LY24-5211-0
File No. S370/4300-30

Program Product

VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 3 DAM and ISAM

Program Number 5746-XE9

Component Numbers 5745-SC-DAM
5745-SC-ISM

Release 2

SUMMARY OF AMENDMENTS

This manual contains information previously published in DOS/VSE LIOCS Volume 3, DAM and ISAM Logic, SY33-8561. Changes reflect support for:

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

The major result of this support is the removal of the DAM OPEN B-transients, the functions of which are supplied by common routines for opening DASD files executing in the SVA. The logic of these routines is described in VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 4, LY24-5212.

In addition, the DAM DTF Extension is modified to allow dynamic initialization of the DTF to reflect the type of DASD the file resides on, no matter what device type the DTFDA was assembled with. Accordingly, the user's program will be modified to use a DAM logic module residing in the SVA that provides the proper support required to access the actual DASD device; that is, user specification of DAMOD or DAMODV macros is not necessary.

First Edition (October 1979)

This edition, LY24-5211-0, applies to Release 2 of IBM VSE/Advanced Functions, Program Number 5746-XE9. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

PREFACE

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

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

Volume 2: SAM, LY24-5210.

Volume 3: DAM and ISAM, LY24-5211.

Volume 4: SAM for DASD, LY24-5212.

This third volume is intended mainly for persons involved in program maintenance and for systems programmers who are altering the program design. Logic information is not necessary for the operation of the programs described.

General routines that apply to more than one access methcd cr more than one file type are described in Volumes 1 and 4. These routines include open/close and a number of transient routines. References to Volumes 1 and 4 are made whenever required for a good understanding of the topics discussed.

This volume of the VSE/Advanced Functions LIOCS manuals consists of three parts:

1. LIOCS support for DAM files
2. LIOCS support for ISAM files
3. Charts.

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

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

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

Effective use of this publication requires an understanding of IBM VSE/Advanced Functions operation and the Assembler language and its associated macro definition language. Reference publications for this information are listed below.

PREREQUISITE PUBLICATIONS

- VSE System Data Management Concepts, GC24-5209.
- VSE/Advanced Functions Macro User's Guide, SC24-5210.
- VSE/Advanced Functions Macro Reference, SC24-5211.
- OS/VS - DOS/VSE - VM/370 Assembler Language, GC33-4010.
- VSE/Advanced Functions Diagnosis Reference LIOCS Volume 1, General Information and Imperative Macros, LY24-5209.

RELATED PUBLICATIONS

- VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 2, SAM, LY24-5210.
- VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 4, SAM for DASI LY24-5212.
- VSE/Advanced Functions Diagnosis Reference: Supervisor, LY33-9091.
- VSE/Advanced Functions Messages, SC33-6098.
- IBM System/370 and 4300 Processors Bibliography, GC20-0001.

CONTENTS

FIGURES

Direct Access (DA) files refer to files contained on 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 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 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.

DEVICE INDEPENDENT SUPPORT

Device independent support is provided in LIOCS by dynamically extending the user DTFDA into the virtual area within the user partition (see Figure 2), and by linking the user DTFDA to the device independent version of the logic module in the SVA (shared virtual area). The user must provide sufficient dynamically allocatable space in his partition for the device independent extension to the DTFDA table.

The device independent versions of the logic module in the SVA are reenterable and therefore sharable between partitions. If the linkage to the original module is already coded read-only, the user supplied save area is not used.

The device independent versions of the logic modules in the SVA are supersets of the functions needed to process the DTFDA being opened. Supersetting of RPS and non-RPS logic modules is not supported.

The CCB CCW address and the module linkage field in each DTFDA are modified to point to the DTFDA extension and the device independent version of the logic module in the SVA. Each DTFDA has two indicators set on by OPEN: one indicates that the DTF has been extended into the virtual area, the other indicates that the device supports RPS (if that is the case).

The DTFDA extension contains a CCW build area necessary to construct channel programs. In addition, it contains a save area to allow reentrant imperative macro calls to the device independent version of the logic modules and to preserve DTFDA information to reestablish the original DTFDA at CLOSE time. The original DTFDA is used for all other information, except the channel program.

DTFDA MACRO

Whenever a file cf 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 lcgical IOCS as shown in Figure 1. The DTF table is completed by the channel program builder subroutine in the DA logic module. This subroutine builds the channel program CCWs to process the file and inserts them into the DTFDA DTF table extension (see VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 4). The number and specific nature of the CCWs varies imperative macros used with the file. See Figures 14 and 15 for a summary of 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 (0-F)		Command Control Block (CCB).
	IJIMOD	16 (10)	0	1 = Trailer labels
			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 (11-13)		Address of logic module.
		20 (14)		DTF type for OPEN/CLOSE (X'22' = direct access files).
	IJISWI	21 (15)	0	1 = Output; 0 = Input.
			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 (16-1C)		Filename (DTF Name).
	IJIDVTP	29 (1D)		Device Type. X'00' = 2311 X'07' = 3350 X'01' = 2314, 2319 X'08' = 3340 general X'04' = 3330-1, 3330-2 X'09' = 3340 35MB X'05' = 3330-11 X'0A' = 3340 70MB
	IJIUNT	30-31 (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).
	IJIRPS	32 (20)	0	Not used
			1	1 = RPS device and RPS=YES in FOPT macro
			2-6	Not used
			7	1 = Extended DTF for RPS
	IJIULB	33-35 (21-23)		Address of user's label routine.
	IJIUXT	36-39 (24-27)		Address of user's routine for processing EXTENT information.
	IJIRELPT	40 (28)		Pointer to relative address area: $\frac{\&Filename.P - \&Filename}{2}$
	IJIERC	41-43 (29-2B)		Address of a 2-byte field in which IOCS can store the error condition or status codes.

Figure 1. DTFDA Table (Part 1 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
	IJITST	44-45 (2C-2D)		Macro code switch for internal use: X'0000' = READ ID X'0001' = READ KEY X'0002' = WRITE ID X'0003' = WRITE KEY X'0004' = WRITE RZERO X'0005' = WRITE AFTER
	IJIBPT	46-47 (2E-2F)		Pointer to channel program build area (&Filename.B) minus 32.
	IJICB2	48-63 (30-3F)		Control seek CCB.
&Filename.Z	IJICCW	64-71 (40-47)		Control Seek CCW for overlap seek routine.
	IJIXMD	72-75 (48-4B)		Channel program builder instruction: XI 36(2),C'0'
	IJIMSZ	76-77 (4C-4D)		Maximum data length for FIXUNB or UNDEF records; BLKSIZE for VARUNB or SPNUNB records.
	IJISPT	78 (4E)		Pointer to READ ID string (Filename.0); X'00' if no READ ID issued.
		79 (4F)		Pointer to READ KEY string (Filename.1); X'00' if no READ KEY issued.
		80 (50)		Pointer to WRITE ID string (Filename.2); X'00' if no WRITE ID issued.
		81 (51)		Pointer to WRITE KEY string (Filename.3); X'00' if no WRITE KEY issued.
		82 (52)		Pointer to WRITE RZERO string (Filename.4); X'00' if no WRITE RZERO issued.
		83 (53)		Pointer to WRITE AFTER string (Filename.5); X'00' if no WRITE AFTER issued.
	IJITRK	84-85		Track constant: Key length = 0 / Key length ≠ 0 H'0' for 2311 / H'20' for 2311 H'0' for 2314/2319 / H'45' for 2314/2319 H'135' for 3330 / H'191' for 3330 H'167' for 3340 / H'242' for 3340 H'185' for 3350 / H'267' for 3350
	IJIRIC	86-87 (56-57)		2311: H'61' 2314/2319: H'101' 3330: H'135' 3340: H'167' 3350: H'185'

Figure 1. DTFDA Table (Part 2 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
	IJILAT	88 (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 (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 (5A-5F)		Label track address, YBCCHH, 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.

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 (60-8F)		Basic CCWs to build channel program (see Figure 9).
		144-183 (90-B7)		Basic CCWs for undefined length or formatting macros (see Figure 9).
	IJIVIT	184-185 (B8-B9)		Instruction to give record length to user if record length is undefined. (NOPR 0 if no RECSIZE specified.)
	IJIFRU	186-187 (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 (BC-C0)		Work area (used for R0 address - CCHH0).
&Filename.K	IJICNT	193-200 (C1-C8)		Work area (used for R0 data field).
&Filename.C	IJICTS	201-208 (C9-D0)		Work area (included only for spanned or variable records for record count field).

Figure 1. DTFDA Table (Part 3 of 6)

The channel program builder strings are generated following the DTFDA table, and preceding the channel program building area. (See Figures 9 and 10 for the channel program builder string to be used for each macro.)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&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.

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: 1. Eight CCWs if AFTER is not specified. 2. Eight CCWs if spanned or variable length records and AFTER=YES is specified. 3. Seven CCWs if undefined or fixed records and AFTER=YES is specified.
		Variable		Area to build: 1. Eight CCWs if AFTER is not specified and VERIFY=YES is specified. 2. Eight CCWs if spanned or variable length records and AFTER=YES and VERIFY=YES are specified. 3. Five CCWs if undefined or fixed records and AFTER=YES and VERIFY=YES are specified.

Figure 1. DTFDA Table (Part 4 of 6)

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.

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: 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):
				2311: X'00000001'
				2314, 2319: X'00000001'
				3330: X'00001300'
				3340: X'00000C00'
				3350: X'00001E00'
		4 bytes		Alteration factor for C2 in SEEKADR (see bytes 112-119):
				2311: X'0000000A'
				2314, 2319: X'00000014'
				3330: X'00000013'
				3340: X'0000000C'
				3350: X'0000001E'

Figure 1. DTFDA Table (Part 5 of 6)

DTF Assembly Label	Mcdule DSECT Label	Bytes	Bits	Function
		4 bytes		Alteration factor for H1 in SEEKADR (see bytes 112-119): 2311: X'00000001' 2314, 2319: X'00000001' 3330: X'00000001' 3340: X'00000001' 3350: X'00000001'
		Variable to end of DTF table		DSKXTNT table composed of a variable number of 8-byte entries containing extent information in the following format: Bytes 0-2 TTT2 - cumulative number of tracks in the DSKXTNT table entries up to and including the current entry. 3 M - volume sequence number. 4 B - bin number (0 for disk devices). Bytes 5-7 TTT1 - relative track number of lower limit of this entry. A 1-byte end-of-table indicator containing X'FF' follows the last entry in the DSKXTNT table.

Numbers in parentheses are displacements in hexadecimal notation.

Figure 1. DTFDA Table (Part 6 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
	IJIXBLD	0 (0)		CCW build area.
	IJIXSPTR	176 (B0)		Address of original channel program.
	IJIXSVMP	180 (B4)		Address of original logic module.
	IJISAVA	184 (B8)		72-byte register save area.
		266 (10A)		Not used.
	IJISECV0	267 (10B)		Sector work byte.
	IJISECV1	268 (10C)		Sector work byte.
	IJISECV2	269 (10D)		Sector work byte.
	IJIXSEC	270 (10E)		RPS CCW.
	IJIXSS0	278 (116)		RPS CCW.
	IJIXSSX	286 (11E)		RPS CCW.
	IJIXSSNF	294 (126)		RPS CCW.
	IJIXSTRG	302 (12E)		PESC byte string area.
	IJIXSPT	382 (17E)		Displacement to strings.
	IJIXMCYL	390 (186)		Maximum cylinders per volume.
	IJIXTFAC	392 (188)		Tolerance factor.
	IJIFLG1	394 (18A)		Flag byte.
	IJIXUSTF	395 (18B)		Indicator needed to use tolerance factor.
	IJIFLG2	396 (18C)		Flag byte

Numbers in parentheses are displacements in hexadecimal notation.

Figure 2. Device Independent DTF Extension for DTFDA

DTFPH MACRO

Figure 3 illustrates the DTF table generated by the DTFPH macro when the parameters DEVICE=xxxx 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.

If the device being opened has RPS

capability and the SYSGEN option RPS=YES has been specified, OPEN will set on bit 1 in byte 32 of the DTF. If the user wishes to make use of the RPS feature of a device, he must provide the appropriate Read Sector and Set Sector CCWs in his channel programs as is done in the Direct Access Method. (See the appropriate hardware manual for your device, for format of Sector CCWs and a write-up on Rotational Position Sensing.)

Bytes	Bits	Function
0-15 (0-F)		CCB.
16 (10)		X'08' indicates DTF relocated by OPENR.
17-19 (11-13)		3X'00'
20 (14)		DTF type (X'23').
21 (15)		Option codes.
	0	1 = Output, 0 = Input.
	1-7	Not Used.
22-28 (16-1C)		Filename.
29 (1D)		Device type code: X'00' = 2311 X'01' = 2314, 2319 X'04' = 3330-1, 3330-2 X'05' = 3330-11 X'07' = 3350 X'08' = 3340 general X'09' = 3340 35MB X'0A' = 3340 70MB.
30-31 (1E-1F)		Logical unit address of first volume containing the file.
32 (20)	0	Not Used.
	1	1 = Device supports RPS.
	2	1 = Version 3 DTF.
	3-7	Reserved for future use.
33-35 (21-23)		Address of user label routine.
36-39 (24-27)		Address of user routine to process EXTENT information.

Numbers in parentheses are displacements in hexadecimal notation.

Figure 3. DTFPH Table for DAM Files

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

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

3. Relative addressing (RELTYPE=DEC) - 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, and so on.

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

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

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

3. Relative addressing (RELTYPE=DEC) - 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 key 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. (Refer to the section "Multiple Track Search".)

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 16 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 17). 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. See Figure 4 for a summary of these conditions and the addresses returned. When the problem program references a record by ID or KEY and does not specify the SRCHM (search multiple tracks) option, IOCS returns the ID of the next record under normal conditions. If the user is 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 before 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.

Read/Write Function	SRCHM = YES			SRCHM ≠ YES				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	EOF record read	
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), each containing X'FF'
Relative addressing (HEX) -- 4 bytes (TTTR), each containing X'FF'
Relative addressing (DEC) -- 10 bytes, each containing decimal 9

Figure 4. Record ID Returned to IDLOC

CONTROL FIELD - SPANNED RECORDS

Figure 5 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 when 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.

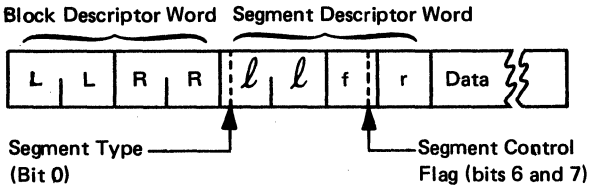
Normal Segment: The term normal segment refers to any segment of the kind described by the segment control flag.

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 6.



Block Descriptor Word

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

RR = Used by the system.

Segment Descriptor Word

= 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 5. 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. UNDEF records: This bit is set on under the following conditions: • 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. • 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. • 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. VARUNB records: This bit is set on under the following conditions: • 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. • 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. • 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.

Figure 6. Error/Status Indicator (Part 1 of 4)

Byte	Bit	Error/Status Indicator	Explanation
			SPNUNB records: This bit is set on under the following conditions: • 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. • When a WRITE ID or KEY macro is issued, the wrong-length record indicator is set if: 1. The LL count for the record to be written exceeds the value specified by the BLKSIZE parameter in the DTFDA macro. 2. The data length of the record to be written exceeds the data length of the original record. In either of the above cases the record is written with the low-order bytes truncated. 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. 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. • 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	Non-data 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.
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.	

Figure 6. Error/Status Indicator (Part 2 of 4)

Byte	Bit	Error/Status Indicator	Explanation
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 VSE/Advanced Function 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. If SRCHM=YES is specified, the end-of-cylinder indicator (bit 2 in byte 1) is also set on. 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 not 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).
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.

Figure 6. Error/Status Indicator (Part 3 of 4)

Byte	Bit	Error/Status Indicator	Explanation
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; cylinder 403, head 18 for a 3330 model 1 or 2; cylinder 807, head 18 for 3330 model 11; cylinder 347 or 695, head 11, for a 3340 with 35MB or 70MB respectively; and cylinder 554, head 29 for a 3350. These limits allow for the reserved alternate track area. 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 6. 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.

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 LIOCS 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) + DL

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 (KL)
- 2-byte data length (DL)

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. (See Figure 5 for the format of the 8-byte control field.) Therefore, the blocksize (BLKSIZE) must be equal to:

Maximum DL + 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 (KL)
- 2-byte data length (DL)

Note: For VARUNB and SPNUNB records, DL includes the 8-byte control field.

DAM LOGIC MODULES

VSE/Advanced Functions Release 2 provides preassembled DAM superset logic modules that are loaded into the SVA during IPL. DAM file open processing automatically links the DTFDA to the proper logic module; therefore, use of either the DAMOD or DAMODV macro, which was required in DOS/VSE, is not necessary. However, if either macro is submitted (and properly specified), it will cause no compilation problems; it will simply be ignored by OPEN.

DAM LOGIC MODULE MACROS

Note: This discussion of the DAMOD and DAMODV macros and their operands pertains to programs written prior to VSE/Advanced Functions Release 2 and is for reference purposes only.

Two macros generate independent logic modules needed to process records under the Direct Access Method. The macros are:

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

Operation	Operand	Remarks
DAMOD or DAMODV	AFTER=YES	When WRITE with the operand AFTER or RZERO is used.
	ERREXT=YES	Required if non-data transfer error conditions are to be indicated in the ERRBYTE status bits
	FEOVD=YES	Required if support for sequential disk end-of-volume records is desired
	HOLD=YES	Required if the track hold function is to be used
	IDLOC=YES	Required if IDLOC specified in DTFDA
	RDONLY=YES	Required if a read only module is to be generated
	RECFORM= {FIXUNB | UNDEF | VARUNB | SPNUNB}	Describes record format
	RELTRK=YES	Required if the module is to process relative identifiers along with physical identifiers
	RPS=SVA	To assemble RPS logic modules
	SEPASMB=YES	If the module is assembled separately

AFTER=YES: This operand generates a logic module that can perform a formating WRITE (count, key, and data). It performs the functions required by WRITE filename,AFTER; and WRITE filename,RZERO. The module also processes any files in which the AFTER operand is not specified in the DTF.

ERREXT=YES: Include this operand if unrecoverable I/O errors (occurring before a data transfer takes place) are to be indicated to your program in the bytes named in the DTF ERRBYTE operand.

FEOVD=YES: This operand is specified if

coding is to handle end-of-volume records. It should be specified only if you are reading a file built using DTFSD and the FEOVD macro.

HOLD=YES: This operand is specified if the track hold function is:

- specified at system generation time, and
- included in the DTFDA macro, and
- used when the file is referenced.

IDLOC=YES: This operand generates a logic module that returns record identifier (ID) information to you. The module also processes any files in which the IDLOC operand is not specified in the DTF.

RDONLY=YES: This operand causes a read-only module to be generated. Whenever this operand is spcified, any DTF used with the module must have the same operand.

RECFORM=(FIXUNB|SPNUNB|UNDEF|VARUNB): If UNDEF is specified, the logic module generated can handle both unblocked fixed-length and undefined records. If the operand is omitted or if FIXUNB is specified, the logic module generated can handle only fixed-length unblocked records. If SPNUNB is specified, the module can handle both format V (variable length) and spanned format records. If VARUNB is specified, the module can handle only format V records.

RELTRK=YES: This operand generates a logic module that can process with both physical and relative identifiers. If the operand is omitted, the module can process only with physical identifiers.

RPS=SVA: This operand causes the RPS logic modules to be assembled.

SEPASMB=YES: Include this operand only if the module will be assembled separately. This causes a CATALR card with the module name (standard or user-specified) to be punched ahead of the object deck and the module name to be defined as an ENTRY point in the assembly. If the operand is omitted, the assembler assumes the module is being assembled with the problem program and no CATALR card is punched.

Standard DAMOD Names

Each name begins with a 3-character prefix (IJI) and continues with a 5-character field corresponding to the options permitted in the generation of the module.

DAMOD name = IJIabcde

a = F RECFORM=FIXUNB
  = B RECFORM=UNDEF (handles both UNDEF and FIXUNB)
  = S RECFORM=SPNUNB
  = V RECFORM=VARUNB

b = A AFTER=YES,RPS=SVA omitted
  = W AFTER=YES,RPS=SVA
  = Z neither is specified

c = E IDLOC=YES and FEOVD=YES
  = I IDLOC=YES
  = R FEOVD=YES
  = Z neither is specified

d = H ERREXT=YES and RELTRK=YES
  = P ERREXT=YES
  = R RELTRK=YES
  = Z neither is specified

e = W HOLD=YES and RDONLY=YES
  = X HOLD=YES
  = Y RDONLY=YES
  = Z neither is specified

Subset/Superset DAMOD Names

The subsetting and supersetting allowed for DAMOD name is shown below. Five parameters allow supersetting. For example, the module IJIBAIZZ is a superset of the module with the name IJIFAZZZ.

			+	+	+	+	+
I	J	I	B	A	E	H	Y
			F	Z	I	P	Z
			+		Z	Z	+
			S		+	+	W
			V		E	H	Y
					R	R	
					Z	Z	

\+ Subsetting/supersetting permitted

Notes:

1. The module IJIBAEHW will cause assembly error message IPK154 TOO MANY ENTRY SYMBOLS. The valid entry points for this module total more than 100, which is the maximum for assembler language. Specify less parameters for DAMOD if you can. Otherwise, you must assemble your own module for your program.
2. Your program can have only one DAMOD for fixed unblocked or undefined records and/or only one DAMOD for variable unblocked or spanned unblocked records; otherwise, duplicate name flagging occurs during assembly time.

