO
H1 SPNUNB SPECIFIED
NO
YES
J1 RELATIVE ADDRESSING
NO
YES
1
IJISOV1
J2 MOVE USER SUPPLIED SEEKADR TO FILENAME.S IN DTF TABLE
PE C3
IJISOV2
1
A3 RELTYPE = DECIMAL
NO
YES
B3 CONVERT TTTTTTTTRR ADDRESS TO TTTR
C3 STORE R VALUE IN R BYTE OF SEEKADR (FILENAME.S)
IJISCMC
D3 INITIALIZE DSKXTNT TABLE POINTER TO START OF TABLE (FILENAME.X)
IJISLBK
E3 SAVE TTT VALUE FROM SEEKADR
F3 ADD TTT2 VALUE FROM PREVIOUS DSKXTNT ENTRY TO TTT VALUE FROM SEEKADR
G3 UPDATE DSKXTNT TABLE POINTER TO NEXT ENTRY
H3 CURRENT ENTRY THE LAST IN TABLE
NO
2
YES
J3 TURN ON RECORD OUT OF EXTENT AREA INDICATOR IN DTF TABLE
PB F4
IJISTOM
2
A4 SUBTRACT TTT2 OF CURRENT DSKXTNT ENTRY FROM UPDATED SEEKADR TTT
B4 SEEKADR WITHIN CORRECT EXTENT
NO
YES
C4 STORE DSKXTAT ENTRY M AND B2 VALUES IN SEEKADR (FILENAME.S)
D4 CALCULATE RECCNVERSION *D5
E4 ADD SEEKADR TTT VALUE TO TTT1 FROM DSKXTNT ENTRY TO GET RECORD TTT *F5
PE C1
IJIOVCH

*D5
RECONVERSION FACTOR = CUMULATIVE TRACK TOTAL OF PREVIOUS EXTENT (TTT2) MINUS NUMBER OF TRACKS TO LOWER LIMIT OF CURRENT EXTENT (TTT1).

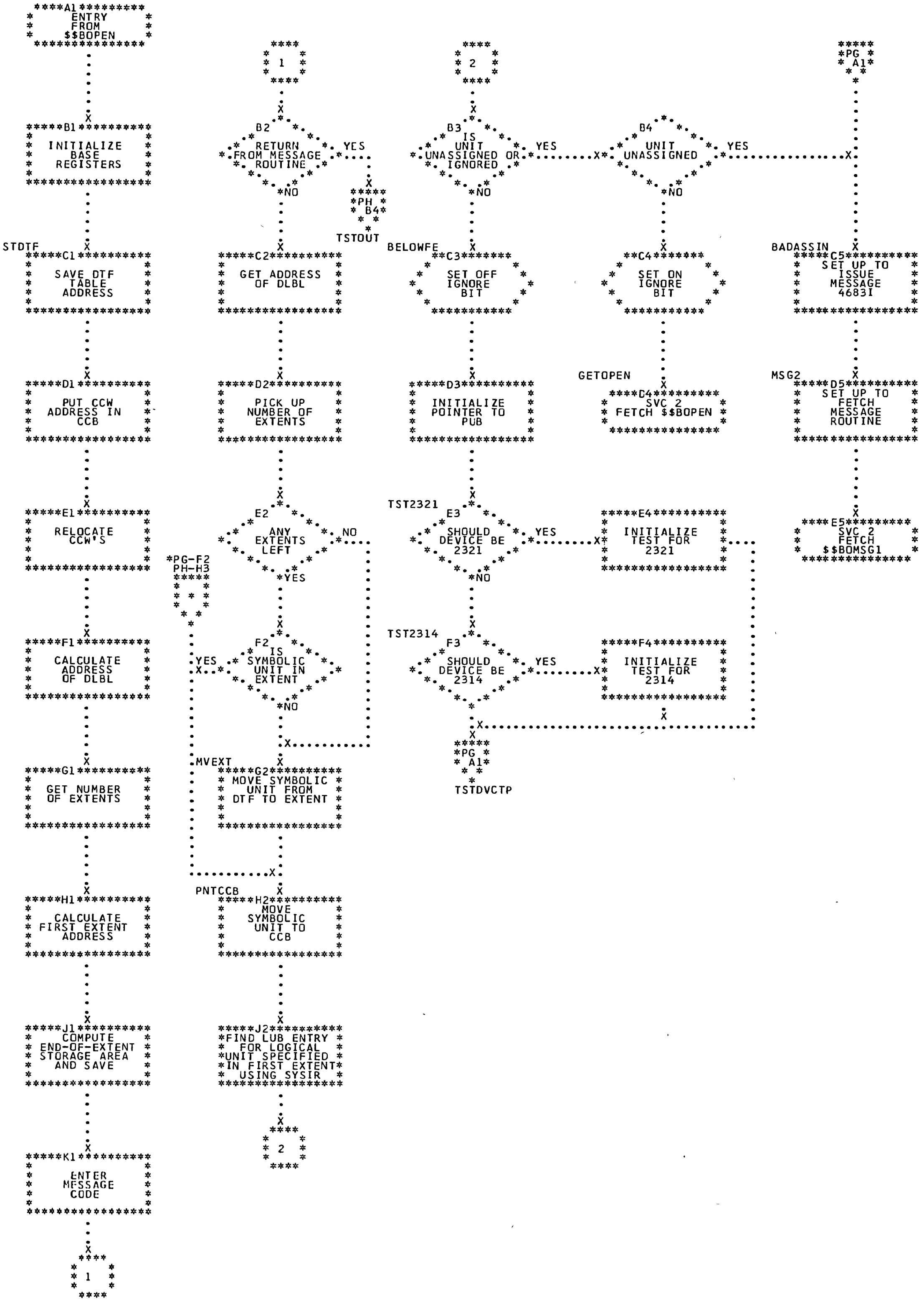
*F5
SEEKADR TTT VALUE SAVED AT BLOCK F3 IS ADDED TO THE LOWER LIMIT OF CURRENT EXTENT.

## Chart PE. DAMODV: IJISOVP Seek Overlap Subroutine (Section 2 of 2)



PD E4
1
3
A1 IJIOVCH
A3 STORE QUOTIENT (HIGH ORDER REG) IN H1 OF SEEK ADDRESS (ACTUAL)
A4 HOLD = BLANK *D2
YES
NO
2
IJIOVCH
B1 C1 ALTERATION FACTOR = 1
YES
NO
IJISDV3
B3 STORE REMAINDER (LOW ORDER REG) AS H2 IN SEEK ADDR (ACTUAL)
B4 RECFORM = SPNUNB *D2
YES
NO
*PD-B1,E1,J2
*C2 C1 ALTERATION FACTOR 8 1 FOR 2311 1 FOR 2314 1000 FOR 2321
IJISOV2
C1 DIVIDE RECORD TTT VALUE (DIVIDEND) BY C1 ALTERATION FACTOR *C2
C3 INITIALIZE CCW BUILD AREA POINTER WITH STARTING ADDR -32 BYTES
C4 SPNUNB SPECIFIED
NO
YES
*D2 DAMODV MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
IJISOV2+6
D1 STORE QUOTIENT (HIGH ORDER REG) IN C1 OF SEEK ADDRESS (ACTUAL)
D3 OBTAIN SEEKADR FROM SEEK CCW
D4 HOLD IGNORE SWITCH SET
YES
NO
E1 RESTORE REMAINDER (LOW ORDER REG) AS NEW DIVIDEND
E3 DETERMINE NEW SYMBOLIC UNIT (M) ADDRESS AND STORE IN CCB
IJIOVH
E4 TRACK HOLD OPTION SPECIFIED
NO
YES
3
IJISOV2
F1 DIVIDE DIVIDEND BY C2 ALTERATION FACTOR *F2
*F2 C2 ALTERATION FACTOR = 10 FOR 2311 20 FOR 2314 100 FOR 2321
F4 CNTRL MACRO
YES
NO
G1 STORE QUOTIENT (HIGH ORDER REG) IN C2 OF SEEK ADDRESS (ACTUAL)
G2 IJISSVCF
G4 FREE MACRO
YES
NO
H1 H1 ALTERATION FACTOR = 1
YES
NO
2
IJISSVCF
H3 SVC 36 FREE TRACK
H4 SVC 35 READ AND HOLD
IJISFGC
H5 SVC 0 EXECUTE CHANNEL PROGRAM
J1 RESTORE REMAINDER (LOW ORDER REG) AS NEW DIVIDEND
IJISLOAD
J4 RESTORE REGISTER 1
K1 DIVIDE DIVIDEND BY H1 ALTERATION FACTOR *K2
*K2 H1 ALTERATION FACTOR = 1 FOR 2311 1 FOR 2314 20 FOR 2321
K4 RETURN TO CALLING ROUTINE
1

Chart PF. $$BODAIN: DA Open Input/Output (Section 1 of 3)



## Chart PG. $$BODAIN: DA Open Input/Output (Section 2 of 3)

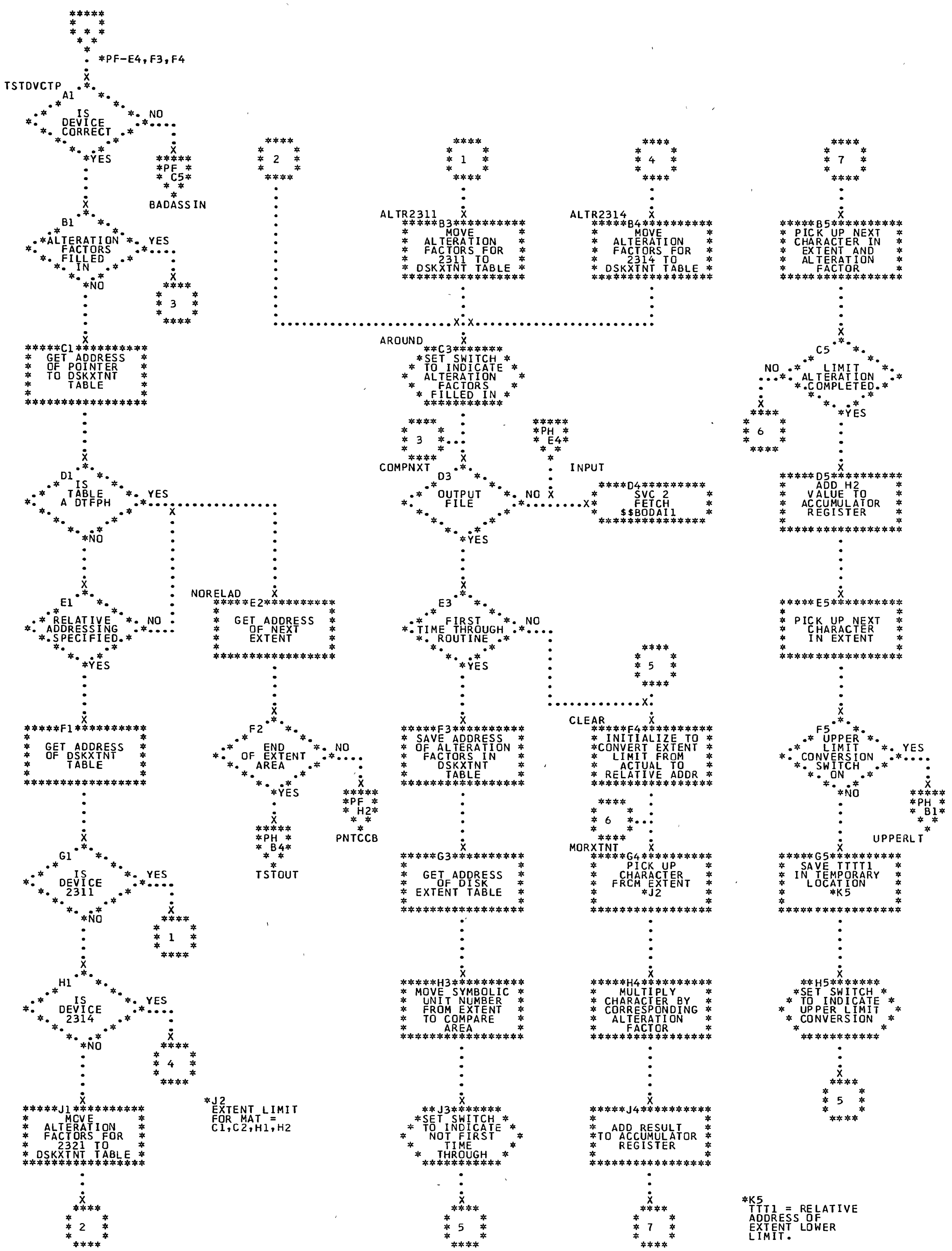
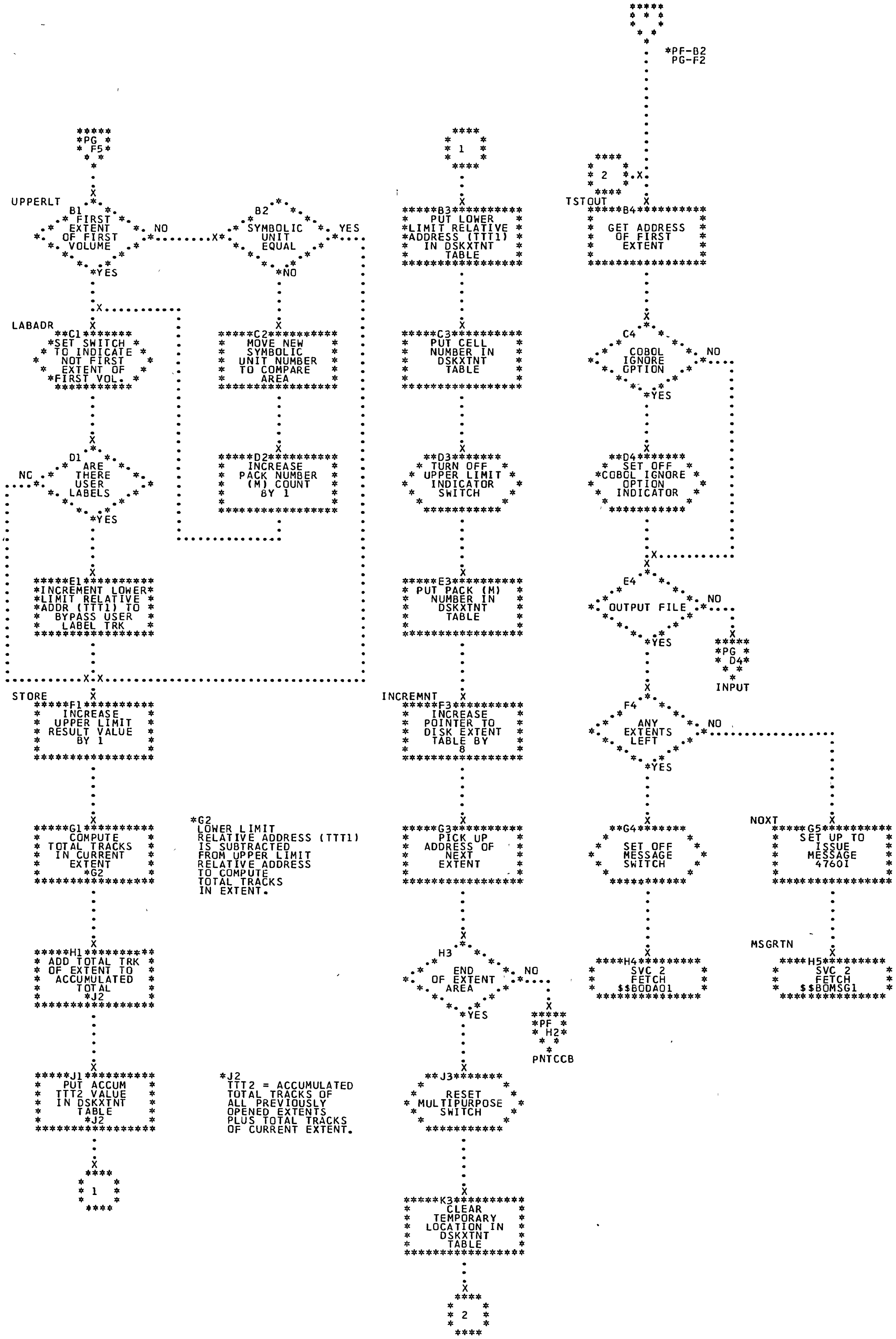
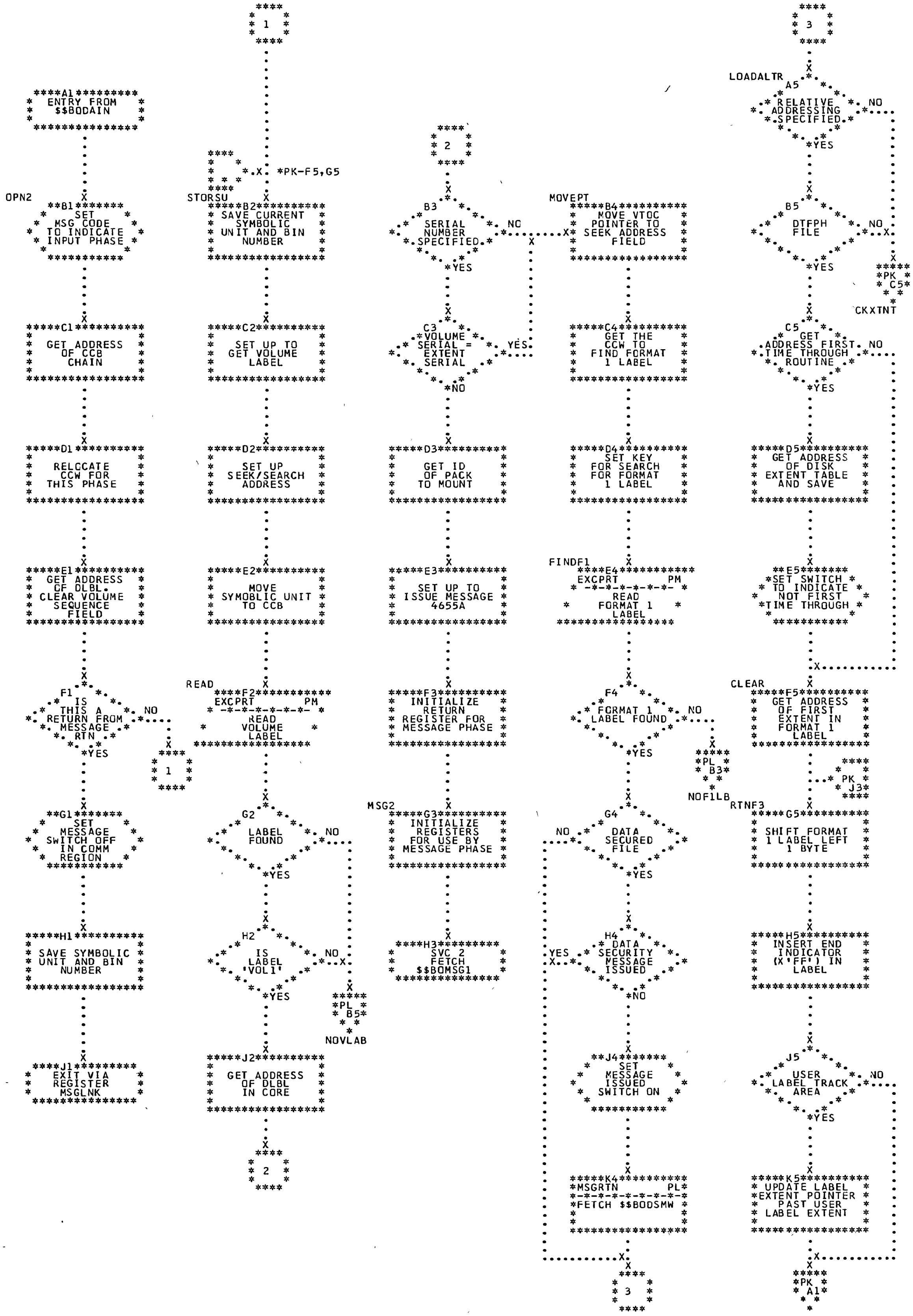

PF-E4,F3,F4
TSTDVCTP
A1
IS DEVICE CORRECT
NO
YES
PF C5
BADASSIN
B1
ALTERATION FACTORS FILLED IN
YES
NO
3
C1
GET ADDRESS OF POINTER TO DSKXTNT TABLE
D1
IS TABLE A DTFPH
YES
NO
E1
RELATIVE ADDRESSING SPECIFIED
NO
YES
F1
GET ADDRESS OF DSKXTNT TABLE
G1
IS DEVICE 2311
YES
NO
1
H1
IS DEVICE 2314
YES
NO
4
J1
MOVE ALTERATION FACTORS FOR 2321 TO DSKXTNT TABLE
2
2
NORELAD
E2
GET ADDRESS OF NEXT EXTENT
F2
END OF EXTENT AREA
NO
YES
PF H2
PNTCCB
PH B4
TSTOUT
*J2
EXTENT LIMIT FOR MAT = C1,C2,H1,H2
1
ALTR2311
B3
MOVE ALTERATION FACTORS FOR 2311 TO DSKXTNT TABLE
AROUND
C3
SET SWITCH TO INDICATE ALTERATION FACTORS FILLED IN
3
COMPNXT
D3
OUTPUT FILE
NO
YES
E3
FIRST TIME THROUGH ROUTINE
NO
YES
F3
SAVE ADDRESS OF ALTERATION FACTORS IN DSKXTNT TABLE
G3
GET ADDRESS OF DISK EXTENT TABLE
H3
MOVE SYMBOLIC UNIT NUMBER FROM EXTENT TO COMPARE AREA
J3
SET SWITCH TO INDICATE NOT FIRST TIME THROUGH
5
4
ALTR2314
B4
MOVE ALTERATION FACTORS FOR 2314 TO DSKXTNT TABLE
PH E4
INPUT
D4
SVC 2 FETCH $$BODAI1
5
CLEAR
F4
INITIALIZE TO CONVERT EXTENT LIMIT FROM ACTUAL TO RELATIVE ADDR
6
MORXTNT
G4
PICK UP CHARACTER FROM EXTENT *J2
H4
MULTIPLY CHARACTER BY CORRESPONDING ALTERATION FACTOR
J4
ADD RESULT TO ACCUMULATOR REGISTER
7
7
B5
PICK UP NEXT CHARACTER IN EXTENT AND ALTERATION FACTOR
C5
LIMIT ALTERATION COMPLETED
NO
YES
6
D5
ADD H2 VALUE TO ACCUMULATOR REGISTER
E5
PICK UP NEXT CHARACTER IN EXTENT
F5
UPPER LIMIT CONVERSION SWITCH ON
YES
NO
PH B1
UPPERLT
G5
SAVE TTTT1 IN TEMPORARY LOCATION *K5
H5
SET SWITCH TO INDICATE UPPER LIMIT CONVERSION
5
*K5
TTT1 = RELATIVE ADDRESS OF EXTENT LOWER LIMIT.

Chart PH. $$BODAIN: DA Open Input/Output (Section 3 of 3)



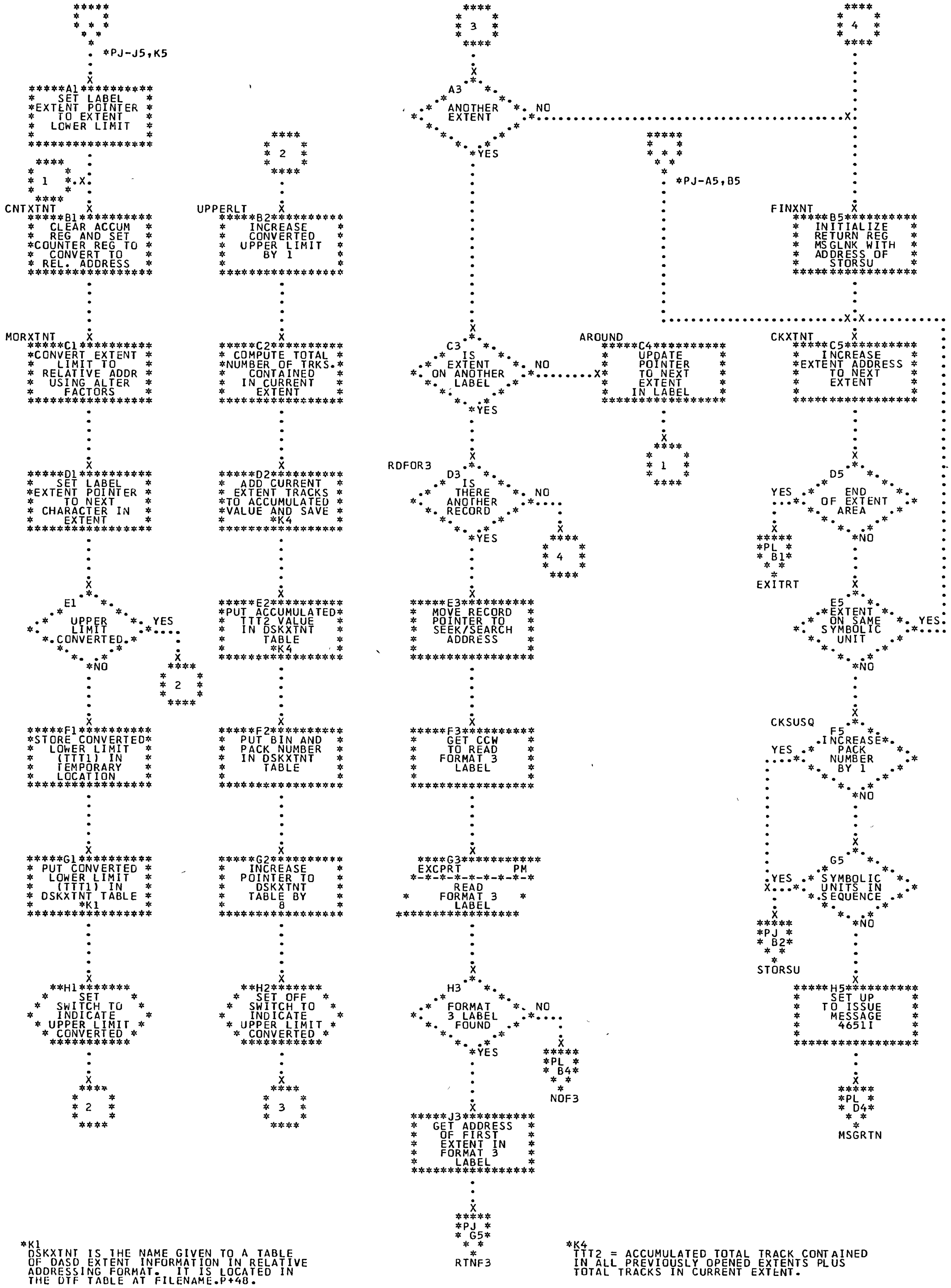
PF-B2
PG-F2
PG F5
UPPERLT
B1
FIRST EXTENT OF FIRST VOLUME
NO
YES
B2
SYMBOLIC UNIT EQUAL
YES
NO
1
B3
PUT LOWER LIMIT RELATIVE ADDRESS (TTT1) IN DSKXTNT TABLE
2
TSTOUT
B4
GET ADDRESS OF FIRST EXTENT
LABADR
C1
SET SWITCH TO INDICATE NOT FIRST EXTENT OF FIRST VOL.
C2
MOVE NEW SYMBOLIC UNIT NUMBER TO COMPARE AREA
C3
PUT CELL NUMBER IN DSKXTNT TABLE
C4
COBOL IGNORE OPTION
NO
YES
D1
ARE THERE USER LABELS
NO
YES
D2
INCREASE PACK NUMBER (M) COUNT BY 1
D3
TURN OFF UPPER LIMIT INDICATOR SWITCH
D4
SET OFF COBOL IGNORE OPTION INDICATOR
E1
INCREMENT LOWER LIMIT RELATIVE ADDR (TTT1) TO BYPASS USER LABEL TRK
E3
PUT PACK (M) NUMBER IN DSKXTNT TABLE
E4
OUTPUT FILE
NO
YES
PG D4
INPUT
STORE
F1
INCREASE UPPER LIMIT RESULT VALUE BY 1
INCREMNT
F3
INCREASE POINTER TO DISK EXTENT TABLE BY 8
F4
ANY EXTENTS LEFT
NO
YES
G1
COMPUTE TOTAL TRACKS IN CURRENT EXTENT
*G2
*G2
LOWER LIMIT RELATIVE ADDRESS (TTT1) IS SUBTRACTED FROM UPPER LIMIT RELATIVE ADDRESS TO COMPUTE TOTAL TRACKS IN EXTENT.
G3
PICK UP ADDRESS OF NEXT EXTENT
G4
SET OFF MESSAGE SWITCH
NOXT
G5
SET UP TO ISSUE MESSAGE 4760I
H1
ADD TOTAL TRK OF EXTENT TO ACCUMULATED TOTAL
*J2
H3
END OF EXTENT AREA
NO
YES
PF H2
PNTCCB
H4
SVC 2 FETCH $$BODAO1
MSGRTN
H5
SVC 2 FETCH $$BOMSG1
J1
PUT ACCUM TTT2 VALUE IN DSKXTNT TABLE
*J2
*J2
TTT2 = ACCUMULATED TOTAL TRACKS OF ALL PREVIOUSLY OPENED EXTENTS PLUS TOTAL TRACKS OF CURRENT EXTENT.
J3
RESET MULTIPURPOSE SWITCH
1
K3
CLEAR TEMPORARY LOCATION IN DSKXTNT TABLE
2

## Chart PJ. $$BODAI1: DA Open Input (Section 1 of 4)



1
2
3
A1 ENTRY FROM $$BODAIN
OPN2
B1 SET MSG CODE TO INDICATE INPUT PHASE
C1 GET ADDRESS OF CCB CHAIN
D1 RELCCATE CCW FOR THIS PHASE
E1 GET ADDRESS OF DLBL. CLEAR VOLUME SEQUENCE FIELD
F1 IS THIS A RETURN FROM MESSAGE RTN
NO
YES
1
G1 SET MESSAGE SWITCH OFF IN COMM REGION
H1 SAVE SYMBOLIC UNIT AND BIN NUMBER
J1 EXIT VIA REGISTER MSGLNK
*PK-F5,G5
STORSU
B2 SAVE CURRENT SYMBOLIC UNIT AND BIN NUMBER
C2 SET UP TO GET VOLUME LABEL
D2 SET UP SEEK/SEARCH ADDRESS
E2 MOVE SYMOBLIC UNIT TO CCB
READ
F2 EXCPRT PM READ VOLUME LABEL
G2 LABEL FOUND
NO
YES
H2 IS LABEL 'VOL1'
NO
YES
PL B5
NOVLAB
J2 GET ADDRESS OF DLBL IN CORE
2
B3 SERIAL NUMBER SPECIFIED
NO
YES
C3 VOLUME SERIAL = EXTENT SERIAL
YES
NO
D3 GET ID OF PACK TO MOUNT
E3 SET UP TO ISSUE MESSAGE 4655A
F3 INITIALIZE RETURN REGISTER FOR MESSAGE PHASE
MSG2
G3 INITIALIZE REGISTERS FOR USE BY MESSAGE PHASE
H3 SVC 2 FETCH $$BOMSG1
MOVEPT
B4 MOVE VTOC POINTER TO SEEK ADDRESS FIELD
C4 GET THE CCW TO FIND FORMAT 1 LABEL
D4 SET KEY FOR SEARCH FOR FORMAT 1 LABEL
FINDF1
E4 EXCPRT PM READ FORMAT 1 LABEL
F4 FORMAT 1 LABEL FOUND
NO
YES
PL B3
NOFILB
G4 DATA SECURED FILE
NO
YES
H4 DATA SECURITY MESSAGE ISSUED
YES
NO
J4 SET MESSAGE ISSUED SWITCH ON
K4 MSGRTN PL FETCH $$BODSMW
3
LOADALTR
A5 RELATIVE ADDRESSING SPECIFIED
NO
YES
B5 DTFPH FILE
NO
YES
PK C5
CKXTNT
C5 GET ADDRESS FIRST TIME THROUGH ROUTINE
NO
YES
D5 GET ADDRESS OF DISK EXTENT TABLE AND SAVE
E5 SET SWITCH TO INDICATE NOT FIRST TIME THROUGH
CLEAR
F5 GET ADDRESS OF FIRST EXTENT IN FORMAT 1 LABEL
PK J3
RTNF3
G5 SHIFT FORMAT 1 LABEL LEFT 1 BYTE
H5 INSERT END INDICATOR (X'FF') IN LABEL
J5 USER LABEL TRACK AREA
NO
YES
K5 UPDATE LABEL EXTENT POINTER PAST USER LABEL EXTENT
PK A1

## Chart PK. $$BODAI1: DA Open Input (Section 2 of 4)


*PJ-J5,K5
A1
SET LABEL EXTENT POINTER TO EXTENT LOWER LIMIT
1
CNTXTNT
B1
CLEAR ACCUM REG AND SET COUNTER REG TO CONVERT TO REL. ADDRESS
MORXTNT
C1
CONVERT EXTENT LIMIT TO RELATIVE ADDR USING ALTER FACTORS
D1
SET LABEL EXTENT POINTER TO NEXT CHARACTER IN EXTENT
E1
UPPER LIMIT CONVERTED.
YES
NO
2
F1
STORE CONVERTED LOWER LIMIT (TTT1) IN TEMPORARY LOCATION
G1
PUT CONVERTED LOWER LIMIT (TTT1) IN DSKXTNT TABLE
*K1
H1
SET SWITCH TO INDICATE UPPER LIMIT CONVERTED
2
2
UPPERLT
B2
INCREASE CONVERTED UPPER LIMIT BY 1
C2
COMPUTE TOTAL NUMBER OF TRKS. CONTAINED IN CURRENT EXTENT
D2
ADD CURRENT EXTENT TRACKS TO ACCUMULATED VALUE AND SAVE
*K4
E2
PUT ACCUMULATED TTT2 VALUE IN DSKXTNT TABLE
*K4
F2
PUT BIN AND PACK NUMBER IN DSKXTNT TABLE
G2
INCREASE POINTER TO DSKXTNT TABLE BY 8
H2
SET OFF SWITCH TO INDICATE UPPER LIMIT CONVERTED
3
3
A3
ANOTHER EXTENT
NO
YES
C3
IS EXTENT ON ANOTHER LABEL
NO
YES
RDFOR3
D3
IS THERE ANOTHER RECORD
NO
YES
4
E3
MOVE RECORD POINTER TO SEEK/SEARCH ADDRESS
F3
GET CCW TO READ FORMAT 3 LABEL
G3
EXCPRT PM
READ FORMAT 3 LABEL
H3
FORMAT 3 LABEL FOUND
NO
YES
PL B4
NOF3
J3
GET ADDRESS OF FIRST EXTENT IN FORMAT 3 LABEL
PJ G5
RTNF3
*PJ-A5,B5
AROUND
C4
UPDATE POINTER TO NEXT EXTENT IN LABEL
1
4
FINXNT
B5
INITIALIZE RETURN REG MSGLNK WITH ADDRESS OF STORSU
CKXTNT
C5
INCREASE EXTENT ADDRESS TO NEXT EXTENT
D5
END OF EXTENT AREA
YES
NO
PL B1
EXITRT
E5
EXTENT ON SAME SYMBOLIC UNIT
YES
NO
CKSUSQ
F5
INCREASE PACK NUMBER BY 1
YES
NO
G5
SYMBOLIC UNITS IN SEQUENCE
YES
NO
PJ B2
STORSU
H5
SET UP TO ISSUE MESSAGE 4651I
PL D4
MSGRTN

*K1
DSKXTNT IS THE NAME GIVEN TO A TABLE OF DASD EXTENT INFORMATION IN RELATIVE ADDRESSING FORMAT. IT IS LOCATED IN THE DTF TABLE AT FILENAME.P+48.

*K4
TTT2 = ACCUMULATED TOTAL TRACK CONTAINED IN ALL PREVIOUSLY OPENED EXTENTS PLUS TOTAL TRACKS IN CURRENT EXTENT.

## Chart PL. $$BODAI1: DA Open Input (Section 3 of 4)

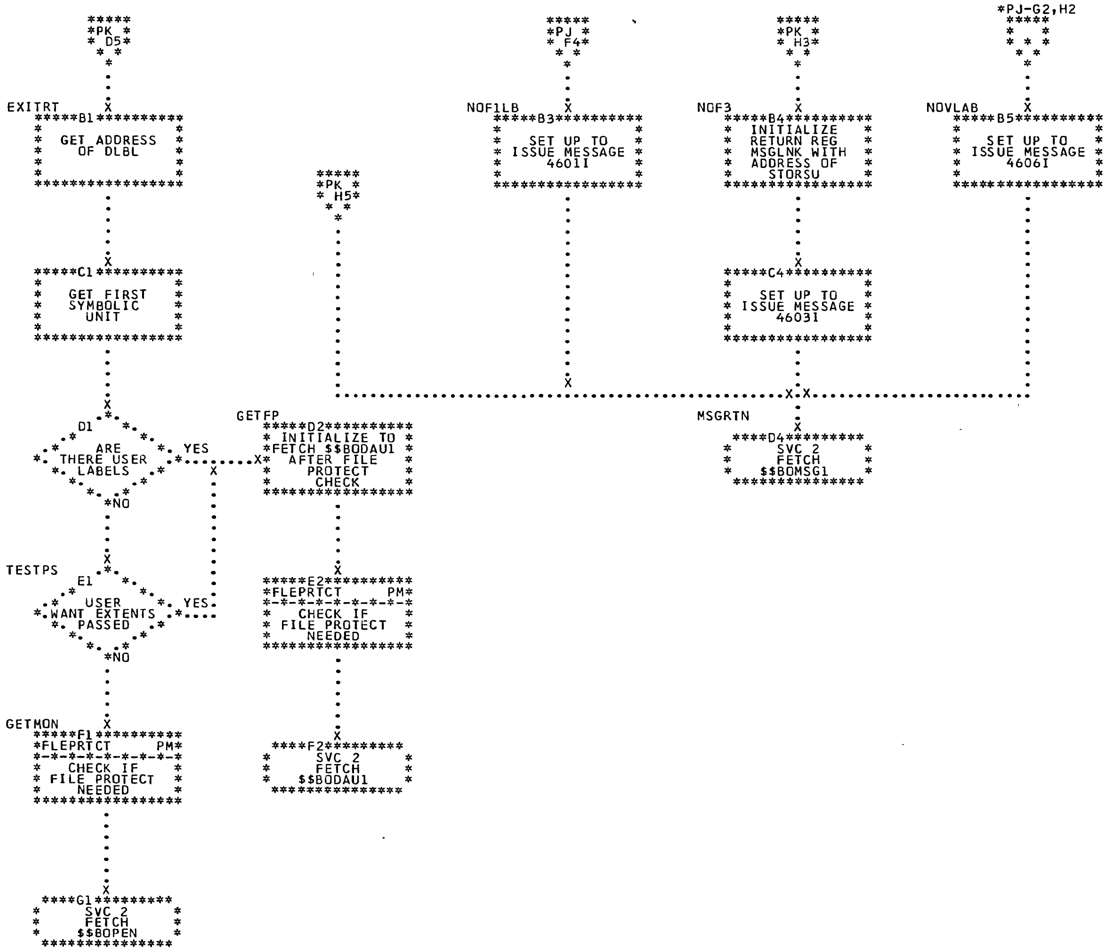


Chart PM. $$BODAI1: DA Open Input (Section 4 of 4)

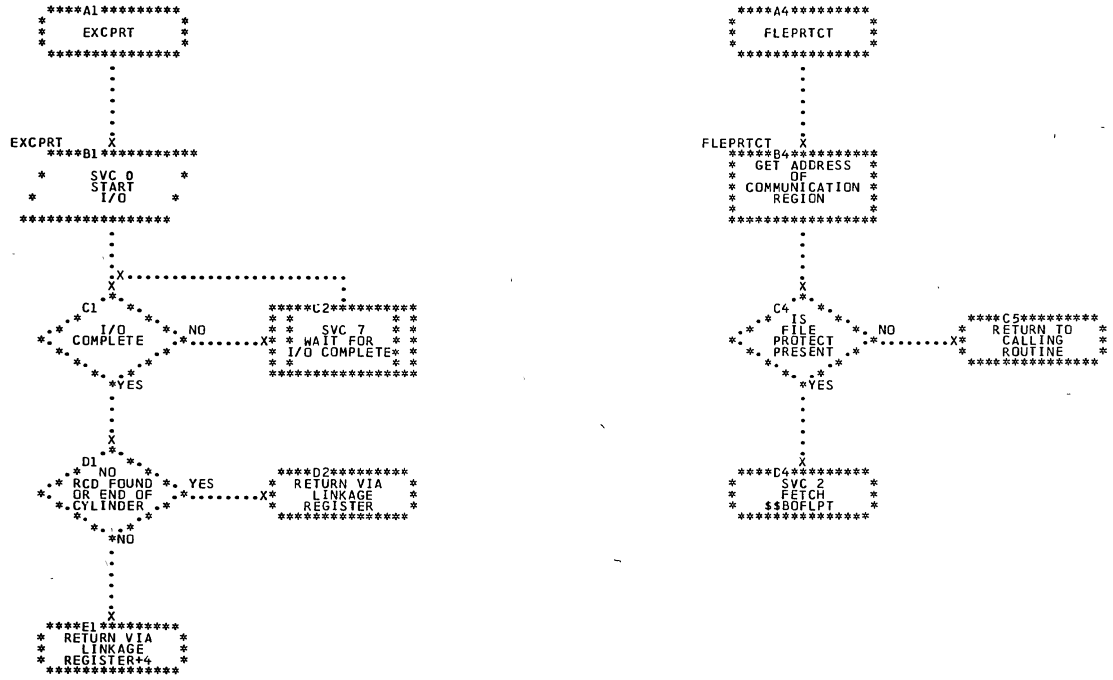

A1
EXCPRT
EXCPRT
B1
SVC 0 START I/O
C1
I/O COMPLETE
NO
YES
C2
SVC 7 WAIT FOR I/O COMPLETE
D1
NO RCD FOUND OR END OF CYLINDER
YES
NO
D2
RETURN VIA LINKAGE REGISTER
E1
RETURN VIA LINKAGE REGISTER+4
A4
FLEPRTCT
FLEPRTCT
B4
GET ADDRESS OF COMMUNICATION REGION
C4
IS FILE PROTECT PRESENT
NO
YES
C5
RETURN TO CALLING ROUTINE
C4
SVC 2 FETCH $$BOFLPT

Chart PN. $$BODA01: DA Open Output, Phase 1 (Section 1 of 4)

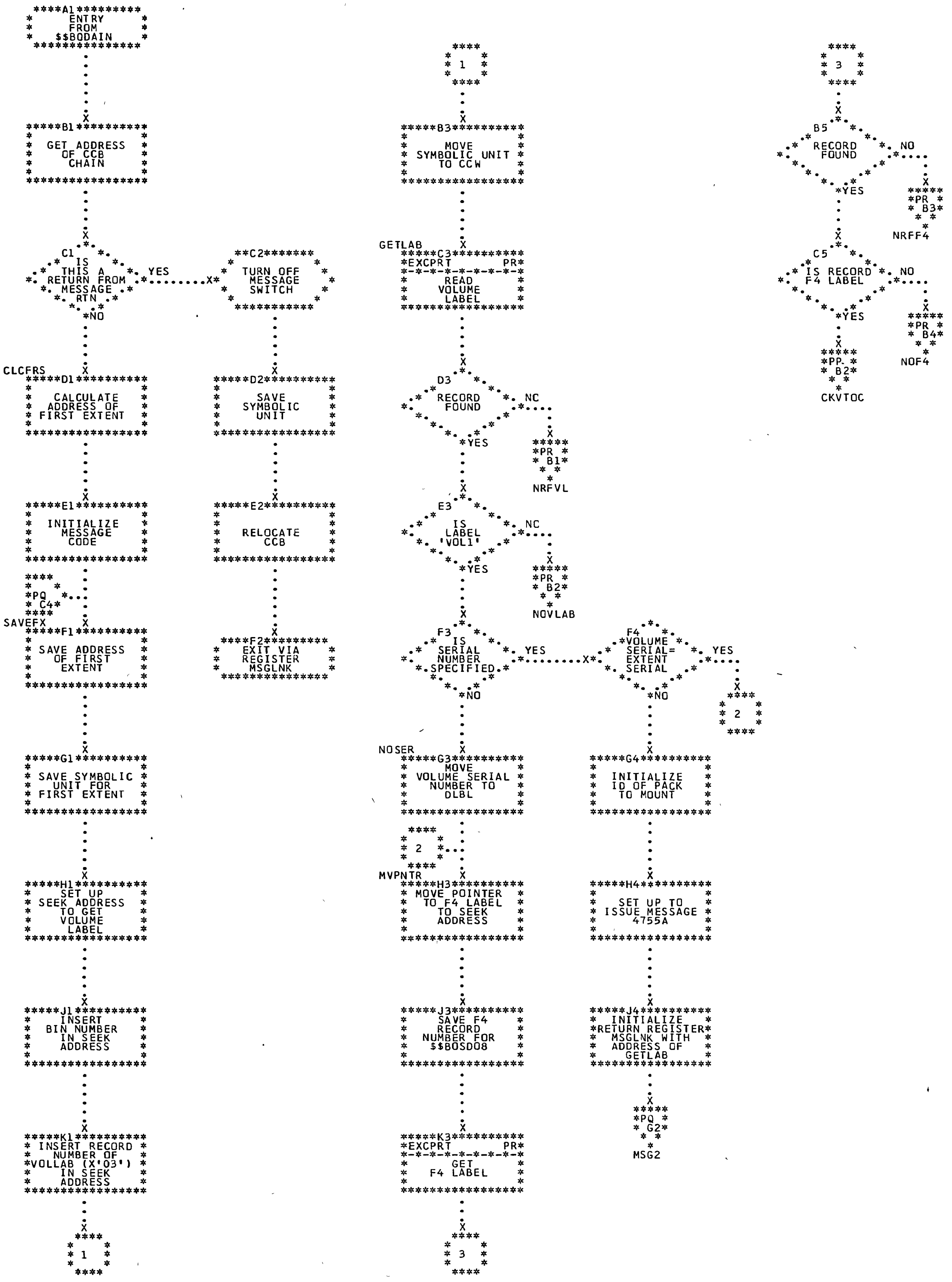
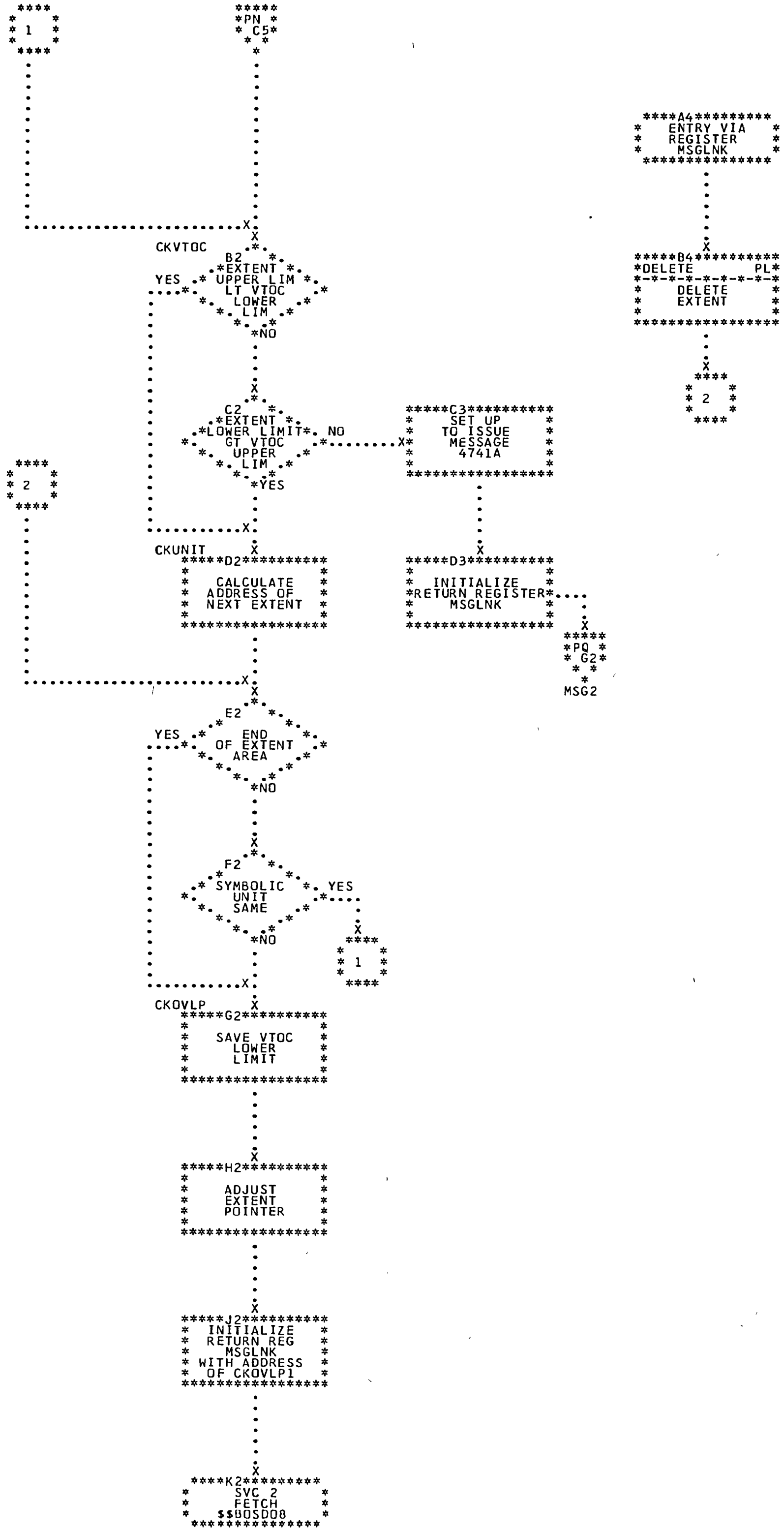
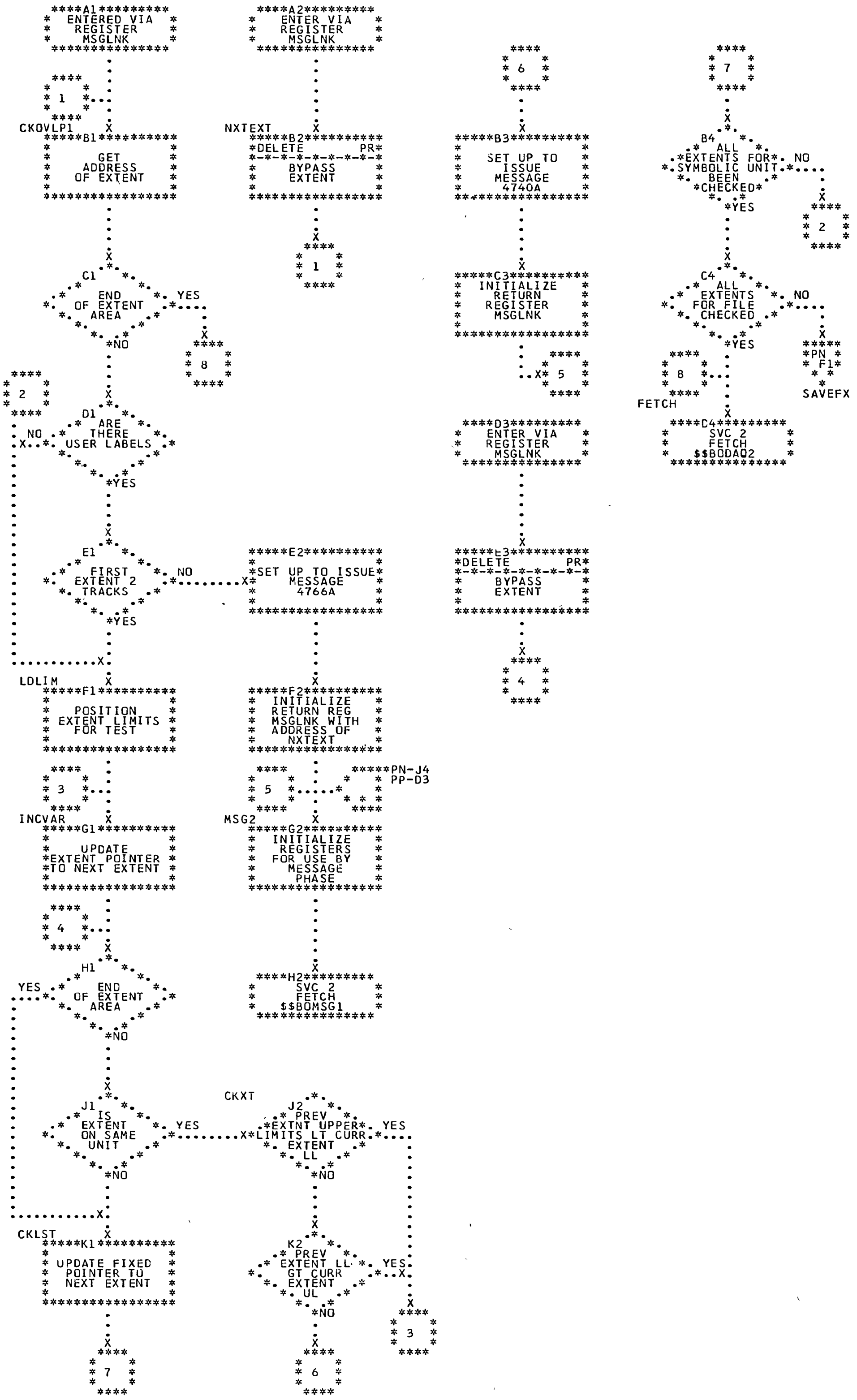

A1
ENTRY FROM $$BODAIN
B1
GET ADDRESS OF CCB CHAIN
C1
IS THIS A RETURN FROM MESSAGE RTN
YES
NO
C2
TURN OFF MESSAGE SWITCH
CLCFRS
D1
CALCULATE ADDRESS OF FIRST EXTENT
D2
SAVE SYMBOLIC UNIT
E1
INITIALIZE MESSAGE CODE
E2
RELOCATE CCB
PQ C4
SAVEFX
F1
SAVE ADDRESS OF FIRST EXTENT
F2
EXIT VIA REGISTER MSGLNK
G1
SAVE SYMBOLIC UNIT FOR FIRST EXTENT
H1
SET UP SEEK ADDRESS TO GET VOLUME LABEL
J1
INSERT BIN NUMBER IN SEEK ADDRESS
K1
INSERT RECORD NUMBER OF VOLLAB (X'03') IN SEEK ADDRESS
1
1
B3
MOVE SYMBOLIC UNIT TO CCW
GETLAB
C3
EXCPRT PR
READ VOLUME LABEL
D3
RECORD FOUND
NC
YES
PR B1
NRFVL
E3
IS LABEL 'VOL1'
NC
YES
PR B2
NOVLAB
F3
IS SERIAL NUMBER SPECIFIED
YES
NO
F4
VOLUME SERIAL= EXTENT SERIAL
YES
NO
2
NOSER
G3
MOVE VOLUME SERIAL NUMBER TO DLBL
G4
INITIALIZE ID OF PACK TO MOUNT
2
MVPNTR
H3
MOVE POINTER TO F4 LABEL TO SEEK ADDRESS
H4
SET UP TO ISSUE MESSAGE 4755A
J3
SAVE F4 RECORD NUMBER FOR $$BOSD08
J4
INITIALIZE RETURN REGISTER MSGLNK WITH ADDRESS OF GETLAB
PQ G2
MSG2
K3
EXCPRT PR
GET F4 LABEL
3
3
B5
RECORD FOUND
NO
YES
PR B3
NRFF4
C5
IS RECORD F4 LABEL
NO
YES
PR B4
NOF4
PP B2
CKVTOC

Chart PP. $$BODA01: DA Open Output, Phase 1 (Section 2 of 4)


1
PN
C5
A4
ENTRY VIA REGISTER MSGLNK
CKVTOC
B2
EXTENT UPPER LIM LT VTOC LOWER LIM
YES
NO
B4
DELETE PL
DELETE EXTENT
2
C2
EXTENT LOWER LIMIT GT VTOC UPPER LIM
NO
YES
C3
SET UP TO ISSUE MESSAGE 4741A
2
CKUNIT
D2
CALCULATE ADDRESS OF NEXT EXTENT
D3
INITIALIZE RETURN REGISTER MSGLNK
PQ
G2
MSG2
E2
END OF EXTENT AREA
YES
NO
F2
SYMBOLIC UNIT SAME
YES
NO
1
CKOVLP
G2
SAVE VTOC LOWER LIMIT
H2
ADJUST EXTENT POINTER
J2
INITIALIZE RETURN REG MSGLNK WITH ADDRESS OF CKOVLP1
K2
SVC 2 FETCH $$BOSD08

## Chart PQ. $$BODA01: DA Open Output, Phase 1 (Section 3 of 4)


A1 ENTERED VIA REGISTER MSGLNK
1
CKOVLP1
B1 GET ADDRESS OF EXTENT
C1 END OF EXTENT AREA
YES
NO
8
2
D1 ARE THERE USER LABELS
NO
YES
E1 FIRST EXTENT 2 TRACKS
NO
YES
LDLIM
F1 POSITION EXTENT LIMITS FOR TEST
3
INCVAR
G1 UPDATE EXTENT POINTER TO NEXT EXTENT
4
H1 END OF EXTENT AREA
YES
NO
J1 IS EXTENT ON SAME UNIT
YES
NO
CKLST
K1 UPDATE FIXED POINTER TO NEXT EXTENT
7
A2 ENTER VIA REGISTER MSGLNK
NXTEXT
B2 DELETE PR BYPASS EXTENT
1
E2 SET UP TO ISSUE MESSAGE 4766A
F2 INITIALIZE RETURN REG MSGLNK WITH ADDRESS OF NXTEXT
5
PN-J4
PP-D3
MSG2
G2 INITIALIZE REGISTERS FOR USE BY MESSAGE PHASE
H2 SVC 2 FETCH $$BOMSG1
CKXT
J2 PREV EXTNT UPPER LIMITS LT CURR. EXTENT LL
YES
NO
K2 PREV EXTENT LL GT CURR EXTENT UL
YES
NO
6
3
6
B3 SET UP TO ISSUE MESSAGE 4740A
C3 INITIALIZE RETURN REGISTER MSGLNK
5
D3 ENTER VIA REGISTER MSGLNK
E3 DELETE PR BYPASS EXTENT
4
7
B4 ALL EXTENTS FOR SYMBOLIC UNIT BEEN CHECKED
NO
YES
2
C4 ALL EXTENTS FOR FILE CHECKED
NO
YES
PN F1
SAVEFX
8
FETCH
C4 SVC 2 FETCH $$BODA02

## Chart PR. $$BODA01: DA Open Output, Phase 1 (Section 4 of 4)

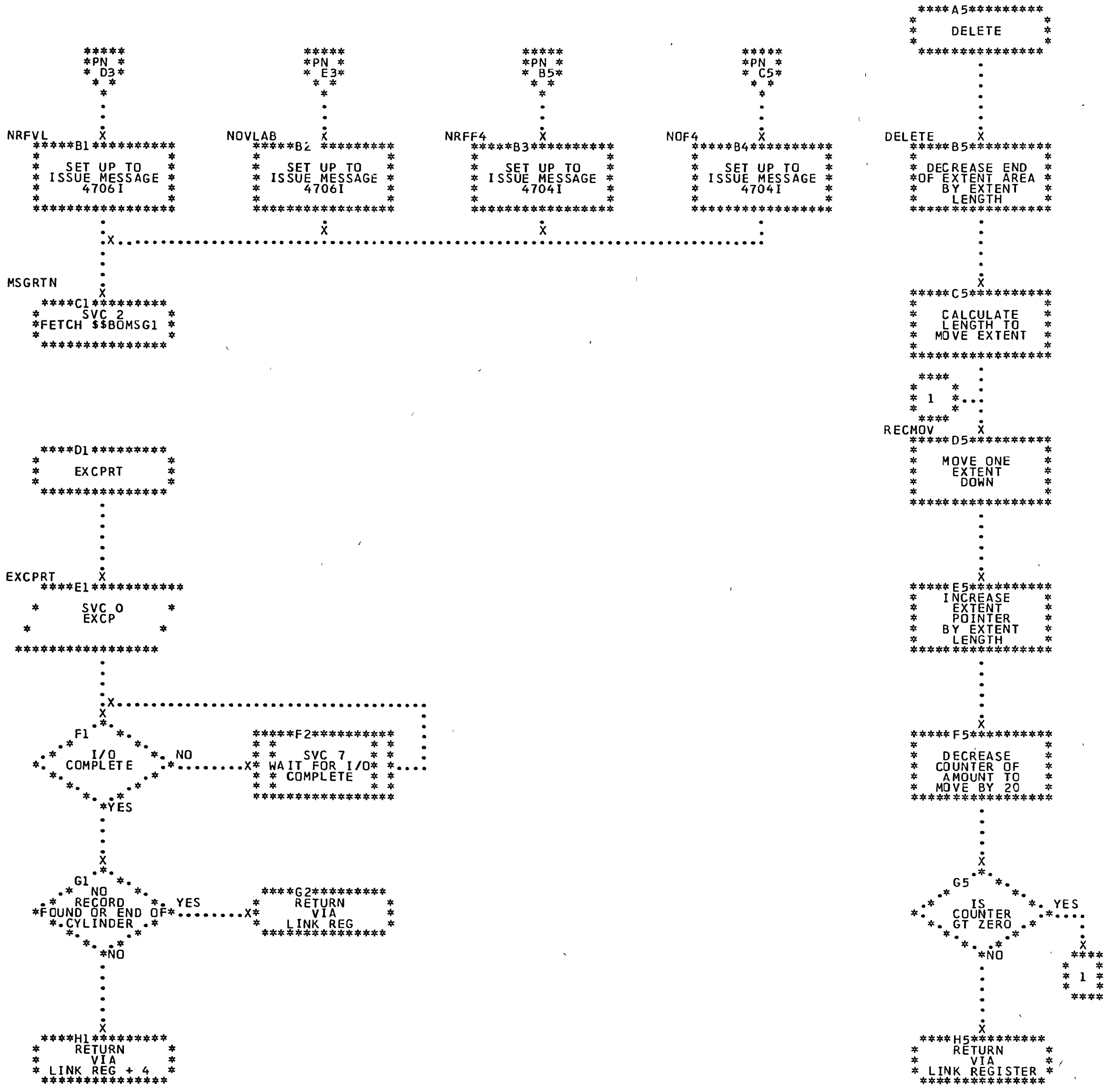

A5 DELETE
PN D3
PN E3
PN B5
PN C5
NRFVL B1 SET UP TO ISSUE MESSAGE 4706I
NOVLAB B2 SET UP TO ISSUE MESSAGE 4706I
NRFF4 B3 SET UP TO ISSUE MESSAGE 4704I
NOF4 B4 SET UP TO ISSUE MESSAGE 4704I
DELETE B5 DECREASE END OF EXTENT AREA BY EXTENT LENGTH
MSGRTN C1 SVC 2 FETCH $$BOMSG1
C5 CALCULATE LENGTH TO MOVE EXTENT
1
RECMOV D5 MOVE ONE EXTENT DOWN
D1 EXCPRT
EXCPRT E1 SVC 0 EXCP
E5 INCREASE EXTENT POINTER BY EXTENT LENGTH
F1 I/O COMPLETE
NO
YES
F2 SVC 7 WAIT FOR I/O COMPLETE
F5 DECREASE COUNTER OF AMOUNT TO MOVE BY 20
G1 NO RECORD FOUND OR END OF CYLINDER
YES
NO
G2 RETURN VIA LINK REG
G5 IS COUNTER GT ZERO
YES
NO
1
H1 RETURN VIA LINK REG + 4
H5 RETURN VIA LINK REGISTER

Chart QA. $$BODA02: DA Open Output, Phase 2 (Section 1 of 4)

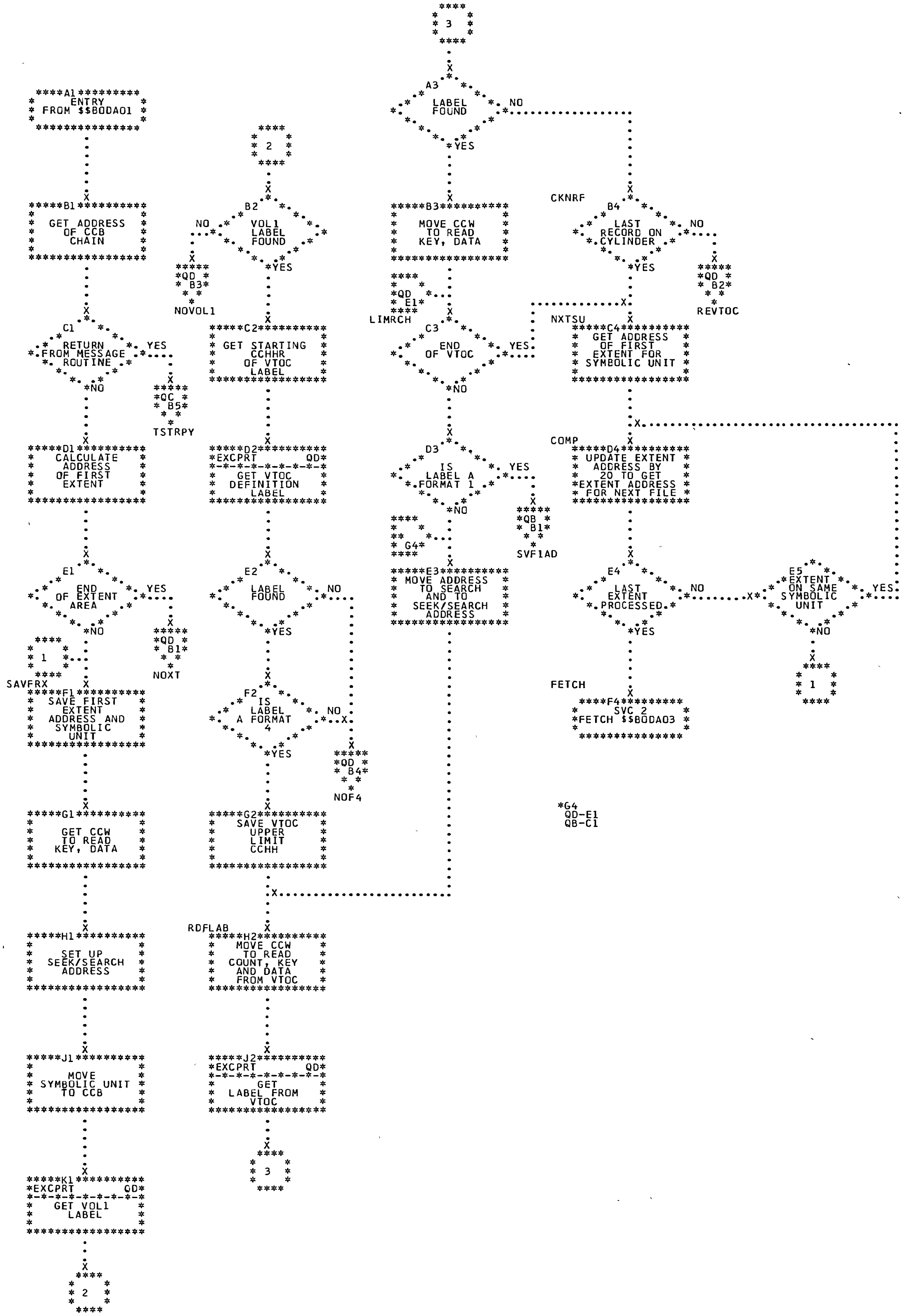
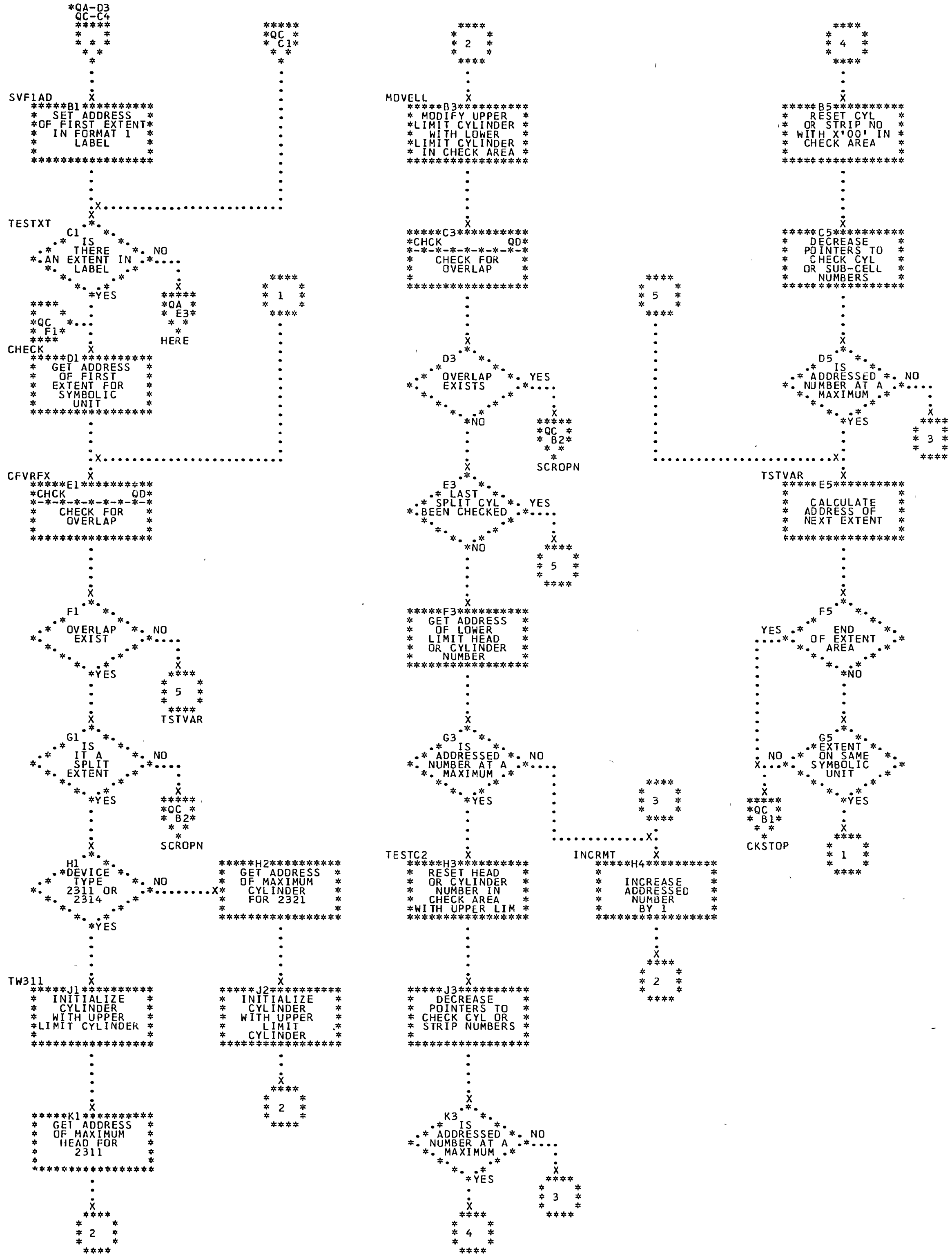
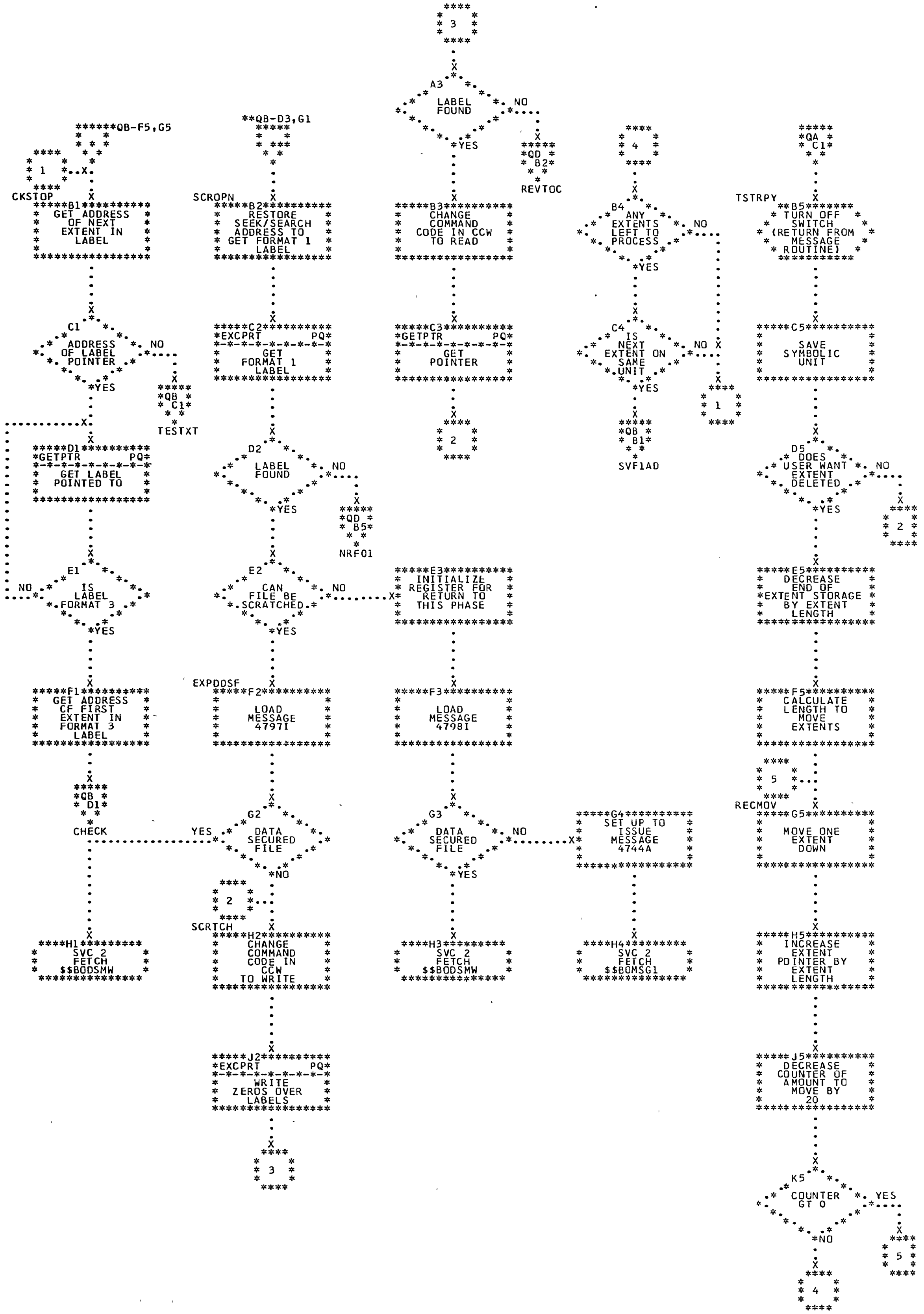

A1 ENTRY FROM $$BODA01
B1 GET ADDRESS OF CCB CHAIN
C1 RETURN FROM MESSAGE ROUTINE
YES
NO
QC B5
TSTRPY
D1 CALCULATE ADDRESS OF FIRST EXTENT
E1 END OF EXTENT AREA
YES
NO
QD B1
NOXT
1
SAVFRX
F1 SAVE FIRST EXTENT ADDRESS AND SYMBOLIC UNIT
G1 GET CCW TO READ KEY, DATA
H1 SET UP SEEK/SEARCH ADDRESS
J1 MOVE SYMBOLIC UNIT TO CCB
K1 EXCPRT QD GET VOL1 LABEL
2
2
B2 VOL1 LABEL FOUND
NO
QD B3
NOVOL1
YES
C2 GET STARTING CCHHR OF VTOC LABEL
D2 EXCPRT QD GET VTOC DEFINITION LABEL
E2 LABEL FOUND
NO
YES
F2 IS LABEL A FORMAT 4
NO
QD B4
NOF4
YES
G2 SAVE VTOC UPPER LIMIT CCHH
RDFLAB
H2 MOVE CCW TO READ COUNT, KEY AND DATA FROM VTOC
J2 EXCPRT QD GET LABEL FROM VTOC
3
3
A3 LABEL FOUND
NO
YES
B3 MOVE CCW TO READ KEY, DATA
QD E1
LIMRCH
C3 END OF VTOC
YES
NO
D3 IS LABEL A FORMAT 1
YES
QB B1
SVF1AD
NO
G4
E3 MOVE ADDRESS TO SEARCH AND TO SEEK/SEARCH ADDRESS
CKNRF
B4 LAST RECORD ON CYLINDER
NO
QD B2
REVTOC
YES
NXTSU
C4 GET ADDRESS OF FIRST EXTENT FOR SYMBOLIC UNIT
COMP
D4 UPDATE EXTENT ADDRESS BY 20 TO GET EXTENT ADDRESS FOR NEXT FILE
E4 LAST EXTENT PROCESSED
NO
YES
FETCH
F4 SVC 2 FETCH $$BODA03
E5 EXTENT ON SAME SYMBOLIC UNIT
YES
NO
1
*G4
QD-E1
QB-C1

Chart QB. $$BODA02: DA Open Output, Phase 2 (Section 2 of 4)