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 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 perfcrming their functions.

When the user issues a READ or WRITE macro instruction for a file, program control transfers to a logical IOCS routine that 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, Chart AA

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 IJISOVP subroutine, to calculate the physical address and to determine the symbolic unit.

Next, this routine branches to the channel program builder to build the CCW chain for the macro that is being processed (refer to Figure 14 or 15). 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, and the macro is READ,ID or READ,KEY, the track held is not freed by DAMOD. This must therefore be done in the user program with the FREE macro. Control then returns to the problem program.

DAMOD: WAITF Macro, Charts AB-AE

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 4). If IDLOC is specified, a test is made for the type of macrc 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 contrcl to the problem program via linkage register 14.

DAMOD: CNTRL Macro, Chart AF

Objective: To perform non-data operations on a file. For a disk device, a seek operation is executed.

Entry: From the CNTRL macro.

Exit: To the prcblem program.

Method: This routine saves the user's registers, and then branches and links to the IJISOVP subroutine, to calculate the physical address and to determine the symbolic unit. When the non-data transfer 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 AF

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 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 AK-AN

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 14 or 15, Chart I for READ ID; Chart J for READ KEY.

The byte count in the basic read data CCW (see Figure 11) 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 calculate the physical address and to determine the symbolic unit. 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. See Figure 14 or 15, Chart I for READ ID; Chart J for READ KEY.

The byte count in the basic read data CCW (see Figure 11) is set equal to the length specified by the user in the BLKSIZE= parameter for the DTFDA macro. The IJISOVP is then entered to calculate the physical address and to determine the symbolic unit. 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 mulitsegment 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 7); these bytes are later restored after the contrcl 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. See Figure 14 or 15, 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 11) 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 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 calculate the physical address and to determine the symbolic unit. 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. See Figure 14 or 15, 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 calculate the physical address and to determine the symbolic unit. 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 (see Figure 14 or 15, 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

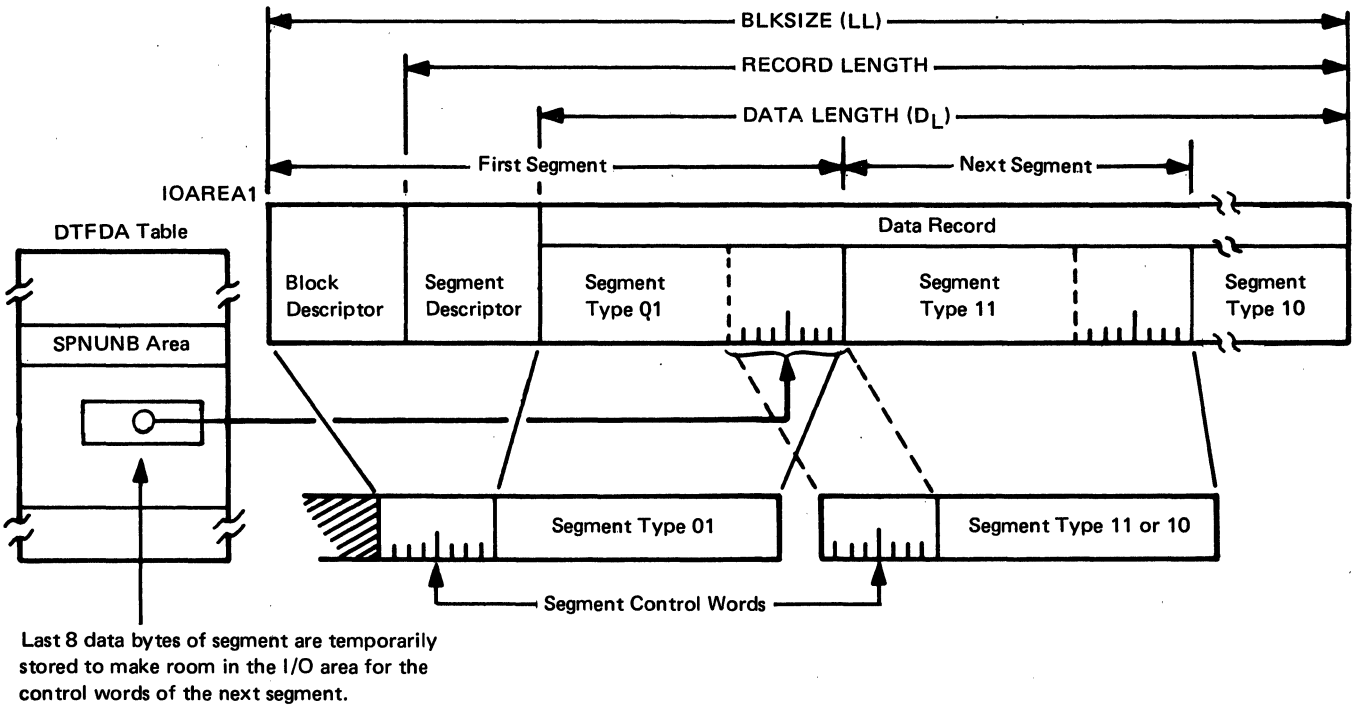


Figure 7. Multisegment Spanned Record

multiple DASD segments and, if in multiple segments, the lengths of the individual segments must be known. 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 ccmpared 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 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 7). 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 - VARUNP Records: The byte count of the basic read data CCW (see Figure 11) is set equal to the block length (LL) of the record to be written, and the IJISOVP subroutine is entered to calculate the physical address and determine the symbolic unit. 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 (see Figure 14 or 15, 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 VSE/Advanced Functions.

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 length of the largest single record that fits on a track 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 IJISBLD subroutine is entered to 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 AF

Objective: To perform non-data operations on a file. For a disk device, a seek 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 IJISOVP subroutine, to calculate the physical address and to determine the symbolic unit. When the non-data transfer 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 AF

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 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 BA - BC

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 4). 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.

DAMOD and DAMODV: Channel Program Builder Subroutine, Chart AJ

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

Note: Figures 14 and 15 provide 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 (see Figures 14 and 15). 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 macrc 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 8.

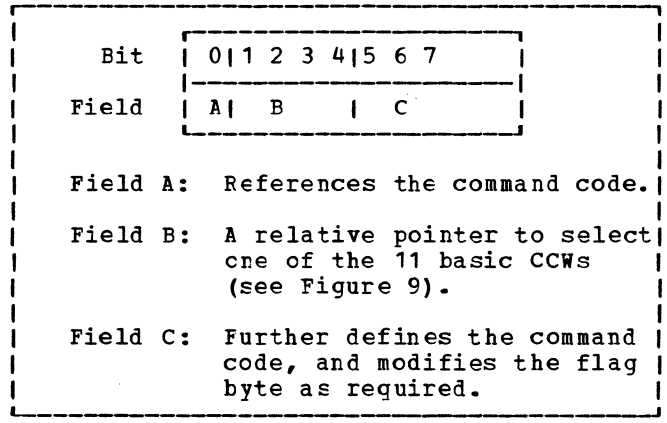


Figure 8. DAM Descriptor Byte

Because the first CCW in a direct access method 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 (see Figures 9 and 10). 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 11) and tests to determine if the CCW is to be a Transfer In Channel (TIC). Figure 11 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 12). 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 12). 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'87181487189l9E'
	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'8718148718918718l5'
	KEYLEN	DC X'8718A987182D'	DC X'8718AB87182D'	DC X'8718AB9187182B15'	DC X'8718148718AB9187182B15'
	IDLOC	DC X'87189187181l9E'	DC X'87189387181l9C'	DC X'87189187181l9E'	DC X'8718148718918718119E'
	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'A7188818910A1815'	DC X'A7189A8A10148A18910A1815'
	IDLOC	DC X'8F18918F18119E'	DC X'8F18938F18139C'	DC X'A7188F18910A18119E'	DC X'A7180A18140A18910A18119E'
	SRCHM,IDLOC	DC X'A7189A8810918F1815'	DC X'A718B8881093 8F1815'	DC X'A7189A8810910A1815'	DC X'A7189A8A10148A18910A1815'

Macros WRITE AFTER and WRITE RZERO use the same strings. If AFTER is not specified in the DTFDA 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'
		KEYLEN	DC X'C718B18718CBAB95'
VERIFY	DC X'C718B18718C9C7183187184D'	No options	DC X'C718B18718CB91C7183187184B15'
		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 9. DAM Channel Program Builder Strings Without RPS Support

Macro	Option	FIXUNB	UNDEF	VARUNB	SPUNB
READ ID	No options	DC X'58871895'	DC X'C718BF14'	DC X'871816'	DC X'871816'
	KEYLEN	DC X'588718AD'	DC X'C718BF2C'	DC X'87182A16'	DC X'87182A16'
	IDLOC	DC X'588718919E'	DC X'C718BF129C'	DC X'8718128E'	DC X'8718129E'
	KEYLEN,IDLOC	DC X'588718A99E'	DC X'C718BF2A9C'	DC X'87182A129E'	DC X'87182A129E'
READ KEY	No options	DC X'8F1814'	DC X'BF8F1014'	DC X'487A188F1816'	DC X'48A7180A1816'
	SRCHM	DC X'48A718881814'	DC X'48A718B8881014'	DC X'48A718881816'	DC X'48A7188A1816'
	IDLOC	DC X'8F18979E'	DC X'BF8F10129C'	DC X'48A7188F18129E'	DC X'48A7180A18129E'
	SRCHM,IDLOC	DC X'48A7189A881014'	DC X'48A718B8881014'	DC X'48A7189A881016'	DC X'48A7189A8A1016'
WRITE ID (No VERIFY)*	No options	DC X'58871814'	DC X'871895'	DC X'871895'	DC X'8718977858871895'
	KEYLEN	DC X'5887182C'	DC X'8718AD'	DC X'8718AB95'	DC X'87189778588718AB95'
	IDLOC	DC X'588718979E'	DC X'8718939C'	DC X'8718919E'	DC X'87189778588718AB919E'
	KEYLEN,IDLOC	DC X'588718AF9E'	DC X'8718AB9C'	DC X'8718AB919E'	DC X'87189778588718AB919E'
WRITE ID (VERIFY)	No options	DC X'5887189158871815'	DC X'8718936858871815'	DC X'8718916858871815'	DC X'87189778588718915887 1815'
	KEYLEN	DC X'588718A95887182D'	DC X'8718AB685887182D'	DC X'8718AB91685887182B15'	DC X'87189778588718AB915887182B15'
	IDLOC	DC X'58871891588718119E'	DC X'87189368588718139C'	DC X'8718916858871 19E'	DC X'87189778588718915887 18119E'
	KEYLEN,IDLOC	DC X'588718A9588718299E'	DC X'8718AB685887182B9C'	DC X'8718AB91685887182B119E'	DC X'87189778588718AB915887182B119E'
WRITE KEY (No VERIFY)	No options	DC X'8F1895'	DC X'8F1895'	DC X'48A7188F1895'	DC X'48A7180A189778580A1895'
	SRCHM	DC X'48A718881895'	DC X'48A718881895'	DC X'48A718881895'	DC X'48A7189A8A109778588A1895'
	IDLOC	DC X'8F18919E'	DC X'8F18939C'	DC X'48A7188F1895'	DC X'48A7180A189778580A18919E'
	SRCHM,IDLOC	DC X'48A7189A881095'	DC X'48A718B8881095'	DC X'48A7189A881095'	DC X"48A7189A8A109778588A1895'
WRITE KEY (VERIFY)	No options	DC X'48C718D752C718B5'	DC X'8F189368588F1815'	DC X'48A7188F189168580A1815'	DC X'48A7180A189778580A1891580A1815'
	SRCHM	DC X'48A718881891 68588F1815'	DC X'48A7188818936 8588F1815'	DC X'48A7188818916 8580A1815'	DC X'48A7189A8A109778588A1891580A1815'
	IDLOC	DC X'8F189168588F18119E'	DC X'8F189368588F181393'	DC X'48A7188F189168580A18119E'	DC X'48A7180A189778580A1891580A18119E'
	SRCHM,IDLOC	DC X'48A7189A88109168588F1815'	DC X'48A718B888109368588F1815'	DC X'48A7189A88109168580A1815'	DC X'48A7189A8A109778588A1891580A1815'

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

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

DC X'48C718D752C718B5' WRITE RZERO
DC X'48C71834' READ RZERO

And one of the following strings:			
No VERIFY	DC X'48C718B18718CD'	No options	DC X'48C718B18718CB95'
		KEYLEN	DC X'48C718B18718CBAB95'
VERIFY	DC X'48C718B18718C9C7183187184D'	No options	DC X'48C718B18718CB91C7183187184B15'
		KEYLEN	DC X'48C718B18718CBAB91C7183187184B2B15'

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 10. DAM Channel Program Builder Strings with RPS Support

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',&Filename.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',5	Read R0 count into work 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 D D) into work area in DTF table (SPNUNB records).
1010	X'11',&Filename.B+32,X'40',Length of the largest single record that fits on a track.	Control erase of track.

Figure 11. Basic CCWs for DAM 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 multitrack, 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 multitrack, 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 multitrack, 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 11 under the column heading Field B. This field contains the relative pointer to the basic CCW (see Figure 11).

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

Figure 12. DAM 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 to 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 13. Example of DAM Channel Program for a WRITE ID Macro

The following discussion describes how the DAM 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 13 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 (see Figure 9 or 10). 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 11 illustrates that the CCW for a descriptor byte with a B-field = 0000 is a Search Identifier Equal CCW. Figure 12 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 (see Figure 13).

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 10 illustrates that a descriptor byte of this kind with a B-field = 0011 supplies the CCW, TIC to * - 8 (see Figure 13 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 (see Figure 11). Because the A-field = 1 and the C-field = 011, Figure 12 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 11, 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 (see Figure 12).

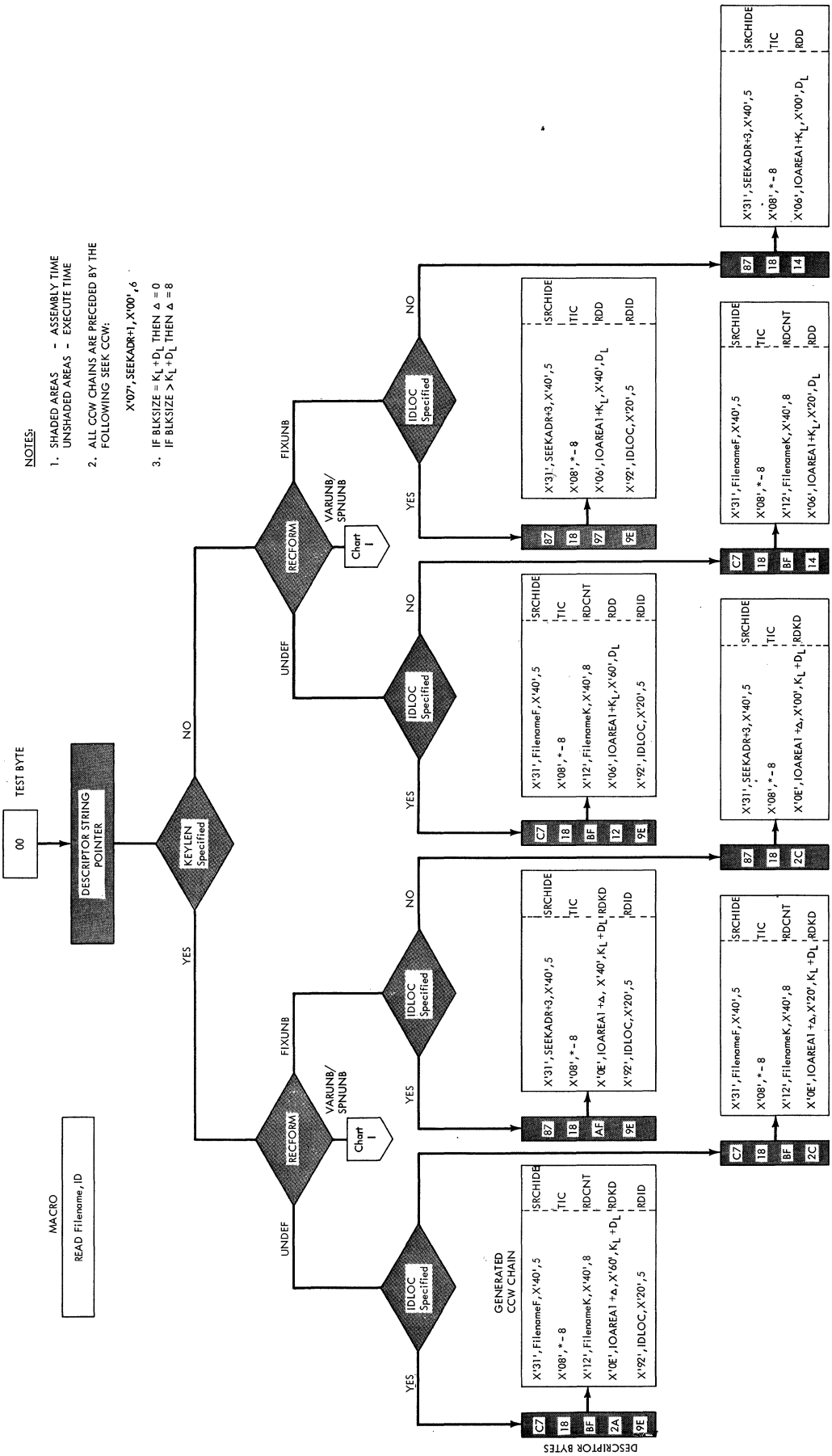


Figure 14. DAM Channel Programs Without RPS Support (Part 1 of 14)

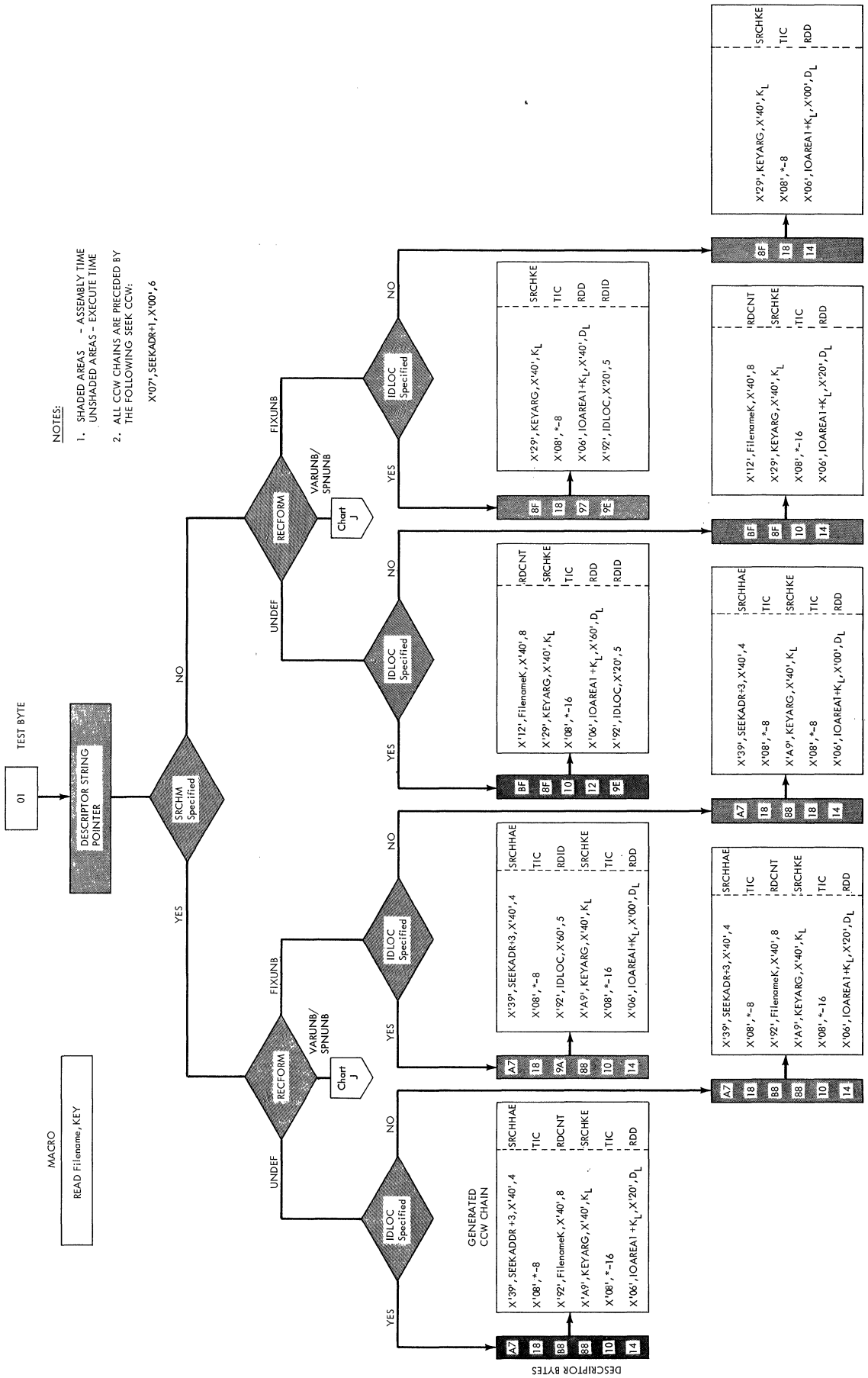


Figure 14. DAM Channel Programs Without RPS Support (Part 2 of 14)