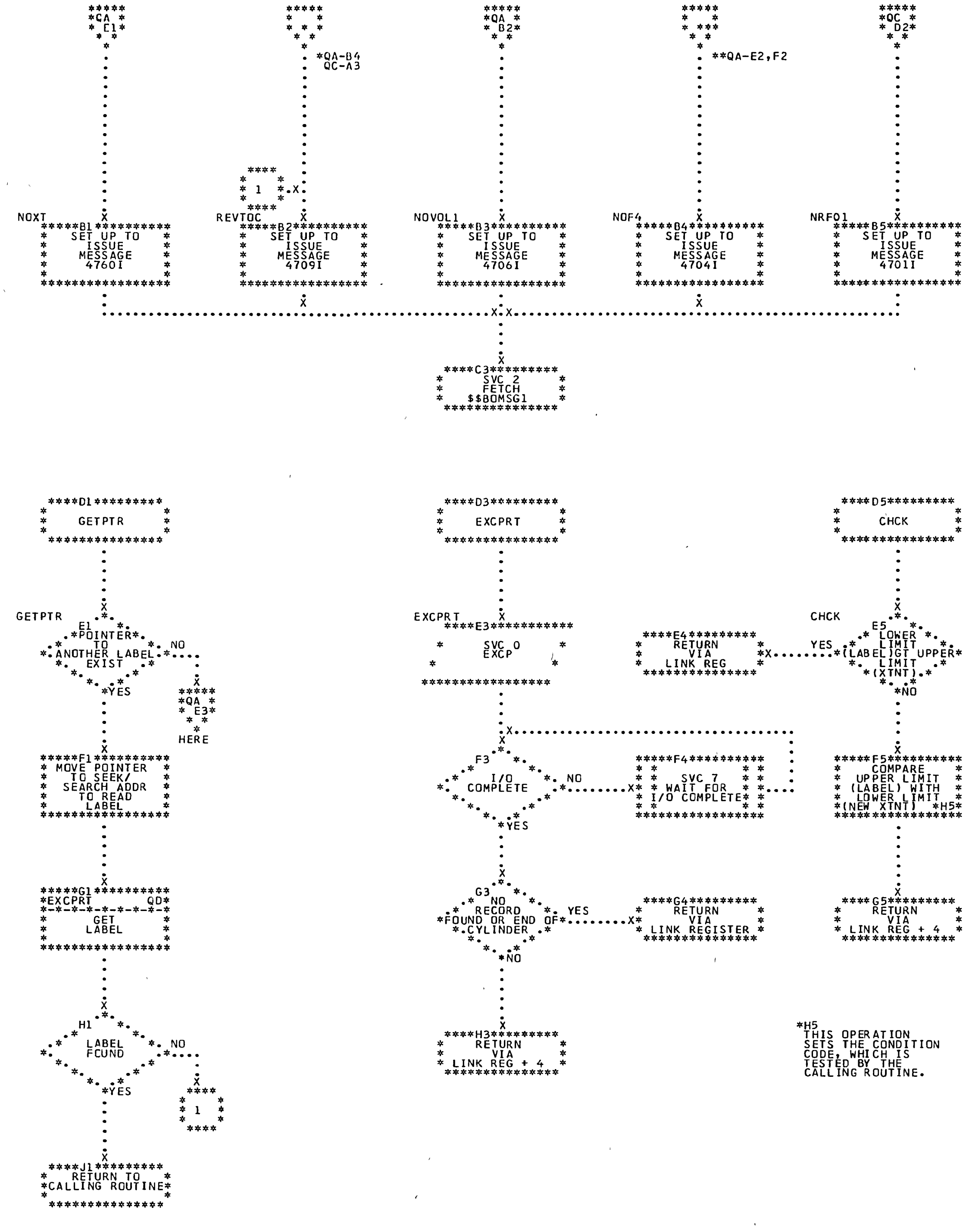

QA-D3
QC-C4
QC C1
2
4
SVF1AD
B1
SET ADDRESS OF FIRST EXTENT IN FORMAT 1 LABEL
MOVELL
D3
MODIFY UPPER LIMIT CYLINDER WITH LOWER LIMIT CYLINDER IN CHECK AREA
B5
RESET CYL OR STRIP NO WITH X'00' IN CHECK AREA
TESTXT
C1
IS THERE AN EXTENT IN LABEL
NO
YES
QA E3
HERE
1
C3
CHCK QD
CHECK FOR OVERLAP
C5
DECREASE POINTERS TO CHECK CYL OR SUB-CELL NUMBERS
5
QC F1
CHECK
D1
GET ADDRESS OF FIRST EXTENT FOR SYMBOLIC UNIT
D3
OVERLAP EXISTS
YES
NO
QC B2
SCROPN
D5
IS ADDRESSED NUMBER AT A MAXIMUM
NO
YES
3
CFVRFX
E1
CHCK QD
CHECK FOR OVERLAP
E3
LAST SPLIT CYL BEEN CHECKED
YES
NO
5
TSTVAR
E5
CALCULATE ADDRESS OF NEXT EXTENT
F1
OVERLAP EXIST
NO
YES
5
TSTVAR
F3
GET ADDRESS OF LOWER LIMIT HEAD OR CYLINDER NUMBER
F5
END OF EXTENT AREA
YES
NO
G1
IS IT A SPLIT EXTENT
NO
YES
QC B2
SCROPN
G3
IS ADDRESSED NUMBER AT A MAXIMUM
NO
YES
3
G5
EXTENT ON SAME SYMBOLIC UNIT
NO
YES
QC B1
CKSTOP
1
H1
DEVICE TYPE 2311 OR 2314
NO
YES
H2
GET ADDRESS OF MAXIMUM CYLINDER FOR 2321
TESTC2
H3
RESET HEAD OR CYLINDER NUMBER IN CHECK AREA WITH UPPER LIM
INCRMT
H4
INCREASE ADDRESSED NUMBER BY 1
2
TW311
J1
INITIALIZE CYLINDER WITH UPPER LIMIT CYLINDER
J2
INITIALIZE CYLINDER WITH UPPER LIMIT CYLINDER
J3
DECREASE POINTERS TO CHECK CYL OR STRIP NUMBERS
2
K1
GET ADDRESS OF MAXIMUM HEAD FOR 2311
K3
IS ADDRESSED NUMBER AT A MAXIMUM
NO
YES
3
2
4

## Chart QC. $$BODAO2: DA Open Output, Phase 2 (Section 3 of 4)



QB-F5,G5
1
CKSTOP
B1 GET ADDRESS OF NEXT EXTENT IN LABEL
C1 ADDRESS OF LABEL POINTER
NO
QB C1
TESTXT
YES
D1 GETPTR PQ GET LABEL POINTED TO
E1 IS LABEL FORMAT 3
NO
YES
F1 GET ADDRESS OF FIRST EXTENT IN FORMAT 3 LABEL
QB D1
CHECK
H1 SVC 2 FETCH $$BODSMW
QB-D3,G1
SCROPN
B2 RESTORE SEEK/SEARCH ADDRESS TO GET FORMAT 1 LABEL
C2 EXCPRT PQ GET FORMAT 1 LABEL
D2 LABEL FOUND
NO
QD B5
NRF01
YES
E2 CAN FILE BE SCRATCHED
NO
YES
EXPDOSF
F2 LOAD MESSAGE 4797I
G2 DATA SECURED FILE
YES
NO
2
SCRTCH
H2 CHANGE COMMAND CODE IN CCW TO WRITE
J2 EXCPRT PQ WRITE ZEROS OVER LABELS
3
3
A3 LABEL FOUND
NO
QD B2
REVTOC
YES
B3 CHANGE COMMAND CODE IN CCW TO READ
C3 GETPTR PQ GET POINTER
2
E3 INITIALIZE REGISTER FOR RETURN TO THIS PHASE
F3 LOAD MESSAGE 4798I
G3 DATA SECURED FILE
NO
YES
H3 SVC 2 FETCH $$BODSMW
4
B4 ANY EXTENTS LEFT TO PROCESS
NO
YES
C4 IS NEXT EXTENT ON SAME UNIT
NO
1
YES
QB B1
SVF1AD
G4 SET UP TO ISSUE MESSAGE 4744A
H4 SVC 2 FETCH $$BOMSG1
QA C1
TSTRPY
B5 TURN OFF SWITCH (RETURN FROM MESSAGE ROUTINE)
C5 SAVE SYMBOLIC UNIT
D5 DOES USER WANT EXTENT DELETED
NO
2
YES
E5 DECREASE END OF EXTENT STORAGE BY EXTENT LENGTH
F5 CALCULATE LENGTH TO MOVE EXTENTS
5
RECMOV
G5 MOVE ONE EXTENT DOWN
H5 INCREASE EXTENT POINTER BY EXTENT LENGTH
J5 DECREASE COUNTER OF AMOUNT TO MOVE BY 20
K5 COUNTER GT 0
YES
5
NO
4

Chart QD. $$BODA02: DA Open Output, Phase 2 (Section 4 of 4)



CA C1
QA B4
QC-A3
QA D2
QA-E2,F2
QC D2
1
NOXT
B1 SET UP TO ISSUE MESSAGE 4760I
REVTOC
B2 SET UP TO ISSUE MESSAGE 4709I
NOVOL1
B3 SET UP TO ISSUE MESSAGE 4706I
NOF4
B4 SET UP TO ISSUE MESSAGE 4704I
NRFO1
B5 SET UP TO ISSUE MESSAGE 4701I
C3 SVC 2 FETCH $$BOMSG1
D1 GETPTR
D3 EXCPRT
D5 CHCK
GETPTR
E1 POINTER TO ANOTHER LABEL EXIST
NO
YES
QA E3 HERE
F1 MOVE POINTER TO SEEK/ SEARCH ADDR TO READ LABEL
G1 EXCPRT QD GET LABEL
H1 LABEL FOUND
NO
YES
1
J1 RETURN TO CALLING ROUTINE
EXCPRT
E3 SVC 0 EXCP
F3 I/O COMPLETE
NO
YES
F4 SVC 7 WAIT FOR I/O COMPLETE
G3 NO RECORD FOUND OR END OF CYLINDER
YES
NO
G4 RETURN VIA LINK REGISTER
H3 RETURN VIA LINK REG + 4
E4 RETURN VIA LINK REG
CHCK
E5 LOWER LIMIT (LABEL)GT UPPER LIMIT (XTNT)
YES
NO
F5 COMPARE UPPER LIMIT (LABEL) WITH LOWER LIMIT (NEW XTNT) H5
G5 RETURN VIA LINK REG + 4
*H5 THIS OPERATION SETS THE CONDITION CODE, WHICH IS TESTED BY THE CALLING ROUTINE.

Chart QE. $$BODA03: DA Open Output, Phase 3 (Section 1 of 4)

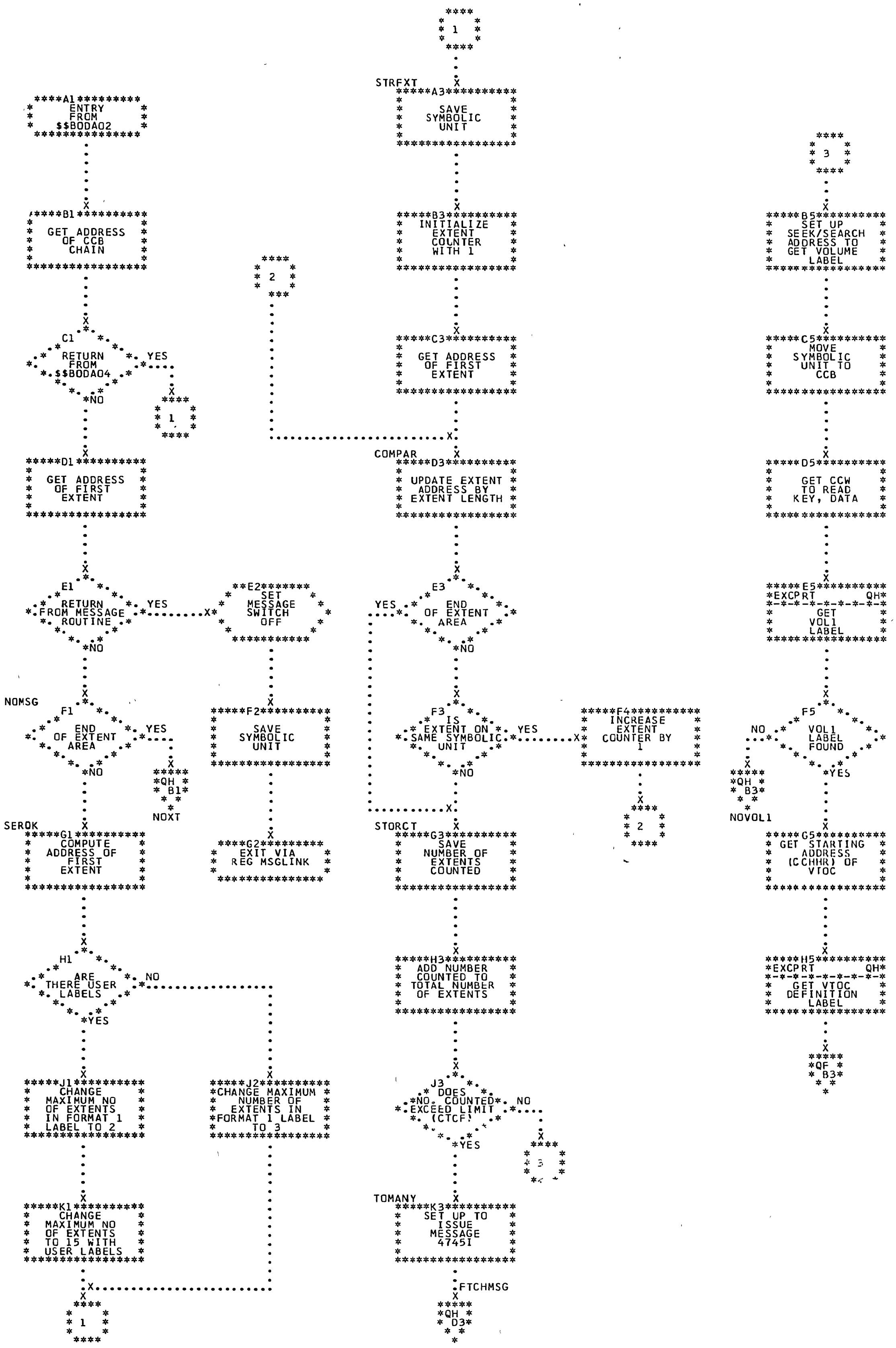
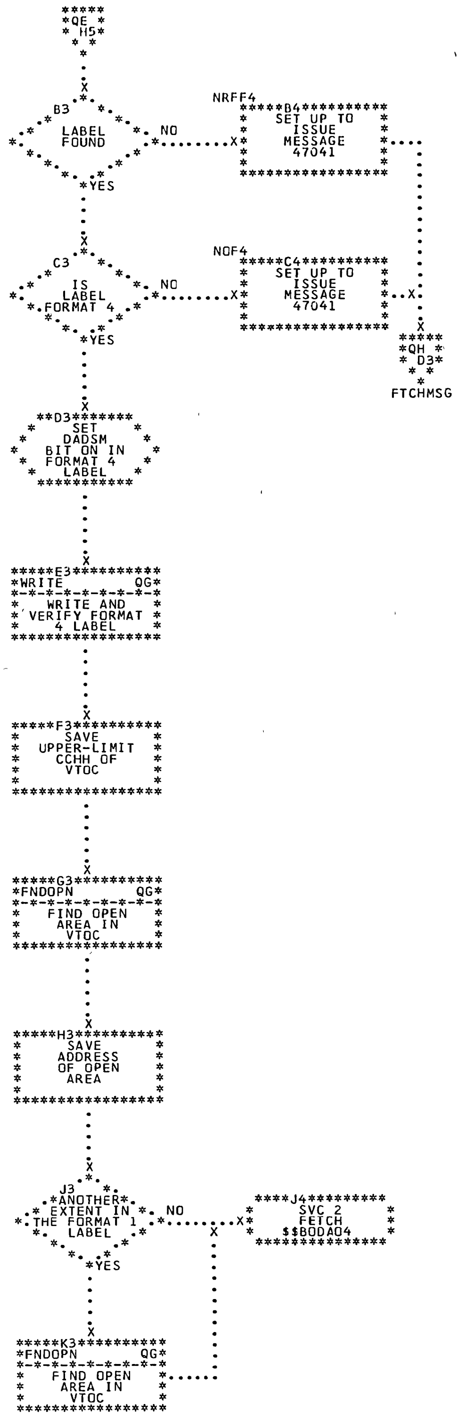

A1
ENTRY FROM $$BODA02
B1
GET ADDRESS OF CCB CHAIN
C1
RETURN FROM $$BODA04
YES
1
NO
D1
GET ADDRESS OF FIRST EXTENT
E1
RETURN FROM MESSAGE ROUTINE
YES
NO
E2
SET MESSAGE SWITCH OFF
NOMSG
F1
END OF EXTENT AREA
YES
QH B1
NOXT
NO
F2
SAVE SYMBOLIC UNIT
SEROK
G1
COMPUTE ADDRESS OF FIRST EXTENT
G2
EXIT VIA REG MSGLINK
H1
ARE THERE USER LABELS
NO
YES
J1
CHANGE MAXIMUM NO OF EXTENTS IN FORMAT 1 LABEL TO 2
J2
CHANGE MAXIMUM NUMBER OF EXTENTS IN FORMAT 1 LABEL TO 3
K1
CHANGE MAXIMUM NO OF EXTENTS TO 15 WITH USER LABELS
1
2
1
STREXT
A3
SAVE SYMBOLIC UNIT
B3
INITIALIZE EXTENT COUNTER WITH 1
C3
GET ADDRESS OF FIRST EXTENT
COMPAR
D3
UPDATE EXTENT ADDRESS BY EXTENT LENGTH
E3
END OF EXTENT AREA
YES
NO
F3
IS EXTENT ON SAME SYMBOLIC UNIT
YES
NO
F4
INCREASE EXTENT COUNTER BY 1
2
STORCT
G3
SAVE NUMBER OF EXTENTS COUNTED
H3
ADD NUMBER COUNTED TO TOTAL NUMBER OF EXTENTS
J3
DOES NO. COUNTED EXCEED LIMIT (CTCF)
NO
3
YES
TOMANY
K3
SET UP TO ISSUE MESSAGE 4745I
FTCHMSG
QH D3
3
B5
SET UP SEEK/SEARCH ADDRESS TO GET VOLUME LABEL
C5
MOVE SYMBOLIC UNIT TO CCB
D5
GET CCW TO READ KEY, DATA
E5
EXCPRT QH
GET VOL1 LABEL
F5
VOL1 LABEL FOUND
NO
YES
QH B3
NOVOL1
G5
GET STARTING ADDRESS (CCHHR) OF VTOC
H5
EXCPRT QH
GET VTOC DEFINITION LABEL
QF B3

Chart QF. $$BODAO3: DA Open Output, Phase 3 (Section 2 of 4)


QE H5
B3 LABEL FOUND
NO
YES
NRFF4
B4 SET UP TO ISSUE MESSAGE 47041
C3 IS LABEL FORMAT 4
NO
YES
NOF4
C4 SET UP TO ISSUE MESSAGE 47041
QH D3
FTCHMSG
D3 SET DADSM BIT ON IN FORMAT 4 LABEL
E3 WRITE QG WRITE AND VERIFY FORMAT 4 LABEL
F3 SAVE UPPER-LIMIT CCHH OF VTOC
G3 FNDOPN QG FIND OPEN AREA IN VTOC
H3 SAVE ADDRESS OF OPEN AREA
J3 ANOTHER EXTENT IN THE FORMAT 1 LABEL
NO
YES
J4 SVC 2 FETCH $$BODAO4
K3 FNDOPN QG FIND OPEN AREA IN VTOC

## Chart QG. $$BODA03: DA Open Output, Phase 3 (Section 3 of 4)

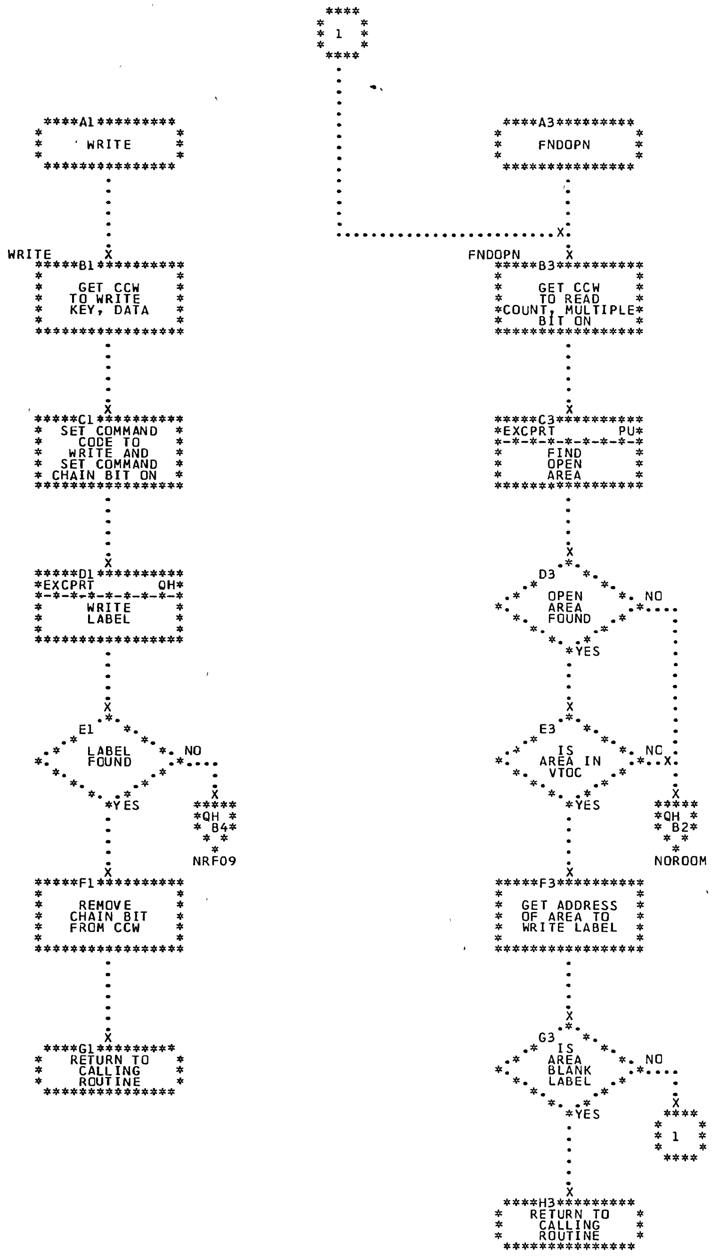

1
A1
WRITE
A3
FNDOPN
WRITE
B1
GET CCW TO WRITE KEY, DATA
FNDOPN
B3
GET CCW TO READ COUNT, MULTIPLE BIT ON
C1
SET COMMAND CODE TO WRITE AND SET COMMAND CHAIN BIT ON
C3
EXCPRT PU
FIND OPEN AREA
D1
EXCPRT QH
WRITE LABEL
D3
OPEN AREA FOUND
NO
YES
E1
LABEL FOUND
NO
YES
QH B4
NRF09
E3
IS AREA IN VTOC
NO
YES
QH B2
NOROOM
F1
REMOVE CHAIN BIT FROM CCW
F3
GET ADDRESS OF AREA TO WRITE LABEL
G1
RETURN TO CALLING ROUTINE
G3
IS AREA BLANK LABEL
NO
YES
1
H3
RETURN TO CALLING ROUTINE

Chart QH. $$BODA03: DA Open Output, Phase 3 (Section 4 of 4)

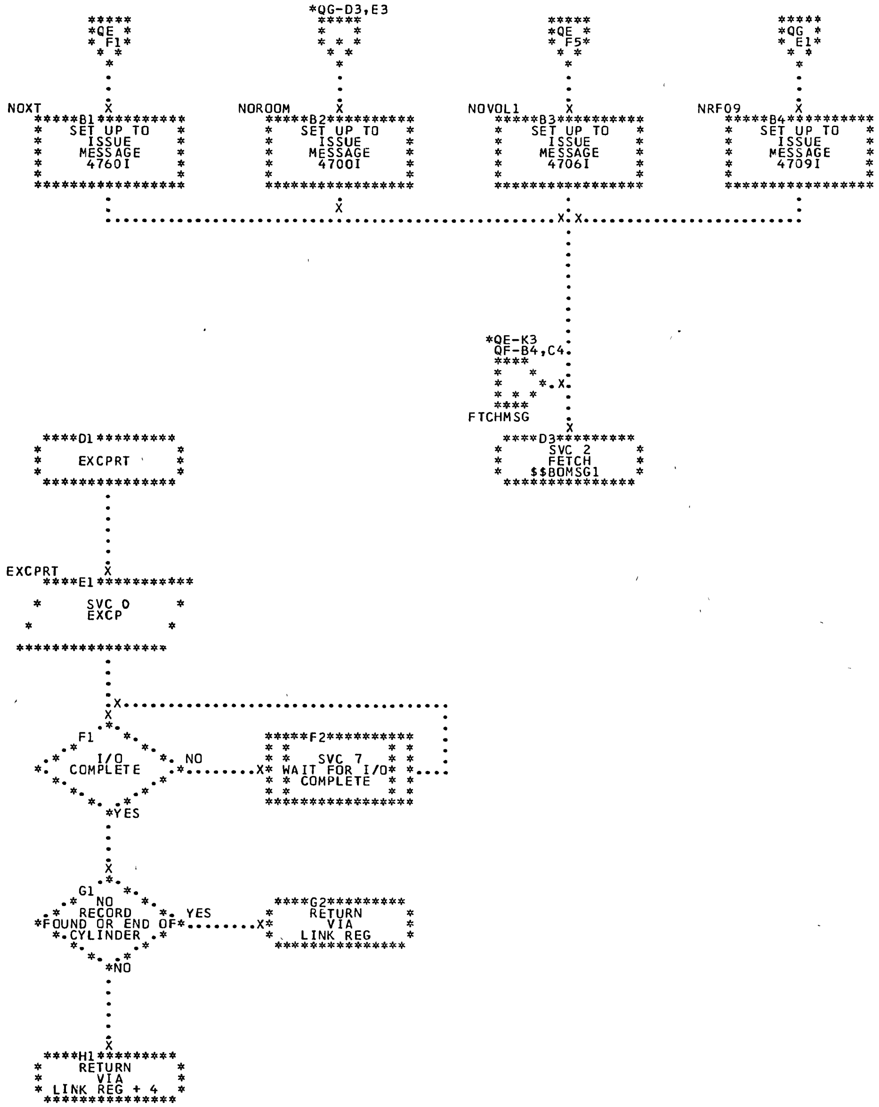


Chart QJ.  $$BODA04:  DA Open Output, Phase 4 (Section 1 of 4)

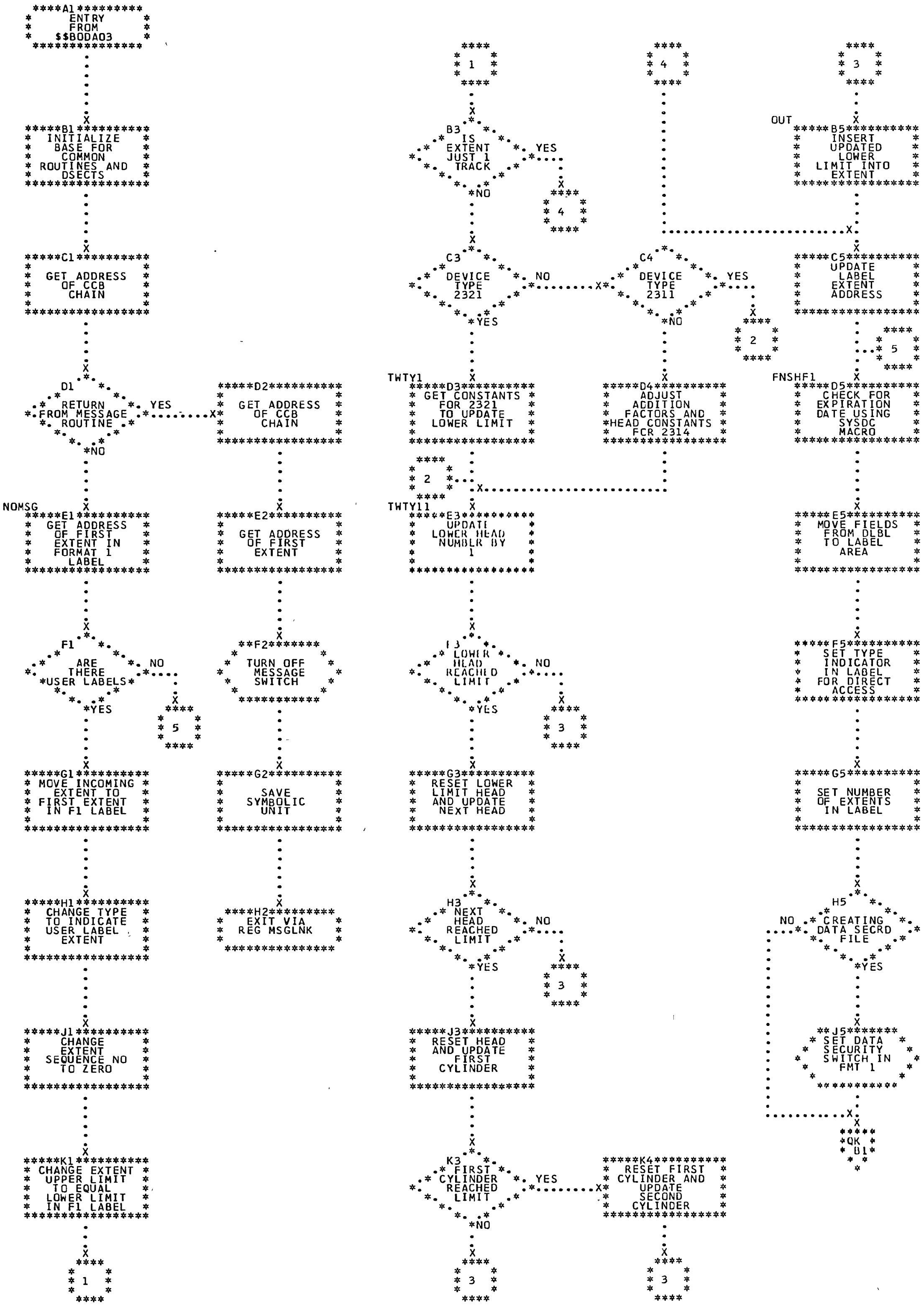
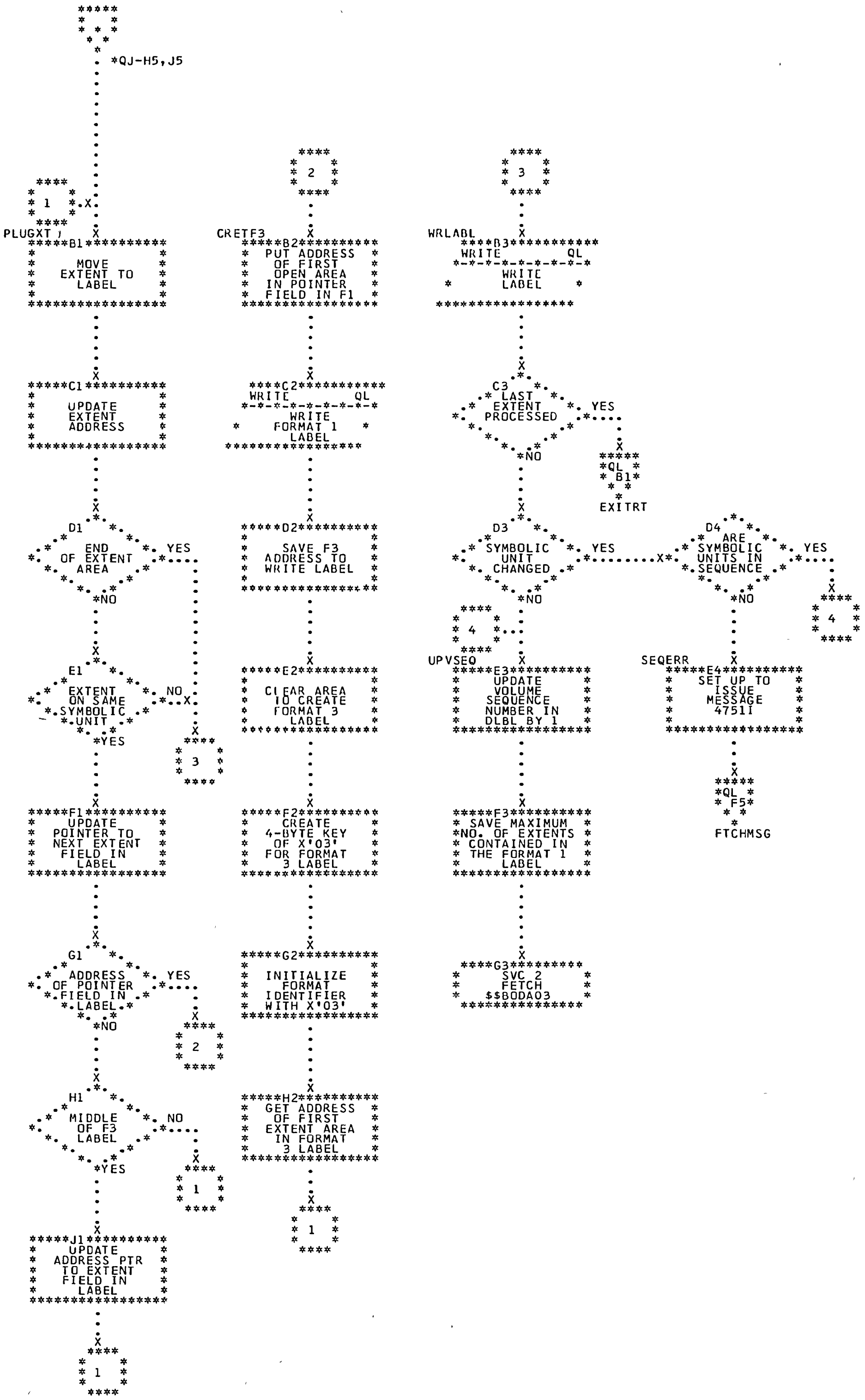

A1 ENTRY FROM $$BODA03
B1 INITIALIZE BASE FOR COMMON ROUTINES AND DSECTS
C1 GET ADDRESS OF CCB CHAIN
D1 RETURN FROM MESSAGE ROUTINE
YES
NO
D2 GET ADDRESS OF CCB CHAIN
NOMSG
E1 GET ADDRESS OF FIRST EXTENT IN FORMAT 1 LABEL
E2 GET ADDRESS OF FIRST EXTENT
F1 ARE THERE USER LABELS
NO
5
YES
F2 TURN OFF MESSAGE SWITCH
G1 MOVE INCOMING EXTENT TO FIRST EXTENT IN F1 LABEL
G2 SAVE SYMBOLIC UNIT
H1 CHANGE TYPE TO INDICATE USER LABEL EXTENT
H2 EXIT VIA REG MSGLNK
J1 CHANGE EXTENT SEQUENCE NO TO ZERO
K1 CHANGE EXTENT UPPER LIMIT TO EQUAL LOWER LIMIT IN F1 LABEL
1
1
B3 IS EXTENT JUST 1 TRACK
YES
4
NO
C3 DEVICE TYPE 2321
NO
YES
TWTY1
D3 GET CONSTANTS FOR 2321 TO UPDATE LOWER LIMIT
2
TWTY11
E3 UPDATE LOWER HEAD NUMBER BY 1
F3 LOWER HEAD REACHED LIMIT
NO
3
YES
G3 RESET LOWER LIMIT HEAD AND UPDATE NEXT HEAD
H3 NEXT HEAD REACHED LIMIT
NO
3
YES
J3 RESET HEAD AND UPDATE FIRST CYLINDER
K3 FIRST CYLINDER REACHED LIMIT
YES
NO
3
4
C4 DEVICE TYPE 2311
YES
2
NO
D4 ADJUST ADDITION FACTORS AND HEAD CONSTANTS FCR 2314
K4 RESET FIRST CYLINDER AND UPDATE SECOND CYLINDER
3
3
OUT
B5 INSERT UPDATED LOWER LIMIT INTO EXTENT
C5 UPDATE LABEL EXTENT ADDRESS
5
FNSHF1
D5 CHECK FOR EXPIRATION DATE USING SYSDC MACRO
E5 MOVE FIELDS FROM DLBL TO LABEL AREA
F5 SET TYPE INDICATOR IN LABEL FOR DIRECT ACCESS
G5 SET NUMBER OF EXTENTS IN LABEL
H5 CREATING DATA SECRD FILE
NO
YES
J5 SET DATA SECURITY SWITCH IN FMT 1
QK B1

Chart QK. $$BODA04: DA Open Output, Phase 4 (Section 2 of 4)


*QJ-H5,J5
1
PLUGXT
B1
MOVE EXTENT TO LABEL
C1
UPDATE EXTENT ADDRESS
D1
END OF EXTENT AREA
YES
NO
E1
EXTENT ON SAME SYMBOLIC UNIT
NO
YES
3
F1
UPDATE POINTER TO NEXT EXTENT FIELD IN LABEL
G1
ADDRESS OF POINTER FIELD IN LABEL
YES
NO
2
H1
MIDDLE OF F3 LABEL
NO
YES
1
J1
UPDATE ADDRESS PTR TO EXTENT FIELD IN LABEL
1
2
CRETF3
B2
PUT ADDRESS OF FIRST OPEN AREA IN POINTER FIELD IN F1
C2
WRITE QL
WRITE FORMAT 1 LABEL
D2
SAVE F3 ADDRESS TO WRITE LABEL
E2
CLEAR AREA TO CREATE FORMAT 3 LABEL
F2
CREATE 4-BYTE KEY OF X'03' FOR FORMAT 3 LABEL
G2
INITIALIZE FORMAT IDENTIFIER WITH X'03'
H2
GET ADDRESS OF FIRST EXTENT AREA IN FORMAT 3 LABEL
1
3
WRLABL
B3
WRITE QL
WRITE LABEL
C3
LAST EXTENT PROCESSED
YES
QL B1
EXITRT
NO
D3
SYMBOLIC UNIT CHANGED
YES
NO
4
UPVSEQ
E3
UPDATE VOLUME SEQUENCE NUMBER IN DLBL BY 1
F3
SAVE MAXIMUM NO. OF EXTENTS CONTAINED IN THE FORMAT 1 LABEL
G3
SVC 2 FETCH $$BODA03
D4
ARE SYMBOLIC UNITS IN SEQUENCE
YES
4
NO
SEQERR
E4
SET UP TO ISSUE MESSAGE 4751I
QL F5
FTCHMSG

## Chart QL. $$BODAO4: DA Open Output, Phase 4 (Section 3 of 4)

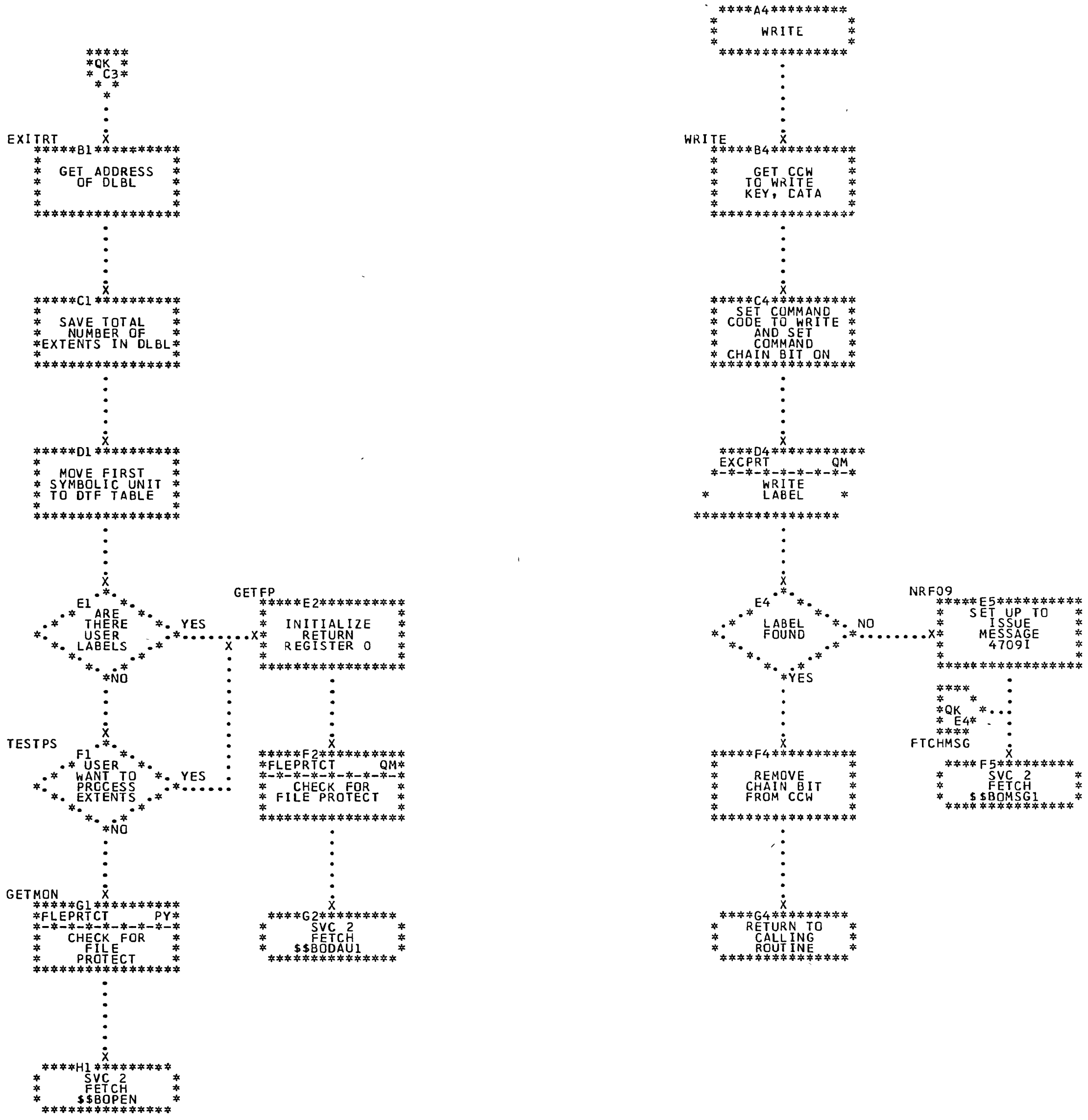


Chart QM. $$BODA04: DA Open Output, Phase 4 (Section 4 of 4)