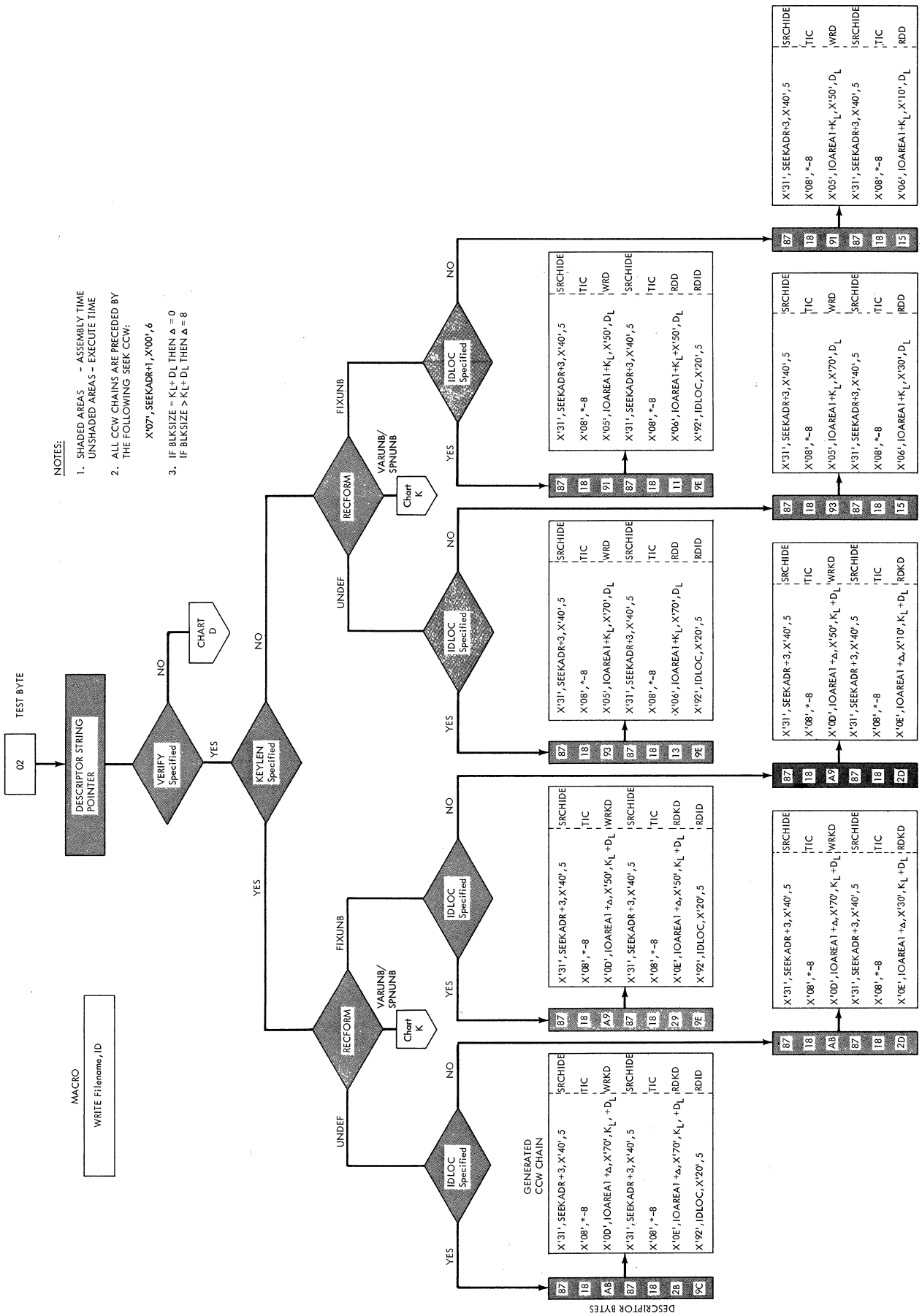


Figure 14. DAM Channel Programs Without RPS Support (Part 3 of 14)

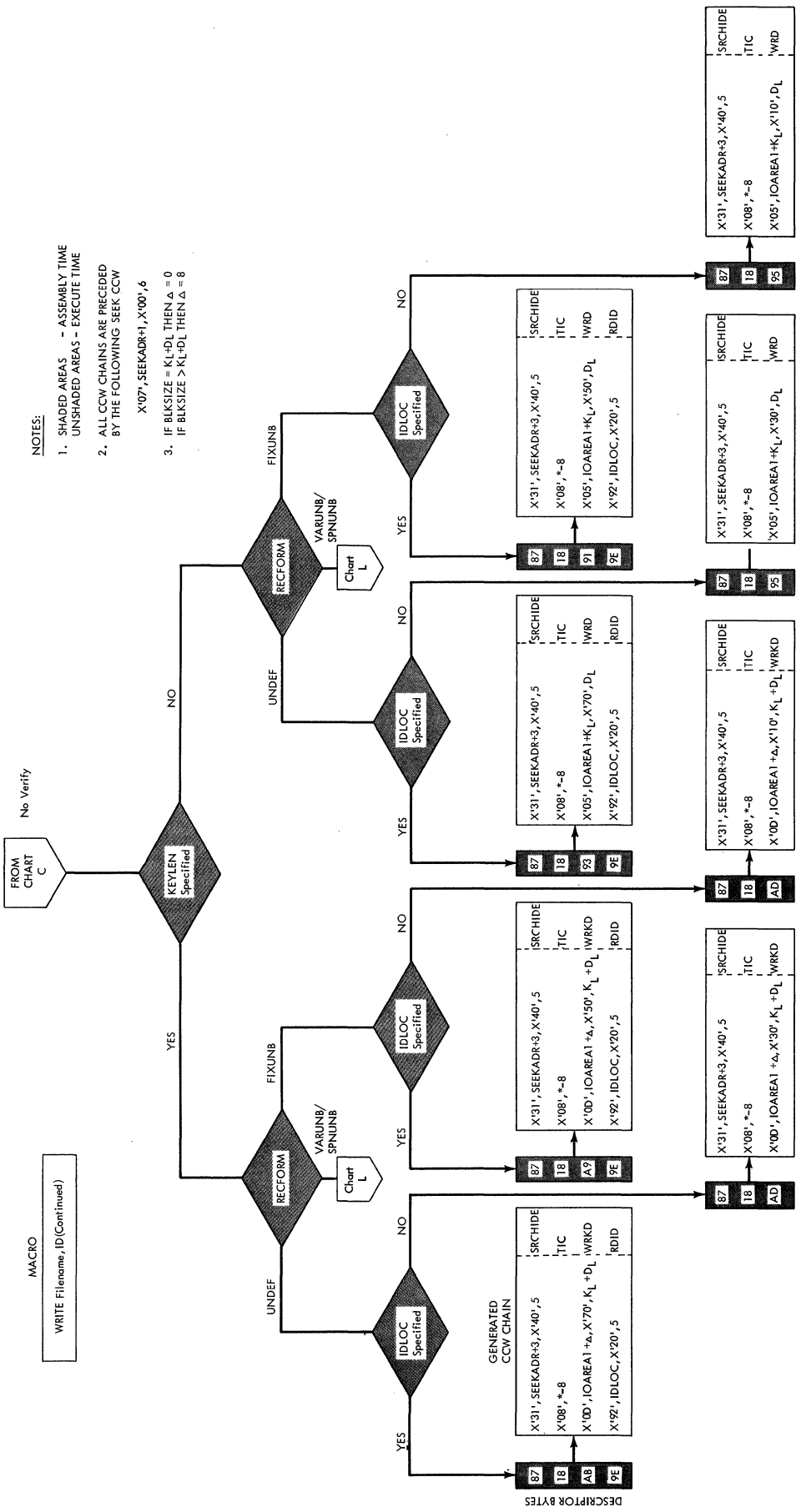


Figure 14. DAM Channel Programs Without RPS Support (Part 4 of 14)

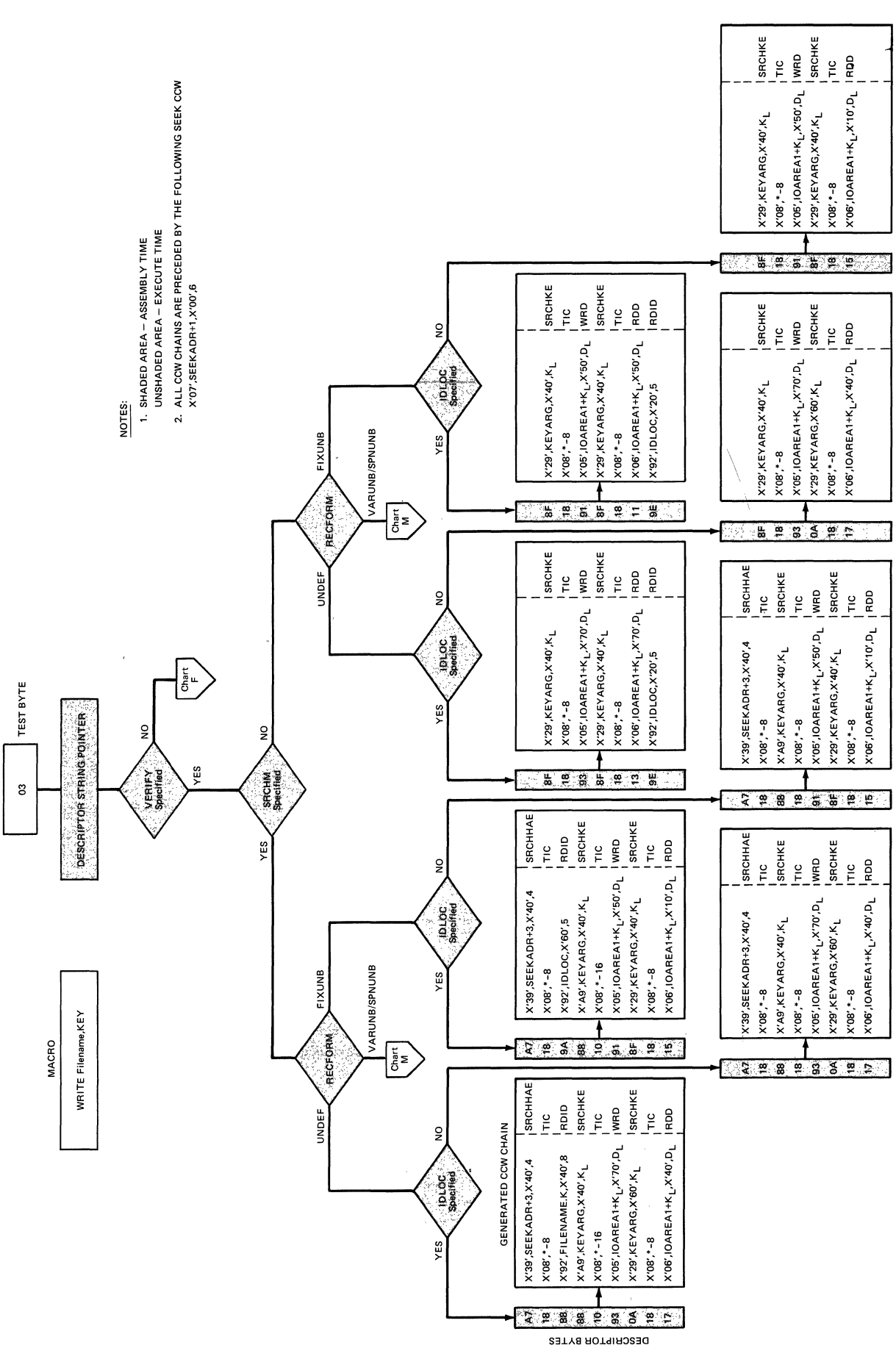


Figure 14. DAM Channel Programs Without RPS Support (Part 5 of 14)

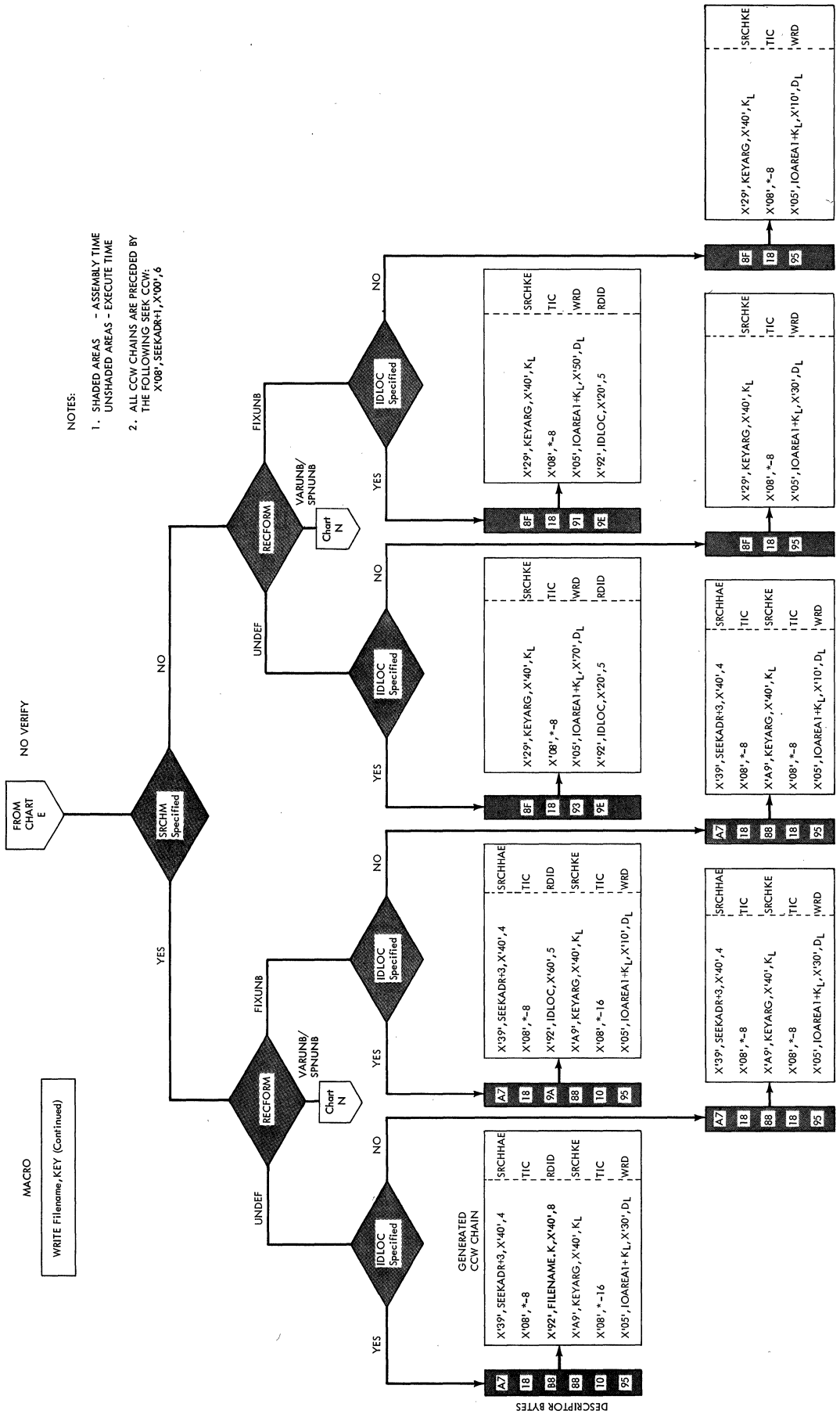


Figure 14. DAM Channel Programs Without RPS Support (Part 6 of 14)

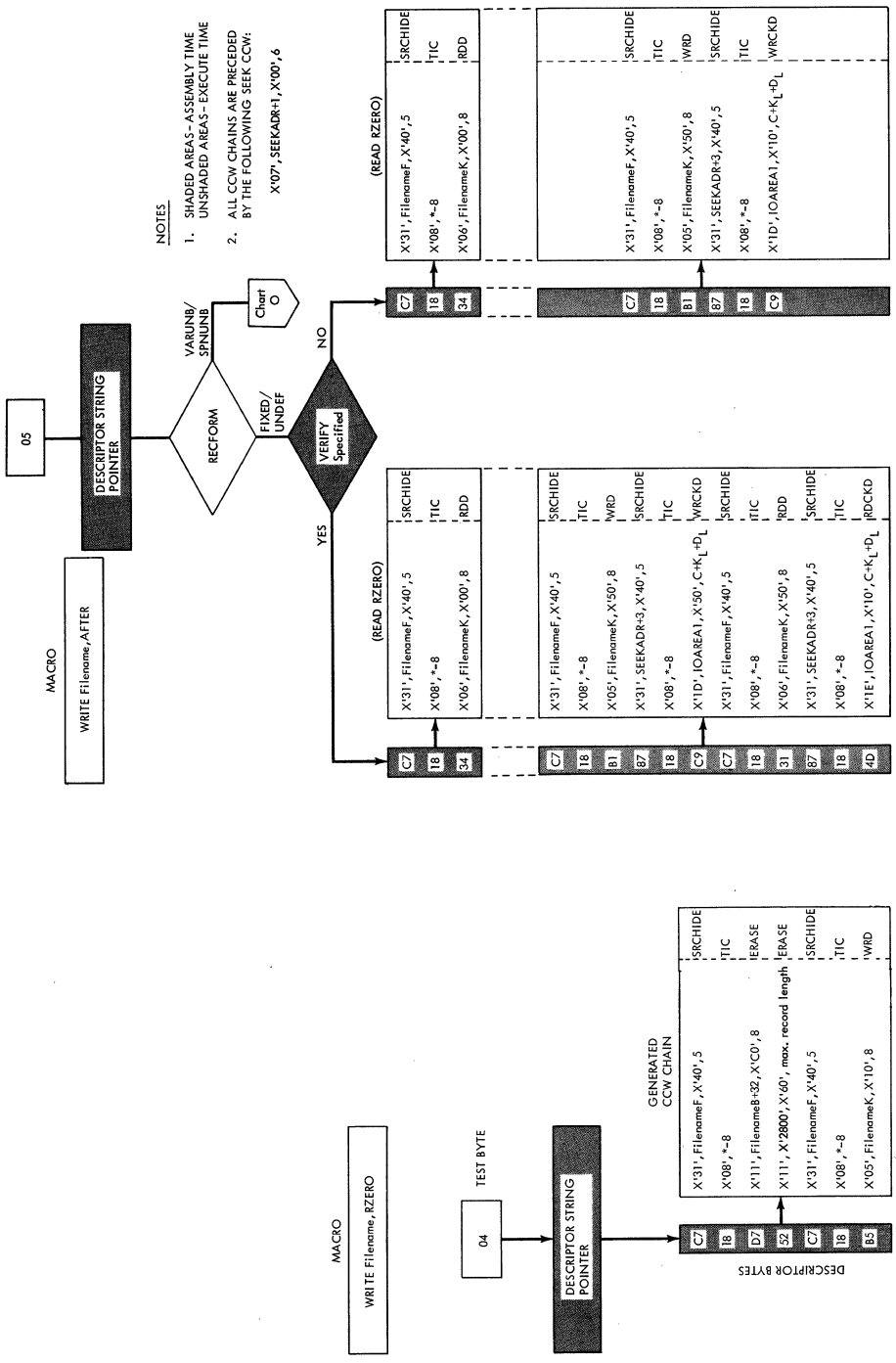


Figure 14. DAM Channel Programs Without RPS Support (Part 7 of 14)

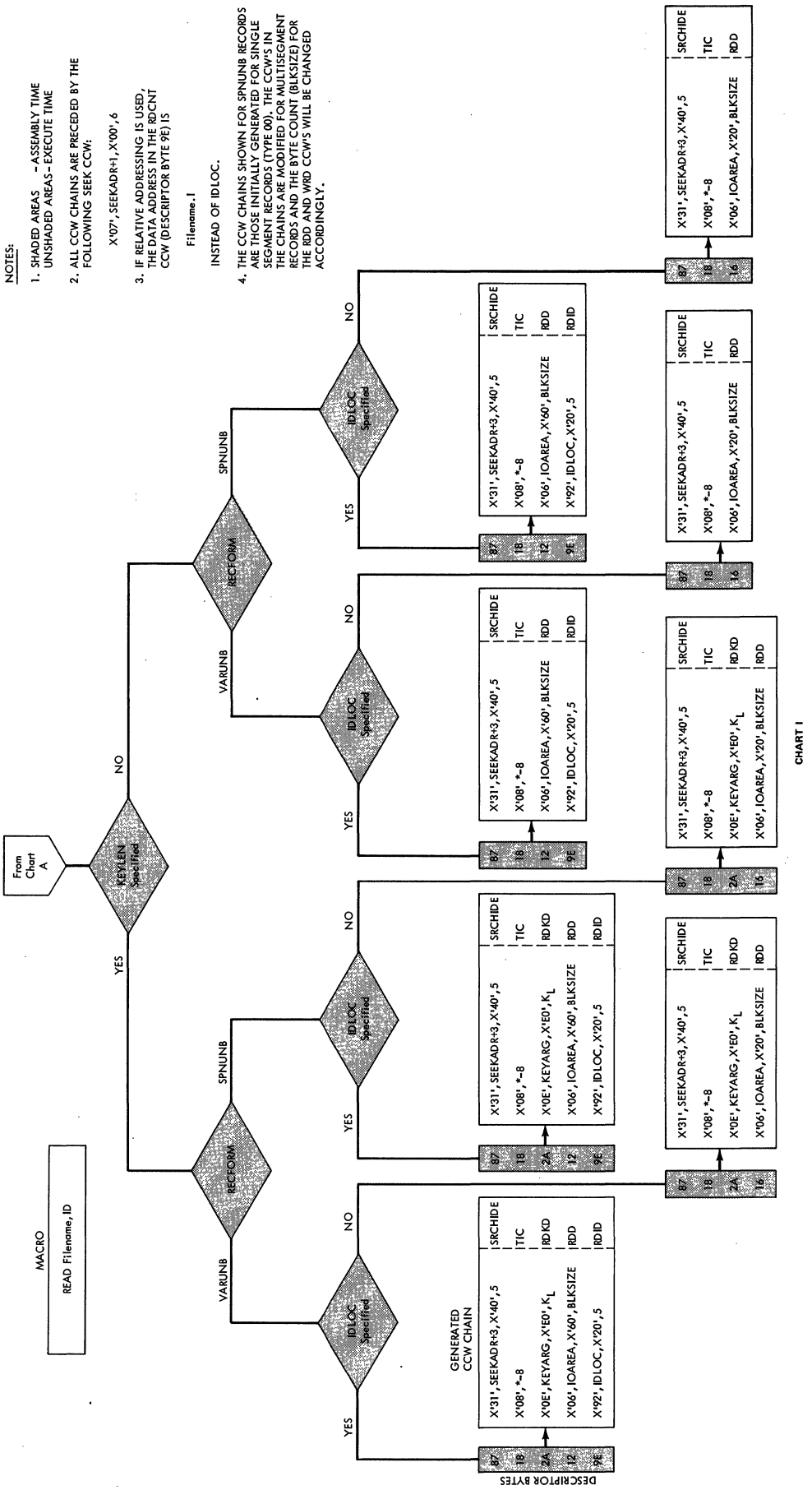


Figure 14. DAM Channel Programs Without RPS Support (Part 8 of 14)

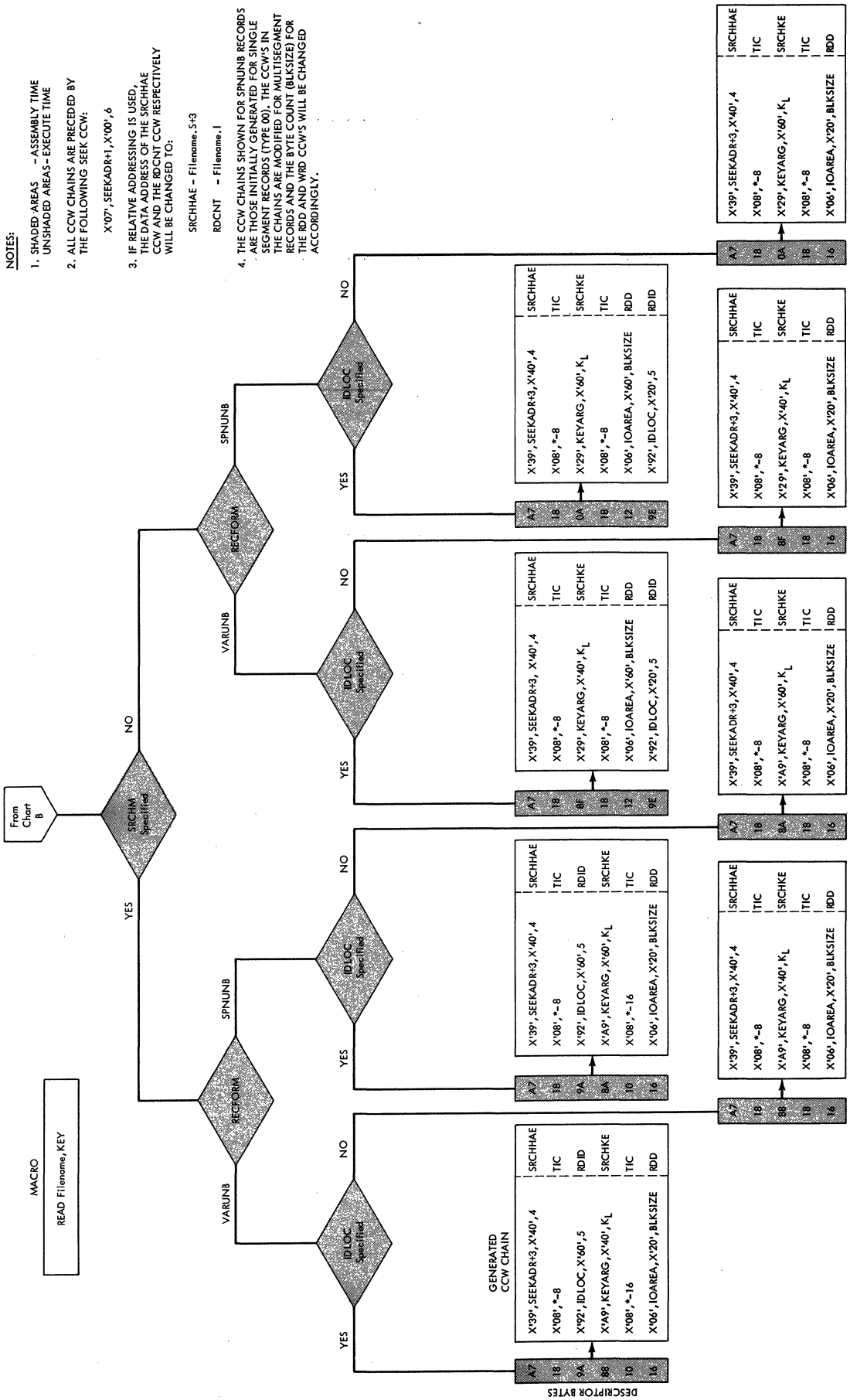


Figure 14. DAM Channel Programs Without RPS Support (Part 9 of 14)

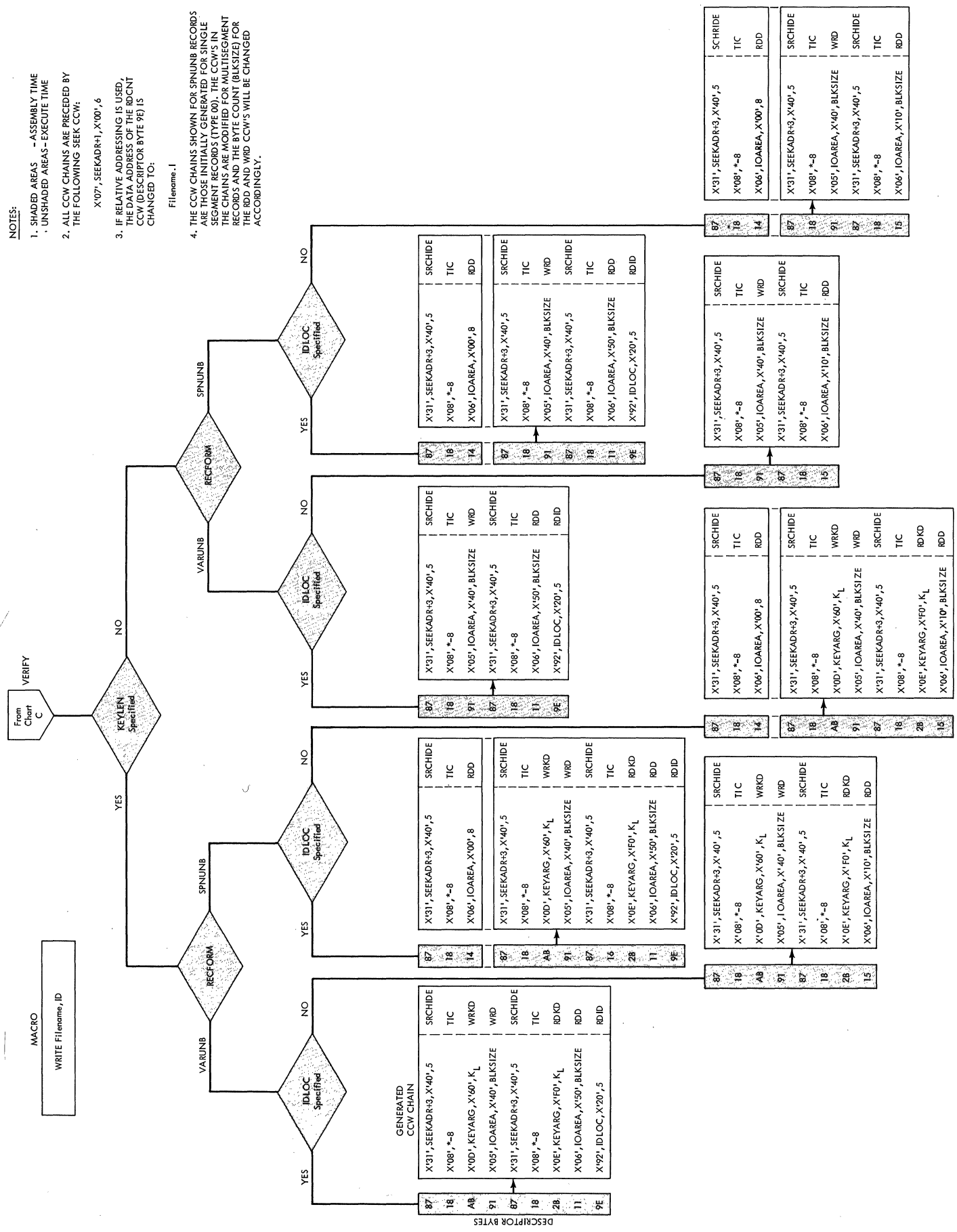


Figure 14. DAM Channel Programs Without RPS Support (Part 10 of 14)

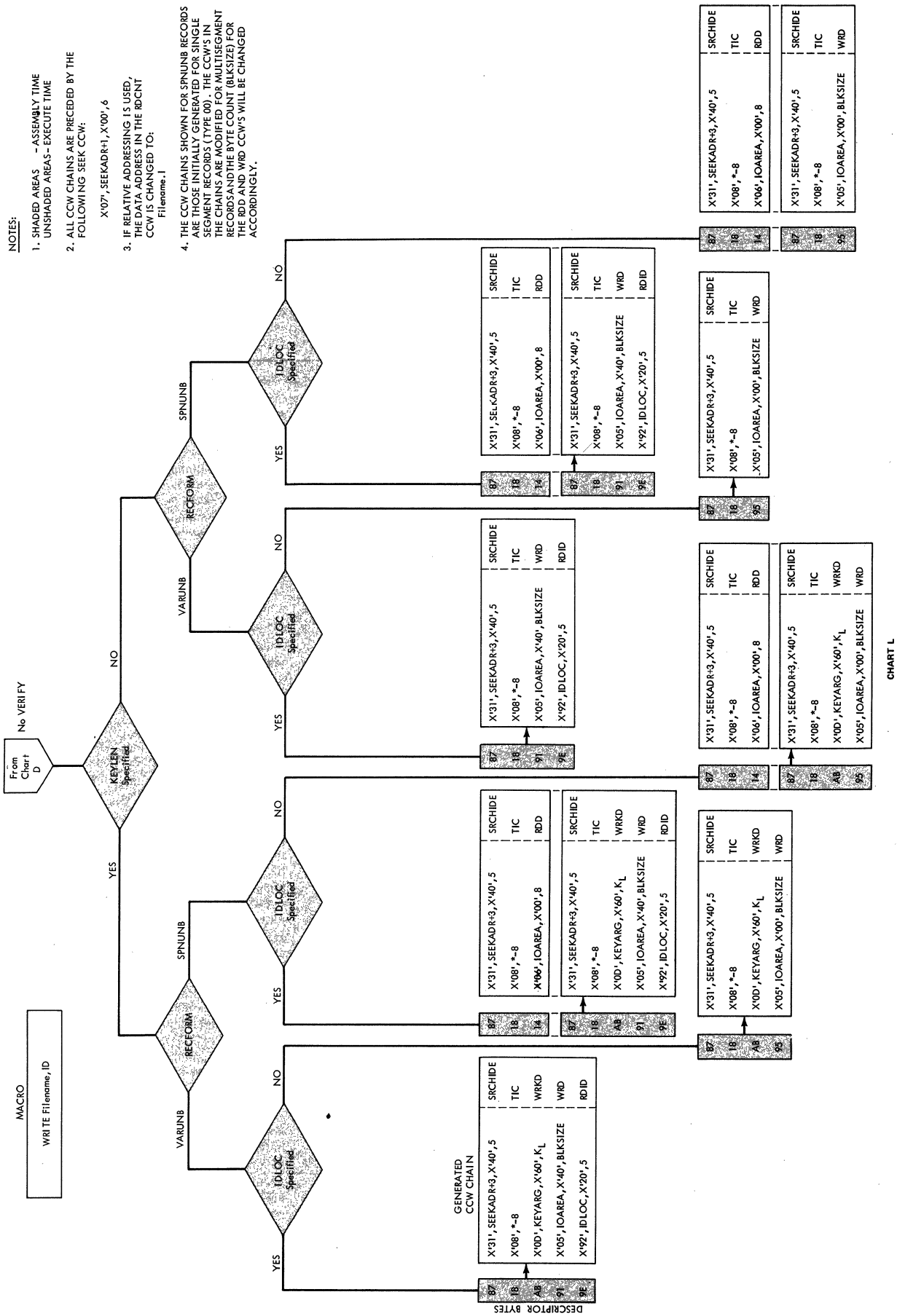


Figure 14. DAM Channel Programs Without RPS Support (Part 11 of 14)

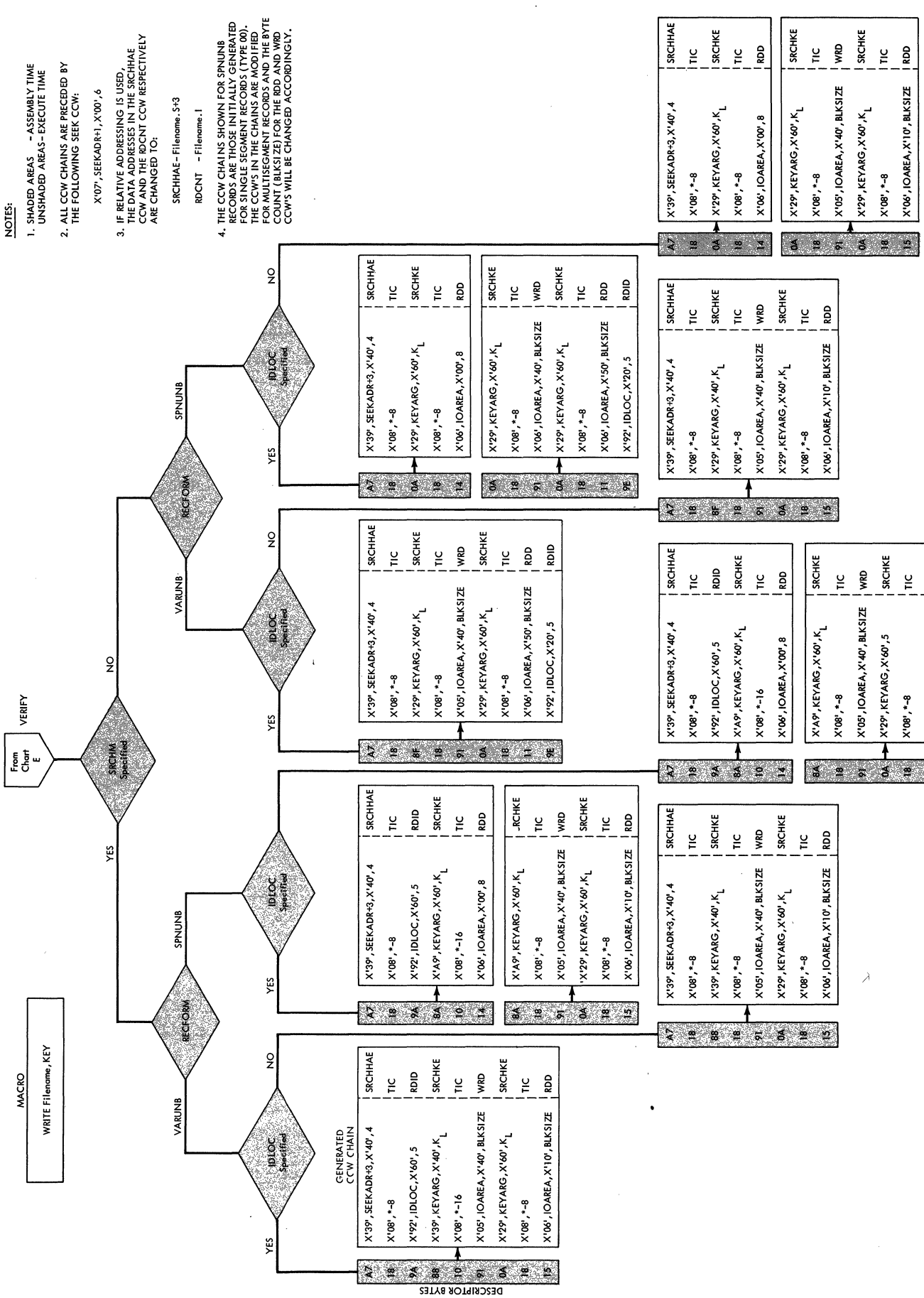


Figure 14. DAM Channel Programs Without RPS Support (Part 12 of 14)

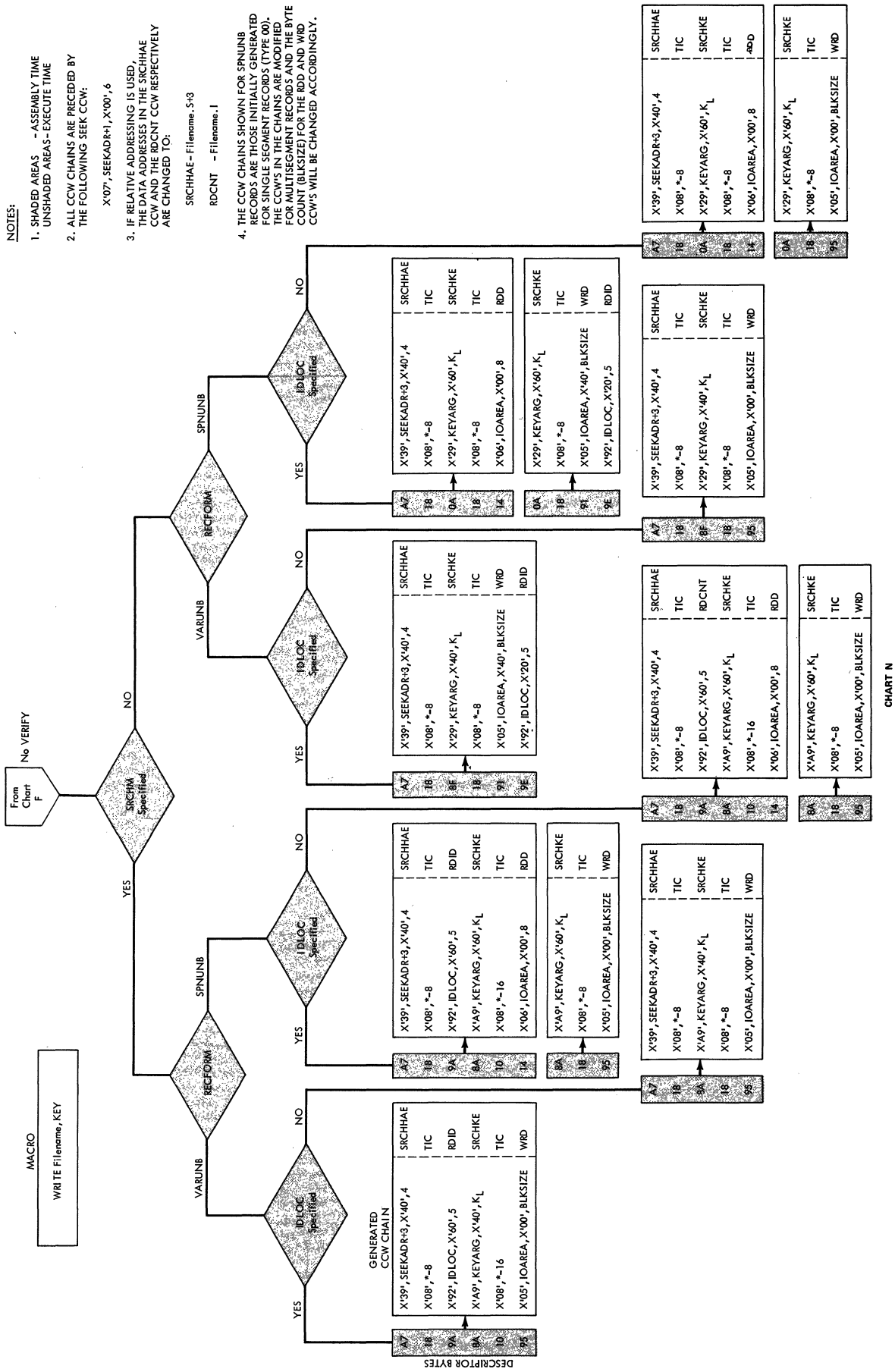


Figure 14. DAM Channel Programs Without RPS Support (Part 13 of 14)