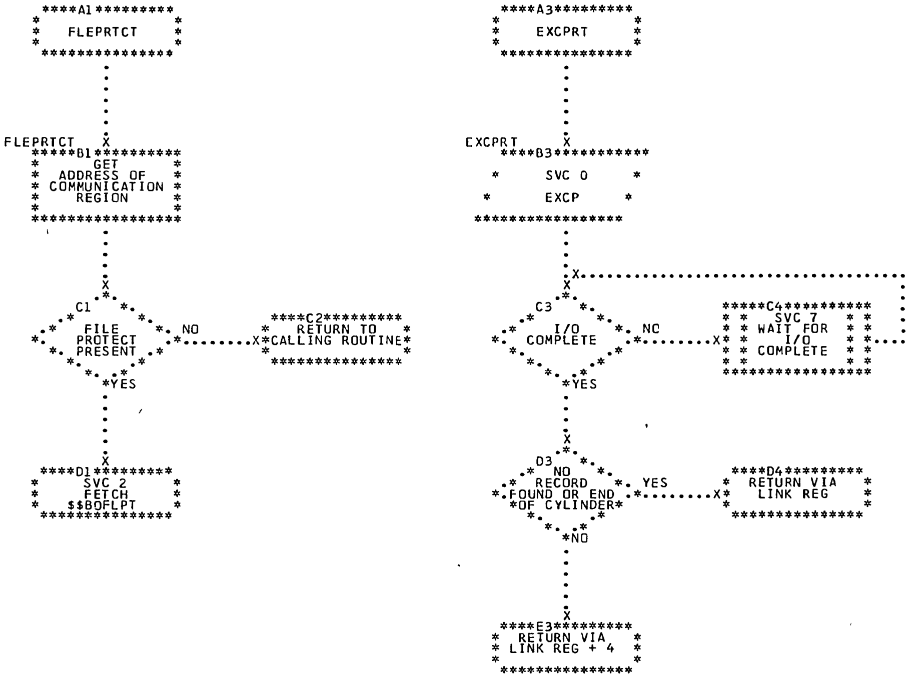


Chart RA. $$BODAU1: DA Open Input, Output (Section 1 of 5)

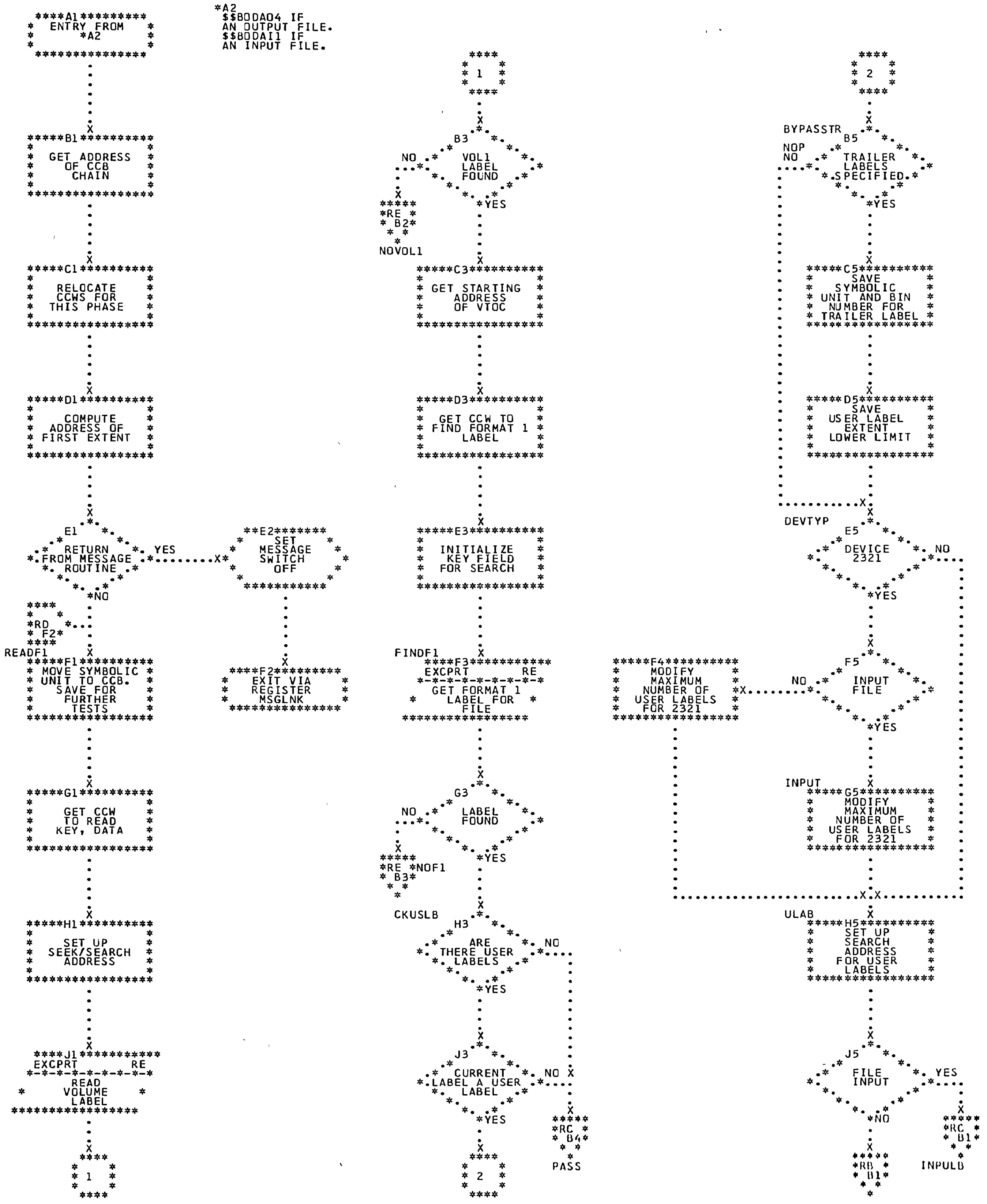
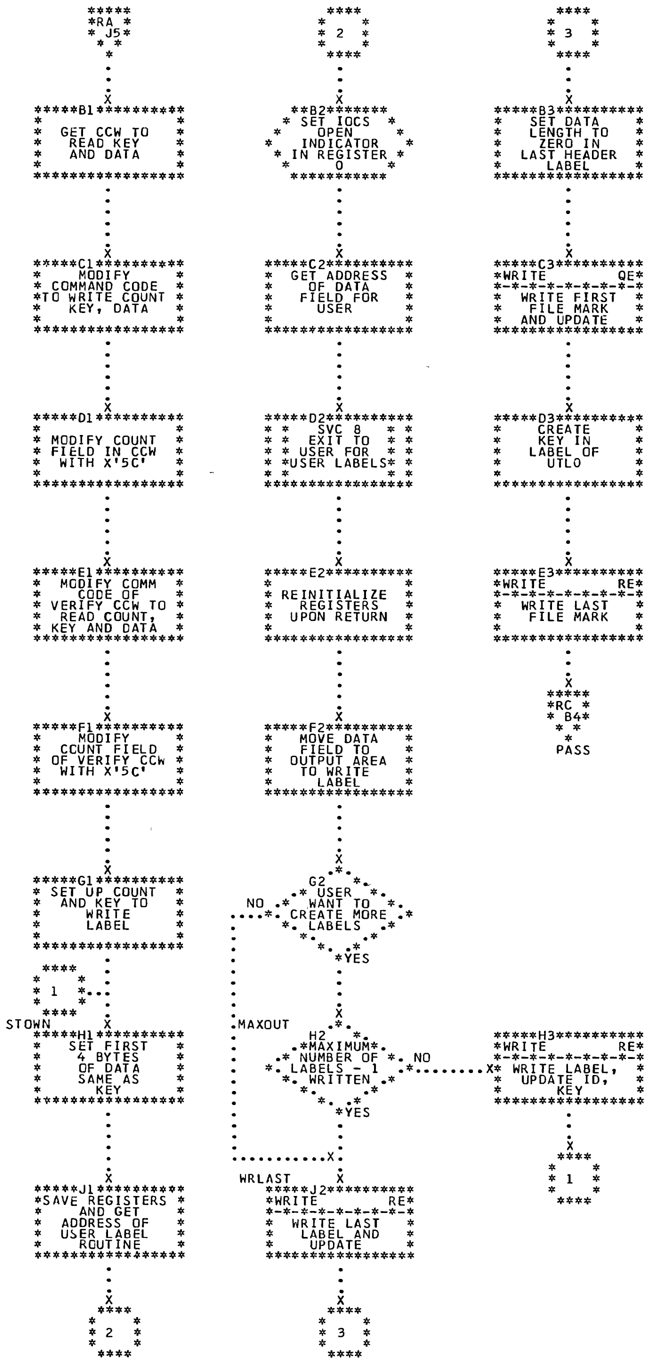

A1 ENTRY FROM *A2
*A2 $$BODA04 IF AN OUTPUT FILE. $$BODAI1 IF AN INPUT FILE.
B1 GET ADDRESS OF CCB CHAIN
C1 RELOCATE CCWS FOR THIS PHASE
D1 COMPUTE ADDRESS OF FIRST EXTENT
E1 RETURN FROM MESSAGE ROUTINE
YES
NO
E2 SET MESSAGE SWITCH OFF
F2 EXIT VIA REGISTER MSGLNK
RD F2
READF1
F1 MOVE SYMBOLIC UNIT TO CCB. SAVE FOR FURTHER TESTS
G1 GET CCW TO READ KEY, DATA
H1 SET UP SEEK/SEARCH ADDRESS
J1 EXCPRT RE READ VOLUME LABEL
1
B3 VOL1 LABEL FOUND
NO
YES
RE B2
NOVOL1
C3 GET STARTING ADDRESS OF VTOC
D3 GET CCW TO FIND FORMAT 1 LABEL
E3 INITIALIZE KEY FIELD FOR SEARCH
FINDF1
F3 EXCPRT RE GET FORMAT 1 LABEL FOR FILE
G3 LABEL FOUND
NO
YES
RE B3 NOF1
CKUSLB
H3 ARE THERE USER LABELS
NO
YES
J3 CURRENT LABEL A USER LABEL
NO
YES
RC B4
PASS
2
2
BYPASSTR
B5 TRAILER LABELS SPECIFIED
NOP NO
YES
C5 SAVE SYMBOLIC UNIT AND BIN NUMBER FOR TRAILER LABEL
D5 SAVE USER LABEL EXTENT LOWER LIMIT
DEVTYP
E5 DEVICE 2321
NO
YES
F5 INPUT FILE
NO
YES
F4 MODIFY MAXIMUM NUMBER OF USER LABELS FOR 2321
INPUT
G5 MODIFY MAXIMUM NUMBER OF USER LABELS FOR 2321
ULAB
H5 SET UP SEARCH ADDRESS FOR USER LABELS
J5 FILE INPUT
YES
NO
RB B1
RC B1
INPULB

Chart RB. $$BODAU1: DA Open Input, Output (Section 2 of 5)


RA J5
2
3
B1 GET CCW TO READ KEY AND DATA
B2 SET IOCS OPEN INDICATOR IN REGISTER 0
B3 SET DATA LENGTH TO ZERO IN LAST HEADER LABEL
C1 MODIFY COMMAND CODE TO WRITE COUNT KEY, DATA
C2 GET ADDRESS OF DATA FIELD FOR USER
C3 WRITE QE
WRITE FIRST FILE MARK AND UPDATE
D1 MODIFY COUNT FIELD IN CCW WITH X'5C'
D2 SVC 8 EXIT TO USER FOR USER LABELS
D3 CREATE KEY IN LABEL OF UTLO
E1 MODIFY COMM CODE OF VERIFY CCW TO READ COUNT, KEY AND DATA
E2 REINITIALIZE REGISTERS UPON RETURN
E3 WRITE RE
WRITE LAST FILE MARK
RC B4
PASS
F1 MODIFY CCUNT FIELD OF VERIFY CCW WITH X'5C'
F2 MOVE DATA FIELD TO OUTPUT AREA TO WRITE LABEL
G1 SET UP COUNT AND KEY TO WRITE LABEL
G2 USER WANT TO CREATE MORE LABELS
NO
YES
1
STOWN
H1 SET FIRST 4 BYTES OF DATA SAME AS KEY
MAXOUT
H2 MAXIMUM NUMBER OF LABELS - 1 WRITTEN
NO
YES
H3 WRITE RE
WRITE LABEL, UPDATE ID, KEY
1
J1 SAVE REGISTERS AND GET ADDRESS OF USER LABEL ROUTINE
WRLAST
J2 WRITE RE
WRITE LAST LABEL AND UPDATE
2
3

## Chart RC. $$BODAU1: DA Open Input, Output (Section 3 of 5)

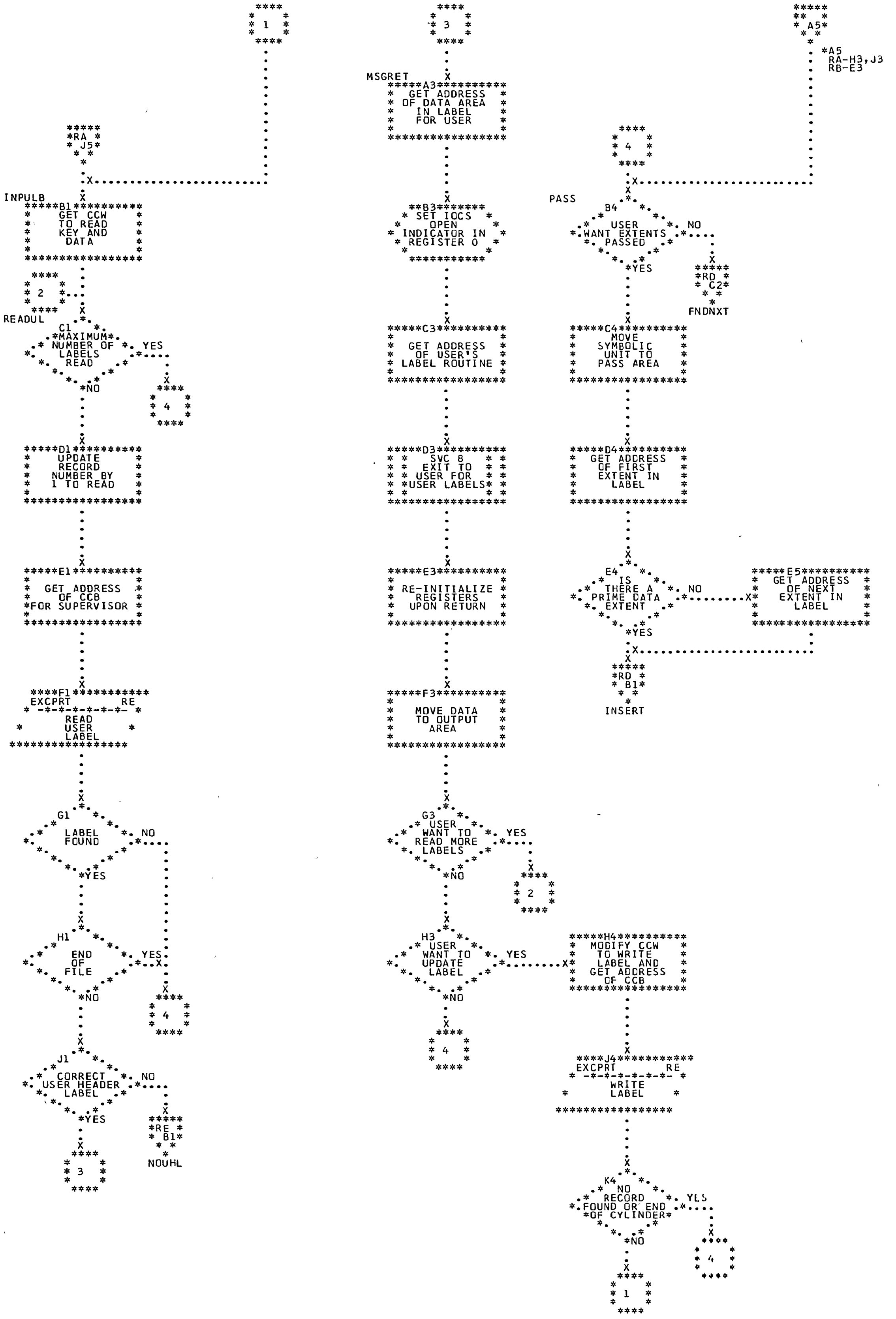

1
RA J5
INPULB
B1
GET CCW TO READ KEY AND DATA
2
READUL
C1
MAXIMUM NUMBER OF LABELS READ
YES
NO
4
D1
UPDATE RECORD NUMBER BY 1 TO READ
E1
GET ADDRESS OF CCB FOR SUPERVISOR
F1
EXCPRT RE
READ USER LABEL
G1
LABEL FOUND
NO
YES
H1
END OF FILE
YES
4
NO
J1
CORRECT USER HEADER LABEL
NO
RE B1
NOUHL
YES
3
3
MSGRET
A3
GET ADDRESS OF DATA AREA IN LABEL FOR USER
B3
SET IOCS OPEN INDICATOR IN REGISTER 0
C3
GET ADDRESS OF USER'S LABEL ROUTINE
D3
SVC 8 EXIT TO USER FOR USER LABELS
E3
RE-INITIALIZE REGISTERS UPON RETURN
F3
MOVE DATA TO OUTPUT AREA
G3
USER WANT TO READ MORE LABELS
YES
2
NO
H3
USER WANT TO UPDATE LABEL
YES
NO
4
A5
A5
RA-H3,J3
RB-E3
4
PASS
B4
USER WANT EXTENTS PASSED
NO
RD C2
FNDNXT
YES
C4
MOVE SYMBOLIC UNIT TO PASS AREA
D4
GET ADDRESS OF FIRST EXTENT IN LABEL
E4
IS THERE A PRIME DATA EXTENT
NO
E5
GET ADDRESS OF NEXT EXTENT IN LABEL
YES
RD B1
INSERT
H4
MODIFY CCW TO WRITE LABEL AND GET ADDRESS OF CCB
J4
EXCPRT RE
WRITE LABEL
K4
NO RECORD FOUND OR END OF CYLINDER
YLS
4
NO
1

Chart RD. $$BODAU1: DA Open Input, Output (Section 4 of 5)

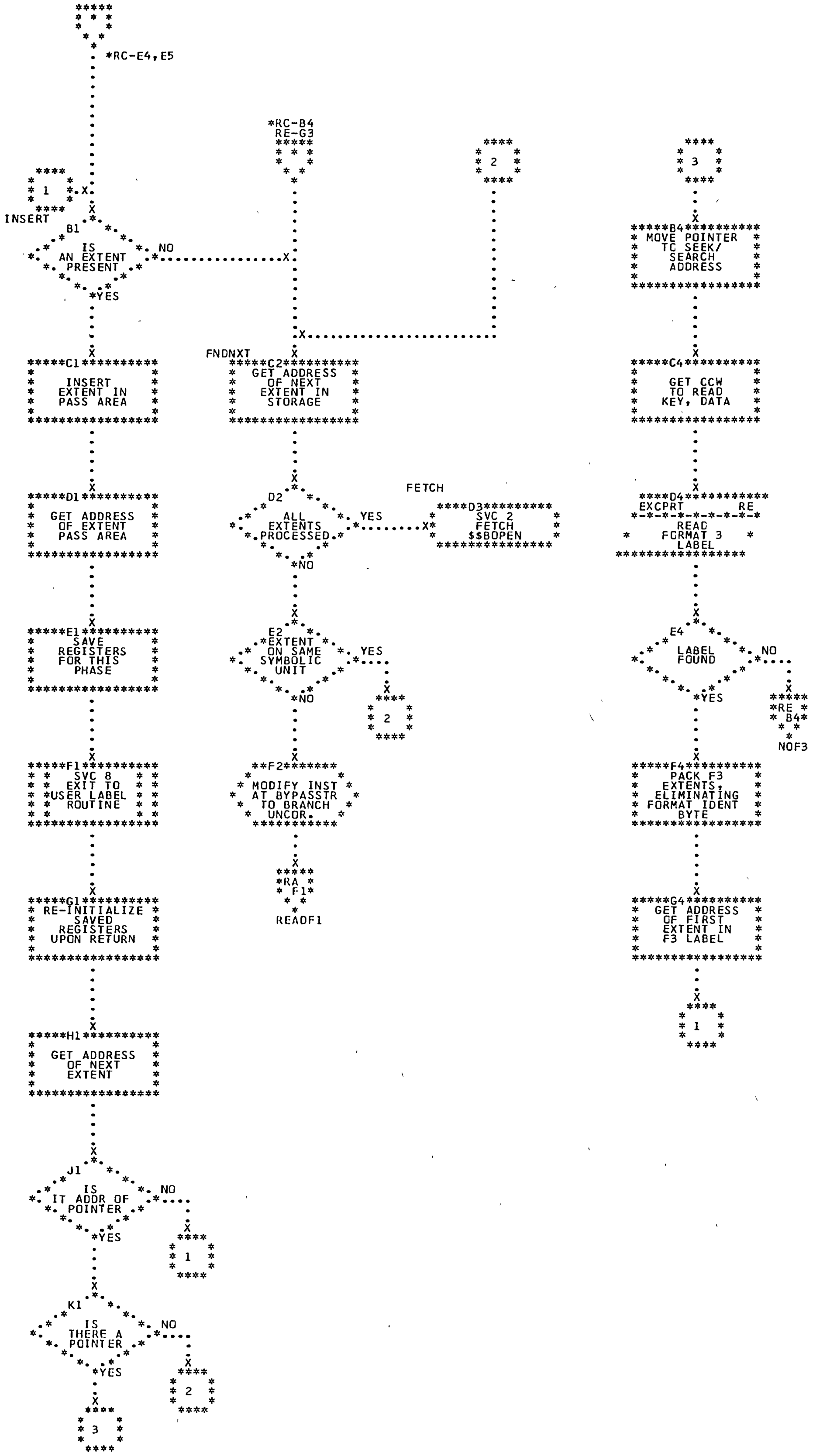
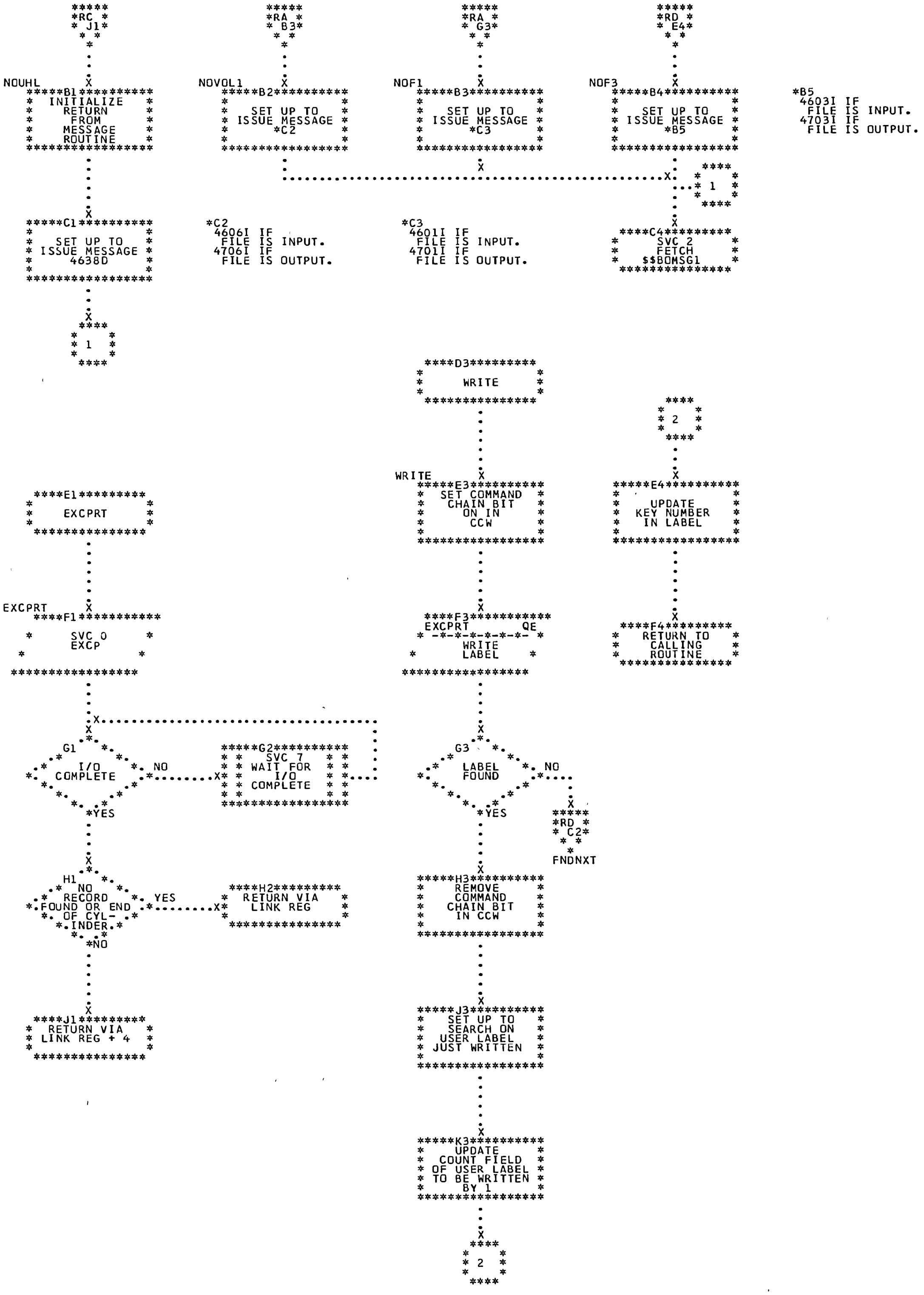
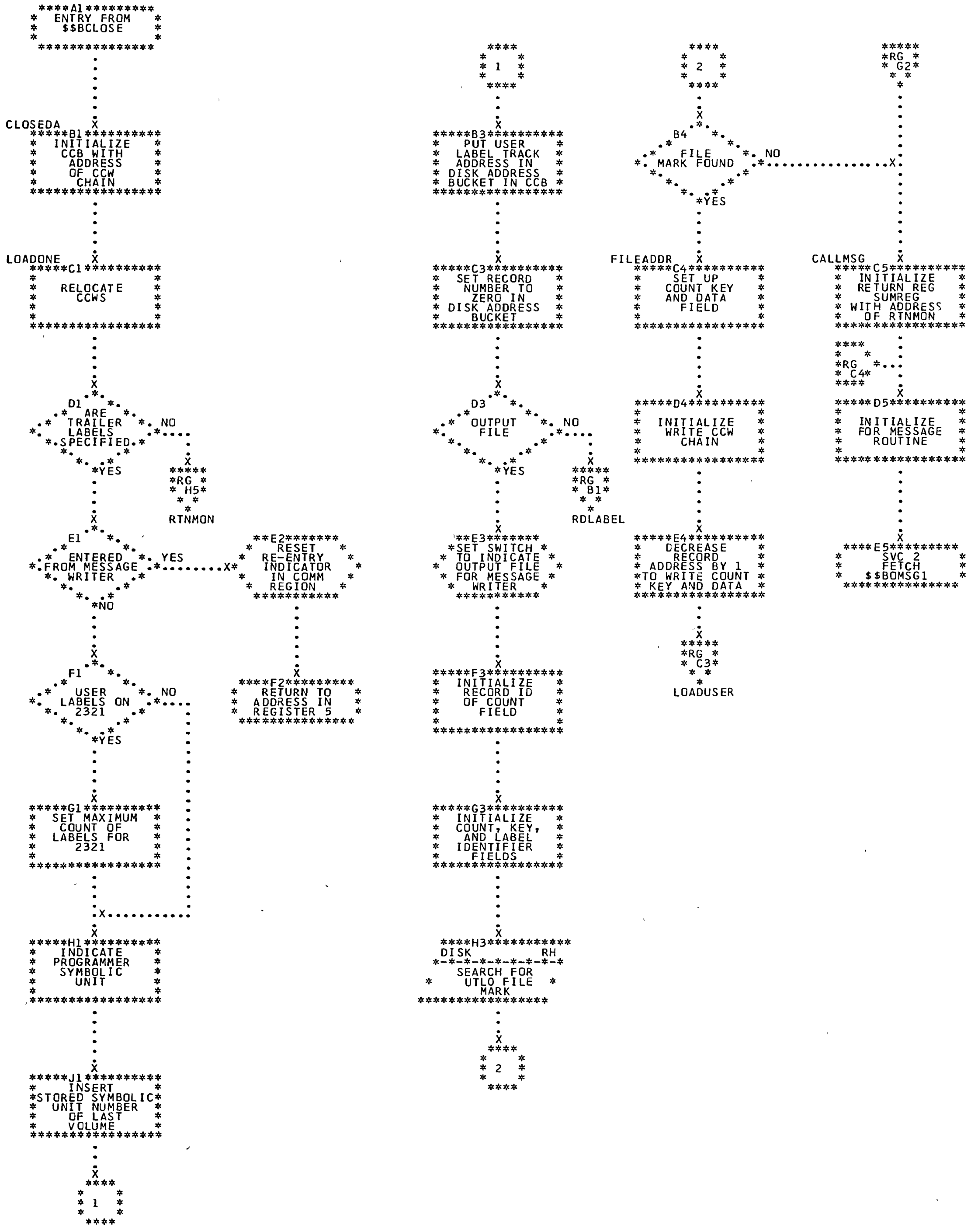

*RC-E4,E5
1
INSERT
B1
IS AN EXTENT PRESENT
NO
YES
C1
INSERT EXTENT IN PASS AREA
D1
GET ADDRESS OF EXTENT PASS AREA
E1
SAVE REGISTERS FOR THIS PHASE
F1
SVC 8 EXIT TO USER LABEL ROUTINE
G1
RE-INITIALIZE SAVED REGISTERS UPON RETURN
H1
GET ADDRESS OF NEXT EXTENT
J1
IS IT ADDR OF POINTER
NO
1
YES
K1
IS THERE A POINTER
NO
2
YES
3
*RC-B4 RE-G3
2
FNDNXT
C2
GET ADDRESS OF NEXT EXTENT IN STORAGE
D2
ALL EXTENTS PROCESSED
YES
FETCH
D3
SVC 2 FETCH $$BOPEN
NO
E2
EXTENT ON SAME SYMBOLIC UNIT
YES
2
NO
F2
MODIFY INST AT BYPASSTR TO BRANCH UNCOR.
RA F1
READF1
3
B4
MOVE POINTER TO SEEK/ SEARCH ADDRESS
C4
GET CCW TO READ KEY, DATA
D4
EXCPRT
RE
READ FORMAT 3 LABEL
E4
LABEL FOUND
NO
RE B4
NOF3
YES
F4
PACK F3 EXTENTS, ELIMINATING FORMAT IDENT BYTE
G4
GET ADDRESS OF FIRST EXTENT IN F3 LABEL
1

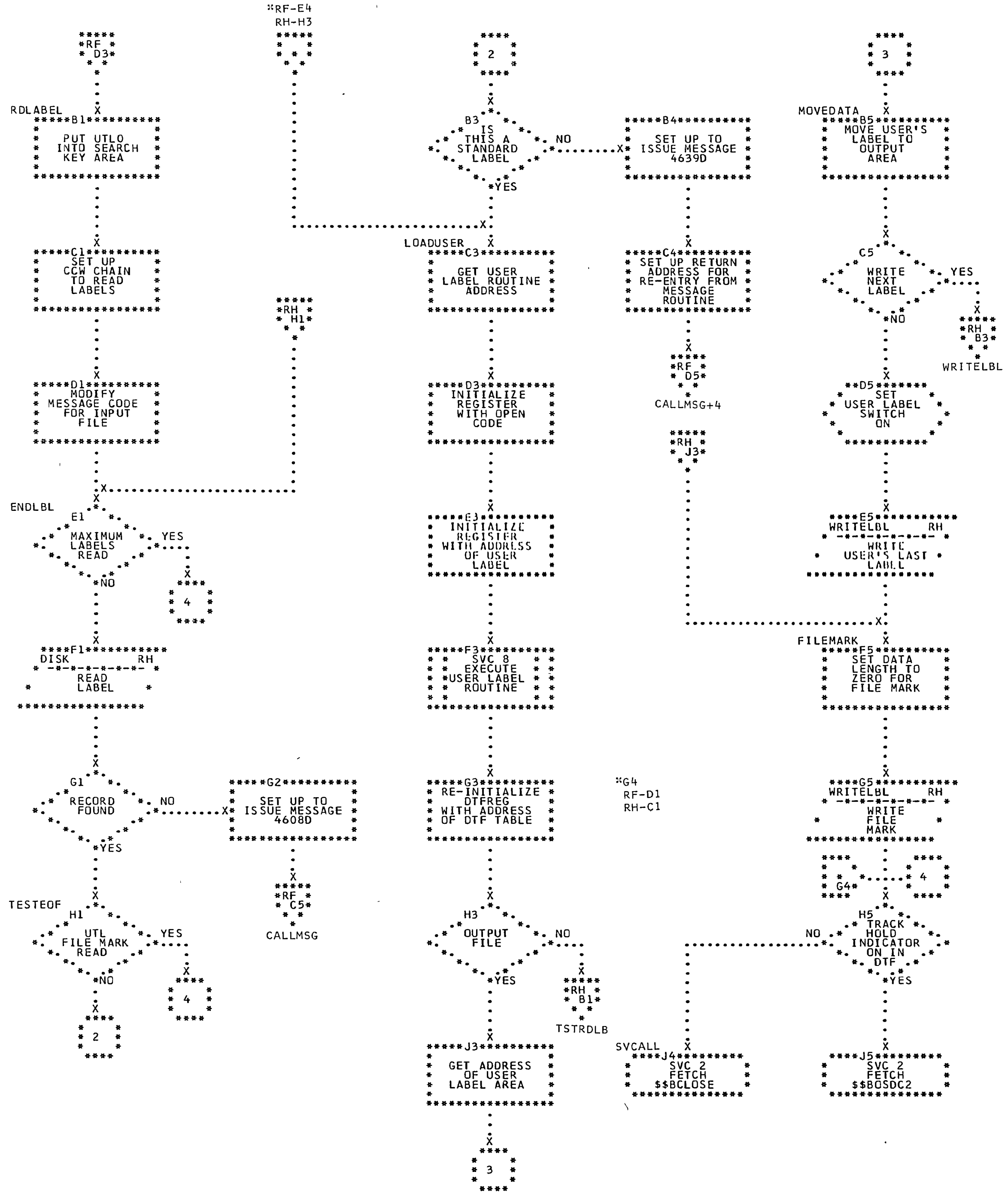
Chart RE. $$BODAU1: DA Open Input, Output (Section 5 of 5)


RC J1
RA B3
RA G3
RD E4
NOUHL
B1 INITIALIZE RETURN FROM MESSAGE ROUTINE
NOVOL1
B2 SET UP TO ISSUE MESSAGE *C2
NOF1
B3 SET UP TO ISSUE MESSAGE *C3
NOF3
B4 SET UP TO ISSUE MESSAGE *B5
*B5 4603I IF FILE IS INPUT. 4703I IF FILE IS OUTPUT.
1
C1 SET UP TO ISSUE MESSAGE 4638D
*C2 4606I IF FILE IS INPUT. 4706I IF FILE IS OUTPUT.
*C3 4601I IF FILE IS INPUT. 4701I IF FILE IS OUTPUT.
C4 SVC 2 FETCH $$BOMSG1
1
D3 WRITE
2
WRITE
E3 SET COMMAND CHAIN BIT ON IN CCW
E4 UPDATE KEY NUMBER IN LABEL
E1 EXCPRT
EXCPRT
F1 SVC 0 EXCP
F3 EXCPRT QE WRITE LABEL
F4 RETURN TO CALLING ROUTINE
G1 I/O COMPLETE
NO
YES
G2 SVC 7 WAIT FOR I/O COMPLETE
G3 LABEL FOUND
NO
YES
RD C2
FNDNXT
H1 NO RECORD FOUND OR END OF CYL-INDER
YES
NO
H2 RETURN VIA LINK REG
H3 REMOVE COMMAND CHAIN BIT IN CCW
J1 RETURN VIA LINK REG + 4
J3 SET UP TO SEARCH ON USER LABEL JUST WRITTEN
K3 UPDATE COUNT FIELD OF USER LABEL TO BE WRITTEN BY 1
2