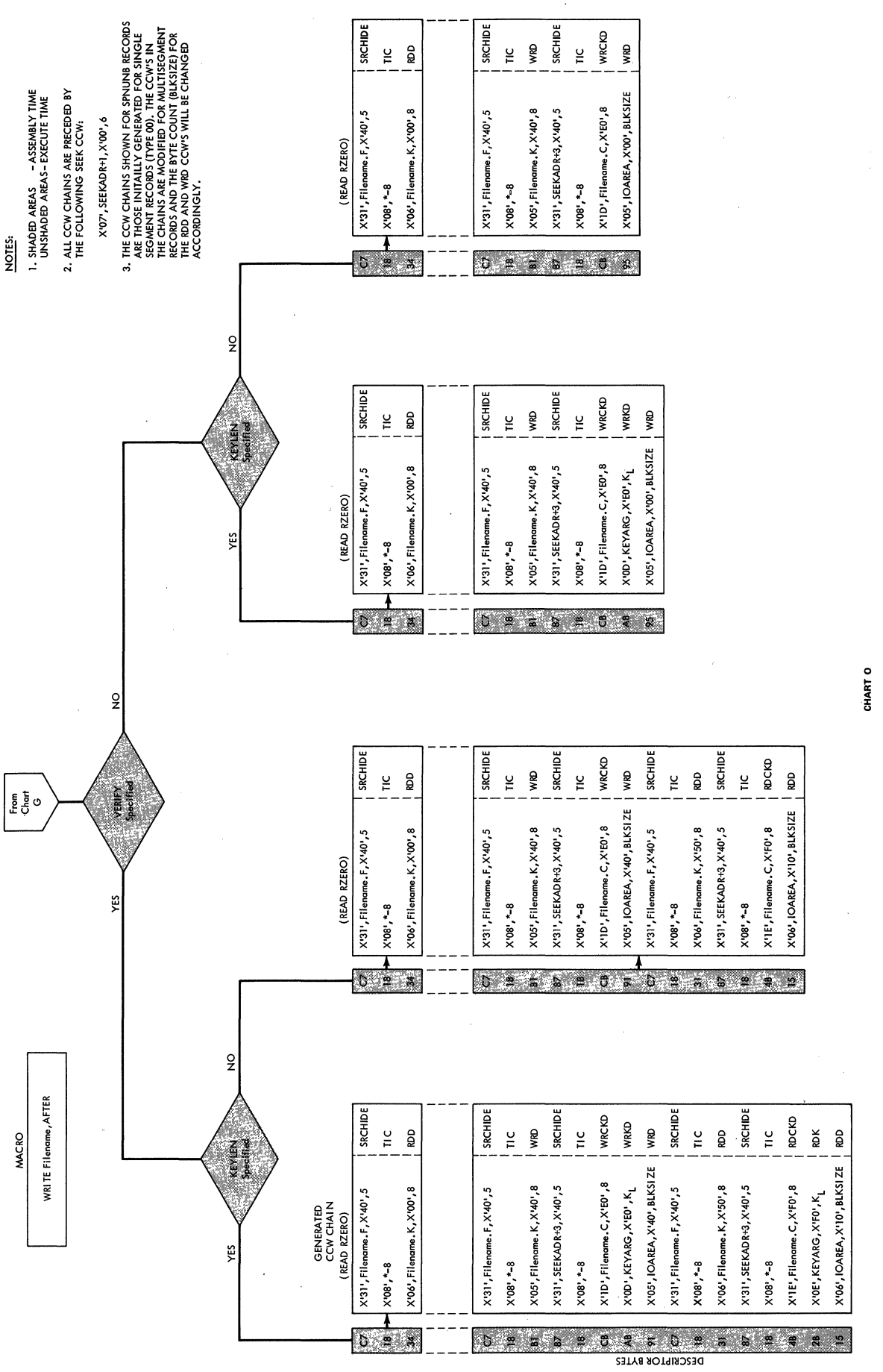


Figure 14. DAM Channel Programs Without RPS Support (Part 14 of 14)

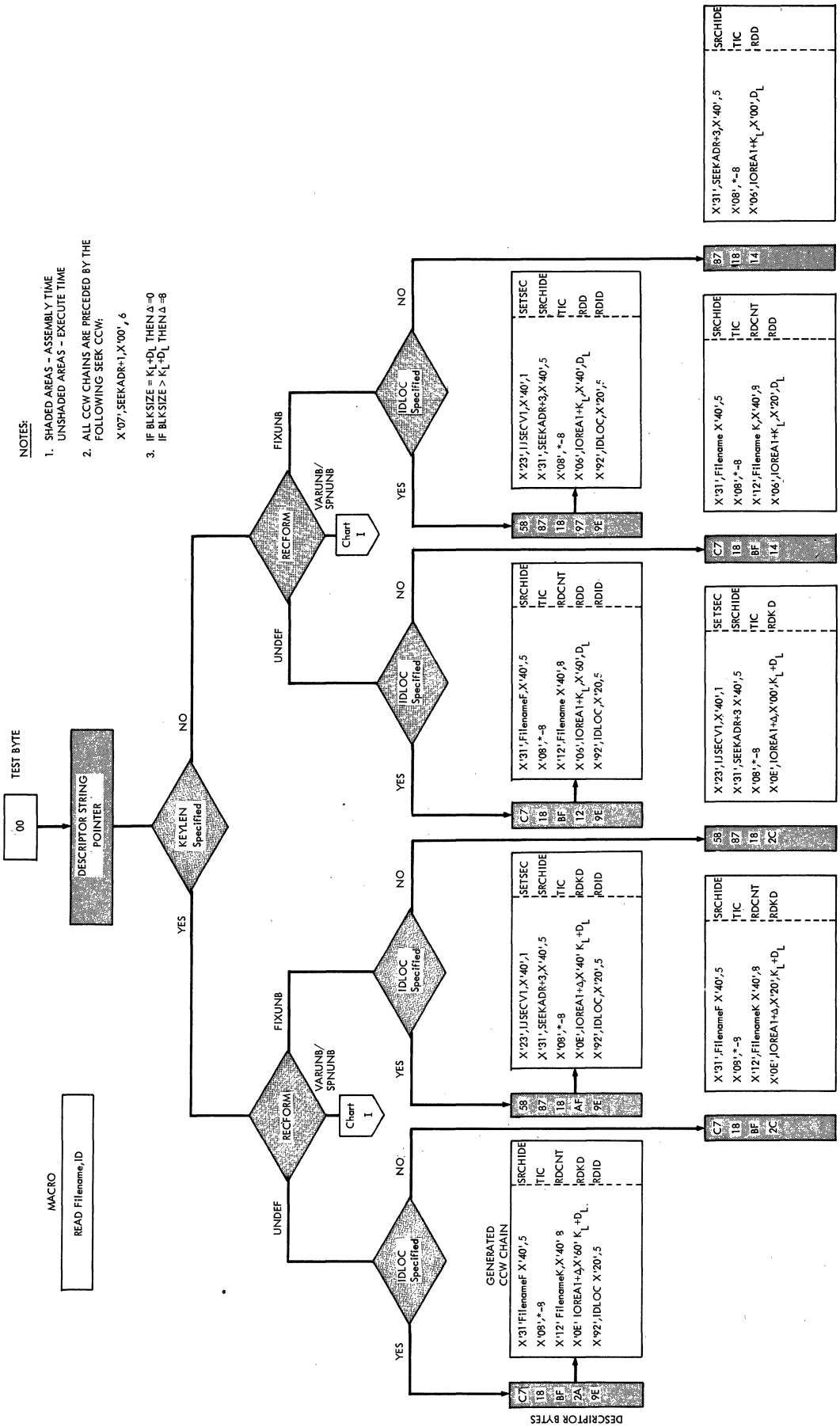


Figure 15. DAM Channel Programs with RPS Support (Part 1 of 14)

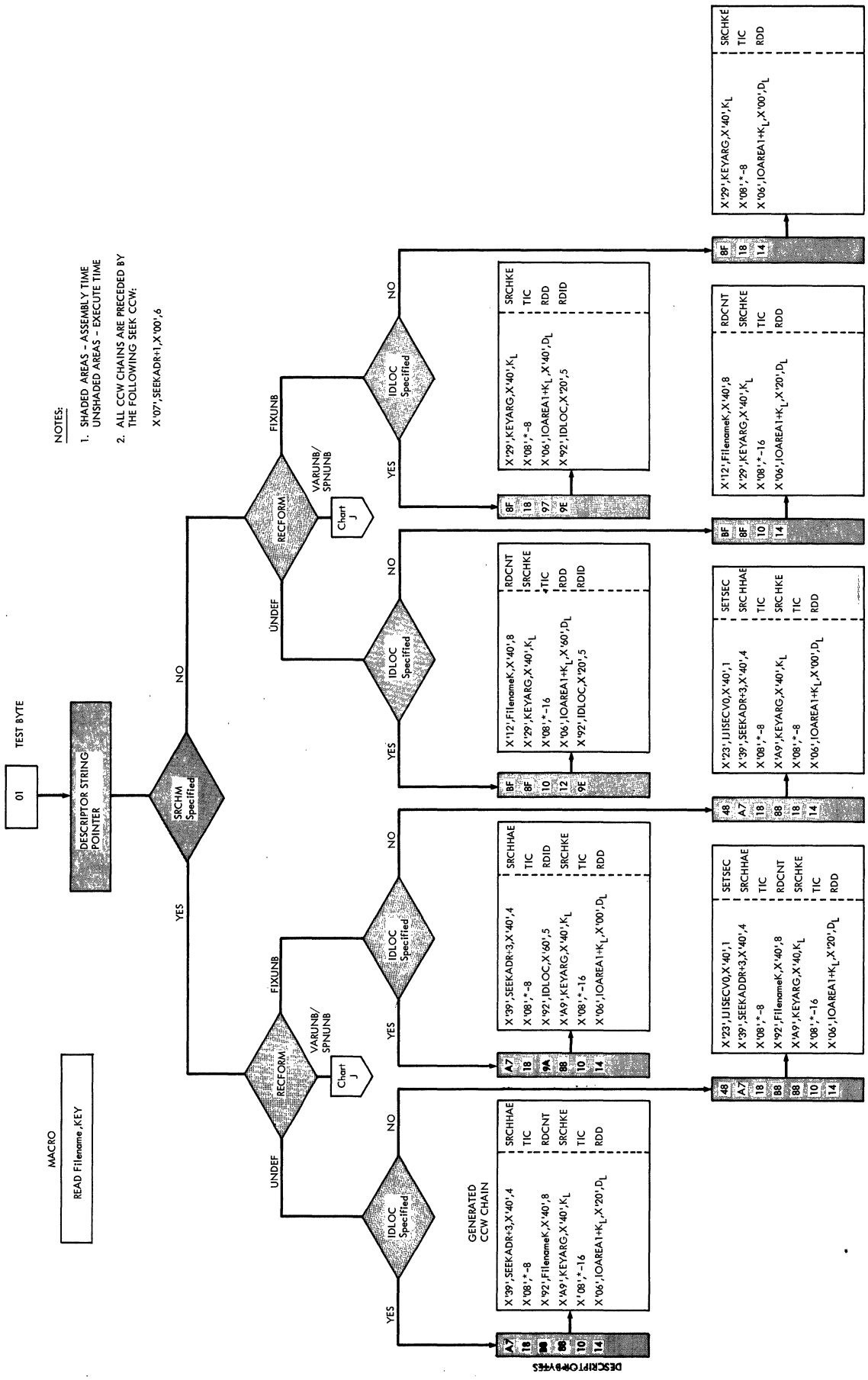


Figure 15. DAM Channel Programs with RPS Support (Part 2 of 14)

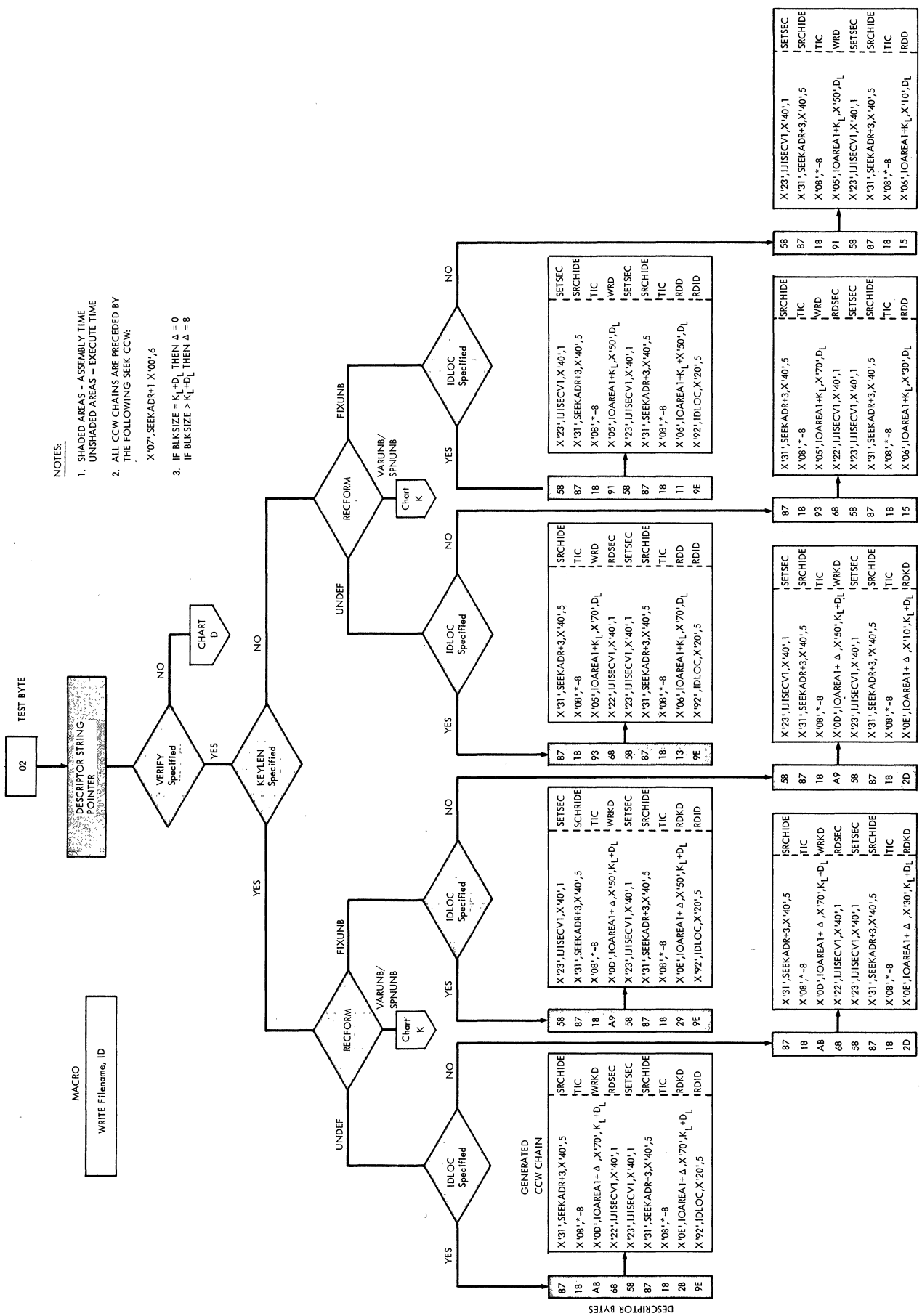


Figure 15. DAM Channel Programs with RPS Support (Part 3 of 14)

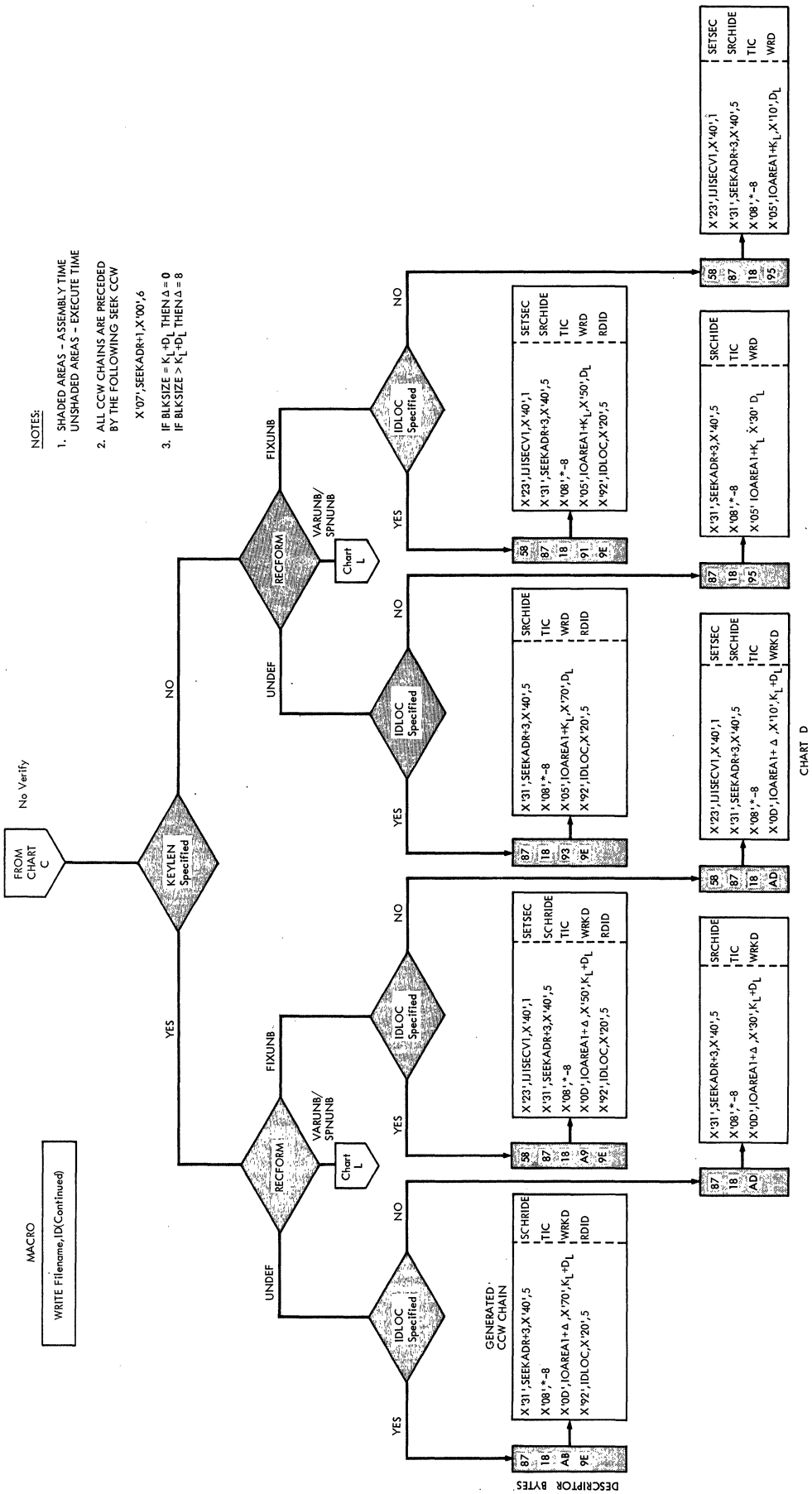


Figure 15. DAM Channel Programs with RPS Support (Part 4 of 14)