## Chart RF. $$BODACL: DA Close Input/Output (Section 1 of 3)

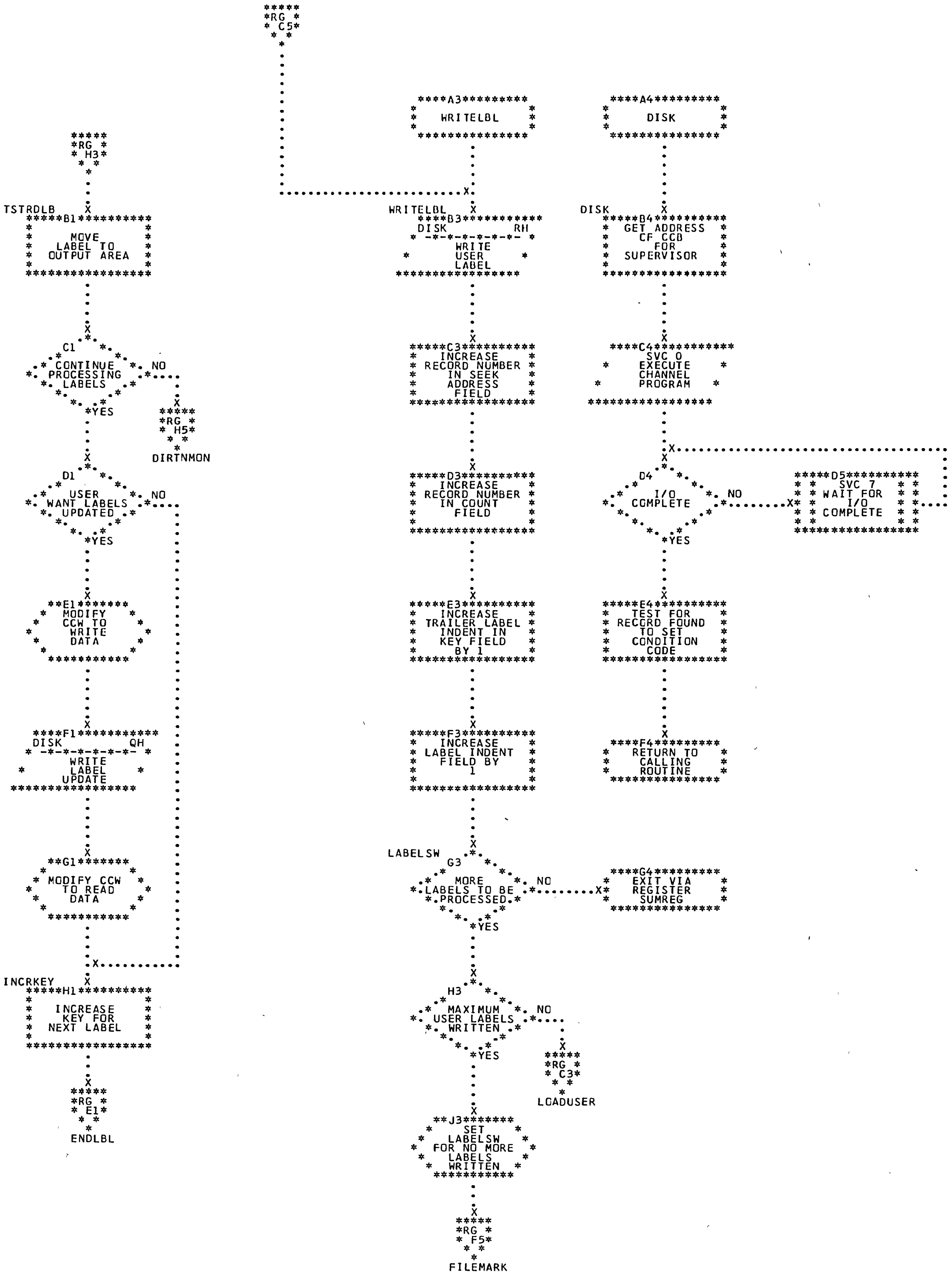

A1
ENTRY FROM $$BCLOSE
CLOSEDA
B1
INITIALIZE CCB WITH ADDRESS OF CCW CHAIN
LOADONE
C1
RELOCATE CCWS
D1
ARE TRAILER LABELS SPECIFIED
NO
RG H5
RTNMON
YES
E1
ENTERED FROM MESSAGE WRITER
YES
E2
RESET RE-ENTRY INDICATOR IN COMM REGION
NO
F1
USER LABELS ON 2321
NO
F2
RETURN TO ADDRESS IN REGISTER 5
YES
G1
SET MAXIMUM COUNT OF LABELS FOR 2321
H1
INDICATE PROGRAMMER SYMBOLIC UNIT
J1
INSERT STORED SYMBOLIC UNIT NUMBER OF LAST VOLUME
1
1
B3
PUT USER LABEL TRACK ADDRESS IN DISK ADDRESS BUCKET IN CCB
C3
SET RECORD NUMBER TO ZERO IN DISK ADDRESS BUCKET
D3
OUTPUT FILE
NO
RG B1
RDLABEL
YES
E3
SET SWITCH TO INDICATE OUTPUT FILE FOR MESSAGE WRITER
F3
INITIALIZE RECORD ID OF COUNT FIELD
G3
INITIALIZE COUNT, KEY, AND LABEL IDENTIFIER FIELDS
H3
DISK RH
SEARCH FOR UTLO FILE MARK
2
2
B4
FILE MARK FOUND
NO
YES
FILEADDR
C4
SET UP COUNT KEY AND DATA FIELD
D4
INITIALIZE WRITE CCW CHAIN
E4
DECREASE RECORD ADDRESS BY 1 TO WRITE COUNT KEY AND DATA
RG C3
LOADUSER
RG G2
CALLMSG
C5
INITIALIZE RETURN REG SUMREG WITH ADDRESS OF RTNMON
RG C4
D5
INITIALIZE FOR MESSAGE ROUTINE
E5
SVC 2 FETCH $$BOMSG1

## Chart RG. $$BODACL: DA Close Input/Output (Section 2 of 3)



RF D3
RDLABEL
B1
PUT UTLO INTO SEARCH KEY AREA
C1
SET UP CCW CHAIN TO READ LABELS
D1
MODIFY MESSAGE CODE FOR INPUT FILE
ENDLBL
E1
MAXIMUM LABELS READ
YES
NO
4
F1
DISK RH
READ LABEL
G1
RECORD FOUND
NO
YES
G2
SET UP TO ISSUE MESSAGE 4608D
RF C5
CALLMSG
TESTEOF
H1
UTL FILE MARK READ
YES
NO
4
2
RF-E4
RH-H3
RH H1
2
B3
IS THIS A STANDARD LABEL
NO
YES
B4
SET UP TO ISSUE MESSAGE 4639D
LOADUSER
C3
GET USER LABEL ROUTINE ADDRESS
C4
SET UP RETURN ADDRESS FOR RE-ENTRY FROM MESSAGE ROUTINE
RF D5
CALLMSG+4
RH J3
D3
INITIALIZE REGISTER WITH OPEN CODE
E3
INITIALIZE REGISTER WITH ADDRESS OF USER LABEL
F3
SVC 8 EXECUTE USER LABEL ROUTINE
G3
RE-INITIALIZE DTFREG WITH ADDRESS OF DTF TABLE
G4
RF-D1
RH-C1
H3
OUTPUT FILE
NO
YES
RH B1
TSTRDLB
J3
GET ADDRESS OF USER LABEL AREA
3
SVCALL
J4
SVC 2 FETCH $$BCLOSE
3
MOVEDATA
B5
MOVE USER'S LABEL TO OUTPUT AREA
C5
WRITE NEXT LABEL
YES
NO
RH B3
WRITELBL
D5
SET USER LABEL SWITCH ON
E5
WRITELBL RH
WRITE USER'S LAST LABEL
FILEMARK
F5
SET DATA LENGTH TO ZERO FOR FILE MARK
G5
WRITELBL RH
WRITE FILE MARK
G4
4
H5
TRACK HOLD INDICATOR ON IN DTF
NO
YES
J5
SVC 2 FETCH $$BOSDC2

Chart RH. $$BODACL: DA Close Input/Output (Section 3 of 3)


RG C5
A3
WRITELBL
A4
DISK
RG H3
TSTRDLB
B1
MOVE LABEL TO OUTPUT AREA
WRITELBL
B3
DISK RH
WRITE USER LABEL
DISK
B4
GET ADDRESS OF CCB FOR SUPERVISOR
C1
CONTINUE PROCESSING LABELS
NO
YES
RG H5
DIRTNMON
C3
INCREASE RECORD NUMBER IN SEEK ADDRESS FIELD
C4
SVC 0 EXECUTE CHANNEL PROGRAM
D1
USER WANT LABELS UPDATED
NO
YES
D3
INCREASE RECORD NUMBER IN COUNT FIELD
D4
I/O COMPLETE
NO
YES
D5
SVC 7 WAIT FOR I/O COMPLETE
E1
MODIFY CCW TO WRITE DATA
E3
INCREASE TRAILER LABEL INDENT IN KEY FIELD BY 1
E4
TEST FOR RECORD FOUND TO SET CONDITION CODE
F1
DISK QH
WRITE LABEL UPDATE
F3
INCREASE LABEL INDENT FIELD BY 1
F4
RETURN TO CALLING ROUTINE
LABELSW
G3
MORE LABELS TO BE PROCESSED
NO
YES
G4
EXIT VIA REGISTER SUMREG
G1
MODIFY CCW TO READ DATA
INCRKEY
H1
INCREASE KEY FOR NEXT LABEL
H3
MAXIMUM USER LABELS WRITTEN
NO
YES
RG C3
LOADUSER
RG E1
ENDLBL
J3
SET LABELSW FOR NO MORE LABELS WRITTEN
RG F5
FILEMARK

| Label | Phase or Routine | Chart |
|---|---|---|
| ABORT | $$BODSMW | AU |
| ABORTJOB | $$BODSPV | AE |
| ABORTJOB | $$BOMSG2 | AR |
| ABORTJOB | $$BOSDO7 | JS |
| ABORTJOB | SD | KC |
| ABORTNOT | $$BOSDW1 | KF |
| ADD1XTNT | $$BOSDW2 | KG |
| ALLBYPS | $$BOSDO1 | JA |
| ALLDONE | $$BODSPW | AG |
| ALLDONE | $$BOWDMP | AL |
| ALTR2311 | $$BODAIN | PG |
| ALTR2314 | $$BODAIN | PG |
| ANYOPEN | $$BOSDW1 | KE |
| AROUND | $$BODAI1 | PK |
| AROUND | $$BODAIN | PG |
| BADASSIN | $$BODAIN | PF |
| BELOWFE | $$BODAIN | PF |
| BELOWFE | $$BOSDW1 | KE |
| BELOWFE | $$BOSIGN | JE |
| BLANKOUT | $$BOWDMP | AM |
| BLD | DAMOD | NL |
| BRSWIT | $$BOSD00 | HA |
| BUILDF1 | $$BOSDO4 | JL |
| BYPASSED | $$BOSDI1 | HC |
| BYPASSTR | $$BODAU1 | RA |
| BYPASSX | $$BOSDI1 | HC |
| BYPASSX | $$BOSDO1 | JA |
| BYPASSX | $$BOSDW1 | KD |
| CALLMSG | $$BODACL | RF |
| CBLCLOSE | $$BOSDI3 | HK |
| CELLTEST | $$BOFLPT | AA |
| CFVRFX | $$BODAO2 | QB |
| CHCK | $$BODAO2 | QD |
| CHECK | $$BODAO2 | QB |
| CHECK | $$BOSDI1 | HD |
| CHECKLL | $$BOSDO8 | KA |
| CHECKXNT | $$BOSDO8 | KA |
| CHEKXTNT | $$BOSDO3 | JH |
| CHKF3CNT | $$BODSPW | AG |
| CHKVTOC | $$BOSDW1 | KF |
| CKLST | $$BODAO1 | PQ |
| CKMORE | $$BOSD01 | HB |
| CKNRF | $$BODAO2 | QA |
| CKOVLP | $$BODAO1 | PP |
| CKOVLP1 | $$BODAO1 | PQ |
| CKSTOP | $$BODAO2 | QC |
| CKSUSQ | $$BODAI1 | PK |
| CKSYS | $$BOSD00 | HA |
| CKUNIT | $$BODAO1 | PP |
| CKUSLB | $$BODAU1 | RA |
| CKVTOC | $$BODAO1 | PP |
| CKXT | $$BODAO1 | PQ |
| CKXTNT | $$BODAI1 | PK |
| CLCFRS | $$BODAO1 | PN |
| CLEAR | $$BODAI1 | PJ |
| CLEAR | $$BODAIN | PG |
| CLEARLBL | $$BOSDO2 | JF |
| CLEARLBL | $$BOSDO8 | KA |
| CLEAROUT | $$BOMSG2 | AS |
| CLEARSW1 | $$BOSDC1 | LB |
| CLEARSW2 | $$BOSDC1 | LB |
| CLOSECBL | $$BOSDI3 | HK |
| CLOSEDA | $$BODACL | RF |
| CLOSEDTF | $$BOSDC1 | LA |
| CLOSEMON | $$BOSDO6 | JQ |
| CLOSFILE | $$BOSDC1 | LC |
| CNTXTNT | $$BODAI1 | PK |
| COBOLBL | $$BOSDC1 | LC |
| COMP | $$BODAO2 | QA |
| COMPAR | $$BODAO3 | QE |
| COMPARE3 | $$BOMSG1 | AP |
| COMPDEV | $$BOSDW1 | KE |
| COMPDEV | $$BOSIGN | JE |
| COMPHEAD | $$BOSDO3 | JH |
| COMPNXT | $$BODAIN | PG |
| COMPXTNT | $$BOSDO7 | JS |
| CONTIN | $$BOSDW1 | KE |
| CONTIN | $$BOSIGN | JE |
| CONVERT | $$BODSMW | AT |
| CPCLOSE | $$BOSDC1 | LA |
| CREATEX1 | $$BOSDO3 | JJ |
| CRETF3 | $$BODAO4 | QK |
| CYLEQUAL | $$BOSDO3 | JJ |
| DELETE | $$BODAO1 | PR |
| DELETION | $$BOSDO3 | JK |
| DEQUERTN | $$BODQUE | LE |
| DEVTYP | $$BODAU1 | RA |
| DISK | $$BODACL | RH |
| DLABOK | $$BOSD01 | HB |
| DONTMOVE | $$BODSMW | AT |
| DONTMOVE | $$BOMSG1 | AN |
| DQEXTNT | $$BOSDO6 | JR |
| DUMMY | $$BOSD01 | HB |
| DUMPIT | $$BODSMW | AU |
| DUMPVTOC | $$BOSDO7 | JS |
| EMTYSPOT | $$BOSDO5 | JN |
| ENDLBL | $$BODACL | RG |
| EOBRESP | $$BOSDO7 | JS |
| EOFEXIT | $$BOSDI3 | HK |
| EOFSWITCH | $$BOSDI3 | HJ |
| EOXADDR | $$BOSDO1 | JB |
| EQXTENT | $$BOSDI2 | HF |
| ERROR | $$BOFLPT | AC |
| EXCPRT | $$BODAI1 | PM |
| EXCPRT | $$BODAO1 | PR |
| EXCPRT | $$BODAO2 | QD |
| EXCPRT | $$BODAO3 | QH |
| EXCPRT | $$BODAO4 | QM |
| EXCPRT | $$BODAU1 | RE |
| EXITRT | $$BODAI1 | PL |
| EXITRT | $$BODAO4 | QL |
| EXITUSER | $$BOSDC2 | LD |
| EXPDDSF | $$BODAO2 | QC |
| EXPIRED | $$BOSDO3 | JK |
| EXPIRED | $$BOSDO8 | KA |
| EXTENT5 | $$BOSDW2 | KG |

| | | |
|---|---|---|
| FCHMSG | $$BOSD00 | HA |
| FEOVRET | $$BOSD05 | JP |
| FETCH | $$BODA01 | PQ |
| FETCH | $$BODA02 | QA |
| FETCH | $$BODAU1 | RD |
| FETCH | $$BOFLPT | AC |
| FETCH | $$BOMSG2 | AR |
| FETCH1 | $$BODSMW | AU |
| FETCHP1 | $$BOSDW1 | KF |
| FILEADD2 | $$BOSD06 | JQ |
| FILEADDR | $$BODACL | RF |
| FILEMARK | $$BODACL | RG |
| FILEMARK | $$BOSD06 | JR |
| FILEOVLP | $$BOMSG1 | AP |
| FILEOVLP | $$BOMSG2 | AQ |
| FILEOVLP | $$BOSD03 | JK |
| FILEPROT | $$BOSDI4 | HM |
| FILEPROT | $$BOSD01 | JC |
| FILEPROT | $$BOSD04 | JM |
| FILEPROT | $$BOSD05 | JP |
| FILEPRTD | $$BOFLPT | AA |
| FILETYPE | $$BOSDC1 | LA |
| FINDF1 | $$BODAI1 | PJ |
| FINDF1 | $$BODAU1 | RA |
| FINDLAST | $$BOSD01 | JB |
| FINDSPOT | $$BOSD04 | JL |
| FINDSPOT | $$BOSD05 | JN |
| FINDSPOT | $$BOSDW2 | KG |
| FINDSPT | $$BOSDW2 | KG |
| FINDXTNT | $$BOSDW3 | KJ |
| FINXNT | $$BODAI1 | PK |
| FIRSTMSG | $$BODSPV | AE |
| FIXED | $$BOSD01 | JB |
| FLEPRTCT | $$BODAI1 | PM |
| FLEPRTCT | $$BODA04 | QM |
| FLG | DAMOD | NL |
| FNDNXT | $$BODAU1 | RD |
| FNDOPN | $$BODA03 | QG |
| FNSHF1 | $$BODA04 | QJ |
| FORMAT3 | $$BOSD05 | JN |
| FORMAT3 | $$BOSDW2 | KG |
| FTCHMSG | $$BODA03 | QH |
| FTCHMSG | $$BODA04 | QL |
| GETCUU | $$BODSMW | AT |
| GETCUU | $$BOMSG1 | AN |
| GETDSPLY | $$BOMSG2 | AR |
| GETDSPLY | $$BOSD07 | JS |
| GETFP | $$BODAI1 | PL |
| GETFP | $$BODA04 | QL |
| GETJIB | $$BOFLPT | AD |
| GETLAB | $$BODA01 | PN |
| GETMON | $$BODAI1 | PL |
| GETMON | $$BODA04 | QL |
| GETNEXT | $$BODQUE | LE |
| GETNEXT | $$BOSDI1 | HD |
| GETNXJIB | $$BOFLPT | AD |
| GETOPEN | $$BODAIN | PF |
| GETPOINT | $$BOSD05 | JN |
| GETPOINT | $$BOSDW2 | KG |
| GETPTR | $$BODA02 | QD |
| HEADERSW | $$BOSD06 | JR |
| IFOPENFD | $$BOSD03 | JG |
| IJBWCONT | SDMODW | GE |

| | | |
|---|---|---|
| IJGBADD2 | SDMODFU | CJ |
| IJGBCHCK | SDMODFU | CL |
| IJGBCTRL | CNTRL | CT |
| IJGBDATE | SDMODFU | CN |
| IJGBDATK | SDMODFU | CN |
| IJGBDEBL | SDMODFU | CH |
| IJGBDECL | SDMODFU | CM |
| IJGBDECR | SDMODFU | CN |
| IJGBDONE | SDMODFU | CJ |
| IJGBEOFT | SDMODFU | CN |
| IJGBEXT | SDMODFU | CJ |
| IJGBEXTN | SDMODFU | CR |
| IJGBEXTN | SDMODFU | CR |
| IJGBFAC1 | SDMODFU | CQ |
| IJGBFACT | SDMODFU | CH |
| IJGBFREE | SDMODFU | CR |
| IJGBGENM | SDMODFU | CM |
| IJGBGET | SDMODFU | CH |
| IJGBGET | SDMODFU | CH |
| IJGBGETA | SDMODFU | CK |
| IJGBGETD | SDMODFU | CK |
| IJGBHOLD | SDMODFU | CK |
| IJGBIORU | SDMODFU | CH |
| IJGBKETH | SDMODFU | CM |
| IJGBKETH | SDMODFU | CM |
| IJGBLOOP | SDMODFU | CJ |
| IJGBNXRC | SDMODFU | CQ |
| IJGBPUT | SDMODFU | CS |
| IJGBRD1 | SDMODFU | CH |
| IJGBRD3 | SDMODFU | CJ |
| IJGBRDER | SDMODFU | CN |
| IJGBRDER | SDMODFU | CP |
| IJGBRDWR | SDMODFU | CQ |
| IJGBREAD | SDMODFU | CK |
| IJGBRELA | SDMODFU | CH |
| IJGBRELS | RELSE | CT |
| IJGBRESI | SDMODFU | CM |
| IJGBRET | SDMODFU | CP |
| IJGBRET | SDMODFU | CP |
| IJGBRTRI | SDMODFU | CP |
| IJGBRTRY | SDMODFU | CP |
| IJGBSKIP | SDMODFU | CQ |
| IJGBSWOF | SDMODFU | CN |
| IJGBSW03 | SDMODFU | CK |
| IJGBTEST | SDMODFU | CH |
| IJGBTOM | SDMODFU | CM |
| IJGBTST2 | SDMODFU | CQ |
| IJGBUNIO | SDMODFU | CN |
| IJGBUSER | SDMODFU | CN |
| IJGBWAIT | SDMODFU | CM |
| IJGBWLRE | SDMODFU | CN |
| IJGBWLRI | SDMODFU | CN |
| IJGBWRIT | SDMODFU | CL |
| IJGBWRT | SDMODFU | CR |
| IJGBWRT1 | SDMODFU | CR |
| IJGCADD1 | SDMODFI | BH |
| IJGCADD2 | SDMODFI | BH |
| IJGCADD3 | SDMODFI | BH |
| IJGCCHCK | SDMODFI | BG |
| IJGCCTRL | CNTRL | CT |
| IJGCDATE | SDMODFI | BG |
| IJGCDATK | SDMODFI | BF |
| IJGCDATK | SDMODFI | BF |
| IJGCDEBL | SDMODFI | BF |

| | | |
|---|---|---|
| IJGCDECT | SDMODFI | BJ |
| IJGCDER | SDMODFI | BF |
| IJGCDONE | SDMODFI | BH |
| IJGCEXTN | SDMODFI | BF |
| IJGCEXTN | SDMODFI | BH |
| IJGCEXTN | SDMODFI | BH |
| IJGCGETA | SDMODFI | BJ |
| IJGCGETD | SDMODFI | BJ |
| IJGCGETX | SDMODFI | BF |
| IJGCIORU | SDMODFI | BF |
| IJGCLOOP | SDMODFI | BH |
| IJGCRD1 | SDMODFI | BF |
| IJGCRDEV | SDMODFI | BG |
| IJGCRDEV | SDMODFI | BH |
| IJGCRELA | SDMODFI | BF |
| IJGCRELS | RELSE | CT |
| IJGCRET | SDMODFI | BG |
| IJGCRET | SDMODFI | BH |
| IJGCSKIP | SDMODFI | BH |
| IJGCSW03 | SDMODFI | BJ |
| IJGCTEST | SDMODFI | BF |
| IJGCUSER | SDMODFI | BG |
| IJGCWLRE | SDMODFI | BJ |
| IJGCWLRI | SDMODFI | BJ |
| IJGDBLKR | SDMODFO | BR |
| IJGDBUFF | SDMODFO | BR |
| IJGDCLOS | SDMODFO | BV |
| IJGDCLOS | SDMODFO | BV |
| IJGDCTRL | CNTRL | CT |
| IJGDDECR | SDMODFO | BR |
| IJGDDONE | SDMODFO | BS |
| IJGDENCY | SDMODFO | BS |
| IJGDERR1 | SDMODFO | BT |
| IJGDEXCH | SDMODFO | BR |
| IJGDEXTN | SDMODFO | BS |
| IJGDEXTN | SDMODFO | BS |
| IJGDFLIP | SDMODFO | BV |
| IJGDIORU | SDMODFO | BR |
| IJGDLOOP | SDMODFO | BS |
| IJGDPUT | SDMODFO | BR |
| IJGDPUT | SDMODFO | BR |
| IJGDSTKY | SDMODFO | BS |
| IJGDSW02 | SDMODFO | BR |
| IJGDSW03 | SDMODFO | BS |
| IJGDUNIO | SDMODFO | BT |
| IJGDWAIT | SDMODFO | BT |
| IJGDWROT | SDMODFO | BR |
| IJGEBLKR | SDMODFO | BK |
| IJGEBUFF | SDMODFO | BL |
| IJGECALF | SDMODFO | BK |
| IJGECALP | SDMODFO | BK |
| IJGECALT | SDMODFO | BK |
| IJGECLOS | SDMODFO | BQ |
| IJGECLOS | SDMODFO | BQ |
| IJGECTRL | CNTRL | CT |
| IJGEDECR | SDMODFO | BK |
| IJGEDONE | SDMODFO | BL |
| IJGEEFFL | SDMODFO | BL |
| IJGEENCY | SDMODFO | BL |
| IJGEEOFS | SDMODFO | BL |
| IJGEERR1 | SDMODFO | BN |
| IJGEESCH | SDMODFO | BM |
| IJGEEXCL | SDMODFO | BL |
| IJGEEXCL | SDMODFO | BQ |
| IJGEEXTN | SDMODFO | BL |
| IJGEEXTN | SDMODFO | BL |
| IJGEFLIP | SDMODFO | BQ |
| IJGEFLIP | SDMODFO | BQ |
| IJGEIORU | SDMODFO | BK |
| IJGELOOP | SDMODFO | BL |
| IJGEPUT | SDMODFO | BK |
| IJGEPUT | SDMODFO | BK |
| IJGESTRY | SDMODFO | BQ |
| IJGESW03 | SDMODFO | BM |
| IJGETRUN | SDMODFO | BQ |
| IJGETUPR | SDMODFO | BL |
| IJGEUNIO | SDMODFO | BN |
| IJGEVRRF | SDMODFO | BL |
| IJGEWAIT | SDMODFO | BN |
| IJGEZERF | SDMODFO | BM |
| IJGFADD3 | SDMODFI | BA |
| IJGFCTRL | CNTRL | CT |
| IJGFDATK | SDMODFI | BC |
| IJGFDATK | SDMODFI | BC |
| IJGFDATR | SDMODFI | BC |
| IJGFDONE | SDMODFI | BB |
| IJGFEXTN | SDMODFI | BA |
| IJGFEXTN | SDMODFI | BA |
| IJGFFACT | SDMODFI | BE |
| IJGFGET | SDMODFI | BA |
| IJGFGET | SDMODFI | BA |
| IJGFGETA | SDMODFI | BB |
| IJGFGETD | SDMODFI | BB |
| IJGFGETX | SDMODFI | BA |
| IJGFIORU | SDMODFI | BA |
| IJGFJOHN | SDMODFI | BE |
| IJGFLIP | SDMODFO | BV |
| IJGFLOOP | SDMODFI | BB |
| IJGFNXRC | SDMODFI | BE |
| IJGFRD1 | SDMODFI | BA |
| IJGFRDER | SDMODFI | BC |
| IJGFRDER | SDMODFI | BD |
| IJGFRELA | SDMODFI | BA |
| IJGFRELS | RELSE | CT |
| IJGFRET | SDMODFI | BD |
| IJGFRET | SDMODFI | BD |
| IJGFRTRI | SDMODFI | BD |
| IJGFRTRY | SDMODFI | BD |
| IJGFSETC | SDMODFI | BA |
| IJGFSKIP | SDMODFI | BC |
| IJGFSW02 | SDMODFI | BA |
| IJGFSW02 | SDMODFO | BK |
| IJGFSW03 | SDMODFI | BB |
| IJGFSW04 | SDMODFI | BB |
| IJGFSW06 | SDMODFI | BB |
| IJGFTEST | SDMODFI | BA |
| IJGFUNIO | SDMODFI | BC |
| IJGFUPDA | SDMODFI | BE |
| IJGFWAIT | SDMODFI | BC |
| IJGFWLRI | SDMODFI | BC |
| IJGGADD2 | SDMODFU | BX |
| IJGGADD3 | SDMODFU | BW |
| IJGGADD4 | SDMODFU | BX |
| IJGGCHCK | SDMODFU | CD |
| IJGGCTRL | CNTRL | CT |
| IJGGDATR | SDMODFU | CB |
| IJGGDEC | SDMODFU | CF |
| IJGGDONE | SDMODFU | BX |

| | | |
|---|---|---|
| IJGGEOFF | SDMODFU | BW |
| IJGGEOFT | SDMODFU | CB |
| IJGGEXT | SDMODFU | BX |
| IJGGEXT | SDMODFU | BX |
| IJGGEXTN | SDMODFU | CE |
| IJGGEXTN | SDMODFU | CE |
| IJGGFAC1 | SDMODFU | CD |
| IJGGFACT | SDMODFU | CF |
| IJGGFREE | SDMODFU | CF |
| IJGGGENM | SDMODFU | CF |
| IJGGGET | SDMODFU | BW |
| IJGGGET | SDMODFU | BW |
| IJGGGETA | SDMODFU | BY |
| IJGGGETD | SDMODFU | BY |
| IJGGGETX | SDMODFU | BW |
| IJGGHOLD | SDMODFU | BY |
| IJGGIORU | SDMODFU | BW |
| IJGGKETH | SDMODFU | CA |
| IJGGKETH | SDMODFU | CA |
| IJGGLOOP | SDMODFU | BX |
| IJGGNXRC | SDMODFU | CE |
| IJGGPUT | SDMODFU | CG |
| IJGGRD1 | SDMODFU | BW |
| IJGGRDER | SDMODFU | CB |
| IJGGRDER | SDMODFU | CC |
| IJGGRDWR | SDMODFU | CD |
| IJGGREAD | SDMODFU | BY |
| IJGGRELA | SDMODFU | BW |
| IJGGRELS | RELSE | CT |
| IJGGRESI | SDMODFU | CF |
| IJGGRET | SDMODFU | CC |
| IJGGRET | SDMODFU | CC |
| IJGGRTRI | SDMODFU | CC |
| IJGGRTRI | SDMODFU | CC |
| IJGGSKIP | SDMODFU | CD |
| IJGGSUPD | SDMODFU | BY |
| IJGGSWOF | SDMODFU | CB |
| IJGGSW02 | SDMODFU | BW |
| IJGGSW03 | SDMODFU | BX |
| IJGGSW04 | SDMODFU | BX |
| IJGGSW05 | SDMODFU | BX |
| IJGGSW06 | SDMODFU | BY |
| IJGGTEST | SDMODFU | BW |
| IJGGTOM | SDMODFU | CA |
| IJGGTST2 | SDMODFU | CD |
| IJGGUSER | SDMODFU | CB |
| IJGGWAIT | SDMODFU | CA |
| IJGGWLPI | SDMODFU | CB |
| IJGGWRT | SDMODFU | BZ |
| IJGGWRT1 | SDMODFU | BZ |
| IJGQU111 | SDMODVU | EK |
| IJGQU112 | SDMODVU | EJ |
| IJGQU114 | SDMODVU | EJ |
| IJGQU115 | SDMODVU | EH |
| IJGQU110 | SDMODVU | EJ |
| IJGQU121 | SDMODVU | EK |
| IJGQU122 | SDMODVU | EK |
| IJGQU123 | SDMODVU | EB |
| IJGQU124 | SDMODVU | EK |
| IJGQU120 | SDMODVU | EK |
| IJGQU156 | SDMODVU | EM |
| IJGQU101 | SDMODVU | EH |
| IJGQU103 | SDMODVU | EH |
| IJGQU104 | SDMODVU | EJ |
| IJGQU105 | SDMODVU | EJ |
| IJGQU106 | SDMODVU | EJ |
| IJGQU107 | SDMODVU | EK |
| IJGQU108 | SDMODVU | EH |
| IJGQU109 | SDMODVU | EJ |
| IJGQU10B | SDMODVU | EJ |
| IJGQU10K | SDMODVU | EK |
| IJGQU440 | SDMODVU | EL |
| IJGQU490 | SDMODVU | ED |
| IJGQU401 | SDMODVU | EH |
| IJGQU400 | SDMODVU | EG |
| IJGQUERR | SDMODVU | EH |
| IJGQUEXT | SDMODVU | EK |
| IJGQUNR | SDMODVU | EM |
| IJGQUPT | SDMODVU | EM |
| IJGQU011 | SDMODVU | EE |
| IJGQU012 | SDMODVU | EE |
| IJGQU013 | SDMODVU | EE |
| IJGQU015 | SDMODVU | EE |
| IJGQU016 | SDMODVU | EE |
| IJGQU017 | SDMODVU | EF |
| IJGQU018 | SDMODVU | EF |
| IJGQU010 | SDMODVU | DW |
| IJGQU021 | SDMODVU | EM |
| IJGQU023 | SDMODVU | EE |
| IJGQU024 | SDMODVU | EE |
| IJGQU020 | SDMODVU | EE |
| IJGQU030 | SDMODVU | DV |
| IJGQU000 | SDMODVU | DV |
| IJGQUSER | SDMODVU | EM |
| IJGQUSKB | SDMODVU | EM |
| IJGSG410 | SDMODVI | DF |
| IJGSG425 | SDMODVI | DF |
| IJGSG420 | SDMODVI | DF |
| IJGSG430 | SDMODVI | DF |
| IJGSG400 | SDMODVI | DA |
| IJGSG500 | SDMODVI | DF |
| IJGSG600 | SDMODVI | DF |
| IJGSG710 | SDMODVI | DC |
| IJGSG720 | SDMODVI | DC |
| IJGSG700 | SDMODVI | DE |
| IJGSG810 | SDMODVI | DF |
| IJGSG805 | SDMODVI | DF |
| IJGSG800 | SDMODVI | DF |
| IJGSG910 | SDMODVI | DF |
| IJGSG030 | SDMODVI | DA |
| IJGSGSKI | SDMODVI | DA |
| IJGSP110 | SDMODVI | DC |
| IJGSP110 | SDMODVO | DK |
| IJGSP120 | SDMODVO | DQ |
| IJGSP130 | SDMODVO | DK |
| IJGSP140 | SDMODVO | DK |
| IJGSP150 | SDMODVO | DM |
| IJGSP100 | SDMODVO | DK |
| IJGSP210 | SDMODVO | DR |
| IJGSP220 | SDMODVO | DR |
| IJGSP245 | SDMODVO | DS |
| IJGSP240 | SDMODVO | DS |
| IJGSP265 | SDMODVO | DS |
| IJGSP260 | SDMODVO | DS |
| IJGSP270 | SDMODVO | DS |
| IJGSP280 | SDMODVO | DS |
| IJGSP202 | SDMODVO | DR |

| | | |
|---|---|---|
| IJGSP203 | SDMODVO | DR |
| IJGSP205 | SDMODVO | DR |
| IJGSP200 | SDMODVO | DR |
| IJGSP530 | SDMODVO | DP |
| IJGSP530 | SDMODVO | DP |
| IJGSP030 | SDMODVO | DG |
| IJGSP045 | SDMODVO | DQ |
| IJGSP040 | SDMODVO | DH |
| IJGSP040 | SDMODVO | DQ |
| IJGSP050 | SDMODVO | DQ |
| IJGSP060 | SDMODVO | DQ |
| IJGSP070 | SDMODVO | DQ |
| IJGSP080 | SDMODVO | DQ |
| IJGSP095 | SDMODVO | DK |
| IJGSP090 | SDMODVO | DK |
| IJGUG170 | SDMODUI | ET |
| IJGUG100 | SDMODUI | ET |
| IJGUG210 | SDMODUI | EQ |
| IJGUG220 | SDMODUI | EU |
| IJGUG240 | SDMODUI | EU |
| IJGUG250 | SDMODUI | EU |
| IJGUG261 | SDMODUI | EU |
| IJGUG260 | SDMODUI | EU |
| IJGUG290 | SDMODUI | EU |
| IJGUG310 | SDMODUI | ET |
| IJGUG310 | SDMODUI | ET |
| IJGUG320 | SDMODUI | EU |
| IJGUG330 | SDMODUI | EU |
| IJGUG330 | SDMODUI | EU |
| IJGUG300 | SDMODUI | EQ |
| IJGUGOPT | SDMODUI | EQ |
| IJGUG020 | CNTRL | GA |
| IJGUG030 | SDMODUI | EQ |
| IJGUG030 | SDMODUI | EQ |
| IJGUG040 | SDMODUI | EQ |
| IJGUG050 | SDMODUI | EQ |
| IJGUG060 | SDMODUI | ET |
| IJGUG060 | SDMODUO | EV |
| IJGUG070 | SDMODUI | ET |
| IJGUG080 | SDMODUI | ET |
| IJGUG090 | SDMODUI | ET |
| IJGUGUNI | SDMODUI | EQ |
| IJGUGUSR | SDMODUI | ES |
| IJGUP120 | SDMODUO | EW |
| IJGUP130 | SDMODUO | EW |
| IJGUP150 | SDMODUO | EV |
| IJGUP160 | SDMODUO | EZ |
| IJGUP170 | SDMODUO | EZ |
| IJGUP100 | SDMODUO | EW |
| IJGUP220 | SDMODUO | EW |
| IJGUP234 | SDMODUO | EW |
| IJGUP230 | SDMODUO | EW |
| IJGUP240 | SDMODUO | EW |
| IJGUP260 | SDMODUO | EW |
| IJGUP270 | SDMODUO | EW |
| IJGUP281 | SDMODUO | EX |
| IJGUP281 | SDMODUO | EX |
| IJGUP28A | SDMODUO | EX |
| IJGUP28B | SDMODUO | EX |
| IJGUP280 | SDMODUO | EX |
| IJGUP290 | SDMODUO | EY |
| IJGUP290 | SDMODUO | EV |
| IJGUP200 | SDMODUO | EZ |
| IJGUP341 | SDMODUO | FA |
| IJGUP34A | SDMODUO | FA |
| IJGUP340 | SDMODUO | FA |
| IJGUP350 | SDMODUO | FA |
| IJGUP020 | CNTRL | GA |
| IJGUP040 | SDMODUO | EV |
| IJGUP060 | SDMODUO | EV |
| IJGUP070 | SDMODUO | EV |
| IJGUP080 | SDMODUO | EW |
| IJGUP090 | SDMODUO | EW |
| IJGUPUNI | SDMODUO | EY |
| IJGUPUNI | SDMODUO | EY |
| IJGUU110 | SDMODUU | FD |
| IJGUU130 | SDMODUU | FD |
| IJGUU140 | SDMODUU | FD |
| IJGUU150 | SDMODUU | FD |
| IJGUU170 | SDMODUU | FD |
| IJGUU180 | SDMODUU | FE |
| IJGUU190 | SDMODUU | FE |
| IJGUU101 | SDMODUU | FC |
| IJGUU102 | SDMODUU | FC |
| IJGUU100 | SDMODUU | FC |
| IJGUU212 | SDMODUU | FE |
| IJGUU213 | SDMODUU | FE |
| IJGUU222 | SDMODUU | FF |
| IJGUU220 | SDMODUU | FD |
| IJGUU220 | SDMODUU | FF |
| IJGUU240 | SDMODUU | FF |
| IJGUU250 | SDMODUU | FG |
| IJGUU260 | SDMODUU | FG |
| IJGUU271 | SDMODUU | FG |
| IJGUU279 | SDMODUU | FG |
| IJGUU270 | SDMODUU | FG |
| IJGUU280 | SDMODUU | FH |
| IJGUU200 | SDMODUU | FE |
| IJGUU310 | SDMODUU | FJ |
| IJGUU310 | SDMODUU | FK |
| IJGUU350 | SDMODUU | FF |
| IJGUU360 | SDMODUU | FL |
| IJGUU370 | SDMODUU | FL |
| IJGUU380 | SDMODUU | FL |
| IJGUU390 | SDMODUU | FL |
| IJGUU410 | SDMODUU | FM |
| IJGUUFR | SDMODUU | FC |
| IJGUUNIO | SDMODUU | FG |
| IJGUUNOE | SDMODUU | FM |
| IJGUU020 | CNTRL | GA |
| IJGUU030 | SDMODUU | FB |
| IJGUU030 | SDMODUU | FB |
| IJGUU040 | SDMODUU | FB |
| IJGUU051 | SDMODUU | FB |
| IJGUU050 | SDMODUU | FB |
| IJGUU060 | SDMODUU | FB |
| IJGUU070 | SDMODUU | FB |
| IJGUU080 | SDMODUU | FC |
| IJGUU090 | SDMODUU | FC |
| IJGUUSER | SDMODUU | FH |
| IJGVG110 | SDMODVI | DC |
| IJGVG120 | SDMODVI | DC |
| IJGVG131 | SDMODVI | DC |
| IJGVG130 | SDMODVI | DC |
| IJGVG100 | SDMODVI | DC |
| IJGVG210 | SDMODVI | DC |
| IJGVG220 | SDMODVI | DC |

| | | |
|---|---|---|
| IJGVG250 | SDMODVI | DD |
| IJGVG261 | SDMODVI | DD |
| IJGVG262 | SDMODVI | DD |
| IJGVG260 | SDMODVI | DD |
| IJGVG272 | SDMODVI | DD |
| IJGVG311 | SDMODVI | DA |
| IJGVG312 | SDMODVI | DE |
| IJGVG310 | SDMODVI | DD |
| IJGVG320 | SDMODVI | DE |
| IJGVG350 | SDMODVI | DD |
| IJGVG370 | SDMODVI | DE |
| IJGVG300 | SDMODVI | DD |
| IJGVGERR | SDMODVI | DA |
| IJGVGRET | SDMODVI | DB |
| IJGVGRET | SDMODVI | DB |
| IJGVGRTR | SDMODVI | DB |
| IJGVGRTW | SDMODVI | DB |
| IJGVG020 | CNTRL | EP |
| IJGVG030 | SDMODVI | DA |
| IJGVG030 | SDMODVI | DA |
| IJGVG040 | SDMODVI | DA |
| IJGVG050 | SDMODVI | DA |
| IJGVG060 | SDMODVI | DA |
| IJGVG070 | SDMODVI | DC |
| IJGVG080 | SDMODVI | DC |
| IJGVGUNR | SDMODVI | DA |
| IJGVGUSE | SDMODVI | DB |
| IJGVGUSE | SDMODVI | DB |
| IJGVGUSR | SDMODVI | DB |
| IJGVGWLE | SDMODVI | DA |
| IJGVP110 | SDMODVO | DJ |
| IJGVP120 | SDMODVO | DJ |
| IJGVP130 | SDMODVO | DJ |
| IJGVP150 | SDMODVO | DM |
| IJGVP100 | SDMODVO | DJ |
| IJGVP210 | SDMODVO | DH |
| IJGVP230 | SDMODVO | DK |
| IJGVP240 | SDMODVO | DK |
| IJGVP260 | SDMODVO | DK |
| IJGVP280 | SDMODVO | DG |
| IJGVP290 | SDMODVO | DK |
| IJGVP200 | SDMODVO | DH |
| IJGVP310 | SDMODVO | DK |
| IJGVP320 | SDMODVO | DU |
| IJGVP331 | SDMODVO | DT |
| IJGVP330 | SDMODVO | DT |
| IJGVP330 | SDMODVO | DT |
| IJGVP351 | SDMODVO | DT |
| IJGVP352 | SDMODVO | DT |
| IJGVP350 | SDMODVO | DT |
| IJGVP360 | SDMODVO | DL |
| IJGVP380 | SDMODVO | DL |
| IJGVP390 | SDMODVO | DL |
| IJGVP440 | SDMODVO | DL |
| IJGVP450 | SDMODVO | DL |
| IJGVP460 | SDMODVO | DL |
| IJGVP470 | SDMODVO | DL |
| IJGVP520 | SDMODVO | DP |
| IJGVP531 | SDMODVO | DP |
| IJGVP53A | SDMODVO | DP |
| IJGVP53B | SDMODVO | DP |
| IJGVP530 | SDMODVO | DP |
| IJGVP550 | SDMODVO | DN |
| IJGVP560 | SDMODVO | DN |
| IJGVP570 | SDMODVO | DN |
| IJGVP580 | SDMODVO | DN |
| IJGVP020 | CNTRL | EP |
| IJGVP030 | SDMODVO | DG |
| IJGVP030 | SDMODVO | DG |
| IJGVP040 | SDMODVO | DG |
| IJGVP050 | SDMODVO | DG |
| IJGVP060 | SDMODVO | DH |
| IJGVP070 | SDMODVO | DH |
| IJGVP080 | SDMODVO | DJ |
| IJGVP090 | SDMODVO | DJ |
| IJGVPUNI | SDMODVO | DM |
| IJGVU140 | SDMODVU | DW |
| IJGVU151 | SDMODVU | DW |
| IJGVU152 | SDMODVU | DW |
| IJGVU153 | SDMODVU | DW |
| IJGVU156 | SDMODVU | DX |
| IJGVU158 | SDMODVU | DX |
| IJGVU159 | SDMODVU | DX |
| IJGVU150 | SDMODVU | DW |
| IJGVU150 | SDMODVU | DW |
| IJGVU161 | SDMODVU | DY |
| IJGVU161 | SDMODVU | DY |
| IJGVU160 | SDMODVU | DX |
| IJGVU100 | SDMODVU | DW |
| IJGVU220 | SDMODVU | EB |
| IJGVU240 | SDMODVU | EB |
| IJGVU250 | SDMODVU | EB |
| IJGVU250 | SDMODVU | EG |
| IJGVU260 | SDMODVU | EB |
| IJGVU270 | SDMODVU | EB |
| IJGVU292 | SDMODVU | EC |
| IJGVU290 | SDMODVU | EC |
| IJGVU310 | SDMODVU | EC |
| IJGVU330 | SDMODVU | EC |
| IJGVU340 | SDMODVU | EC |
| IJGVU360 | SDMODVU | ED |
| IJGVU370 | SDMODVU | ED |
| IJGVU380 | SDMODVU | ED |
| IJGVU300 | SDMODVU | EC |
| IJGVU415 | SDMODVU | EG |
| IJGVU410 | SDMODVU | EG |
| IJGVU420 | SDMODVU | EG |
| IJGVU430 | SDMODVU | EG |
| IJGVU442 | SDMODVU | EL |
| IJGVU444 | SDMODVU | EL |
| IJGVU440 | SDMODVU | EG |
| IJGVU450 | SDMODVU | EL |
| IJGVU460 | SDMODVU | EM |
| IJGVU470 | SDMODVU | EL |
| IJGVU490 | SDMODVU | ED |
| IJGVU400 | SDMODVU | EG |
| IJGVU400 | SDMODVU | EG |
| IJGVU510 | SDMODVU | ED |
| IJGVU520 | SDMODVU | EN |
| IJGVUHFT | SDMODVU | EA |
| IJGVUINM | SDMODVU | EL |
| IJGVUOPT | SDMODVU | DY |
| IJGVUOPW | SDMODVU | DY |
| IJGVUOPW | SDMODVU | EA |
| IJGVURD | SDMODVU | DZ |
| IJGVURD | SDMODVU | EA |
| IJGVURET | SDMODVU | DZ |

| | | |
|---|---|---|
| IJGVURRH | SDMODVU | EN |
| IJGVURTR | SDMODVU | DZ |
| IJGVURTR | SDMODVU | EA |
| IJGVU020 | CNTRL | EP |
| IJGVU030 | SDMODVU | DV |
| IJGVU030 | SDMODVU | DV |
| IJGVU040 | SDMODVU | DV |
| IJGVU050 | SDMODVU | DV |
| IJGVU065 | SDMODVU | DV |
| IJGVU075 | SDMODVU | DW |
| IJGVU070 | SDMODVU | DW |
| IJGVU090 | SDMODVU | DW |
| IJGVU0P | SDMODVU | EL |
| IJGVU0P | SDMODVU | EL |
| IJGVUSER | SDMODVU | DY |
| IJGVUSER | SDMODVU | EA |
| IJGVUSKB | SDMODVU | EA |
| IJGVUUNR | SDMODVU | DY |
| IJGVUWLR | SDMODVU | DY |
| IJGVUWT | SDMODVU | EL |
| IJGWBLD | SDMODW | GG |
| IJGWCHEK | SDMODW | GF |
| IJGWCKRD | SDMODW | GF |
| IJGWCLOS | SDMODW | GE |
| IJGWCOMP | SDMODW | GG |
| IJGWCTL | SDMODW | GM |
| IJGWDECR | SDMODW | GC |
| IJGWETRY | SDMODW | GD |
| IJGWFOPN | SDMODW | GL |
| IJGWFRE | SDMODW | GG |
| IJGWFREE | SDMODW | GM |
| IJGWFTST | SDMODW | GK |
| IJGWHOLD | SDMODW | GB |
| IJGWIGN | SDMODW | GJ |
| IJGWINST | SDMODW | GC |
| IJGWLCTY | SDMODW | GL |
| IJGWLOAD | SDMODW | GC |
| IJGWLREG | SDMODW | GE |
| IJGWNOTE | SDMODW | GL |
| IJGWOPT | SDMODW | GF |
| IJGWOPT | SDMODW | GF |
| IJGWPASS | SDMODW | GL |
| IJGWPNT | SDMODW | GL |
| IJGWPNTS | SDMODW | GM |
| IJGWRDSK | SDMODW | GG |
| IJGWREAD | SDMODW | GB |
| IJGWRW | SDMODW | GE |
| IJGWSLD | SDMODW | GD |
| IJGWSOFF | SDMODW | GD |
| IJGWSSET | SDMODW | GC |
| IJGWSSTR | SDMODW | GD |
| IJGWSTAR | SDMODW | GB |
| IJGWSTCH | SDMODW | GD |
| IJGWSVC0 | SDMODW | GB |
| IJGWTEST | SDMODW | GC |
| IJGWTLIM | SDMODW | GL |
| IJGWUND | SDMODW | GE |
| IJGWUNIO | SDMODW | GF |
| IJGWUNRC | SDMODW | GG |
| IJGWUPOF | SDMODW | GK |
| IJGWUSER | SDMODW | GH |
| IJGWWRTE | SDMODW | GC |
| IJGWWTER | SDMODW | GF |
| IJGWWTER | SDMODW | GF |
| IJGWWTER | SDMODW | GF |
| IJGWWTER | SDMODW | GF |
| IJGWWXIT | SDMODW | GC |
| IJHIHRE2 | DAMOD | NF |
| IJIAE11 | DAMODV | NW |
| IJIAFG | DAMODV | NX |
| IJIAFRD | DAMODV | NU |
| IJIAFS | DAMODV | NW |
| IJIAFT | DAMOD | NB |
| IJIAFT2 | DAMODV | NW |
| IJIAFTA | DAMODV | NW |
| IJIAFX | DAMODV | NW |
| IJIANT | DAMODV | NV |
| IJIANT1 | DAMODV | NV |
| IJIANV | DAMODV | NV |
| IJIANVL | DAMODV | NX |
| IJIARD | DAMODV | NV |
| IJIAWR | DAMODV | NW |
| IJICMC | DAMOD | NK |
| IJICTL | DAMOD | NH |
| IJICTL1 | DAMOD | NH |
| IJIDIC | DAMOD | NG |
| IJIDUMY | DAMOD | ND |
| IJIDV2 | DAMOD | NK |
| IJIDV3 | DAMOD | NK |
| IJIEOF | DAMOD | NA |
| IJIEOF1 | DAMOD | NE |
| IJIEOV | DAMOD | ND |
| IJIFGC | DAMOD | NJ |
| IJIFIN | DAMOD | NC |
| IJIFREE | DAMOD | NH |
| IJIFRM | DAMOD | ND |
| IJIFXL | DAMOD | NA |
| IJIGCH | DAMODV | PC |
| IJIGET | DAMODV | PC |
| IJIGET1 | DAMODV | PC |
| IJIGNXT | DAMODV | PC |
| IJIGSKB | DAMODV | PC |
| IJIGSKB1 | DAMODV | PC |
| IJIGSKE | DAMODV | PC |
| IJIGSKM | DAMODV | PC |
| IJIGTTT | DAMODV | PC |
| IJIJCK | DAMOD | NF |
| IJILADR | DAMOD | NE |
| IJILBK | DAMOD | NK |
| IJILCH | DAMOD | NC |
| IJILOAD | DAMOD | NJ |
| IJIMXK | DAMOD | NA |
| IJINCD | DAMOD | NK |
| IJINCT | DAMOD | NG |
| IJINEF | DAMOD | ND |
| IJINFIT | DAMODV | NV |
| IJINID | DAMOD | NF |
| IJINRF | DAMODV | NT |
| IJINRM | DAMOD | NB |
| IJINRMU | DAMODV | NT |
| IJINRTO | DAMOD | ND |
| IJIOV2 | DAMOD | NJ |
| IJIOVCH | DAMODV | PE |
| IJIOVH | DAMODV | PE |
| IJIOVP | DAMOD | NJ |
| IJIPRI | DAMOD | NC |
| IJIPSI | DAMOD | NB |

| | | |
|---|---|---|
| IJIREAD | DAMOD | NE |
| IJIREAD | DAMODV | PA |
| IJIRND | DAMOD | NB |
| IJIRON | DAMOD | NB |
| IJIRTER | DAMOD | ND |
| IJIRZO | DAMOD | NB |
| IJISAFIT | DAMODV | NU |
| IJISAFT | DAMODV | NU |
| IJISANUL | DAMODV | NM |
| IJISASR0 | DAMODV | NV |
| IJISBK | DAMOD | NB |
| IJISCMC | DAMODV | PD |
| IJISCTL | DAMODV | NY |
| IJISCTL1 | DAMODV | NY |
| IJISCTL1 | DAMODV | NY |
| IJISDOM | DAMODV | NN |
| IJISDV3 | DAMODV | PE |
| IJISEOV | DAMODV | NZ |
| IJISERF | DAMODV | NT |
| IJISFGC | DAMODV | PE |
| IJISFIN | DAMODV | NU |
| IJISFREE | DAMODV | NY |
| IJISFRM | DAMODV | NZ |
| IJISGDL | DAMODV | NM |
| IJISGS | DAMODV | NS |
| IJISICL | DAMODV | NZ |
| IJISING | DAMODV | NT |
| IJISJCK | DAMODV | PB |
| IJISKI | DAMODV | NR |
| IJISKI1 | DAMODV | NR |
| IJISKID | DAMODV | NR |
| IJISKIF | DAMODV | NR |
| IJISKIF1 | DAMODV | NR |
| IJISKIR | DAMODV | NR |
| IJISKS | DAMODV | NR |
| IJISKS1 | DAMODV | NR |
| IJISLBK | DAMODV | PD |
| IJISLOAD | DAMODV | PE |
| IJISMID | DAMODV | PA |
| IJISNEF | DAMODV | PA |
| IJISNF | DAMODV | NQ |
| IJISNID1 | DAMODV | PA |
| IJISNL | DAMODV | NP |
| IJISNL1 | DAMODV | NP |
| IJISNRM | DAMODV | NT |
| IJISNRTO | DAMODV | NZ |
| IJISOV1 | DAMODV | PD |
| IJISOV2 | DAMODV | PE |
| IJISOV2 | DAMODV | PE |
| IJISOV2&6 | DAMODV | PE |
| IJISOVP | DAMODV | PD |
| IJISPW | DAMODV | NP |
| IJISRFD | DAMODV | NN |
| IJISRON | DAMODV | NN |
| IJISRTER | DAMODV | PB |
| IJISSF | DAMODV | NP |
| IJISSTO | DAMODV | NM |
| IJISSTO | DAMODV | NM |
| IJISSTRT | DAMODV | NM |
| IJISSVCF | DAMODV | PE |
| IJISSWL | DAMODV | NM |
| IJISTDL | DAMODV | NS |
| IJISTI | DAMODV | NP |
| IJISTK | DAMODV | NP |
| IJISTKW | DAMODV | NQ |
| IJISTNR | DAMODV | PB |
| IJISTO | DAMOD | NA |
| IJISTO | DAMOD | NA |
| IJISTRT | DAMOD | NA |
| IJISTT | DAMODV | NP |
| IJISUBRS | DAMOD | NC |
| IJISUID | DAMODV | PA |
| IJISVR | DAMODV | NS |
| IJISVR1 | DAMODV | NS |
| IJISVT | DAMODV | NP |
| IJISVT1 | DAMODV | NQ |
| IJISVW | DAMODV | NQ |
| IJISWC | DAMODV | NQ |
| IJISWLL | DAMODV | NT |
| IJISWND | DAMODV | NT |
| IJISWSB | DAMODV | NN |
| IJISWTM | DAMODV | NZ |
| IJISXTFD | DAMODV | NZ |
| IJISYID | DAMODV | PA |
| IJISZER | DAMODV | NU |
| IJITNR | DAMOD | NF |
| IJITOM | DAMOD | NF |
| IJITRHLD | DAMOD | NB |
| IJIUID | DAMOD | NF |
| IJIWAFT | DAMODV | NW |
| IJIWFTR | DAMODV | PA |
| IJIWLR | DAMODV | NT |
| IJIWND | DAMOD | NB |
| IJIWND | DAMOD | NH |
| IJIWRT | DAMODV | NM |
| IJIWRU | DAMOD | NA |
| IJIWTM | DAMOD | ND |
| IJIXNF | DAMOD | NK |
| IJIXTFD | DAMOD | ND |
| IJIYID | DAMOD | ND |
| IJIZER | DAMOD | NB |
| IJOPEN | $$BOSDO4 | JL |
| IJUP330 | SDMODUO | EZ |
| IJWUPDT | SDMODW | GC |
| ILLEGAL | $$BOMSG2 | AS |
| INCORE | $$BOMSG2 | AQ |
| INCREASE | $$BOSDI2 | HG |
| INCREMEN | $$BOSDO3 | JH |
| INCREMNT | $$BODAIN | PH |
| INCRKEY | $$BODACL | RH |
| INCRKEY | $$BOSDI3 | HJ |
| INCRMT | $$BODAO2 | QB |
| INCVAR | $$BODAO1 | PQ |
| INITBS | $$BOSD01 | HB |
| INITCCB | $$BOSDO1 | JB |
| INITDTF | $$BOSDI4 | HL |
| INITDTF | $$BOSDO4 | JL |
| INITDTF | $$BOSDO5 | JP |
| INITDUMP | $$BOMSG2 | AR |
| INITPSW | $$BOSDI1 | HC |
| INITREG | $$BOMSG1 | AP |
| INITSEEK | $$BOSDO1 | JB |
| INITUTLO | $$BOSDO6 | JR |
| INPULB | $$BODAU1 | RC |
| INPUT | $$BODAIN | PG |
| INPUT | $$BODAU1 | RA |
| INPUTFLE | $$BOSDC1 | LA |
| INRANGE | $$BOSDI2 | HG |

| | | |
|---|---|---|
| INSERT | $$BODAU1 | RD |
| INSERTX | $$BOSDO5 | JP |
| INSERTX | $$BOSDW2 | KH |
| ISAMTYPE | $$BOFLPT | AA |
| JCTEXIT | $$BOMSG2 | AR |
| JJISSEDF | DAMODV | NM |
| LABADR | $$BODAIN | PH |
| LABELSW | $$BODACL | RH |
| LABELSW | $$BOSDO6 | JQ |
| LAF1XTNT | $$BOSDI2 | HG |
| LASTIND | $$BOSDO5 | JN |
| LASTIND | $$BOSDW2 | KG |
| LASTLINE | $$BODSPW | AG |
| LASTSW | $$BOSDO1 | JB |
| LASTVOL | $$BOSDI1 | HC |
| LASTXTNT | $$BOSDO5 | JP |
| LASTXTNT | $$BOSDW1 | KD |
| LASTXTNT | $$BOSDW2 | KH |
| LDLIM | $$BODAO1 | PQ |
| LIMRCH | $$BODAO2 | QA |
| LOADALTR | $$BODAI1 | PJ |
| LOADEXIT | $$BOSDI3 | HK |
| LOADEXIT | $$BOSDO4 | JM |
| LOADEXIT | $$BOSDO5 | JP |
| LOADEXIT | $$BOSDO6 | JR |
| LOADINP | $$BOSD00 | HA |
| LOADIO | $$BOSDEV | LG |
| LOADMSG | $$BOSDI1 | HD |
| LOADMSG | $$BOSDO1 | JB |
| LOADMSG | $$BOSDW1 | KE |
| LOADONE | $$BODACL | RF |
| LOADPHAS | $$BOMSG2 | AQ |
| LOADREG | $$BOMSG1 | AP |
| LOADREG0 | $$BOSDO6 | JR |
| LOADUSER | $$BODACL | RG |
| LOADUSER | $$BOSDO6 | JQ |
| LOADWORK | $$BOMSG1 | AN |
| LOADWORK | $$BOSDI1 | HC |
| LOADWORK | $$BOSDO7 | JS |
| LOOP | $$BODSMW | AT |
| LUBXXX | $$BOMSG1 | AN |
| MARSWITCH | $$BOSDO3 | JJ |
| MAXOUT | $$BODAU1 | RB |
| MESSG17 | $$BOSDI3 | HJ |
| MESSG47 | $$BOSDI1 | HC |
| MESSG5 | $$BOMSG2 | AQ |
| MODCCWLG | $$BODSPW | AF |
| MODIFY3 | $$BOSDI1 | HC |
| MODRESP | $$BOMSG2 | AQ |
| MONORTN | $$BODQUE | LE |
| MONORTN | $$BOFLPT | AC |
| MORXTNT | $$BODAI1 | PK |
| MORXTNT | $$BODAIN | PG |
| MORXTNTS | $$BODSPW | AG |
| MORXTNTS | $$BOSDI1 | HC |
| MOVE | $$BODSPW | AG |
| MOVEADDR | $$BOSDW3 | KJ |
| MOVECCW | $$BOSDI2 | HG |
| MOVECCW | $$BOSDW1 | KD |
| MOVECCW9 | $$BOSDW1 | KE |
| MOVEDATA | $$BODACL | RG |
| MOVELL | $$BODAO2 | QB |
| MOVEMSG | $$BOMSG2 | AQ |
| MOVEMSG1 | $$BODSPV | AE |
| MOVENAME | $$BODSMW | AT |
| MOVENAME | $$BOFLPT | AC |
| MOVEPT | $$BODAI1 | PJ |
| MOVERCM | $$BODSPW | AG |
| MOVERCM | $$BOSDO3 | JG |
| MOVERCM | $$BOSDO8 | KA |
| MOVESYM | $$BOSDI1 | HD |
| MOVESYM | $$BOSDW1 | KE |
| MOVEUNIT | $$BOSDC1 | LB |
| MOVEUNIT | $$BOSDI1 | HD |
| MOVEUNIT | $$BOSDW1 | KE |
| MOVEUTL0 | $$BOSDO6 | JQ |
| MOVLOLIM | $$BOSDO4 | JL |
| MOVSYM | $$BOSD01 | HB |
| MOVSYM | $$BOSD01 | HB |
| MSG2 | $$BODAI1 | PJ |
| MSG2 | $$BODAIN | PF |
| MSG2 | $$BODAO1 | PQ |
| MSGEXIT | $$BOSDO3 | JG |
| MSGPHAS1 | $$BOSDI1 | HE |
| MSGPHAS1 | $$BOSDO1 | JC |
| MSGPHAS1 | $$BOSDW1 | KF |
| MSGRCUT2 | $$BOMSG2 | AQ |
| MSGRET | $$BODAU1 | RC |
| MSGRITE | $$BOSDO3 | JK |
| MSGROUT1 | $$BOMSG1 | AN |
| MSGRTN | $$BODAI1 | PL |
| MSGRTN | $$BODAIN | PH |
| MSGRTN | $$BODAO1 | PR |
| MSG05SW | $$BOMSG2 | AS |
| MULT4 | $$BOSDEV | LG |
| MVEXT | $$BODAIN | PF |
| MVPNTR | $$BODAO1 | PN |
| NDRUTINE | $$BOFLPT | AC |
| NEWLIMIT | $$BOSDI2 | HF |
| NEWVOLUM | $$BOSDO1 | JB |
| NEXT | $$BODSPW | AF |
| NEXT1 | $$BODSMW | AU |
| NEXT2 | $$BODSMW | AU |
| NEXTADR | $$BOWDMP | AL |
| NEXTLBL | $$BOSDO3 | JG |
| NEXTLBL | $$BOSDO8 | KA |
| NEXTPHAS | $$BODSPW | AH |
| NEXTPHAS | $$BOSDI1 | HE |
| NEXTPHAS | $$BOSDI2 | HG |
| NEXTPHAS | $$BOSDO1 | JC |
| NEXTPHAS | $$BOSDO3 | JG |
| NEXTPHAS | $$BOSDO6 | JR |
| NEXTPHAS | $$BOSDW1 | KF |
| NEXTREAD | $$BOSDO3 | JK |
| NEXTTEST | $$BOMSG2 | AS |
| NEXTVOL | $$BOSDO1 | JA |
| NEXTXTNT | $$BOSDI1 | HD |
| NEXTXTNT | $$BOSDO1 | JB |
| NOF1 | $$BODAU1 | RE |
| NOF1LB | $$BODAI1 | PL |
| NOF3 | $$BODAI1 | PL |
| NOF3 | $$BODAU1 | RE |
| NOF4 | $$BODAO1 | PR |
| NOF4 | $$BODAO2 | QD |
| NOF4 | $$BODAO3 | QF |
| NOHOLD | $$BOSDC2 | LD |
| NOMATCH | $$BOSDW3 | KJ |
| NOMSG | $$BODAO3 | QE |

| | | |
|---|---|---|
| NOMSG | $$BODAO4 | QJ |
| NONSDTY | $$BOFLPT | AA |
| NONXTENT | $$BOFLPT | AA |
| NORELAD | $$BODAIN | PG |
| NOROOM | $$BODAO3 | QH |
| NOSER | $$BODAO1 | PN |
| NOTAP | $$BOSDEV | LF |
| NOTF1 | $$BOWDMP | AM |
| NOTRKHLD | $$BOSDC1 | LC |
| NOTRKHLD | $$BOSDI3 | HK |
| NOUHL | $$BODAU1 | RE |
| NOVLAB | $$BODAI1 | PL |
| NOVLAB | $$BODAO1 | PR |
| NOVOL1 | $$BODAO2 | QD |
| NOVOL1 | $$BODAO3 | QH |
| NOVOL1 | $$BODAU1 | RE |
| NOXNTSW1 | $$BOSDI2 | HG |
| NOXT | $$BODAIN | PH |
| NOXT | $$BODAO2 | QD |
| NOXT | $$BODAO3 | QH |
| NRFF4 | $$BODAO1 | PR |
| NRFF4 | $$BODAO3 | QF |
| NRF01 | $$BODAO2 | QD |
| NRF09 | $$BODAO3 | QH |
| NRF09 | $$BODAO4 | QL |
| NRFVL | $$BODAO1 | PR |
| NXTEXT | $$BODAO1 | PQ |
| NXTSU | $$BODAO2 | QA |
| ONLYONE | $$BOSDO1 | JA |
| OPENCODE | $$BOSDI3 | HJ |
| OPENCUT8 | $$BOSDO8 | KA |
| OPENFILE | $$BOSDO1 | JA |
| OPENINP1 | $$BOSDI1 | HC |
| OPENINP2 | $$BOSDI2 | HF |
| OPENOUT1 | $$BOSDO1 | JA |
| OPENOUT2 | $$BOSDO2 | JF |
| OPENOUT3 | $$BOSDO3 | JG |
| OPENOUT4 | $$BOSDO4 | JL |
| OPENOUT5 | $$BOSDO5 | JN |
| OPENOUT6 | $$BOSDO6 | JQ |
| OPENOUT7 | $$BOSDO7 | JS |
| OPENWRK1 | $$BOSDW1 | KD |
| OPENWRK2 | $$BOSDW2 | KG |
| OPENWRK3 | $$BOSDW3 | KJ |
| OPN2 | $$BODAI1 | PJ |
| OTHER | $$BOWDMP | AM |
| OUT | $$BODAO4 | QJ |
| OUTPTFL | $$BOSDC1 | LA |
| OUTPUT | $$BOSIGN | JE |
| OUTPUTFL | $$BOSDEV | LF |
| OUTRUTIN | $$BODQUE | LE |
| PACKXTNT | $$BOSDI2 | HG |
| PACKXTNT | $$BOSDO3 | JH |
| PASS | $$BODAU1 | RC |
| PASSXTNT | $$BOSDI1 | HD |
| PAXENTS | $$BOSDW3 | KJ |
| PLUGXT | $$BODAO4 | QK |
| PNTCCB | $$BODAIN | PF |
| POINTERX | $$BOSDW3 | KJ |
| POSTD18 | $$BOSDI1 | HC |
| POSTDIB | $$BOSDO1 | JA |
| POSTEDX | $$BOSDEV | LG |
| POSTLMTS | $$BOSDO4 | JL |
| POSTOPEN | $$BOSDI4 | HL |
| POSTOPEN | $$BOSDW1 | KD |
| POSTXTNT | $$BOSDI2 | HH |
| POSTXTNT | $$BOSDI4 | HL |
| POSTXTNT | $$BOSDW3 | KJ |
| PRBGN | $$BOSD01 | HB |
| PRECONVT | $$BOSD01 | HB |
| PRINTER1 | $$BODSPV | AE |
| PRINTER1 | $$BODSPW | AH |
| PRINTER1 | $$BOVDMP | AK |
| PRINTER1 | $$BOWDMP | AL |
| PROGNAME | $$BODSPV | AE |
| PUT21LL | $$BOFLPT | AB |
| PUT21UL | $$BOFLPT | AB |
| PUTTER2 | $$BOWDMP | AM |
| PVTLIBRY | $$BOSDC1 | LA |
| QTAMFLPT | $$BOSDI4 | HM |
| QTAMOPEN | $$BOSDI4 | HM |
| RDFLAB | $$BODAO2 | QA |
| RDFOR3 | $$BODAI1 | PK |
| RDLABEL | $$BODACL | RG |
| RDY | DAMOD | NL |
| READ | $$BODAI1 | PJ |
| READDISK | $$BODSPW | AH |
| READDISK | $$BOSDC1 | LC |
| READDISK | $$BOSDI1 | HE |
| READDISK | $$BOSDI2 | HH |
| READDISK | $$BOSDO1 | JD |
| READDISK | $$BOSDW2 | KH |
| READDISK | $$BOSDW3 | KK |
| READDISK | $$BOVDMP | AK |
| READDISK | $$BOWDMP | AL |
| READDISK | SD | KB |
| READDISK | SD | KC |
| READF1 | $$BODAU1 | RA |
| READF3 | $$BODSPW | AG |
| READFMT4 | $$BOVDMP | AJ |
| READMSG | $$BODSPV | AE |
| READMSG | $$BOMSG2 | AR |
| READUL | $$BODAU1 | RC |
| READVOL1 | $$BOSDI1 | HD |
| READVOL1 | $$BOSDO2 | JF |
| READXTNT | $$BOSDO1 | JB |
| RECMOV | $$BODAO1 | PR |
| RECMOV | $$BODAO2 | QC |
| REDUCE | $$BODSMW | AT |
| REENTRY | $$BOSDI2 | HF |
| REENTRY | $$BOSDI3 | HJ |
| REGSAVE | $$BOSDEV | LF |
| RELOAD8 | $$BODSPW | AG |
| RELOCATE | $$BOVDMP | AJ |
| REOPEN1 | $$BOSDC1 | LA |
| REOPEN1 | $$BOSDC1 | LB |
| REREAD | $$BODSPV | AE |
| REREAD1 | $$BOSD01 | HB |
| REREADF1 | $$BOSDW2 | KG |
| RESET | $$BOFLPT | AC |
| RESET9 | $$BOSDC1 | LA |
| RESETCP | $$BOSD00 | HA |
| RESETEOF | $$BOSDI1 | HC |
| RESETIND | $$BOSDI3 | HK |
| RESETIND | $$BOSDO1 | JC |
| RESETLBL | $$BOSDO6 | JQ |
| RESETMON | $$BOSDI1 | HE |
| RESETPNT | $$BOSDW3 | KJ |

| | | |
|---|---|---|
| RESTORE | $$BODQUE | LE |
| RESTORE | $$BODSMW | AT |
| RESTORE | $$BOFLPT | AC |
| RETMON | $$BOSDI1 | HC |
| RETMON | $$BOSDO1 | JA |
| RETURN | $$BOVDMP | AJ |
| RETURN | $$BOWDMP | AL |
| RETURNF3 | $$BODSPW | AG |
| REVTOC | $$BODAO2 | QD |
| RTNF3 | $$BODAI1 | PJ |
| RTNMON | $$BODACL | RG |
| SAVDIB | $$BOSD00 | HA |
| SAVE | $$BODSMW | AT |
| SAVECON | $$BOSDEV | LF |
| SAVEF1 | $$BOSDO1 | JB |
| SAVEFX | $$BODAO1 | PN |
| SAVELIMIT | $$BOSDW1 | KF |
| SAVESEQ | $$BOSDI2 | HF |
| SAVESEQNO | $$BOSDO1 | JB |
| SAVESYS | $$BODSPV | AE |
| SAVEUNIT | $$BOSDW1 | KE |
| SAVFRX | $$BODAO2 | QA |
| SAVLIMIT | $$BOSDO2 | JF |
| SCROPN | $$BODAO2 | QC |
| SCRTCH | $$BODAO2 | QC |
| SD | $$BOSD01 | HB |
| SEARCH | SD | KB |
| SECDOVLP | $$BOMSG2 | AQ |
| SEQERR | $$BODAO4 | QK |
| SEROK | $$BODAO3 | QE |
| SETEXIT | $$BOSDO4 | JM |
| SETIND | $$BOSD01 | HB |
| SETINTSW | $$BOMSG2 | AR |
| SETLAST | $$BOSDO1 | JA |
| SETON | $$BOFLPT | AA |
| SETRECD | $$BOSDI4 | HL |
| SETSW1 | $$BOSDC1 | LB |
| SHIFTMSG | $$BOMSG1 | AP |
| SKIP16 | $$BOFLPT | AA |
| SKIP24 | $$BOFLPT | AA |
| SKIP28 | $$BOFLPT | AB |
| SKIP29 | $$BOFLPT | AB |
| SKIP20 | $$BOFLPT | AA |
| SKIP32 | $$BOFLPT | AB |
| SKIP36 | $$BOFLPT | AB |
| SKIP30 | $$BOFLPT | AB |
| SKIP40 | $$BOFLPT | AB |
| SKIP90 | $$BOFLPT | AD |
| SKIP04 | $$BODQUE | LE |
| SKIP04 | $$BOFLPT | AA |
| SKIP06 | $$BOFLPT | AA |
| SKIP08 | $$BOFLPT | AA |
| SPLITCYL | $$BOSDO3 | JH |
| SQEZJIB | $$BOFLPT | AD |
| STARTFRE | $$BOSDC2 | LD |
| STDTF | $$BODAIN | PF |
| STORCT | $$FODAO3 | QE |
| STORE | $$BODAIN | PH |
| STORSU | $$BODAI1 | PJ |
| STOWN | $$BODAU1 | RB |
| STRFXT | $$BODAO3 | QE |
| SUMROUT | $$BOMSG2 | AS |
| SUMROUT | SD | KC |
| SVF1AD | $$BODAO2 | QB |
| SYSPRINT | $$BODSPV | AE |
| SYSTEMSW | $$BOSDO8 | KA |
| TESTC2 | $$BODAO2 | QB |
| TESTCCB | $$BOSDI1 | HC |
| TESTDTF | $$BOSDW1 | KE |
| TESTEOB | $$BODSMW | AU |
| TESTEOF | $$BODACL | RG |
| TESTEOF | $$BODSMW | AU |
| TESTEOF | $$BOSDI3 | HJ |
| TESTEOV | $$BOSDO1 | JB |
| TESTIPT | $$BOSDEV | LF |
| TESTLAST | $$BOSDI1 | HD |
| TESTNEXT | $$BOSDO7 | JS |
| TESTOPEN | $$BOSDI1 | HC |
| TESTOPEN | $$BOSDO3 | JG |
| TESTPS | $$BODAI1 | PL |
| TESTPS | $$BODAO4 | QL |
| TESTPTR | $$BODQUE | LE |
| TESTPTR | $$BOFLPT | AA |
| TESTSW | $$BOSDI1 | HC |
| TESTSYM | $$BOSDI1 | HD |
| TESTWORK | $$BOSD00 | HA |
| TESTXT | $$BODAO2 | QB |
| TIC | DAMOD | NL |
| TOMANY | $$BODAO3 | QE |
| TRAILER | $$BOSDO1 | JC |
| TRKLOOP | $$BOSDC2 | LD |
| TST2314 | $$BODAIN | PF |
| TST2321 | $$BODAIN | PF |
| TSTDVCTP | $$BODAIN | PG |
| TSTLIM | $$BOSD00 | HA |
| TSTMONT | $$BOFLPT | AC |
| TSTOUT | $$BODAIN | PH |
| TSTPOINT | $$BOSDO3 | JH |
| TSTPRIME | $$BOSDI2 | HG |
| TSTRDLB | $$BODACL | RH |
| TSTRPY | $$BODAO2 | QC |
| TSTVAR | $$BODAO2 | QB |
| TW311 | $$BODAO2 | QB |
| TWTY1 | $$BODAO4 | QJ |
| TWTY11 | $$BODAO4 | QJ |
| TYPEASW | $$BOMSG2 | AS |
| TYPEEOB | $$BOMSG2 | AR |
| TYPEMSG | $$BOSDI2 | HG |
| TYPEMSG | $$BOSDO3 | JK |
| TYPEMSG1 | $$BOSDO7 | JS |
| TYPERROR | $$BOSDO7 | JS |
| ULAB | $$BODAU1 | RA |
| UNDEF | $$BOSDO1 | JB |
| UNEXDEF | $$BOSDC1 | LC |
| UNPACK | $$BOWDMP | AM |
| UNPACK3 | $$BODSPW | AG |
| UPPERLT | $$BODAI1 | PK |
| UPPERLT | $$BODAIN | PH |
| UPVSEQ | $$BODAO4 | QK |
| USEREXIT | $$BOSDC1 | LC |
| USEREXIT | $$BOSDEV | LF |
| USEREXIT | $$BOSDEV | LG |
| USEREXIT | $$BOSDW3 | KJ |
| USERLBLS | $$BOSDO1 | JC |
| USERXTNT | $$BOSDI2 | HF |
| VERIFY | $$BOSDO1 | JB |
| VERIFY | $$BOSDW1 | KE |
| VERIFY | $$BOSIGN | JE |

| | | |
|---|---|---|
| VERIFYCP | $$BOSDO4 | JM |
| VIA1052 | $$BOSDO4 | JM |
| VIA1052 | $$BOSDO5 | JP |
| VTOCADDR | $$BOSDI1 | HE |
| VTOCADDR | $$BOSDO2 | JF |
| VTOCADDR | $$BOSDW1 | KF |
| VTOCDSPZ | $$BODSPW | AF |
| WORKCLOS | $$BOSDC1 | LB |
| WORKFILE | $$BOSDC1 | LA |
| WRI | DAMOD | NL |
| WRITDISK | $$BOSDC1 | LC |
| WRITDISK | $$BOSDO6 | JR |
| WRITDISK | $$BOSDW2 | KH |
| WRITDISK | SD | KB |
| WRITDISK | SD | KC |
| WRITE | $$BODAO3 | QG |
| WRITE | $$BODAO4 | QL |
| WRITE | $$BODAU1 | RE |
| WRITE | $$BODSMW | AT |
| WRITEF1 | $$BOSDC1 | LB |
| WRITELBL | $$BODACL | RH |
| WRITELBL | $$BOSDO6 | JQ |
| WRITEMSG | $$BOMSG2 | AQ |
| WRLABL | $$BODAO4 | QK |
| WRLAST | $$BODAU1 | RB |
| WRNGPKSW | $$BOMSG2 | AS |
| WRONGPAK | $$BOSDO2 | JF |
| XTNTCONV | $$BOSD01 | HB |
| XTNTOK | $$BOSD01 | HB |
| XTNTOKAY | $$BOSDO2 | JF |

For explanations and actions to be taken for the various messages, refer to DOS Messages, GC24-5074.