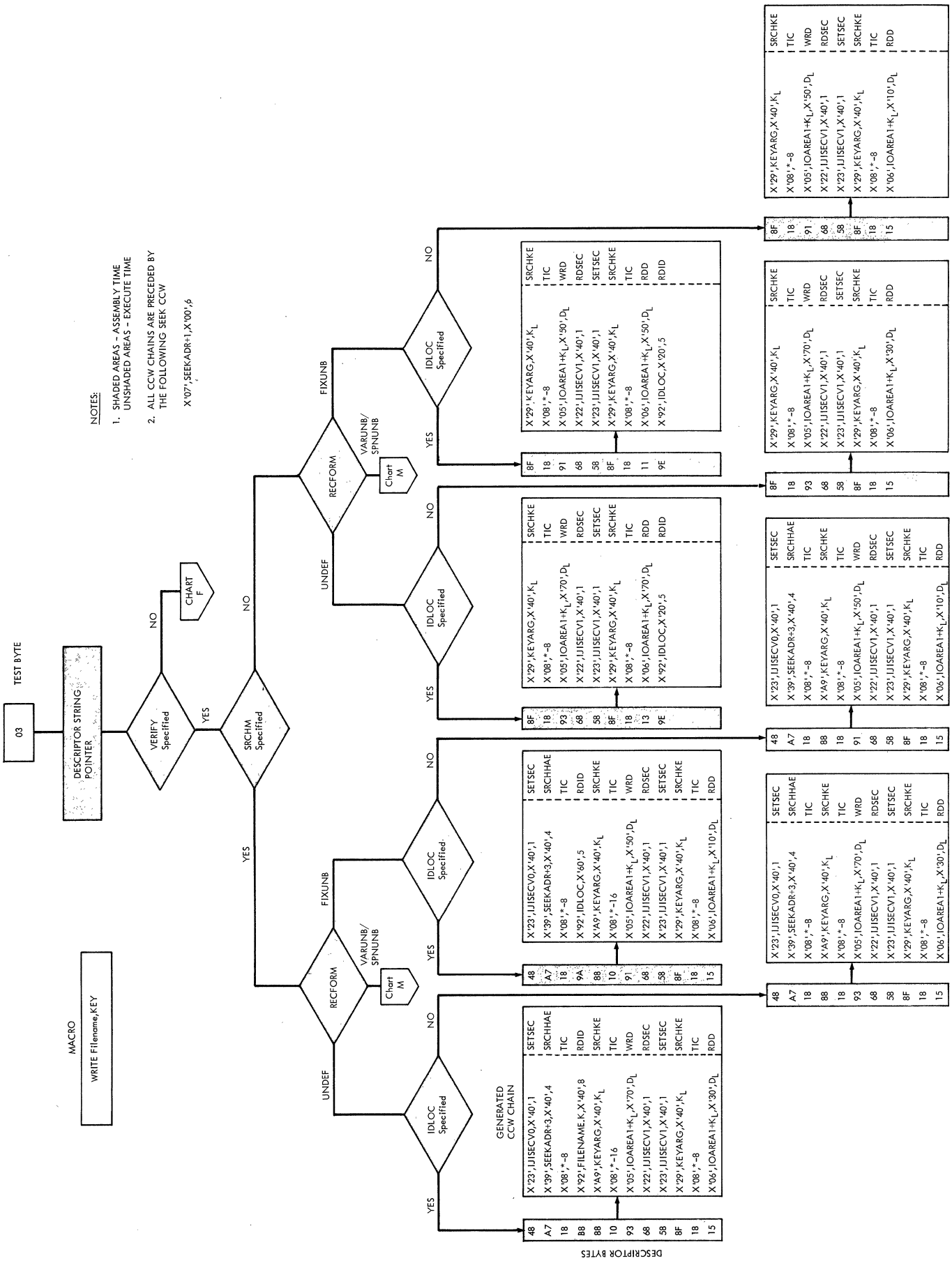


Figure 15. DAM Channel Programs with RPS Support (Part 5 of 14)

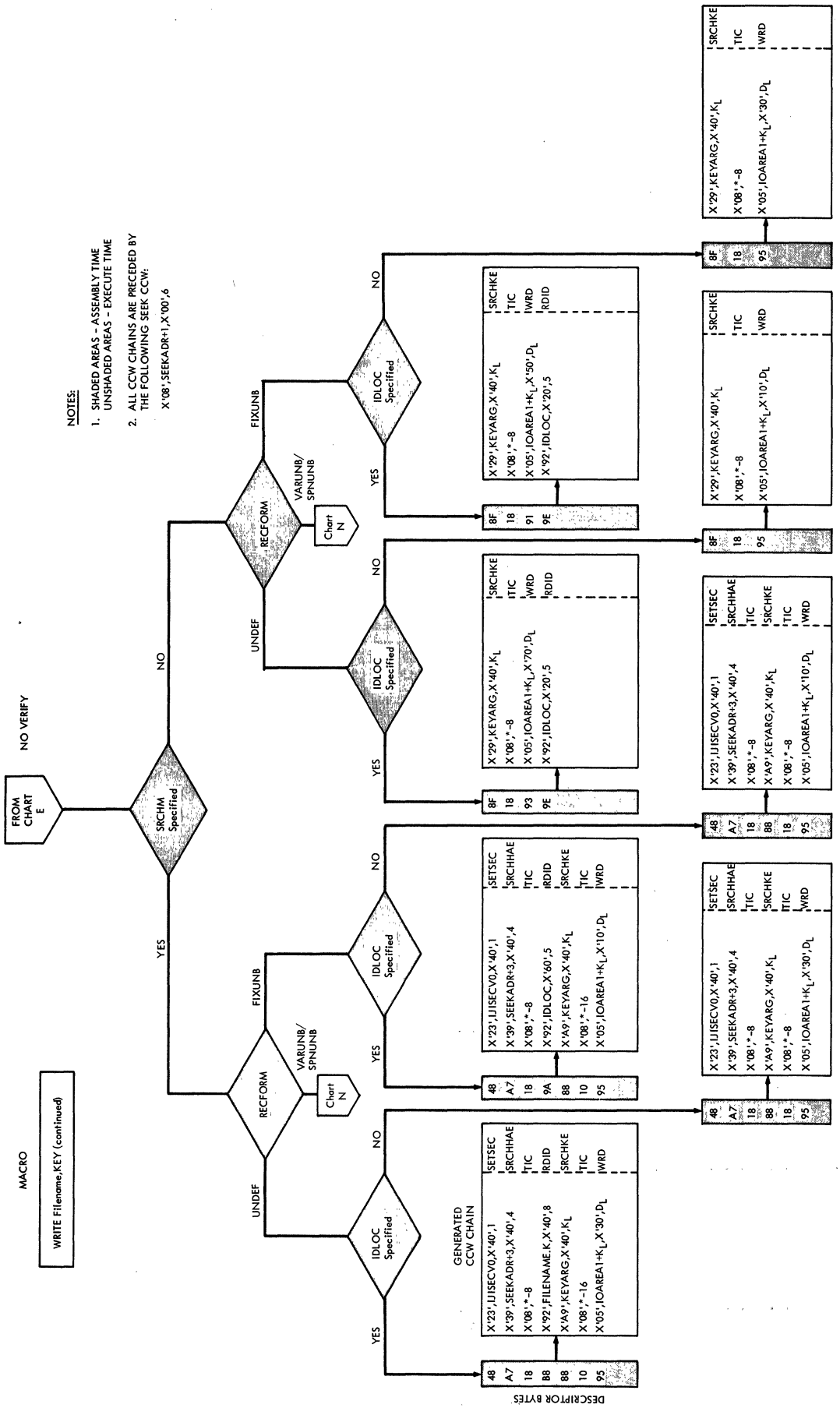


Figure 15. DAM Channel Programs with RPS Support (Part 6 of 14)

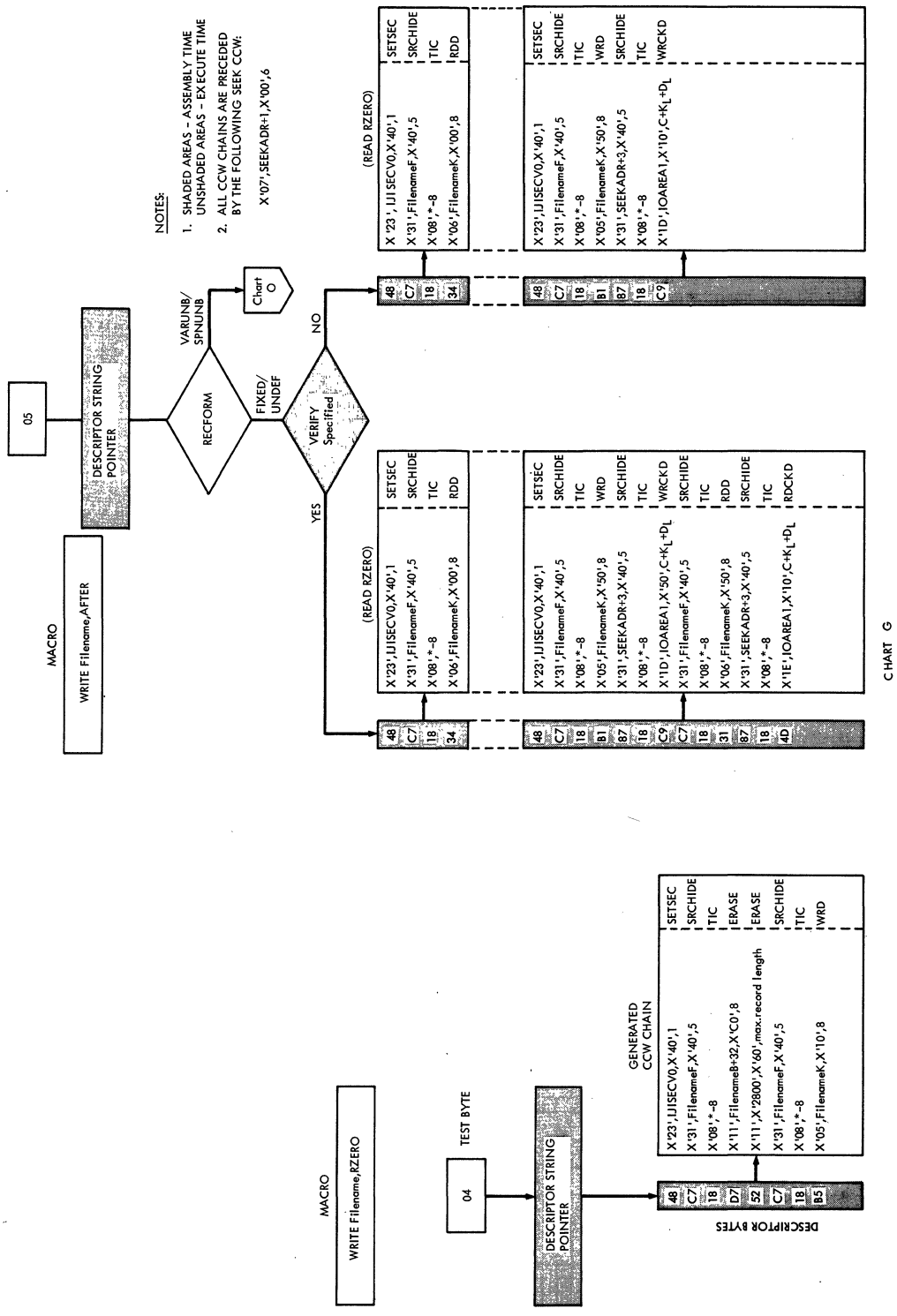


Figure 15. DAM Channel Programs with RPS Support (Part 7 of 14)

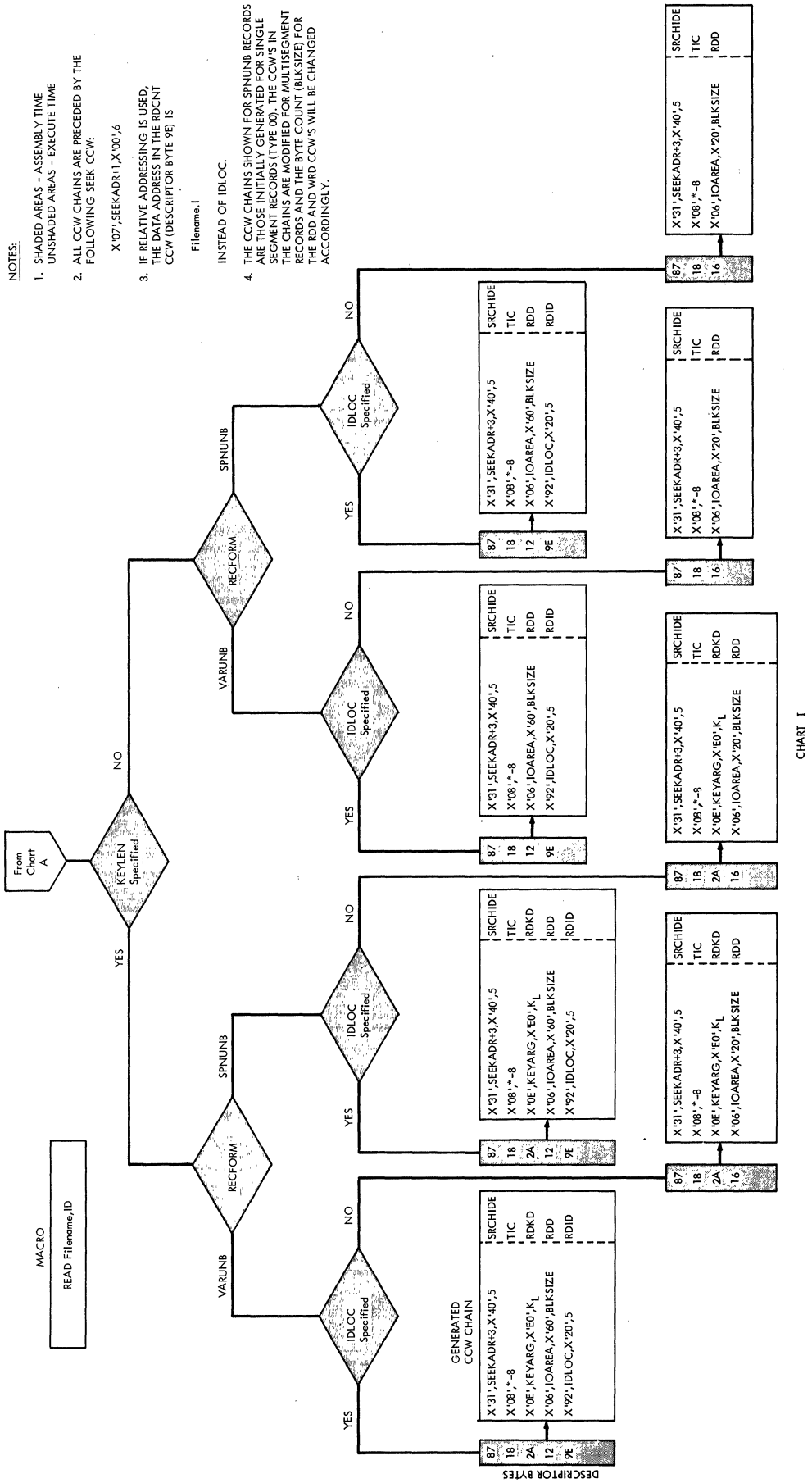


Figure 15. DAM Channel Programs with RPS Support (Part 8 of 14)

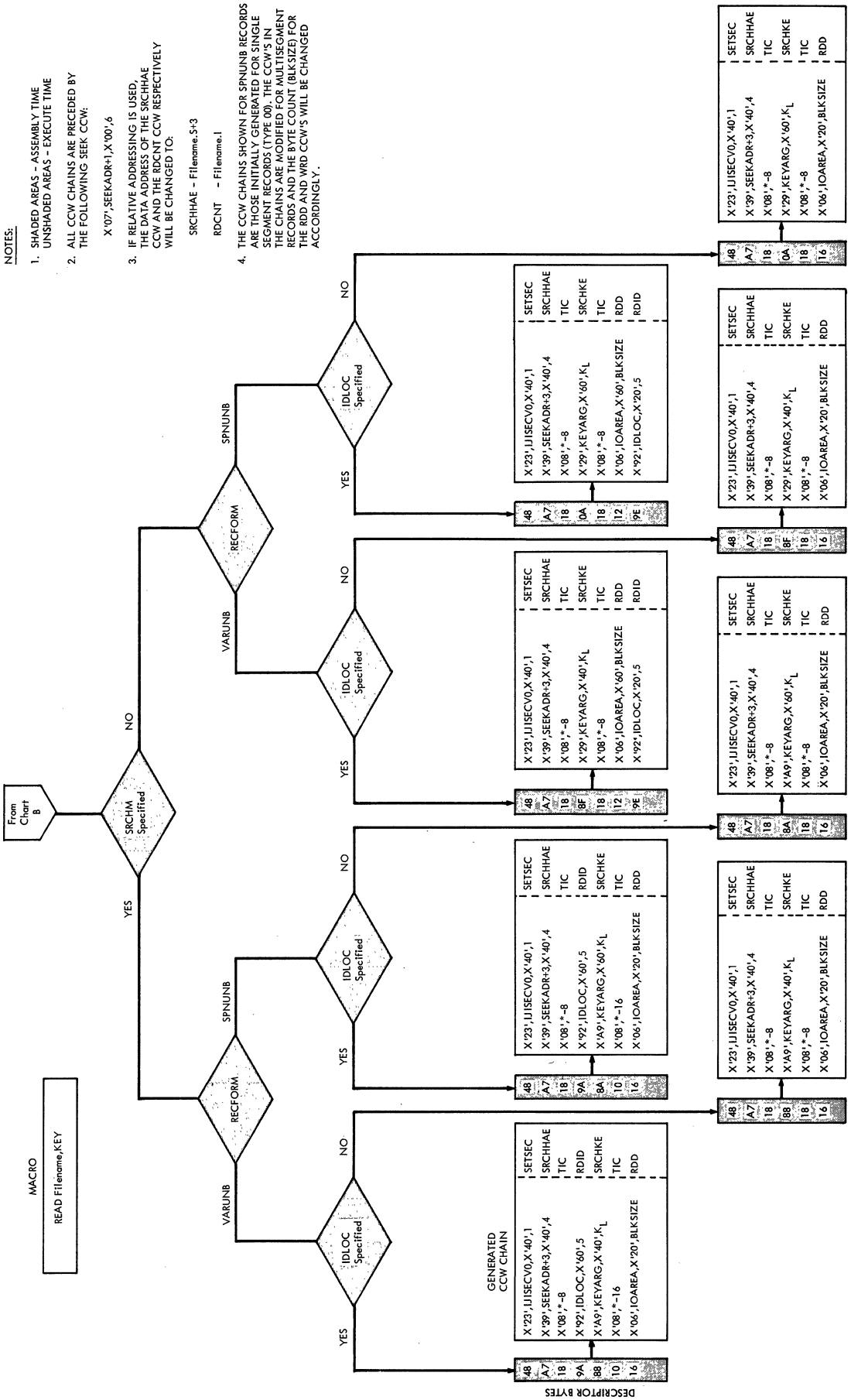


Figure 15. DAM Channel Programs with RPS Support (Part 9 of 14)

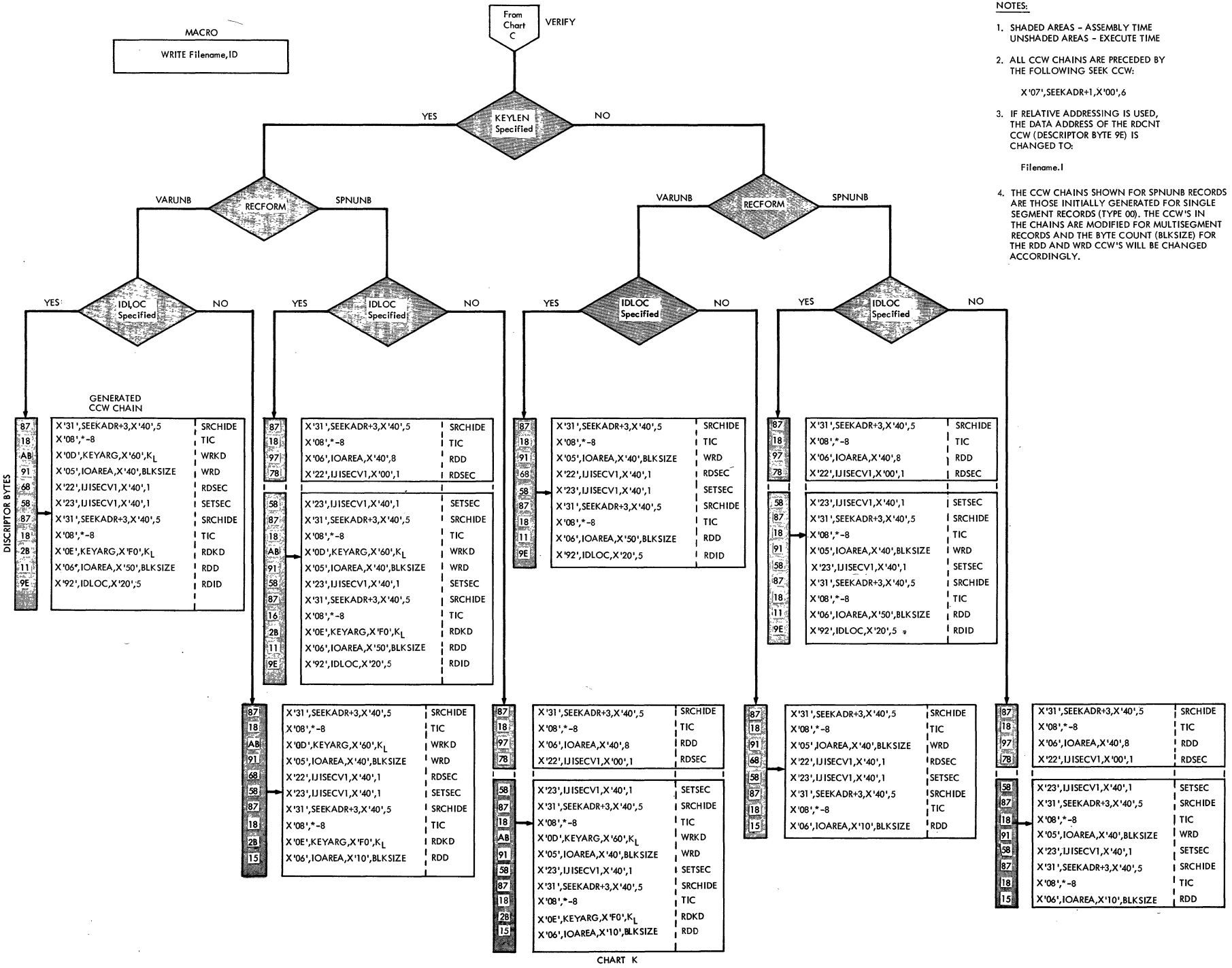


Figure 15. DAM Channel Programs with RPS Support (Part 10 of 14)