Note: The second digit of the message number indicates the type of DASD file issuing the message. The file types are:

3 = Sequential DASD - Open input

4 = Sequential DASD - Open output

5 = Sequential DASD - Close

6 = Direct Access - Input

7 = Direct Access - Output

8 = Common Open/Close routines

9 = Sequential DASD - Work file

V = VTOC Display routines

| Message Number | Phase | Chart | Message |
|---|---|---|---|
| 4400I | | KB | NO LABEL SPACE IN VTOC |
| 4700I | $$BODAO3 | QH | or |
| 4900I | | KB | NO RECORD FOUND |
| 4301I | $$BOSDI2 | HF | NO FORMAT 1 LABEL |
| 4401I | $$BOSDO4<br>$$BOSDO5 | KB<br>JM<br>JN | or<br>NO RECORD FOUND |
| 4501I | $$BOSDC1 | LB | |
| 4601I | $$BODAI1<br>$$BODAU1 | PL<br>RE | |
| 4701I | $$BODAO2<br>$$BODAU1 | QD<br>RE | |
| 4901I | $$BOSDW2<br>$$BOSDW3 | KB<br>KG<br>KJ | |
| 4303I | $$BOSDI2 | HG | NO FORMAT 3 LABEL FOUND |
| 4403I | $$BOSDO5 | JN | |
| 4503I | $$BOSDC1 | LB | |
| 4603I | $$BODAI1<br>$$BODAU1 | PL<br>RE | |
| 4703I | $$BODAU1 | RE | |
| 4903I | $$BOSDW2<br>$$BOSDW3 | KG<br>KJ | |

Figure 40. Message Cross-Reference List (Part 1 of 5)

| Message Number | Phase | Chart | Message |
|---|---|---|---|
| 4304I | $$BOSDI1 | HE | NO FORMAT 4 LBL IN VTOC<br>or<br>NO RECORD FOUND |
| 4404I | $$BOSDO2 | JE | |
| 4704I | $$BODAO1<br>$$BODAO2<br>$$BODAO3 | PR<br>QD<br>QF | |
| 4904I | $$BOSDW1 | KF | |
| 4V04I | $$BODSPW<br>$$BOVDMP | AF<br>AJ | |
| 4306I | $$BOSDI1 | FL | NO STANDARD VOL1 LABEL<br>or<br>NO RECORD FOUND |
| 4406I | $$BOSDO2 | JE | |
| 4506I | $$BOSDC1 | LB | |
| 4606I | $$BODAI1<br>$$BODAU1 | PL<br>RE | |
| 4706I | $$BODAO1<br>$$BODAO2<br>$$BODAO3<br>$$BODAU1 | PR<br>QD<br>QH<br>RE | |
| 4906I | $$BOSDW1 | KE | |
| 4V06I | $$BODSPW<br>$$BOVDMP | AF<br>AJ | |
| 4307I | $$BOSDI1 | HD | NO RECORD FOUND |
| 4407I | $$BOSDO1 | JB | |
| 4907I | $$BOSDW1 | KD | |
| 4308D | $$BOSDI3<br>$$BODACL | HJ<br>RG | NO RECORD FOUND<br>or<br>NO UTL0 FILE MARK FOUND |
| 4408D | $$BOSDO6 | JQ | |
| 4608D | $$BODACL | RG | |
| 4708D | $$BODACL | RF | |

Figure 40. Message Cross-Reference List (Part 2 of 5)

| Message Number | Phase | Chart | Message |
|---|---|---|---|
| 4409I | $$BOSDO3 | JG | NO RECORD FOUND |
| | | JH | |
| | | JK | |
| | $$BOSDO8 | KA | |
| 4709I | $$BODAO2 | QB | |
| | $$BODAO3 | QH | |
| | $$BODAO4 | QL | |
| | $$BOSDO8 | KA | |
| 4909I | $$BOSDO3 | JG | |
| | | JH | |
| | | JK | |
| | $$BOSDO8 | KA | |
| 4V09I | $$BODSPW | AG | |
| | $$BOVDMP | AL | |
| 4330D | $$BOSDI2 | JD | FMT1-DLAB UNEQUAL |
| 4331D | $$BOSDI2 | HF | VOLUME SEQUENCE ERROR |
| 4433A | $$BOSDO8 | KA | EQUAL FILE ID IN VTOC |
| 4733A | $$BOSDO8 | KA | |
| 4933A | $$BOSDO8 | KA | |
| 4434I | $$BOSDO5 | JN | CURRENT FILE LBL DELETED |
| 4935I | $$BOSDW2 | KG | DELETED WORKFILE LABEL |
| | $$BOSDW3 | KJ | |
| 4936I | $$BOSDW3 | KJ | NO MORE AVAIL/MATCH EXTENT |
| 4338D | $$BOSDI3 | HJ | USER HDR LBL IS NOT STD |
| 4638D | $$BODAU1 | RE | |
| 4639D | $$BODACL | RG | USER TRL LBL IS NOT STD |
| 4440A | $$BOSDO3 | JK | EXTENT OVERLAP ON ANOTHER |
| 4740A | $$BODAO1 | PQ | |
| 4940A | $$BOSDO3 | JK | |
| 4441A | $$BOSDO2 | JF | EXTENT OVERLAP ON VTOC |
| 4741A | $$BODAO1 | PP | |
| 4941A | $$BOSDW1 | KF | |
| 4342A | $$BOSDI2 | HG | NO MATCHING EXTENT |
| 4243I | $$BOSDO8 | KA | INVALID EXTENT HI/LO LIMIT |

Figure 40. Message Cross-Reference List (Part 3 of 5)

| Message Number | Phase | Chart | Message |
|---|---|---|---|
| 4444A | $$BOSDO3 | JK | OVERLAP ON UNEXPRD FILE |
| 4744A | $$BODAO2 | QC | |
| 4944A | $$BOSDO3 | JK | |
| 4445I | $$BOSD01 | JA | TOO MANY EXTENTS |
| 4745I | $$BODAO3 | QE | |
| 4947A | $$BOSDW1 | KD | EXTENTS NOT ON SAME UNIT |
| 4348I | $$BOSD00 | HA | SYSIN/SYSOUT UNSUPPORTED |
| 4448I | $$BOSD00 | HA | |
| 4450A | $$BOSDO7 | JS | NO MORE AVAILABLE EXTENTS |
| 4651I | $$BODAI1 | PK | SYSUNITS NOT IN SEQUENCE |
| 4751I | $$BODAO4 | QK | |
| 4355A | $$BOSDI1 | HD | WRONG PACK, MOUNT nnnnnn |
| 4455A | $$BOSDO2 | JF | |
| 4655A | $$BODAI1 | PJ | |
| 4755A | $$BODAO1 | PN | |
| 4955A | $$BOSDW1 | KE | |
| 4358I | $$BOSD01 | HB | NO EXTENT FOR OUTPUT FILE |
| 4458I | $$BOSD01 | HB | |
| 4359I | $$BOSD01 | HB | INVALID EXTENT |
| 4459I | $$BOSD01 | HB | |

Figure 40. Message Cross-Reference List (Part 4 of 5)

| Message Number | Phase | Chart | Message |
|---|---|---|---|
| 4360I | $$BOSDI1 | HC | NO EXTENTS, ALL BYPASSED |
| 4460I | $$BOSDO1 | JA | |
| 4760I | $$BODAIN<br>$$BODAO2<br>$$BODAO3 | PH<br>QD<br>QH | |
| 4960I | $$BOSDW1 | KD | |
| 4361I | $$BOSD01 | HB | INVALID DLBL FUNCTION |
| 4461I | $$BOSD01 | HB | |
| 4366A | $$BOSDI2 | HF | 1 TRACK USER LBL EXTENT |
| 4466A | $$BOSDO1 | JC | |
| 4766A | $$BODAO1 | PQ | |
| 4477A | $$BOSDO7 | JS | EXTENT ENTRY ERROR-RETRY |
| 4383I | $$BOSDI1 | HD | INVALID LOGICAL UNIT |
| 4483I | $$BOSDO1 | JB | |
| 4683I | $$BODAIN | PF | |
| 4983I | $$BOSDW1 | KE | |
| 4890I | $$BOFLPT | AC | NO JIBS AVAILABLE |
| 4V95A | $$BODSPV | AE | DISPLAY VTOC ON SYSLOG OR SYSLST |
| 4V96A | $$BODSPV | AE | SYSLST NOT A PRINTER |
| 4497I | $$BOSDO3 | JK | OVLAP EXPIRED SECRD FILE |
| 4797I | $$BODAO2 | QC | |
| 4498I | $$BOSDO3 | JK | OVLAP UNEXPRD SECRD FILE |
| 4798I | $$BODAO2 | QC | |
| 4699D | $$BODAI1 | PJ | DATA SECURED FILE ACCESSED |
| 4399D | $$BOSDI2 | HF | |

Figure 40. Message Cross-Reference List (Part 5 of 5)

## APPENDIX C: SUPERVISOR CALLS (SVC'S)

| Macro Supported | SVC | Function |
|---|---|---|
| EXCP | 0 | Execute channel programs. |
| FETCH | 1<br>2<br>3 | Fetch any phase.<br>Fetch a logical transient (B-transient).<br>Fetch or return from a physical transient (A-transient). |
| LOAD | 4 | Load any phase. |
| MVCOM | 5 | Modify supervisor communications region. |
| CANCEL | 6 | Cancel a problem program or task. |
| WAIT | 7 | Wait for a CCB or TECB. |
|  | 8 | Transfer control to the problem program from a logical transient (B-transient). |
| LBRET | 9 | Return to a logical transient (B-transient) from the problem program after a SVC 8. |
| SETIME | 10* | Set timer interval. |
|  | 11<br>12<br>13 | Return from a logical transient (B-transient).<br>Logical AND (Reset) to second job control byte (displacement 57 in communications region).<br>Logical OR (Set) to second job control byte (displacement 57 in communications region). |
| EOJ | 14 | Cancel job and go to job control for end of job step. |
|  | 15 | Same as SVC 0 except ignored if CHANQ table is full. (Primarily used by ERP). |
| STXIT (PC) | 16* | Provide supervisor with linkage to user's PC routine for program check interrupts. |
| EXIT (PC) | 17* | Return from user's PC routine. |
| STXIT (IT) | 18* | Provide supervisor with linkage to user's IT routine for interval timer interrupts. |
| EXIT (IT) | 19* | Return from user's IT routine. |
| STXIT (OC) | 20* | Provide supervisor with linkage to user's OC routine for external or attention interrupts (operator communications). |
| EXIT (OC) | 21* | Return from user's OC routine. |
|  | 22*<br>23* | The first SVC 22 seizes the system for the issuing program by disabling multiprogram operation. The second SVC 22 releases the system (enables multiprogram operation).<br>Load phase header. Phase load address is stored at user's address. |
| SETIME | 24* | Provide supervisor with linkage to user's TECB and set timer interval. |
|  | 25*<br>26*<br>27* | Issue HALT I/O on a teleprocessing device.<br>Validate address limits.<br>Special HIO on teleprocessing devices. |
| EXIT (MR) | 28* | Return from user's stacker select routine (MICR type devices only). |
|  | 29* | Provide return from multiple wait macros WAITF and WAITM (except MICR type devices). |
| QWAIT | 30* | Wait for a QTAM element. |
| QPOST | 31*<br>32<br>33<br>34 | Post a QTAM element.<br>Reserved.<br>Reserved for internal macro COMRG.<br>Reserved for internal macro GETIME. |

* = optional

Figure 41. Supervisor Calls (Part 1 of 2)

| Macro Supported | SVC | Function |
|---|---|---|
| HOLD | 35* | Hold a track for use by the requesting task only. |
| FREE | 36* | Free a track held by the task issuing the FREE. |
| STXIT (AB) | 37* | Provide supervisor with linkage to user's AB routine for abnormal termination of a task. |
| ATTACH | 38* | Initialize a subtask and establish its priority. |
| DETACH | 39* | Perform normal termination of a subtask. It includes calling the FREE routine to free any tracks held by the subtask. |
| POST | 40* | Inform the system of the termination of an event and ready any waiting tasks. |
| DEQ | 41* | Inform the system that a previously enqueued resource is now available. |
| ENQ | 42* | Prevent tasks from simultaneous manipulation of a shared data area (resource). |
|  | 43* | Provide supervisor support for external creation and updating of SDR records. |
|  | 44* | Provide supervisor support for external creation of OBR records. |
|  | 45* | Provide emulator interface. |
|  | 46* | Provide OLTEP with the facility to operate in supervisory state. |
|  | 47* | Provide return from wait multiple WAITF for MICR type devices. |
|  | 48 | Reserved. |
|  | 49 | Reserved. |
|  | 50 | Reserved for LIOCS error recovery. |

* = optional

Figure 41. Supervisor Calls (Part 2 of 2)

# APPENDIX D: MICROFICHE CROSS-REFERENCE INDEX

The index gives the relationship of core-image phase names, relocatable module names, microfiche labels, and microfiche identification numbers to each other.

An asterisk indicates the microfiche label. If the microfiche label differs from both the phase and the module name, it is so indicated in parentheses.

When a phase or module takes up more than one microfiche card, the identification number of only the first card is shown.

The index includes all the macro names and identification numbers. The position of each macro on the microfiche card is further identified by rows A to E. The rows are not indicated for macros that use more than one microfiche card.

For the complete microfiche cross-reference index, see Introduction to DOS Logic listed in the Preface.

## Logical IOCS Macros

| Macro Name | Row | Card ID |
|---|---|---|
| CHECK | C | 453.002 |
| CLOSE(R) | A,B | 453.003 |
| CNTRL | C | 453.003 |
| DAMOD | A,B | 454.001 |
| DTFDA | A-C | 454.002 |
| DTFPH | C | 453.005 |
| DTFSD | A,B | 455.001 |
| DTFSR | E | 453.005 |
| FREE | B | 453.006.50 |
| GET | E | 453.007 |
| NOTE | B | 453.009 |
| OPEN(R) | C,D | 453.009 |
| POINTR | B | 453.010 |
| POINTS | C | 453.010 |
| POINTW | D | 453.010 |
| PUT | B | 453.011 |
| READ | C | 453.011 |
| RELSE | D | 453.011 |
| SDMOD | C,D | 455.001 |
| SDMODFI | A,B | 455.001.01 |
| SDMODFO | C,D | 455.001.01 |
| SDMODFU | A-C | 455.002 |
| SDMODUI | D | 455.002 |
| SDMODUO | E | 455.002 |
| SDMODUU | A,B | 455.002.01 |
| SDMODVI | A,B | 455.003 |
| SDMODVO | C,D | 455.003 |
| SDMODVU | A-C | 455.004 |
| SDMODW | D,E | 455.004 |
| TRUNC | A | 453.016 |
| WAITF | C | 453.016 |
| WRITE | D | 453.016 |

## Logical Transient Phases

| Core Image Phase Name | Relocatable Module Name | Card ID |
|---|---|---|
| $$BODACL* | None | CTL.104.00 |
| $$BODAIN* | None | CTL.105.00 |
| $$BODAI1* | None | CTL.106.00 |
| $$BODAO1* | None | CTL.107.00 |
| $$BODAO2* | None | CTL.108.00 |
| $$BODAO3* | None | CTL.109.00 |
| $$BODAO4* | None | CTL.110.00 |
| $$BODAU1* | None | CTL.111.00 |
| $$BODQUE* | None | CTL.112.00 |
| $$BODSMW* | None | CTL.112.50 |
| $$BODSPV* | None | CTL.113.00 |
| $$BODSPW* | None | CTL.114.00 |
| $$BOFLPT* | None | CTL.115.00 |
| $$BOMSG1* | None | CTL.126.00 |
| $$BOMSG2* | None | CTL.127.00 |
| $$BOMSG3* | None | CTL.128.00 |
| $$BOMSG4* | None | CTL.129.00 |
| $$BOMSG5* | None | CTL.130.00 |
| $$BOMSG6* | None | CTL.131.00 |
| $$BOMSG7* | None | CTL.131.50 |
| $$BOSDC1* | None | CTL.149.00 |
| $$BOSDC2* | None | CTL.150.00 |
| $$BOSDI1* | None | CTL.152.00 |
| $$BOSDI2* | None | CTL.153.00 |
| $$BOSDI3* | None | CTL.154.00 |
| $$BOSDI4* | None | CTL.154.50 |
| $$BOSDO1* | None | CTL.155.00 |
| $$BOSDO2* | None | CTL.156.00 |
| $$BOSDO3* | None | CTL.157.00 |
| $$BOSDO4* | None | CTL.158.00 |
| $$BOSDO5* | None | CTL.159.00 |
| $$BOSDO6* | None | CTL.160.00 |
| $$BOSDO7* | None | CTL.161.00 |
| $$BOSDO8* | None | CTL.162.00 |

| | | |
|---|---|---|
| $$BOSDW1* | None | CTL.164.00 |
| $$BOSDW2* | None | CTL.165.00 |
| $$BOSDW3* | None | CTL.166.00 |
| $$BOSD00* | None | CTL.167.00 |
| $$BOSD01* | None | CTL.168.00 |
| $$BOSIGN* | None | CTL.169.00 |
| $$BOVDMP* | None | CTL.171.00 |
| $$BOWDMP* | None | CTL.171.50 |

| | Count | Transmission Information | CSW Status Bits | Symbolic Unit Address | Reserved for Logical IOCS | CCW Address | Reserved for Physical IOCS | CCW Address in CSW | Optional Sense CCW |
|---|---|---|---|---|---|---|---|---|---|
| Bytes | 0 1 | 2 3 | 4 5 | 6 7 | 8 | 9 11 | 12 | 13 15 | 16 23 |
| Used for | Residual Count | Transmitting Information Between Physical IOCS and Problem Program | (Note 1)<br>Byte 4: see BIT DESIGNATION<br>Byte 5: see BIT DESIGNATION | Hexadecimal Representation of SYSnnn<br>SYSRDR =0000<br>SYSIPT =0001<br>SYSPCH =0002<br>SYSLST =0003<br>SYSLOG =0004<br>SYSLNK =0005<br>SYSRES =0006<br>SYSSLB =0007<br>SYSRLB =0008<br>SYSUSE =0009<br>SYS000 =0100<br>SYS001 =0101<br>SYS221 =01DD | Buffer Offset:<br>ASCII Input Tapes X'00'-X'63'<br>ASCII Output Tapes<br>Fixed: X'00'<br>Variable: X'00' or X'04'<br>Undefined: X'00' | Address of CCW Associated with this CCB | X'80' - CCB being used by ERP<br>X'40' - Channel Appendage Routine Present for Teleprocessing Device<br>X'20' - Sense Information Desired<br>X'10' - Message Writer<br>X'08' - EU Tape Error<br>X'02' - Tape ERP Read Opposite Recovery<br>X'01' - Seek Separation or Console Buffering | Address of CCW in the CSW Stored at Channel End, or Address of the Channel Appendage Routine for Teleprocessing Devices | 8 Bytes Appended to the CCB when Sense Information is Desired |

| Byte 4 BIT DESIGNATION | Byte 5 BIT DESIGNATION |
|---|---|
| 32 Attention | 40 Program-controlled interruption |
| 33 Status modifier | 41 Incorrect length |
| 34 Control unit end | 42 Program check |
| 35 Busy | 43 Protection check |
| 36 Channel end | 44 Channel data check |
| 37 Device end | 45 Channel control check |
| 38 Unit check | 46 Interface control check |
| 39 Unit exception | 47 Chaining check |

Byte 2

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Traffic Bit (Wait)<br>(Note 5) | End - of - File ( / * or / & )<br>(Note 2) | Unrecoverable I/O Error | Accept Unrecoverable I/O Error | Return DASD Data Checks 2671 errors, or 1017/1018 errors to the user | Post at Device End<br>(Note 5) | Return Tape Read Data Check, 1018 Data Check, 2540 or 2520 Equipment Check or DASD Data Checks on Read or Verify Command<br>(Notes 3 & 6) | User Error Routine |
| Bits | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Set On By | PIOCS | PIOCS | PIOCS | Pr. Pr. | Pr. Pr. | Pr. Pr. | Pr. Pr. | Pr. Pr. |

Byte 3

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | DASD -<br>Data Check in Count Area<br>MICR -<br>SCU Not Operational<br>1285/1287/1288 Data Check | DASD -<br>Track Overrun<br>MICR -<br>Intervention Required<br>1285/1287 -<br>Keyboard Correction in Journal Tape Mode<br>1017 -<br>Broken Tape | DASD -<br>End of Cylinder<br>MICR -<br>(Note 4)<br>1287/1288 -<br>Hopper Empty in Document Mode | 2540, 2520 -<br>Equipment Check<br>Tape -<br>Read Data Check<br>DASD -<br>Any Data Check<br>1285/1287 -<br>Equipment Check<br>1017/1018 -<br>Data Check | Non - Recovery Questionable Condition:<br>Card -<br>Unusual Command Sequence<br>DASD -<br>No Record Found<br>1285/1287/1288 -<br>Document Jam or Torn Tape | No - Record - Found Condition (Retry on 2311, 2314, or 2319) | Carriage Channel 9 Overflow or Verify Error for DASD.<br>1287 Document Mode - Late Stacker Select<br>1288 - End of Page | Command Chaining Retry from the next CCW to be executed |
| Bits | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Set On By | PIOCS | PIOCS | PIOCS | PIOCS | PIOCS | Pr. Pr. | Pr. Pr. | Pr. Pr. |

PIOCS = Physical IOCS
Pr. Pr. = Problem Program

Note 1. Bytes 4 and 5 contain the status bytes of the Channel Status Word (Bits 32-47). If byte 2, bit 5 is on and device end results as a separate interrupt, device end will be ORed into CCB byte 4.
Note 2. Indicates / * or / & statement encountered on SYSRDR or SYSIPT. Byte 4, bit 7 (unit exception) is also on.
Note 3. DASD data checks on count not returned.
Note 4. For 1270/1275/1412/1419, Disengage. For 1275/1419D, I/O Error in external interrupt routine (channel data check or busout check).
Note 5. The traffic bit (Byte 2, bit 0) is normally set on at channel end to signify that the I/O was completed. If byte 2, bit 5 has been set on, the traffic bit and bits 2 and 6 in byte 3 will be set on at device end. Also see Note 1.
Note 6. 1018 ERP does not support the Error Correction Function.

Figure 42. Command Control Block (Part 1 of 3)

| Byte | Bit | | Condition Indicated | | Mask for Test Under Mask Instruction |
|---|---|---|---|---|---|
| | | | 1 (ON) | 0 (OFF) | |
| 2 | 0 | Traffic Bit (WAIT) | I/O Completed. Normally set at Channel End. Set at Device End if Bit 5 is ON. | I/O Requested and not completed. | X'80' |
| | 1 | End of File on System Input | /* or /& on SYSRDR or SYSIPT. Byte 4, Unit Exception Bit is also ON. | | X'40' |
| | 2 | Unrecoverable I/O Error | I/O error passed back due to program option or operator option | No program or operator option error was passed back | X'20' |
| | 3* | Accept Unrecoverable I/O Error (Bit 2 is ON) | Return to User after Physical IOCS Attempts to correct I/O Error. + | Operator Option: Dependent of the Error. | X'10' |
| | 4* | 2671 Data Check | Operator Options: Ignore, Retry, or Cancel | Operator Option: Retry or Cancel | X'08' |
| | | 1017/1018 Data Checks | Ignore or Cancel | Cancel | |
| | | Return any DASD Data Checks | Return to User | | |
| | 5* | Post at Device End | Device End Condition will be posted i.e., byte 2, bit 0 and byte 3, bits 2 and 6 set at Device End. Also byte 4, bit 5 is set. | Device End Conditions will not be posted. Traffic Bit is set at Channel End. | X'04' |
| | 6* | Return: Uncorrectable tape read data check; 1018 data check; 2540 or 2520 punch equipment check; or DASD read or verify data check. (Data checks on count not returned.) | Return to user after physical IOCS attempts to correct error.<br>Return to user when 1018 data check. | Operator Option: Ignore or Cancel for Tapes, Punches, and Paper Tape Punch (1018). Retry or cancel for DASD. | X'02' |
| | 7* | User Error Routine | User will handle error recovery (Test Bit 2).¤ | A Physical IOCS Error routine will be used. | X'01' |
| 3 | 0 | Data check in DASD count field. | Yes-Byte 3, bit 3 is OFF; Byte 2, bit 2 is ON. | No | X'80' |
| | | MICR-SCU not operational. | Yes | No | |
| | | Data Check-1285, 1287 or 1288 | Yes | No | |
| | 1 | DASD Track overrun. | Yes | No | X'40' |
| | | 1017 Broken Tape | Yes | No | |
| | | MICR Intervention required. | Yes | No | |
| | | Keyboard correction 1285 or 1287 in Journal Tape Mode | Yes | No | |
| | 2 | End of DASD Cylinder | Yes | No | X'20' |
| | | MICR- 1270/1275/1412/1419, disengage. | Document feeding Stopped. | No | |
| | | - 1275/1419D, I/O error in external interrupt routine. | Channel data check or Busout check | | |
| | | Hopper Empty 1287 or 1288 Document Mode | Yes | No | |

Figure 42. Command Control Block (Part 2 of 3)

| Byte | Bit | | Condition Indicated | | Mask for Test Under Mask Instruction |
|---|---|---|---|---|---|
| | | | 1 (ON) | 0 (OFF) | |
| 3 | 3 | Tape read data check; 2540 or 2520 punch equipment check; or any DASD data check | Operation was unsuccessful. Byte 2, Bit 2 is also ON. Byte 3, Bit 0 is OFF. | No | X'10' |
| | | 1017/1018 Data Check | Yes | No | |
| | | 1285, 1287, or 1288 equipment check | Yes | No | |
| | 4 | Questionable Condition | Card: Unusual Command sequence (2540). DASD: No record found. | | X'08' |
| | | Non-recovery | 1285/1287/1288: Document Jam or Torn Tape | | |
| | 5 | No record found condition | Retry command if no record found condition occurs (2311, 2314, or 2319) | Set ON questionable condition bit and return to user. | X'04' |
| | 6 | Verify Error for DASD or Carriage Channel 9 Overflow | Yes. (Set ON when Channel 9 is reached only if Byte 2, Bit 5 is ON.) | No | X'02' |
| | | 1287 Document Mode-Late Stacker Select | Yes | No | |
| | | 1288 End of Page | Yes | No | |
| | 7* | Command Chain Retry | Retry begins at last CCW executed. | Retry begins at first CCW of channel program. | X'01' |

* User Option Bits. Set in CCB macro. Physical IOCS sets the other bits OFF at EXCP time and ON when the condition specified above occurs.

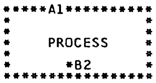
+ I/O program check, command reject, or tape equipment check always terminates the program.

¤ For System/360, the user must handle all error or exceptional conditions except Channel Control Check, Interface Control Check, I/O Program Check, and I/O Protection Check. For System/370, the user may handle Channel Control Checks and Interface Control Checks.

Figure 42. Command Control Block (Part 3 of 3)

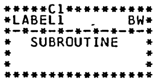
# APPENDIX F: EXPLANATION OF FLOWCHART SYMBOLS

DESCRIPTION

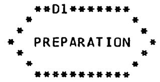


A GROUP OF PROGRAM INSTRUCTIONS THAT PERFORM A PROCESSING FUNCTION OF THE PROGRAM. THE LABEL, IF ANY, IS SHOWN ABOVE THE BLOCK.

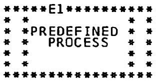
*B2
IF ANY ADDITIONAL EXPLANATION IS REQUIRED, ITS LOCATION ON THE CHART IS IDENTIFIED BY AN ASTERISK AND THE BLOCK ID.



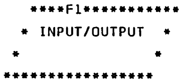
DESCRIPTION OR TITLE OF A ROUTINE THAT IS DETAILED ON ANOTHER FLOWCHART. THE STARTING LABEL OF THE ROUTINE AND THE FLOWCHART ID APPEAR ABOVE THE STRIPE.



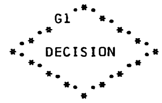
AN INSTRUCTION, OR GROUP OF INSTRUCTIONS, THAT CHANGES PORTIONS OF A ROUTINE OR INITIALIZES A ROUTINE FOR GIVEN CONDITIONS.



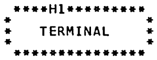
A GROUP OF OPERATIONS NOT DETAILED IN THE FLOWCHARTS IN THIS MANUAL, SUCH AS USER'S ROUTINES.



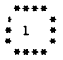
ANY FUNCTION OF AN INPUT/OUTPUT DEVICE OR PROGRAM, USUALLY BRANCHING TO AN I/O ROUTINE TO PERFORM THE FUNCTION STATED IN THE BLOCK.



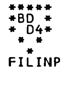
POINTS WHERE THE PROGRAM BRANCHES TO ALTERNATE PROCESSING, BASED UPON VARIABLE CONDITIONS SUCH AS PROGRAM SWITCH SETTINGS AND TEST RESULTS.



THE BEGINNING, END, OR POINT OF INTERRUPTION IN A PROGRAM.

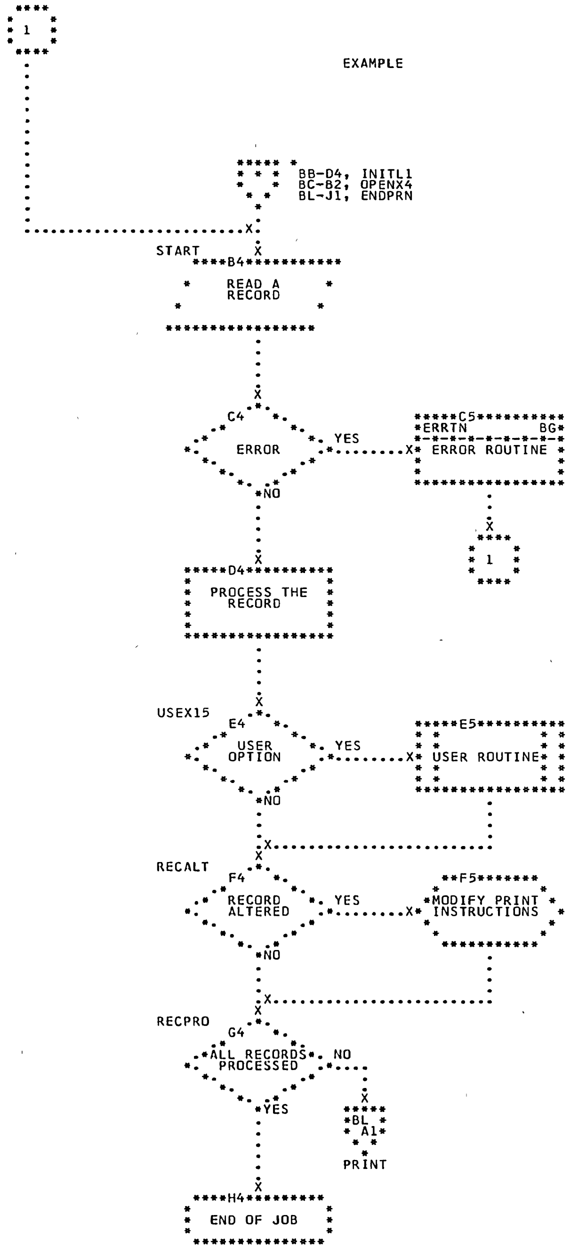


ON-PAGE CONNECTOR. AN ENTRY FROM, OR AN EXIT TO, ANOTHER FUNCTION ON THE SAME FLOWCHART. THE NUMBER IN THE CONNECTOR IDENTIFIES THE CORRESPONDING ENTRY OR EXIT ON THE CHART.



OFF-PAGE CONNECTOR. AN ENTRY FROM, OR AN EXIT TO, A GIVEN POINT ON ANOTHER FLOWCHART. THE CHARACTERS IN THE CONNECTOR IDENTIFY THE CHART AND BLOCK. THE CORRESPONDING LABEL, IF ANY, IS PLACED OUTSIDE THE CONNECTOR. FOR MULTIPLE ENTRIES AND EXITS, AN ASTERISK APPEARS IN THE CONNECTOR AND THE CHARACTERS ARE LISTED NEARBY.

EXAMPLE



# APPENDIX G: DASD LABEL INFORMATION

## DASD LABELS

Whenever files of records are written on DASD, each volume must contain standard labels to identify the pack or cell and the logical file(s) on it. When logical IOCS is used for a file, the IOCS routines read, check and/or write standard labels. When physical IOCS is used, IOCS processes the labels if the DTFPH macro instruction is included in the user's program. The entry TYPEFLE must be specified to indicate whether the file is an input file (read and check labels) or an output file (read and check old labels and write new labels).

The standard labels include one volume label for each pack or cell and one or more file labels for each logical file on the DASD. The following paragraphs briefly describe the organization of labels on disk packs or data cells. Additional information about labels is given in the Data Management Concepts publication, listed in the Preface of this publication.

### Volume Labels

The standard volume label identifies the entire volume and offers volume protection. For systems residence, the volume label is always the third record on cylinder 0, track 0. The first two records on this track of SYSRES are Initial Program Load (IPL) records. On all other volumes, these records contain binary zeros. The volume label record consists of a count area, a 4-byte key area, and an 80-byte data area. Both the key area and the first four bytes of the data area contain the label identifier VOL1. The remaining 76 bytes of the data area contain other identifying information such as the volume serial number, and the address of the file labels for the pack or cell (see Standard File Labels). An IBM-supplied utility program writes the volume label when the direct access device is received by the installation.

The standard volume label may be followed by one to seven additional volume labels (starting with record 4 on cylinder 0, track 0). These labels must contain the label identifier VOL2, VOL3, etc. in the four-byte key areas and in the first four bytes of the data areas. The other 76 bytes may contain whatever information the user requires. The additional volume labels are also written by the utility program that writes the standard volume label. However, IOCS does not make them available to the user for checking or rewriting when problem programs are executed. These labels are provided for use with Operating System/360 and are always bypassed by the Disk Operating System OPEN routines. Figure 43 shows the format of the standard DASD volume label.

### Standard File Labels

The standard file labels identify the logical file, give its location(s) on the disk pack or data cell, and offer file protection. The labels for all logical files on a volume are grouped together and stored in the Volume Table of Contents of DASD.

The number and format of labels required for any one logical file depends on the file organization (see Standard File Label Formats) and the number of separate areas (extents) of the pack or cell used by the file. The data records for a logical file may be contained within one area of the pack or cell, or they may be scattered in different areas of it. The limits (starting and ending addresses) of each area used by the file are specified by the standard file label(s).

Because each file label contains file limits, the group of labels on the volume is essentially a directory of all files on the volume. Therefore, it is known as the Volume Table of Contents (VTOC). The VTOC itself becomes a file of records (one or more standard-label records per logical file) and, in turn, has a label. The label of the VTOC is the first record in the VTOC. This label identifies the file as the VTOC file, and gives the file limits of the VTOC file. The Volume Table of Contents is contained within one cylinder of a disk pack or data cell. It does not overflow onto another cylinder.

If a logical file of data records is recorded on more than one volume, standard labels for the file must be included in the VTOC of each volume used. The label(s) on each volume identifies the portion of the logical file on the pack or cell and specifies the extent(s) used on it.

## Standard File Label Formats

All standard file label records have a count area and a 140-byte key/data area. Five standard-label formats are provided.

Format 1: This format is used for all logical files, and it has a 44-byte key area and a 96-byte data area. It is always the first of the series of labels when a file requires more than one label on a disk pack or cell (as discussed in Format 2 and Format 3).

The format-1 label identifies the logical file (by a file name assigned by the user and included in the 44-byte key area), and it contains file and data record specifications. It also provides the addresses for three separate DASD areas (extents) for the file. If the file is scattered over more than three separate areas on one pack or cell, a format-3 label is also required. In this case, the format-1 label points to the second label set up for the file on this volume.

If a logical file is recorded on more than one volume, a format-1 label is always created in the VTOC for each volume. Figure 44 shows the format of the format-1 label.

Format 2: This format is required for any file that is organized by the Indexed Sequential File Management System. The 44-byte key area and the 96-byte data area contain specifications unique to this type of file organization.

If an indexed sequential file is recorded on two or more volumes, the format-2 label is used only on the volume containing the cylinder index. This volume may, or may not, contain data records. The format-2 label is not repeated on the additional packs (as the format-1 label is). See Figure 45 for the format of the format-2 label.

Format 3: If a logical file uses more than three extents on any pack or cell, this format specifies the addresses of the additional extents. It is used only for extent information. It has a 44-byte key area and a 96-byte data area that provide for 13 extents.

The format-3 label is pointed to by the format-1 label for the logical file. In a DTFSD file, it may also be pointed to another format-3 label. It is included as required on the first pack or cell, or on additional volumes if the logical file is recorded on two or more volumes. See Figure 46 for the format of the format-3 label.

Format 4: The format-4 label defines the VTOC itself. This is always the first label in the VTOC. This label provides the location and number of available tracks in the alternate track area. Figure 47 shows the format of the format-4 label.

Format 5: The format-5 label is used by the Operating System/360 for Direct Access Device Space Management. Figure 48 shows the format of the format-5 label.

## User-Standard DASD File Labels

The user may include additional labels to define his file further, provided the file is processed sequentially (DTFSR or DTFSD macro specified), by the Direct Access Method (DTFDA macro specified) or by physical IOCS (DTFPH macro specified). User standard file labels are not processed by the Indexed Sequential File Management System (DTFIS specified). A file may have up to eight user header labels and eight user trailer labels for a 2311 or 2314 file. If the device is a 2321, the file may have up to five user header labels and five user trailer labels. The trailer labels can be written to indicate an end-of-volume or end-of-file condition. That is, when the end of an extent on one volume is reached and the next extent is on a different volume, or when the end of the file is reached, user trailer labels can be included. The labels may contain whatever trailer information the user desires (for example, a record count for the completed volume).

User-standard labels are not stored in the Volume Table of Contents. Instead, they are written on the first track of the first extent allotted for the logical-file data records. In this case, the user's data records start with the second track in the extent, regardless of whether the labels require a full track. If a file is written on two or more packs or cells, the additional labels are written on each of the packs or cells.

All user-standard labels must be 80 bytes long, and they must contain standard information in the first four bytes. The remaining 76 bytes may contain whatever information the user wants.

The standard information in the first four bytes is used as a record key when reading or writing header labels. The header labels are identified by UHL1, UHL2,...,UHL8 (or UHL5). The trailer labels, when applicable, are identified in the key field by UTL0, UTL1,...,UTL7 (or UTL4) although the first four bytes of the labels will contain UTL1-UTL8 (or UTL5). Each user-label set (header or trailer) is terminated by a file mark (a record with data length 0), which is written by IOCS.

For example, if a file has five header labels and four trailer labels, the contents of the user-label track are:

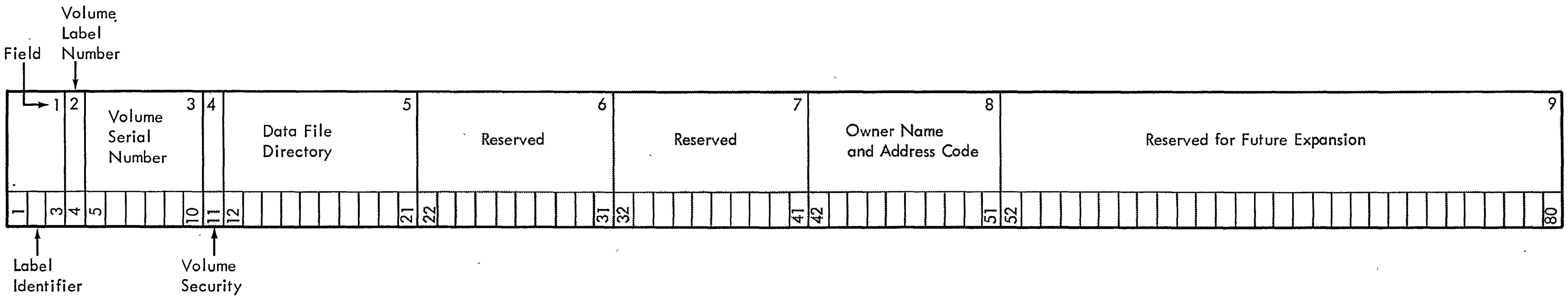
| | |
|---|---|
| R0 | Standard information |
| R1 | UHL1--user's 1st header label |
| R2 | UHL2--user's 2nd header label |
| R3 | UHL3--user's 3rd header label |
| R4 | UHL4--user's 4th header label |
| R5 | UHL5--user's 5th header label |
| R6 | UHL6--file mark |
| R7 | UTL1--user's 1st trailer label |
| R8 | UTL2--user's 2nd trailer label |
| R9 | UTL3--user's 3rd trailer label |
| R10 | UTL4--user's 4th trailer label |
| R11 | UTL5--file mark |

If only <u>header</u> labels are used, the user-label track contains:

| | |
|---|---|
| R0 | Standard information |
| R1 | UHL1--user's 1st header label |
| R2 | UHL2--user's 2nd header label |
| | . |
| | . |
| | . |
| R(n) | UHL(n)--user's nth header label where n is ≤ 8 |
| R(n+1) | UHL(n+1) |
| R(n+2) | UTL0--file mark |

The user's label routine can determine if a label is a header or trailer label by testing the first four bytes of the label.