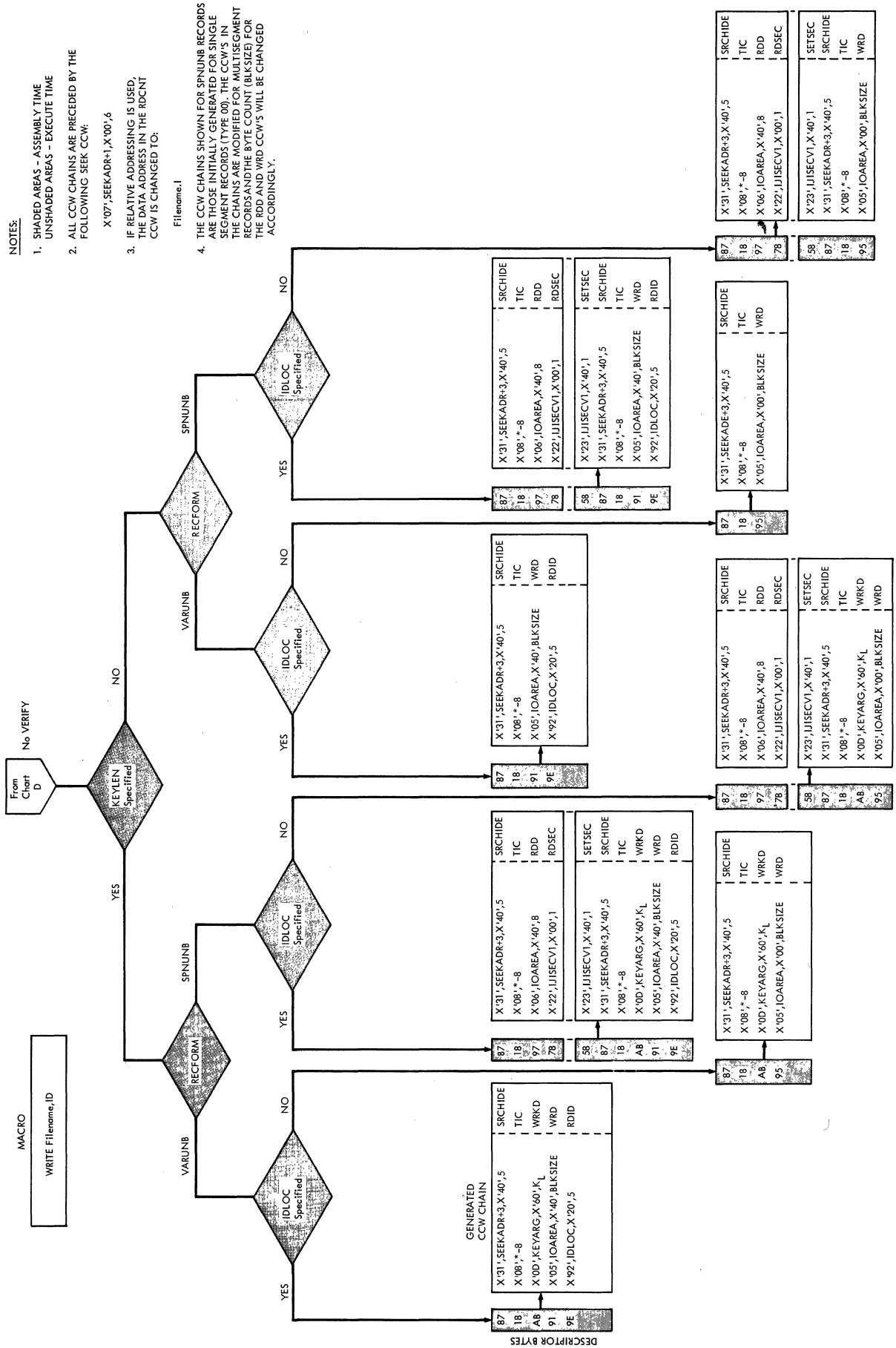


Figure 15. DAM Channel Programs with RPS Support (Part 11 of 14)

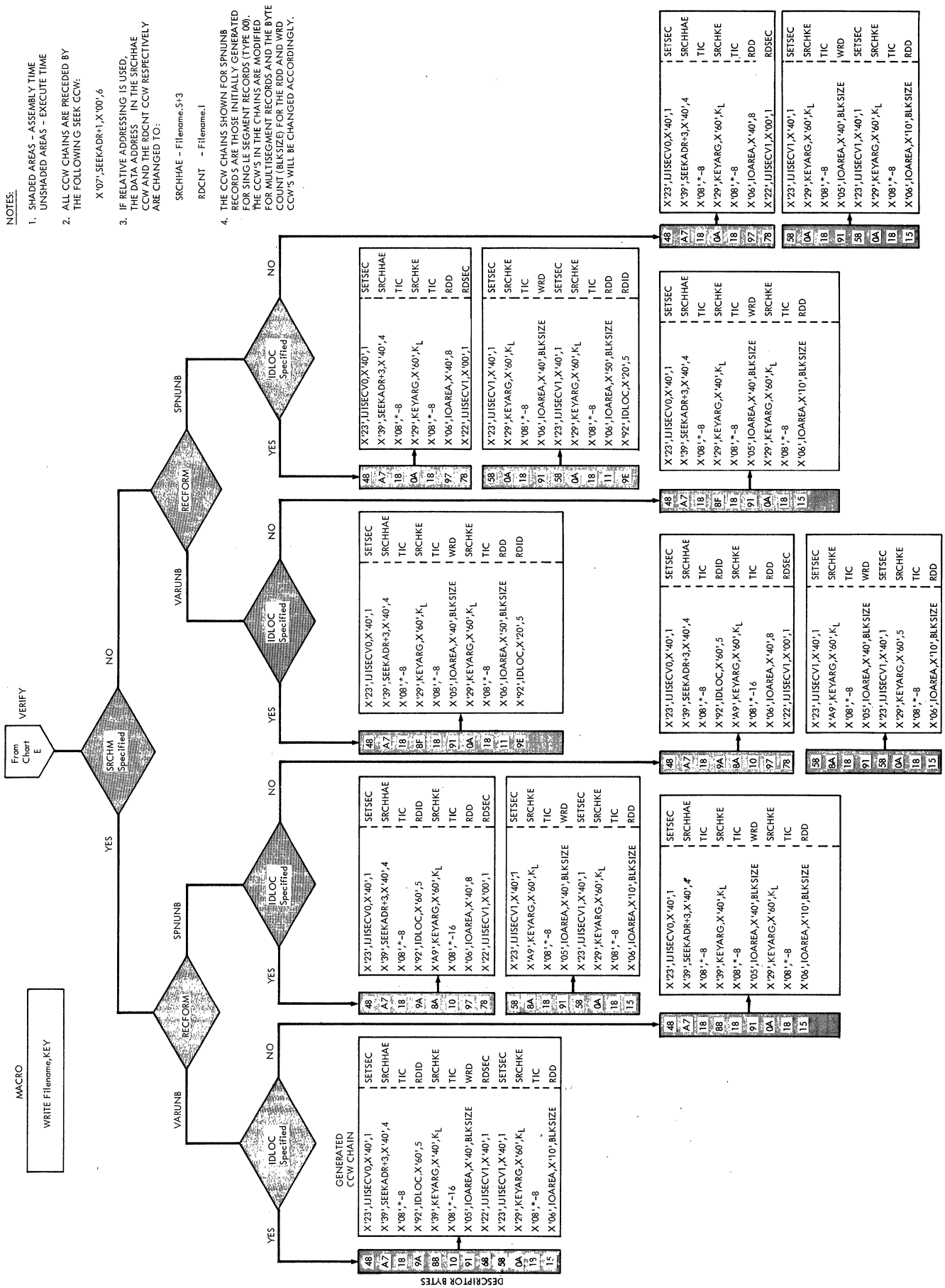


Figure 15. DAM Channel Programs with RPS Support (Part 12 of 14)

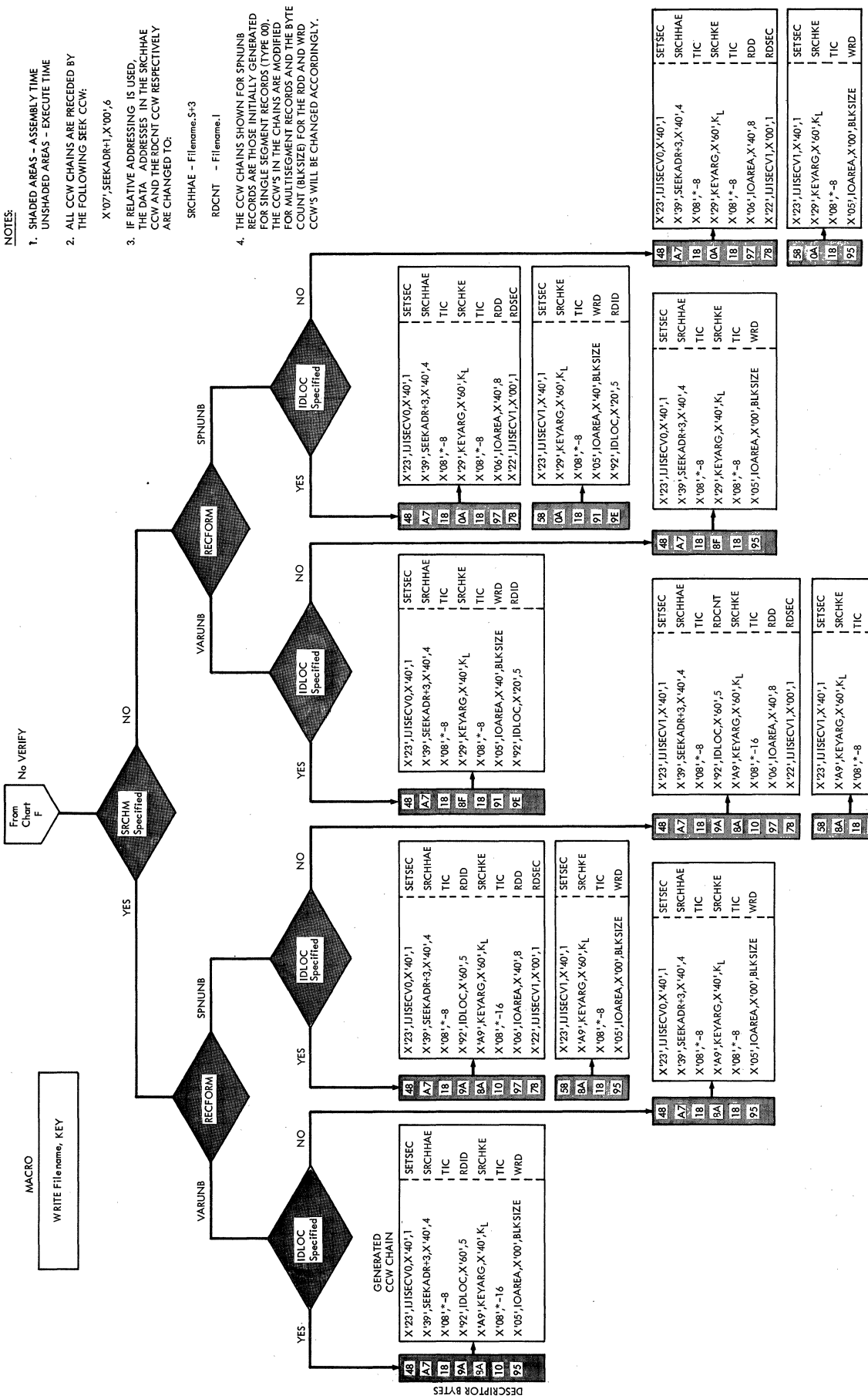


Figure 15. DAM Channel Programs with RPS Support (Part 13 of 14)

Figure 15. DAM Channel Programs with RPS Support (Part 14 of 14)

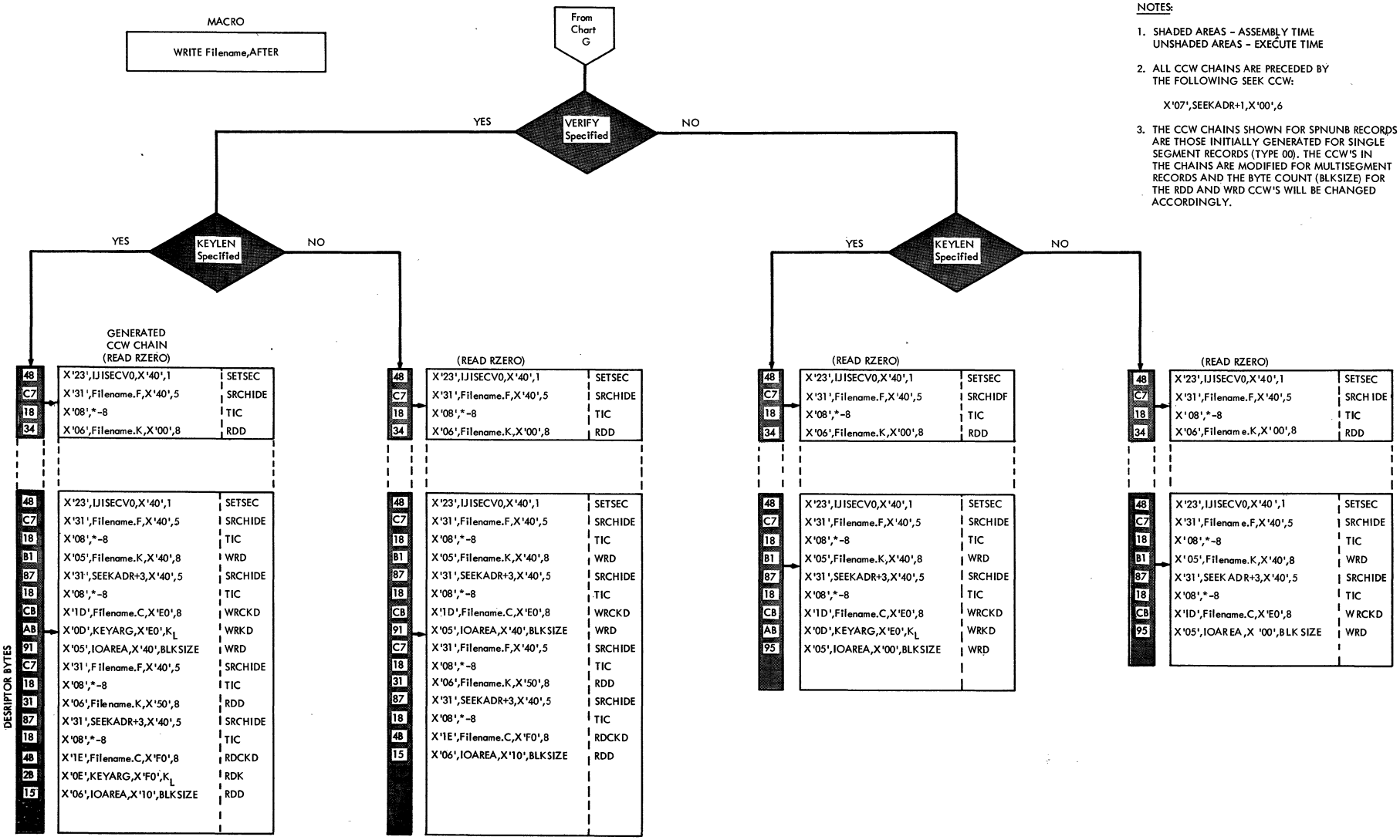


CHART O

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 DAM 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 in the label information area. This information is used either to check or create the actual file labels in the VTOC (Volume Table of Contents) on the DASD volume containing the file. Refer to VSE/Advanced Functions DASD Labels for details of the label information area and the standard DASD file labels processed by logical IOCS.

Close is required for DASD files processed by the Direct Access Method to invoke a FREEVIS for the DTFDA extension and handle the user standard trailer labels if specified.

DAM OPEN CHART 01

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 that 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 (see Figure 16).

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 (0 for disk)
Byte 13	Present Bin Number (0 for disk)

Figure 16. Format of Extent Information to User

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.

Refer to VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 4 for detailed descriptions of common DASD routines that open direct access method files.

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 17. The alteration factors are summarized in Figure 18.

	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'		
	TTT2	M	B2	TTT1

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 a disk device. (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 17. DSKXTNT Table for Relative Addressing

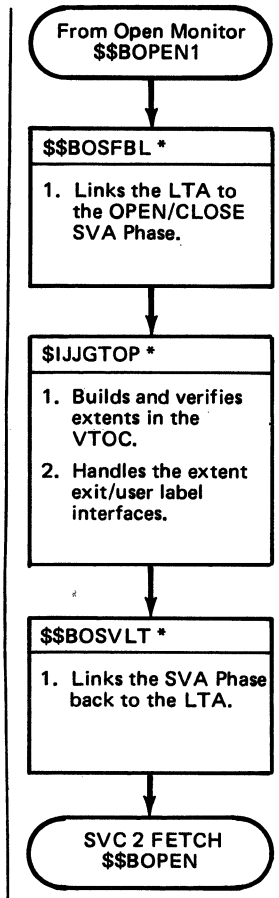
Factor	2311	2314/2319	3330	3340	3350
C1	1	1	4864	3072	7680
C2	10	20	19	12	30
H1	1	1	1	1	1

Figure 18. 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 (see Figure 18). 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 (see Figure 17).

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 (see Figure 17).

Chart 01. DAM Cpen



* Documented in VSE/Advanced Functions Diagnosis Reference LIOCS Volume 4 SAM for DASD.

$$BODACI: DA Close, Input/Output, Charts CA-CC

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 read (8 for disk devices), 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 8 for a disk device.

The indexed sequential access method (ISAM) permits processing DASD records in both random and/or sequential order by control information. For random processing, the user supplies the control information (record key) of the desired record to ISAM, and then issues READ or WRITE macro instructions to transfer the specified record. For sequential processing, the user specifies the first record to be processed, and then issues GET and/or PUT macro instructions to retrieve or insert records in sequential order by record key. Variations in macro instructions permit:

- A logical file of records to be loaded onto DASD (created).
- Individual records to be read from, added to, or updated in the file.

RECORD TYPES

Logical records in an ISAM-organized file must be fixed-length records either blocked or unblocked. Each physical record in the file must contain a key area. If the records are blocked, the record key (control information) of the highest (last) logical record in the block is stored in the key area of the block.

STORAGE AREAS

I/O Areas

An I/O area must be specified for each ISAM file to be processed in a problem program. This I/O area must be defined to contain sufficient space for the data area. If unblocked records are to be retrieved sequentially or records are to be loaded or added, space for a key field is required. Space for the count area must be provided when the file is being loaded or additions to the file are being made. Space for a sequence-link field is required when additions are to be made to the file or when records are retrieved from a file. The sequence-link field is used for overflow records (refer to the section "Add Records to a File").

Figure 19 shows the ISAM I/O area requirements.

Furction	Count	Key	Sequence Link	Data
Load - Unblocked Records	8	Key Length	--	Record Length
Load - Blocked Records	8	Key Length	--	Record Length X Blocking Factor
Add - Unblccked Records	8	Key Length	10	Record Length
Add - Blocked Records	8	Key Length	--	Record Length X Blocking Factor
			or*	
	8	Key Length	10	Record Length
Random Retrieve - Unblocked Records	-	--	10	Record Length
Sequential Retrieve - Unblocked Records	-	Key Length	10	Record Length
Random or Sequential Retrieve - Blocked Records	-	--	--	Record Length (Including Keys) X Blocking Factor
			or*	
	-	--	10	Record Length

* Whichever is larger.

Figure 19. ISAM I/O Area Requirements (in bytes)

The format of the sequence-link field of an overflow record or the index-level pointer is MBBCCHHRFP:

2311/2314/2319/3330/3340 Disk

M = Extent Sequence Number

BB = 00

CC = Cylinder Number

HH = Head (Track) Number

R = Record Number

F = (ccccciii) Entry Type and Index Level. See Note 1.

P = Pointer type. See Note 2.

Note 1: F = ccccciii

Entry Type (ccccc)	Index Level (iii)	DASD Address Information
00000 - Normal Entry (Unshared Track)	000 - Track Index 001 - Cylinder Index 010 - Master Index	R = 0
00001 - Normal Entry (Shared Track)	000 - Track Index	R = N (Points to First Data Record on the Track)
00010 - Overflow Entry (End)	000 - Track Index or Sequence-Link Field	R = 255
00011 - Overflow Entry (Chained)	000 - Track Index or Sequence-Link Field	R = N (Actual Record Address)
00100 - Dummy Entry (End)	000 - Track Index 001 - Cylinder Index 010 - Master Index	M through R = 0
00101 - Dummy Entry (Chained)	001 - Cylinder Index 010 - Master Index	M through H Points to First Track on Next Cylinder, R = 0
00110 - Inactive Entry	000 - Track Index 001 - Cylinder Index 010 - Master Index	M through R = 0

Note 2:

P = Seek Op-Code

Seek Op-Code	Meaning	Index Level
1B	Entry Points to Cylinder Index Track on Same Cylinder	Master Index
0B	Entry Points to Cylinder Index Track on Different Cylinder	Master Index
07	Entry Points to Track Index	Cylinder Index
1B	Normal Entry (Shared or Unshared)	Track Index
07	Overflow Entry (End)	Track Index or Sequence-Link Field
07	Overflow Entry (Chained)	Track Index or Sequence-Link Field
07	Dummy Entry (End)	Master, Cylinder and Track Indexes
07	Dummy Entry (Chained)	Master and Cylinder Indexes
07	Inactive Entry	Master, Cylinder and Track Indexes

Figure 20. Format of Sequence-Link Field/Index Level Pointer

LOAD: To create or extend a disk file of blocked or unblocked records. This area must be defined with enough capacity for an 8-byte count field, a control information field (key area), and the data record(s).