Figure 43. Standard DASD Volume Label

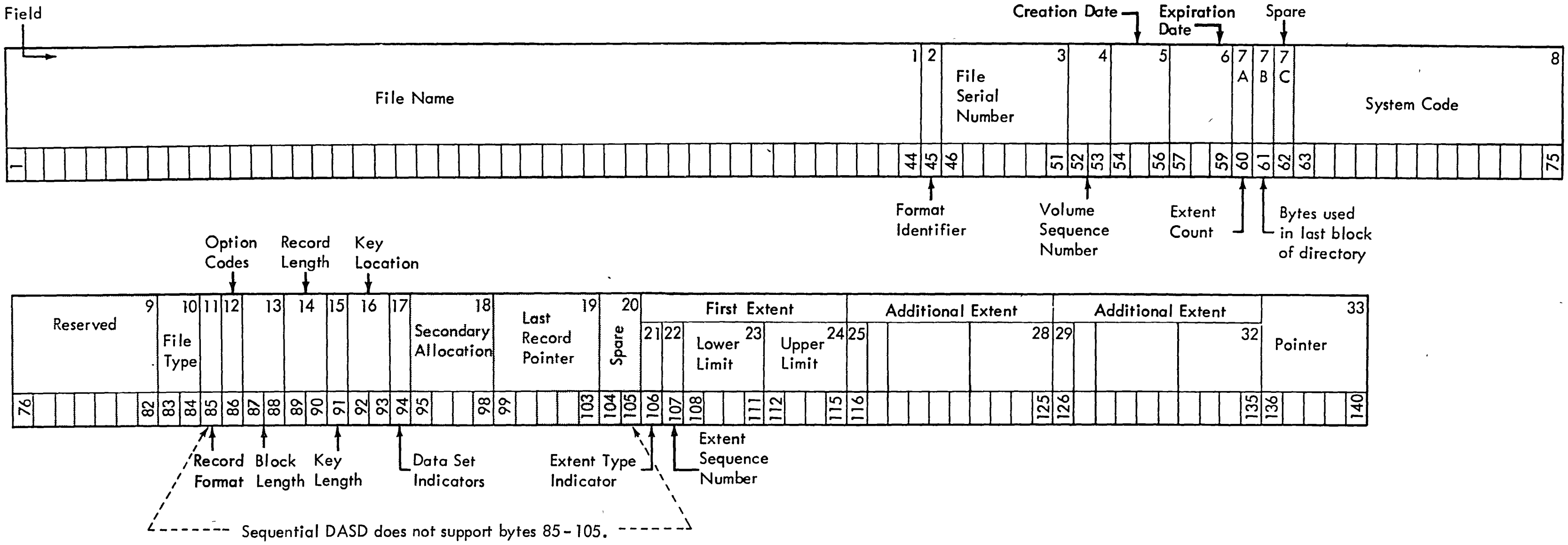


Volume Label Format (80 bytes) for DASD (Shaded areas are not processed by DOS/360)

| FIELD | NAME AND LENGTH | DESCRIPTION |
|---|---|---|
| 1. | LABEL IDENTIFIER<br>3 bytes | Must contain VOL to indicate that this is a Volume Label. |
| 2. | VOLUME LABEL NUMBER<br>1 byte | Indicates the relative position (1 - 8) of a volume label within a group of volume labels. |
| 3. | VOLUME SERIAL NUMBER<br>6 bytes | A unique identification code which is assigned to a volume when it enters an installation. This code may also appear on the external surface of the volume for visual identification. It is normally a numeric field 000001 to 999999, however any or all of the 6 bytes may be alphameric. |
| 4. | VOLUME SECURITY<br>1 byte | Indicates security status of the volume:<br>0=no further identification for each file of the volume is required.<br>1=further identification for each file of the volume is required before processing. |
| 5. | DATA FILE DIRECTORY<br>10 bytes | The first 5 bytes contain the starting address (CCHHR) of the VTOC. The last 5 bytes are blank. |
| 6. | RESERVED<br>10 bytes | Reserved. |
| 7. | RESERVED<br>10 bytes | Reserved. |
| 8. | OWNER NAME AND ADDRESS CODE<br>10 bytes | Indicates a specific customer, installation and/or system to which the volume belongs. This field may be a standardized code, name, address, etc. |
| 9. | RESERVED<br>29 bytes | Reserved. |

Note: All reserved fields should contain blanks to facilitate their use in the future. Any information appearing in these fields at the present time will be ignored by the Disk Operating System programs and Operating System/360 programs.

Figure 44. Standard DASD File Label, Format 1 (Part 1 of 3)



Format 1: This format is common to all data files on Direct Access Storage Devices. (Shaded areas are not processed by DOS/360)

| FIELD | NAME AND LENGTH | DESCRIPTION |
|---|---|---|
| 1. | FILE NAME<br>44 bytes, alphameric EBCDIC | This field serves as the key portion of the file label.<br>Each file must have a unique file name. Duplication of file names will cause retrieval errors. The file name can consist of three sections:<br>1. File ID is an alphameric name assigned by the user and identifies the file. Can be 1-35 bytes if generation and version numbers are used, or 1-44 bytes if they are not used.<br>2. Generation Number. If used, this field is separated from File ID by a period. It has the format Gnnnn, where G identifies the field as the generation number and nnnn (in decimal) identifies the generation of the file.<br>3. Version Number of Generation. If used, this section immediately follows the generation number and has the format Vnn, where V identifies the field as the version of generation number and nn (in decimal) identifies the version of generation of the file.<br>Note: The Disk Operating System compares the entire field against the file name given in the DLBL card. The generation and version numbers are treated differently by Operating System/360. |

The remaining fields comprise the DATA portion of the file label:

| FIELD | NAME AND LENGTH | DESCRIPTION |
|---|---|---|
| 2. | FORMAT IDENTIFIER<br>1 byte, EBCDIC numeric | 1 = Format 1 |
| 3. | FILE SERIAL NUMBER<br>6 bytes, alphameric EBCDIC | Uniquely identifies a file/volume relationship. It is identical to the Volume Serial Number of the first or only volume of a multivolume file. |
| 4. | VOLUME SEQUENCE NUMBER 2 bytes, binary | Indicates the order of a volume relative to the first volume on which the data file resides. |
| 5. | CREATION DATE<br>3 bytes, discontinuous binary | Indicates the year and the day of the year the file was created. It is of the form YDD, where Y signifies the year (0-99) and DD the day of the year (1-366). |

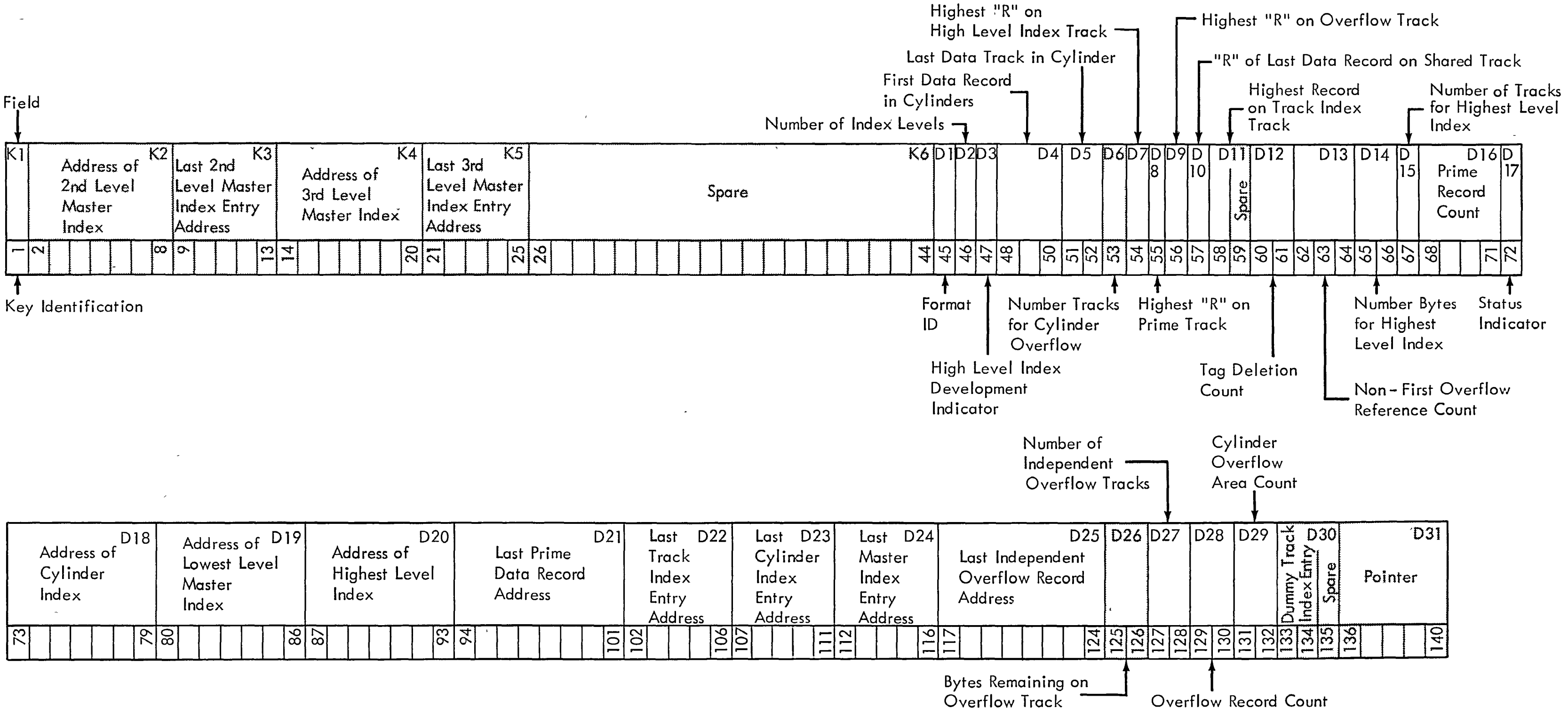
| FIELD | NAME AND LENGTH | DESCRIPTION |
|---|---|---|
| 6. | EXPIRATION DATE<br>3 bytes, discontinuous binary | Indicates the year and the day of the year the file may be deleted. The form of this field is identical to that of Field 5. |
| 7A. | EXTENT COUNT<br>1 byte | Contains a count of the number of extents for this file on this volume. If user labels are used, the count does not include the user label track. This field is maintained by the Disk Operating System programs. |
| 7B. | BYTES USED IN LAST BLOCK OF DIRECTORY<br>1 byte, binary | Used by O/S. |
| 7C. | SPARE<br>1 byte | Reserved |
| 8. | SYSTEM CODE<br>13 bytes | Uniquely identifies the programming system. The character codes that can be used in this field are limited to EBCDIC characters. On input, IOCS ignores this field. On output, IOCS writes the information supplied in DLAB. If you use DLAB, IOCS writes: DOS/360 Ver 3. |
| 9. | RESERVED<br>7 bytes | Reserved |
| 10. | FILE TYPE<br>2 bytes | The contents of this field uniquely identify the type of data file:<br>Hex 4000=Consecutive organization<br>Hex 2000=Direct-access organization<br>Hex 8000=Indexed-sequential organization<br>Hex 0200=Library organization<br>Hex 0000=Organization not defined in the file label. |
| 11. | RECORD FORMAT<br>1 byte | Used by O/S. |
| 12. | OPTION CODES<br>1 byte | Bits within this field indicate various options used in building the file.<br>Bit<br>0=0<br>1=Reserved<br>2=Master index present (ISFMS)<br>3=Independent overflow present (ISFMS)<br>4=Cylinder overflow present (ISFMS)<br>5=Reserved<br>6=Used by O/S<br>7=Used by O/S |
| 13. | BLOCK LENGTH<br>2 bytes, binary | Indicates the block length for fixed length records or maximum block size for variable length blocks. |
| 14. | RECORD LENGTH<br>2 bytes, binary | Indicates the record length for fixed length records or the maximum record length for variable length records. |
| 15. | KEY LENGTH<br>1 byte, binary | Indicates the length of the key portion of the data records in the file. |
| 16. | KEY LOCATION<br>2 bytes, binary | Indicates the high order position of the data record. |
| 17. | DATA SET INDICATORS<br>1 byte | Bits within this field are used to indicate the following:<br>Bit<br>0 If on, indicates that this is the last volume on which this file normally resides.<br>1, 2, 4, 6, 7: 0 for DOS<br>Used by O/S<br>3 If on, DOS data security is invoked.<br>5 Used by DOS |
| 18. | SECONDARY ALLOCATION<br>4 bytes, binary | Used by O/S |
| 19. | LAST RECORD POINTER<br>5 bytes, discontinuous binary | Used by O/S |

Figure 44. Standard DASD File Label, Format 1 (Part 2 of 3)

| FIELD | NAME AND LENGTH | DESCRIPTION |
|---|---|---|
| 20. | SPARE<br>2 bytes | Reserved |
| 21. | EXTENT TYPE INDICATOR<br>1 byte | Indicates the type of extent with which the following fields are associated:<br>HEX CODE<br>00 Next three fields do not indicate any extent.<br>01 Prime data area (Indexed Sequential); or Consecutive area, etc., (i.e., the extent containing the user's data records.)<br>02 Overflow area of an Indexed Sequential file.<br>04 Cylinder index or master index area of an Indexed Sequential file.<br>40 User label track area.<br>80 Shared cylinder indicator. |
| 22. | EXTENT SEQUENCE NUMBER<br>1 byte, binary | Indicates the extent sequence in a multi-extent file. |
| 23. | LOWER LIMIT<br>4 bytes, discontinuous binary | The cylinder and the track address specifying the starting point (lower limit) of this extent component. This field has the format CCHH. |
| 24. | UPPER LIMIT<br>4 bytes | The cylinder and the track address specifying the ending point (upper limit) of this extent component. This field has the format CCHH. |
| 25-28. | ADDITIONAL EXTENT<br>10 bytes | These fields have the same format as the fields 21-24 above. |
| 29-32. | ADDITIONAL EXTENT<br>10 bytes | These fields have the same format as the fields 21-24 above. |
| 33. | POINTER TO NEXT FILE LABEL WITHIN THIS LABEL SET<br>5 bytes, discontinuous binary | The address (format CCHHR) of a continuation label if needed to further describe the file. If field 10 indicates Indexed Sequential organization, this field points to a Format 2 file label within this label set. Otherwise, it points to a Format 3 file label, and then only if the file contains more than three extent segments. This field contains all binary zeros if no additional file label is pointed to. |

Figure 44. Standard DASD File Label, Format 1 (Part 3 of 3)

Figure 45. Standard DASD File Label, Format 2 (Part 1 of 2)


Field
K1
Address of 2nd Level Master Index
K2
Last 2nd Level Master Index Entry Address
K3
Address of 3rd Level Master Index
K4
Last 3rd Level Master Index Entry Address
K5
Spare
K6
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
Spare
D12
D13
D14
D15
D16 Prime Record Count
D17
1 2 8 9 13 14 20 21 25 26 44 45 46 47 48 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 71 72
Key Identification
Number of Index Levels
First Data Record in Cylinders
Last Data Track in Cylinder
Highest "R" on High Level Index Track
Highest "R" on Overflow Track
"R" of Last Data Record on Shared Track
Highest Record on Track Index Track
Number of Tracks for Highest Level Index
Format ID
High Level Index Development Indicator
Number Tracks for Cylinder Overflow
Highest "R" on Prime Track
Tag Deletion Count
Non - First Overflow Reference Count
Number Bytes for Highest Level Index
Status Indicator
Number of Independent Overflow Tracks
Cylinder Overflow Area Count
D18 Address of Cylinder Index
D19 Address of Lowest Level Master Index
D20 Address of Highest Level Index
D21 Last Prime Data Record Address
Last D22 Track Index Entry Address
Last D23 Cylinder Index Entry Address
Last D24 Master Index Entry Address
D25 Last Independent Overflow Record Address
D26
D27
D28
D29
Dummy Track Index Entry
D30
Spare
D31 Pointer
73 79 80 86 87 93 94 101 102 106 107 111 112 116 117 124 125 126 127 128 129 130 131 132 133 134 135 136 140
Bytes Remaining on Overflow Track
Overflow Record Count

Format 2: This format is applicable only to Indexed Sequential data files. It is always pointed to by a Format 1 label. (Shaded areas are not processed by DOS/360)

| FIELD | NAME AND LENGTH | DESCRIPTION |
|---|---|---|
| K1 | KEY IDENTIFICATION<br>1 byte | Contains the Hex Code 02 in order to avoid conflict with a file name. |
| K2 | ADDRESS OF 2ND LEVEL MASTER INDEX<br>7 bytes, discontinuous binary | Contains the address of the first track of the second level of the master index, in the form MBBCCHHR. (OS/360only) |
| K3 | LAST 2ND LEVEL MASTER INDEX ENTRY<br>5 bytes, discontinuous binary | Contains the address of the last index entry in the second level of the master index, in the form CCHHR. (OS/360 only) |
| K4 | ADDRESS OF 3RD LEVEL MASTER INDEX<br>7 bytes, discontinuous binary | Contains the address of the first track of the third level of the master index, in the form MBBCCHH. (OS/360 only) |
| K5 | LAST 3RD LEVEL MASTER INDEX ENTRY<br>5 bytes, discontinuous binary | Contains the address of the last entry in the third level of the master index, in the form CCHHR. (OS/360 only) |
| K6 | SPARE<br>19 bytes | Reserved |
| D1 | FORMAT IDENTIFIER<br>1 byte, EBCDIC numeric | 2 = Format 2 |
| D2 | NUMBER OF INDEX LEVELS<br>1 byte, binary | Indicates how many levels of index are present with an Indexed Sequential file. |

Figure 45. Standard DASD File Label, Format 2 (Part 2 of 2)

| FIELD | NAME AND LENGTH | DESCRIPTION |
|---|---|---|
| D3 | HIGH LEVEL INDEX DEVELOPMENT INDICATOR 1 byte, binary | Contains the number of tracks determining development of master index. |
| D4 | FIRST DATA RECORD IN CYLINDER 3 bytes | Contains the address of the first data record on each cylinder, in the form HHR. |
| D5 | LAST DATA TRACK IN CYLINDERS 2 bytes | Contains the address of the last data track on each cylinder, in the form HH. |
| D6 | NUMBER OF TRACKS FOR CYLINDER OVERFLOW 1 byte, binary | Contains the number of tracks in cylinder overflow area. (Applies to OS/360 only) |
| D7 | HIGHEST "R" ON HIGH LEVEL INDEX TRACK 1 byte | Contains the highest possible R on track containing high-level index entries. |
| D8 | HIGHEST "R" ON PRIME TRACK 1 byte | Contains the highest possible R on prime data tracks for fixed length records. |
| D9 | HIGHEST "R" ON OVERFLOW TRACK | Contains the highest possible R on overflow data tracks for fixed length records. |
| D10 | "R" OF LAST DATA RECORD ON SHARED TRACK 1 byte | Contains the R of the last data record on a shared track. |
| D11 | HIGH RECORD ON TRACK INDEX TRACK 1 byte | The first byte of this two-byte field indicates the high (0-256) record on the track index track. The second byte is reserved. |
| D12 | TAG DELETION COUNT 2 bytes, binary | Contains the number of records that have been tagged for deletion. |
| D13 | NON-FIRST REFERENCE COUNT 3 bytes, binary | Contains a count of the number of random references to a non-first overflow record. |
| D14 | NUMBER OF BYTES FOR HIGHEST-LEVEL INDEX 2 bytes, binary | Indicates how many bytes are needed to hold the highest-level index in main storage. |
| D15 | NUMBER OF TRACKS FOR HIGHEST-LEVEL INDEX 1 byte, binary | Contains a count of the number of tracks occupied by the highest-level index. |
| D16 | PRIME RECORD COUNT 4 bytes, binary | Contains a count of the number of records in the prime data area. |
| D17 | STATUS INDICATOR 1 byte | The eight bits of this byte are used for the following indications:<br>Bit — Description<br>0-1 — must remain off<br>2 — file closed for ADD or ADDRTR<br>3-5 — must remain off<br>6 — last block full<br>7 — last track full |
| D18 | ADDRESS OF CYLINDER INDEX 7 bytes | Contains the address of the first track of the cylinder index, in the form MBBCCHH. |
| D19 | ADDRESS OF LOWEST-LEVEL MASTER INDEX 7 bytes | Contains the address of the first track of the lowest-level index of the high level indexes, in the form MBBCCHH. |
| D20 | ADDRESS OF HIGHEST-LEVEL INDEX 7 bytes | Contains the address of the first track of the highest level master index, in the form MBBCCHH. |
| D21 | LAST PRIME DATA RECORD ADDRESS 8 bytes | Contains the address of the last data record in the prime data area, in the form MBBCCHH. |
| D22 | LAST TRACK INDEX ENTRY ADDRESS 5 bytes | Contains the address of the last normal entry in the track index on the last cylinder in the form CCHHR. |
| D23 | LAST CYLINDER INDEX ENTRY ADDRESS 5 bytes | Contains the address of the last index entry in the cylinder index in the form CCHHR. |
| D24 | LAST MASTER INDEX ENTRY ADDRESS 5 bytes | Contains the address of the last index entry in the master index in the form CCHHR. |
| D25 | LAST INDEPENDENT OVERFLOW RECORD ADDRESS 8 bytes | Contains the address of the last record written in the current independent overflow area, in the form MBBCCHHR. |
| D26 | BYTES REMAINING ON OVERFLOW TRACK 2 bytes, binary | Contains the number of bytes remaining on current independent overflow track. (Applies to OS/360 only) |
| D27 | NUMBER OF INDEPENDENT OVERFLOW TRACKS 2 bytes, binary | Contains the number of tracks remaining in independent overflow area. |
| D28 | OVERFLOW RECORD COUNT 2 bytes, binary | Contains a count of the number of records in the overflow area. |
| D29 | CYLINDER OVERFLOW AREA COUNT 2 bytes, binary | Contains the number of cylinder overflow areas full. |
| D30 | DUMMY TRACK INDEX ENTRY 3 bytes | This field contains the HHR portion of the dummy track index entry. (Applies to OS/360 only) |
| D31 | POINTER TO FORMAT 3 FILE LABEL 5 bytes | Contains the address (in the form CCHHR) of a Format 3 file label if more than 3 extent segments exist for the data file within this volume. Otherwise it contains binary zeros. (Applies to OS 360 only) |

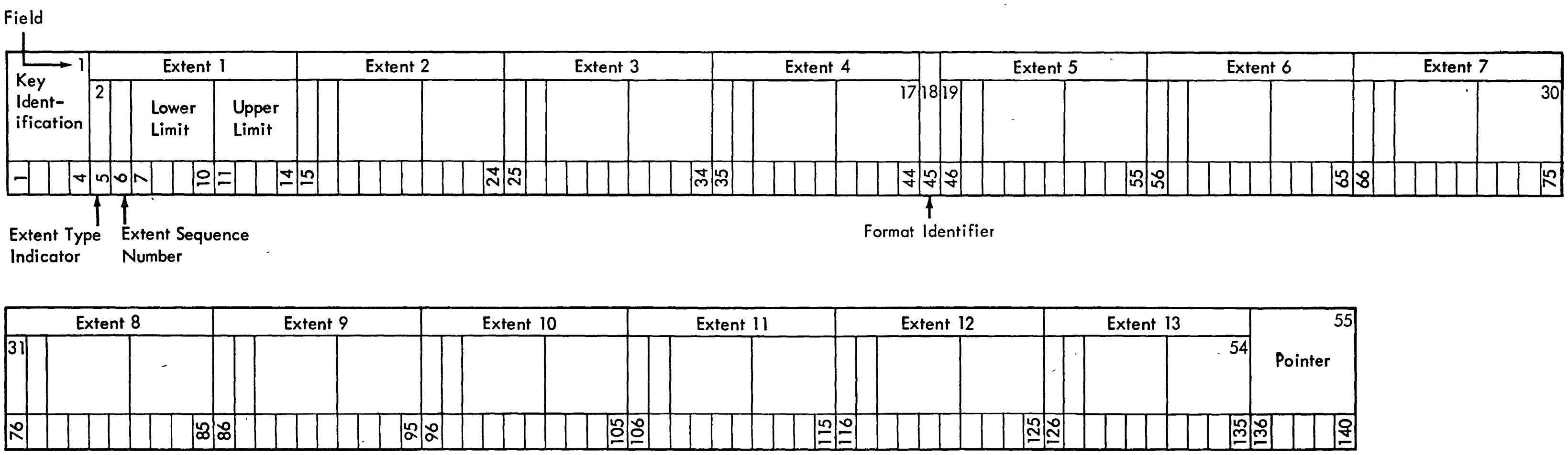
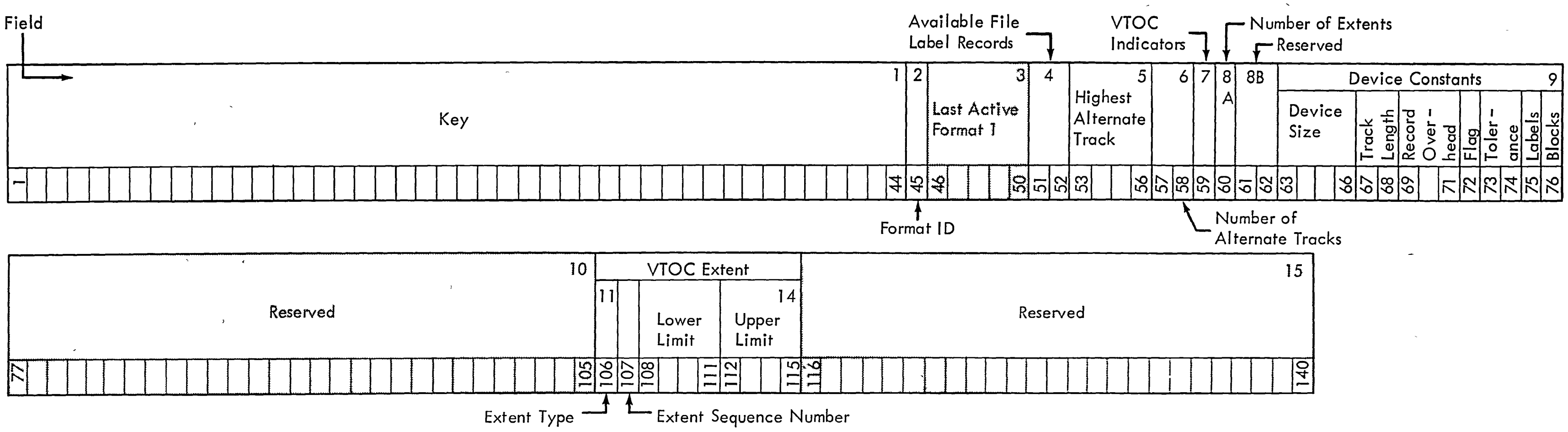


Figure 46. Standard DASD File Label, Format 3

Format 3: This format is used to describe extra extent segments on the volume if there are more than can be described in the Format 1 (and Format 2 if it exists) file label. This file label is pointed to by a Format 1, Format 2, or another Format 3 file label.

| FIELD | NAME AND LENGTH | DESCRIPTION |
|---|---|---|
| 1. | KEY IDENTIFICATION<br>4 byte | Each byte of this field contains the Hex Code 03 in order to avoid conflict with a data file name. |
| 2-17 | EXTENTS (in KEY)<br>40 bytes | Contains four groups of fields identical in format to fields 21 - 24 in the Format 1 label. |
| 18. | FORMAT IDENTIFIER<br>1 byte, EBCDIC numeric | 3 = Format 3 |
| 19-54 | ADDITIONAL EXTENTS<br>90 bytes | Contains nine groups of fields identical in format to fields 21 - 24 in the Format 1label. |
| 55. | POINTER TO NEXT FILE LABEL<br>5 bytes | Contains the address (in the form CCHHR) of another Format 3 label if additional extents must be described. Otherwise, it is all binary zeros. |

Figure 47. Standard DASD File Label, Format 4 (Part 1 of 2)



Format 4: This format describes the Volume Table of Contents and is always the first file label in the VTOC. There must be one and only one of these Format 4 file labels per volume. (Shaded areas are not processed by DOS/360)

| FIELD | NAME AND LENGTH | DESCRIPTION |
|---|---|---|
| 1. | KEY FIELD<br>44 bytes, binary | Each byte of this field contains the Hex Code 04 in order to provide a unique key. |
| 2. | FORMAT ID<br>1 byte, EBCDIC numeric | 4 = Format 4 |
| 3. | LAST ACTIVE FORMAT 1<br>5 bytes | Contains the address (in the form CCHHR) of the last active Format 1 file label. It is used to stop a search on a file name. |
| 4. | AVAILABLE FILE LABEL RECORDS<br>2 bytes, binary | Contains a count of the number of unused records in the VTOC. |
| 5. | HIGHEST ALTERNATE TRACK<br>4 bytes | Contains the highest address (in the form CCHH) of a block of tracks set aside as alternates for bad tracks. |
| 6. | NUMBER OF ALTERNATE TRACKS<br>2 bytes, binary | Contains the number of alternate tracks available. |
| 7. | VTOC INDICATORS<br>1 byte | Bit 0, if on, indicates no DADSM (O/S Direct Access Device Space Management - Format 5) label, or DADSM label does not reflect true status of volume.<br>Bit 1-2 not used.<br>3 1401/1440/1460 Emulator Pack.<br>4-7 not used. |
| 8A. | NUMBER OF EXTENTS<br>1 byte | Contains the hexadecimal constant 01, to indicate one extent in the VTOC. |
| 8B. | RESERVED<br>2 bytes | Reserved. |
| 9. | DEVICE CONSTANTS<br>14 bytes | Contains constants describing the device on which the volume was mounted when the VTOC was created. The following describes each of the subfields. |

Device Size (4 bytes) - The number of cylinders (CC) and tracks per cylinder (HH).

Track Length (2 bytes) - The number of available bytes on a track exclusive of home address and record zero (record zero is assumed to be a non-keyed record with an eight byte data field).

Record Overhead (3 bytes) - The number of bytes required for gaps, check bits, and count field for each record. This value varies according to the record characteristics and thus is broken down into three subfields.

I - Overhead required for a keyed record other than the last record on the track.
L - Overhead required for a keyed record that is the last record on the track.
K - Overhead bytes to be subtracted from I or L if the record does not have a key field.

Flag (1 byte) - Further defines unique characteristics of the device.

| bits | meaning |
|---|---|
| 0-5 | reserved |
| 6 | CC and HH must be used as 1-byte values, as in the case of the 2321. |
| 7 | A tolerance factor must be applied to all but the last record on the track. |

FIELD NAME AND LENGTH DESCRIPTION

Tolerance (2 bytes) - A value that is to be used to determine the effective length of the record on the track. The effective length of a record is calculated in the following manner:

1. Add the key length to the data length of the record

2. Test bit 7 in the flag byte:
   a. if 0, go to step 3
   b. multiply value from step 1 by the tolerance factor
   c. shift result 9 bits to the right

3. Add overhead bytes to the result.

NOTE: Step 2 is not required if the calculation is for the last record on the track.

Labels/Track (1 byte) - A count of the number of labels that can be written on each track in the VTOC. (Number of full records of 44-byte key and 96-byte data lengths that can be contained on one track of this device).

Directory Blocks/Track (1 byte) - A count of the number of directory blocks that can be written on each track for an Operating System/360 partitioned data set. (Number of full records of 8-byte key and 256-byte data lengths that can be contained on one track of this device).

The following illustrates the device constants field for the various direct access devices:

| Device | CC | HH | Track Length | I | L | K | Flag | Tolerance | Labels/ Track | Dir Blk/ Track |
|---|---|---|---|---|---|---|---|---|---|---|
| 2311 | 203 | 10 | 3625 | 81 | 20 | 20 | 1 | 537 | 16 | 10 |
| 2314/2319 | 203 | 20 | 7294 | 146 | 45 | 45 | 1 | 534 | 25 | 17 |
| 2321 | 20/10 | 5/20 | 2000 | 100 | 16 | 16 | 3 | 537 | 8 | 5 |
| 2301 | 0 | 200 | 20616 | 186 | 186 | 53 | 0 | 512 | 63 | 45 |
| 2302 | 250 | 46 | 5070 | 82 | 55 | 20 | 1 | 537 | 22 | 14 |
| 7320 | 0 | 400 | 2129 | 111 | 43 | 14 | 1 | 537 | 8 | 5 |

NOTE: CCHH for the 2321 above are separate 1 byte quantities.

| | | |
|---|---|---|
| 10. | RESERVED<br>29 bytes | Reserved. |
| 11-14. | VTOC EXTENT | These fields describe the extent of the VTOC, and are identical in format to fields 21-24 of the Format 1 file label. Extent type 01 (prime data area). |
| 15. | RESERVED<br>25 bytes | Reserved. |

Figure 47. Standard DASD File Label, Format 4 (Part 2 of 2)

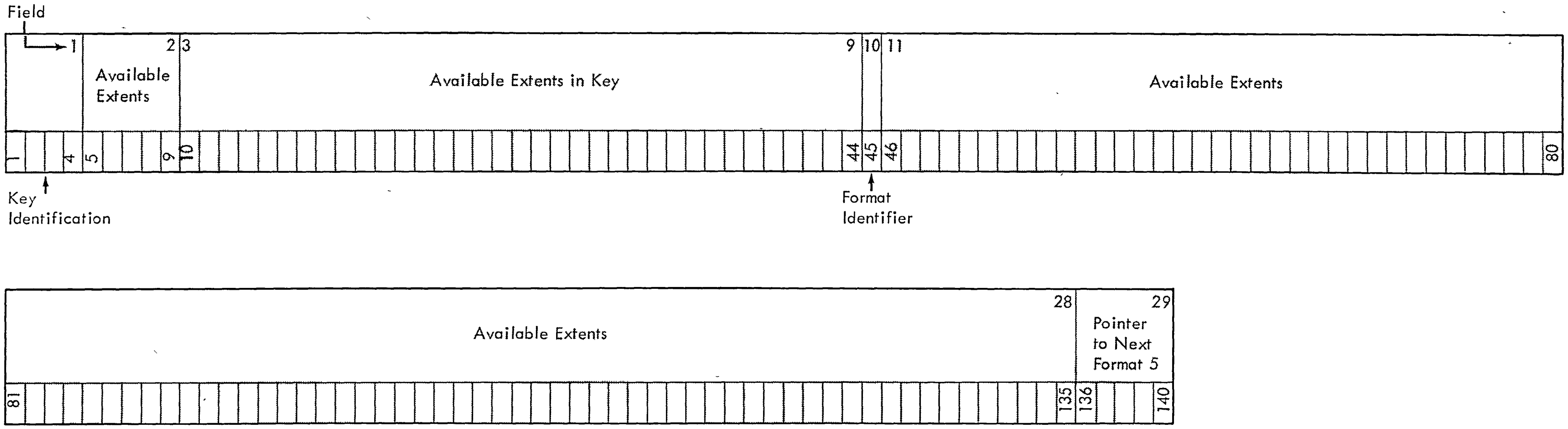


Figure 48. Standard DASD File Label, Format 5

Format 5: This format is used for Direct Access Device Space Management (DADSM) only. (Shaded areas are not processed by DOS/360)

| FIELD | NAME AND LENGTH | DESCRIPTION |
|---|---|---|
| 1. | KEY IDENTIFICATION<br>4 bytes | Each of these four bytes is a hex 05. |
| 2. | AVAILABLE EXTENT<br>5 bytes | Indicates an extent of space available for allocation to a data file. The first two bytes are relative track address. The next two are the number of full cylinders included in the extent. The last byte is the number of tracks in addition to the cylinders in the extent. |
| 3-9 | AVAILABLE EXTENTS IN KEY<br>35 bytes | These fields are identical to Field 2. They are in relative track address sequence. |
| 10. | FORMAT IDENTIFIER<br>1 byte EBCDIC numeric | 5 = Format 5 |
| 11-28 | AVAILABLE EXTENTS<br>90 bytes | These fields are the same as Field 2. There are 26 available extent fields in the Format 5 label. |
| 29. | POINTER TO NEXT FORMAT 5 | Contains the address (in the form CCHHR) of the next Format 5 file label if one exists. |

Note: Format 5 label used by OS/360 only.

Indexes to systems reference library manuals are consolidated in the publication DOS Master Index, GC24-5063. For additional information about any subject listed below, refer to other publications for the same subject in the Master Index. For a consolidated index of all the DOS LIOCS Volumes, refer to DOS LIOCS Volume 1, Introduction, GY24-5020.

GY24-5088-4

DOS LIOCS Volume 3: SAM and DAM for DASD (S360-30) Printed in U.S.A. GY24-5088-4

**International Business Machines Corporation**
**Data Processing Division**
**1133 Westchester Avenue, White Plains, New York 10604**
**[U.S.A. only]**

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

## Reader's Comments Form

DOS LIOCS Volume 3
SAM and DAM for DASD

GY24-5088-4

Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Please give specific page and line references with your comments when appropriate. All comments will be handled on a non-confidential basis. Copies of this and other IBM publications can be obtained through IBM branch offices.

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

● What is your occupation? ______________________________

● How do you use this publication:

| | | | |
|---|---|---|---|
| As an introduction to the subject? | □ | As an instructor in a class? | □ |
| For advanced knowledge of the subject? | □ | As a student in a class? | □ |
| For information about operating procedures? | □ | As a reference manual? | □ |

### Your comments:

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. *If you would like a reply, please supply your name and address on the reverse side of this form.*

GY24-5088-4

Staple

### Your comments, please . . .

This publication is one of a series that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, help us produce better publications for your use. Each reply is carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

*Please note:* Requests for copies of publications and for assistance in using your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Cut Along Line

Fold Fold

FIRST CLASS
PERMIT NO. 170
ENDICOTT, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

**IBM Corporation**
**P. O. Box 6**
**Endicott, N. Y. 13760**

Attention: Programming Publications, Dept. G60

Fold Fold

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

*Your Name* ______________________

*Company Name* ______________________

*Department* ______________________

*Street Address* ______________________

*City* ______________________

*State* ______________ *Zip Code* ______________

IBM

**International Business Machines Corporation**
**Data Processing Division**
**1133 Westchester Avenue, White Plains, New York 10604**
**[U.S.A. only]**

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

DOS LIOCS Volume 3: SAM and DAM for DASD (S360-30) Printed in U.S.A. GY24-5088-4