ADD, UNBLOCKED RECORDS: The output area for adding unblocked records to an ISAM organized file must be defined with enough capacity for an 8-byte count field, a control information field (key area), and a

data record area. The data record area must have space for a 10-byte sequence-link field that is used in conjunction with overflow records (refer to the section "Add Records to a File"). The sequence-link field is required when a record is written on an overflow track. ISAM determines the correct sequence link and stores this information at the beginning of the data section of the I/O area. When the sequence-link field is not used, the ten unused bytes fall at the end of the data section and are ignored. Figure 20 shows the format of the 10-byte sequence-link field.

ADD, BLOCKED RECORDS: The output area for adding blocked records to an ISFMS organized file must contain enough space for an 8-byte count field, a control information field (key area) and a data section large enough to contain the block of logical records. The minimum size for the data section is one logical record plus 10 bytes to be used for a sequence-link field when required.

SEQUENTIAL RETRIEVE, UNBLOCKED RECORDS: The input area for reading unblocked records must contain sufficient capacity for a key area and a data area. The data area must include enough space for the logical record plus 10 bytes for the sequence-link field of overflow records. If a record does not have a sequence-link field, the extra 10 bytes in the I/O area fall at the end of the data section and are ignored by the program.

RANDOM RETRIEVE, UNBLOCKED RECORDS: The input area for reading unblocked records must contain space for a data area. The data area must include enough space for the logical record plus 10 bytes for the sequence-link field of an overflow record. If a record does not have a sequence-link field, the extra 10 bytes in the I/O area fall at the end of the data section and are ignored by the program.

RETRIEVE, BLOCKED RECORDS: The input area for reading blocked records must contain space for a data area. The data area must be large enough to contain a full block of records. The minimum size of the data area is one logical record plus 10 bytes for the sequence-link field used with overflow records.

When blocked or unblocked records are to be retrieved and processed directly in the I/O area, a register must be specified. This register is used for indexing, to point to the beginning of each logical record when it is needed for processing.

Work Areas

When a work area is specified on input, ISAM moves each record from the I/O area to the work area. The problem program can then process the record in the work area. When a work area is specified on output, ISAM moves the record from the work area to the I/O area in preparation for transferring the record to DASD storage. If a work area is specified, an I/O register is not required. Figure 21 shows the ISAM work area requirements.

	Unblocked Records	Blocked Records
Load	KL+DL or 10*	DL or 10*
Add	KL+DL or 10*	DL or (KL+10)*
Random Retrieve	DL	DL
Sequential Retrieve	KL+DL	DL

Where: K = Key
D = Data
L = Length

* Whichever is larger.

Figure 21. ISAM Work Area Requirements (in bytes)

OVERFLOW AREAS

The location of the overflow area(s) for a logical file may be specified by the user. The overflow areas may be built by one of three methods:

1. Overflow areas for records may be located on each cylinder within the prime data area that is specified by a job control extent card for the data file. In this case, the user must specify the number of tracks to be reserved for overflow on each cylinder occupied by the file. The overflow records that occur within a particular cylinder are written in the cylinder overflow area for that cylinder. The number of tracks to be reserved for each cylinder overflow area must be specified in the DTFIS entry CYLOFL when a file of records is to be loaded and when records are to be added to an organized file.

2. An independent overflow area may be specified for storing all overflow records for the logical file. In this case, a job control EXTENT card must be included when the program is executed to specify the area of the volume to be used for the overflow area. This area may be on the same volume with the data records, or on a different volume that is online. However, it must be contained within one volume. (It must be the same kind of device as that containing the prime data area.)

3. Cylinder overflow areas (method 1) and an independent overflow area (method 2) may be used in combination. In this case, overflow records are placed first in the cylinder overflow areas within the data file. When any cylinder overflow area becomes filled, the additional overflow records from that cylinder are written in the independent overflow area. The specifications required for both methods 1 and 2 must be included for this combined method of handling overflows.

All records placed in the overflow area will be in the unblocked format and will have a sequence-link field prefixed to each record. There must always be one prime data track available (for a DASD record that has a data length of 0) when additions are being made to the last track in the prime data area containing records. The format of the overflow area upper limits (MBBCCHHR) is shown in Figure 29.

INDEXES

As ISAM loads the records, it creates a set of two or three indexes to be used to control the processing and location of the data records. Two indexes, the track index and the cylinder index, are always built for each file. The third, a master index, is built only when specified by the user. A master index should be specified only for large files. As a guideline, if a cylinder index occupies less than five tracks, it is usually faster to search only the cylinder index (followed by a search on the track index) than to search also a master index.

Indexes are developed as a series of entries, each including the address of a DASD track and the highest (last) record key on that track or cylinder. Each entry is a separate DASD record composed of a key area and a data area. The key area contains the highest key on the track or cylinder, and its length (number of bytes) is the same as the key-area length specified by the user for the data records. The data area of each index record is 10 bytes in size and contains the physical address of the logical record or of another index. Figure 20 shows the format of the 10-byte index level pointer (index data area).

Track Index (TI)

The lowest level index for logical file is the track index. This index has two important functions.

- Point to the correct track in the cylinder that contains the specified key.
- Provide direct linkage to the record overflow areas.

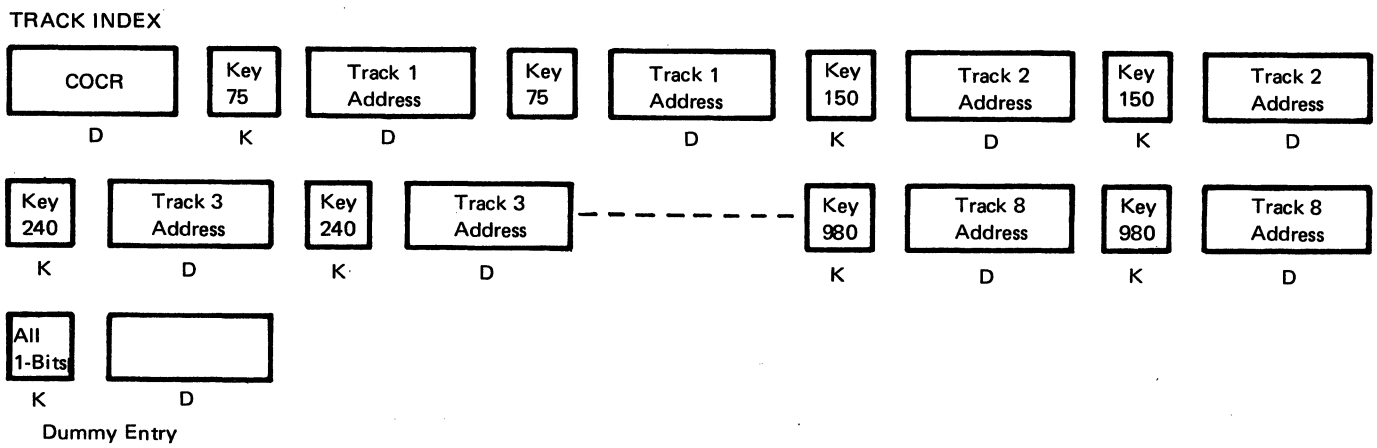
Each track index is built on the cylinder that it is indexing. The track index is located on the first track of each cylinder. The index can occupy a partial track, a full track, or more than one track. If the track index does not fill a track and if the remaining portion is large enough to hold any prime data records, then prime data records are stored on the remaining portion of the track.

The track index can contain the following types of entries:

- Normal Entry - Unshared
- Normal Entry - Shared
- Overflow Entry - Chained
- Overflow Entry - End
- Dummy Entry - End
- Inactive Entry.

When first created, the track index is formatted with two entries for each track used on the cylinder. These two entries are the normal entry and the overflow entry. Each entry is a DASD record containing a key area and a data area. Figure 22 is an example of a track index built for the prime data area of a logical file utilizing eight tracks on a cylinder.

The normal entry is the first of the two entries. After a track is loaded with records for a file, this entry has in its data area the address of the track referenced by the entry. The key of the last record on the track is maintained in the key area of the normal entry. The key area is changed each time a record is added to the track, so that it always reflects the key of the last record on the track. (Refer to the section "Add Records to a File".) When the first track containing prime data records is shared with the last or only track of the track index, the data area of the track index normal entry is modified to indicate a shared entry.



K = Key Area
D = Data Area
COCR = Cylinder Overflow Control Record (R0)

Figure 22. Schematic Example of a Track Index

The overflow entry is used both in the track index and in the sequence-link field of an overflow record. Refer to the section "Add Records to a File" for a description of the overflow entry in the sequence-link field. The overflow entry is required for handling overflow chaining when additional records are inserted into the file. Before a record is added to a track, the track index overflow entry for that track is similar to the normal entry in that they both contain the key of the last record on the track and the address of the track. Note that, at this point, the last record on the track is the last record placed on the track when the file was originally loaded. With overflow records, the data area of the overflow entry is changed to reflect the address of the lowest record in the overflow chain. An overflow chain is developed for each track. The key area cf the overflow entry is not changed, but always contains the key of the highest record, because records added to a track always have keys lower than the highest key originally loaded onto the track. The technique used to add records is explained in the section "Add Records to a File".

The two types of overflow entries in the track index are overflow chained entries and overflow end entries (see Figure 20). The data field cf the track index overflow entry is initially set to indicate an overflow end entry. If an overflow chain is later built, the overflow end entry indicates the last overflow record in the chain. An overflow chained entry is built to indicate an overflow chain exists. The data field of an overflow chained entry contains a pcinter to the lowest record in the overflow chain.

The last entry on a track index is always a dummy end entry. The dummy end entry indicates the end of the track index and indicates that any following records are logical file data records.

The key area of the dummy record is the same length as the user's key length and is filled with X'F's. The data field is the same length as the normal entries but is a null field.

Inactive track index entries are built during the load operation. For a 2311 DASD device type, inactive entries are written for the unused portion of the prime data extent. For all other DASD device types, inactive entries are written only for the unused portion of the last cylinder containing prime data records. The key area of inactive entry is filled with X'F's and is the same length as the user's key length. The data field is the same length as the normal entry. See Figure 20 for the format of the track index data area entries.

When the cylinder overflow option is specified by the user, record zero (track descriptor record) of track zero in the track index is used as a Cylinder Overflow Control Record (COCR). This entry is set up in the data area of record zero (R0). The address of the last overflow record on the cylinder and the number of tracks remaining in the cylinder overflow area are maintained by ISAM in this record. The format of the COCR is HHR00T00, where HHR = Address of last overflow record on cylinder. T = Number of tracks remaining in the cylinder overflow area. The COCR format is shown in Figure 23.

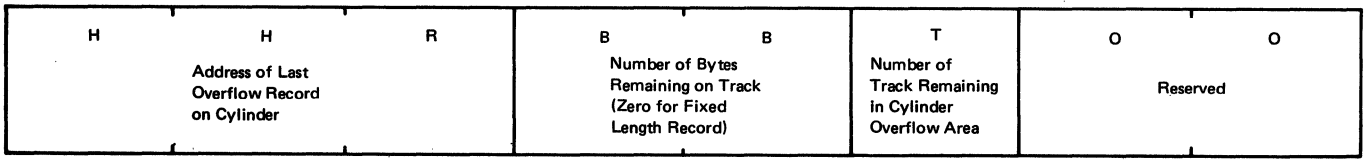


Figure 23. Cylinder Overflow Control Record (COCR)

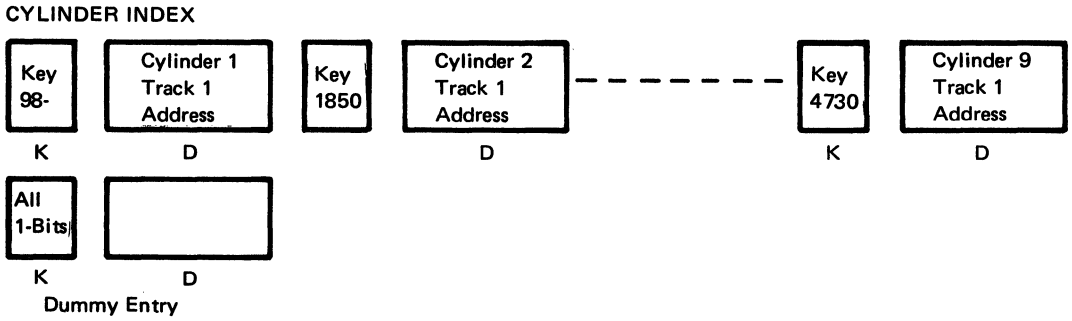


Figure 24. Schematic Example of a Cylinder Index

Cylinder Index (CI)

The cylinder index is present for all ISAM-organized files. It is an intermediate level index used to point to the correct track index.

The cylinder index can contain the following types of entries:

- Normal Entry
- Dummy Entry - Chained
- Dummy Entry - End
- Inactive Entry.

A cylinder index is built by ISAM to contain one index entry for each cylinder in the prime data area of the file. This entry contains the highest record key associated (in the cylinder or a corresponding overflow area) with the cylinder, and the address of the track index for that cylinder. Figure 24 is an example of a cylinder index built for a file requiring nine cylinders. The cylinder index can be located wherever the user chooses except on one of the cylinders that contain data records for the file. It must be on a separate cylinder or it can be placed on a separate volume that will be online whenever the logical file is processed. The cylinder index can also be located on one or more successive cylinders. When more than one cylinder is required, the last entry on each cylinder is a dummy chained entry that points to the first track of the next cylinder. However, the cylinder index cannot be continued from one volume to another. A job control EXTENT card must be used to specify the correct location for this index.

The last entry in the cylinder index is a dummy end entry. The key of the dummy entry is the same length as the user's key length and contains bytes of all one-bits. The data field is of the same length as the normal entries, but is a null field.

Inactive cylinder index entries have the same format as the track index inactive entries. They are written to provide for future expansion of the file and for OS/VS1 and OS/VS2 compatibility. An inactive cylinder index entry is written for each track in the cylinder containing track index inactive entries. See Figure 20 for the format of the cylinder index data area entries.

Master Index (MI)

The master index is the highest level index for a logical file built by ISAM. This index is optional; and if required, must be specified by the user in the DTFIS entry MSTIND.

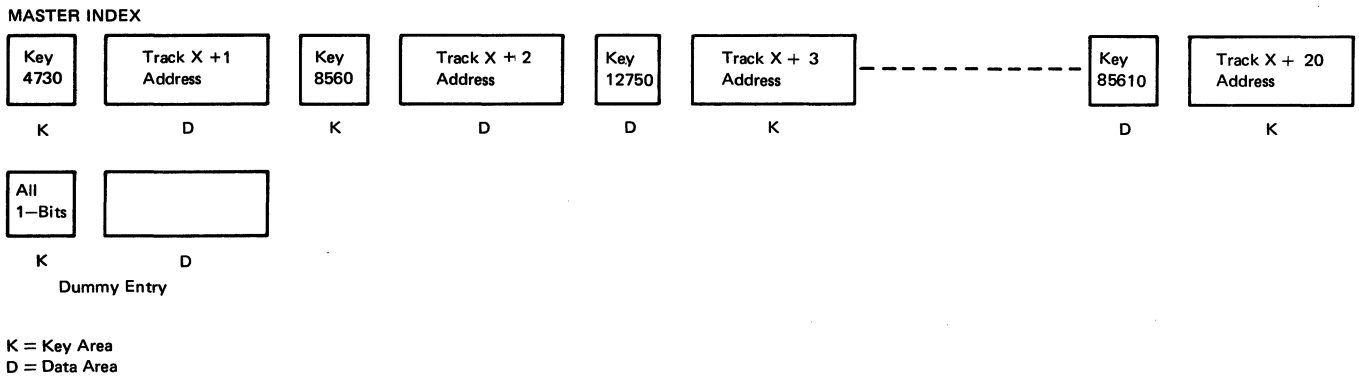


Figure 25. Schematic Example of a Master Index

The master index can contain the following types of entries:

- Normal Entry
- Dummy Entry - Chained
- Dummy Entry - End
- Inactive Entry.

The master index must immediately precede the cylinder index on a volume, and it may be located on one or more successive cylinders. Whenever it is continued from one cylinder to another, the last index entry on the first cylinder contains a linkage field that points to the first track of the next cylinder. This type of entry is a dummy chained entry. A master index may not be continued from one volume to another. It must be completely contained within one volume. The last track assigned to the master index area must be contiguous to the first track of the cylinder index area. A job control EXTENT card must be used to specify the correct location. Like the cylinder index, it can be located on the same volume with the data records or on a different volume that will be online whenever the records are processed.

The entries in this index point to each track of the cylinder index. Each entry contains the highest record key on the cylinder index track and the address of that track. For example, if a master index is located on track x and a cylinder index is located on tracks x+1 through x+20, the master index might contain the entries shown in Figure 25.

The last entry on the master index is a dummy end entry. The key of the dummy end entry is the same length as the user's key length and is filled with X'F's. The data field is of the same length as the normal entries, but is a null field.

Inactive master index entries have the same format as the track index inactive entries. They are written to provide for future expansion of the file and for OS/VS compatibility. An inactive entry is written for each track of the cylinder index containing inactive entries. See Figure 20 for the format of the master index data area entries.

FUNCTIONS PERFORMED BY ISAM

ISAM performs the following four basic functions as specified in the DTFIS entry, IOROUT:

- LOAD. To build a logical file on DASD or to extend a file beyond the highest record presently in an organized file.
- ADD. To insert new records into an organized file.
- RETRVE. To retrieve records from a file for either random or sequential processing and/or updating.
- ADDRTR. Both to insert new records into a file (ADD) and to retrieve records for processing and/or updating (RTR).

LOAD OR EXTEND A DASD FILE

Data records to be loaded onto a DASD file must be sorted into sequence by record key, before being presented to the ISAM load routines.

The data records are written by ISAM onto a DASD track in an area of the file (called the prime data area) specified by the user. The position of each logical record is a function of the record key used in the presort operation. That is, each record is written one after the other onto the prime data area of the logical file. The user must specify one extent for the prime data area on one pack. If a file is to be loaded onto more than one pack, the prime data area must continue from the last track of one pack to the first track of another pack. Extents must be adjacent.

The starting and ending limits of the prime data area are specified by the user in job control EXTENT cards. In addition, all packs to be used for a multipack file must be online throughout the load operation.

ADD RECORDS TO A FILE

After a logical file has been organized on DASD, it may subsequently become necessary to add records to the file. These records may contain keys that are above the highest key presently in the file and, thus, constitute an extension of the file. They may also contain keys that fall between or below keys already in the file and therefore require insertion in the proper sequence in the organized file.

If all records to be added have keys that are higher than the highest key in the organized file, the upper limit of the prime data area of the file can be adjusted (if necessary) by the specification in a job control EXTENT card, and the new records can be added by presorting them and loading them into the file. No overflow area is required. The file is merely extended further on the volume. However, new records can be batched with the normal additions and added to the end of the file.

If records must be inserted among those already organized, an overflow area is required. ISAM uses the overflow area to permit the insertion of records without necessitating a complete reorganization of the established file. The fast random and sequential retrieval of records is maintained by inserting references to the overflow chains in the track indexes, and by using a chaining technique in the overflow records. For chaining, a sequence-link field is prefixed to the user's data record in the overflow area. The sequence-link field enables ISAM to follow a chain of sequential records in a search for a particular record. This 10-byte sequence-link field has two types of entries: an overflow chained entry and an overflow end entry (see Figure 20).

The overflow chained entry contains the address of the record in the overflow area that has the next higher key. The overflow end entry indicates the end of the chain. All records in the overflow area are unblocked, regardless of the specification (in DTFIS RECFORM) for the data records in the logical file.

To add a record by insertion, ISAM searches the established indexes first to determine on which track the record must be inserted. After the proper track index entries are located, the point of insertion can then be determined. The keys of the last records on the tracks in the originally organized file determine the track where an inserted record belongs. A record is always inserted on the track where:

1. The last key is higher than the insertion, and
2. The last key of the preceding track is lower than the insertion.

After the proper track is determined, ISAM searches the individual records on the track or overflow area (if necessary) to find where the record belongs in key order. This results in either of two conditions:

1. The record falls between two records presently on the track. ISAM adds the record by inserting it in the proper sequence and shifting each succeeding record one record location higher on the track, until the end record is forced off the track. ISAM transfers the end record to the overflow area, and prefixes the record (data area) with a sequence-link field. The first time a record is inserted on a track, the sequence-link of the overflow record indicates that this is the highest record associated with the track. Thereafter, the sequence-link field of each overflow records points to the next higher record for that track. ISAM also updates the track index to reflect this change. The normal entry for the track has the key field changed to indicate the new last record located on the track. The overflow entry for the track has the track address (in the data area) changed to point to the address of the overflow record.
2. The record falls between the last record presently on the track and the last record originally on the track. Thus, it belongs in the overflow area. ISAM writes the record in the overflow area following the last record previously written. ISAM searches through the chain of records associated with the corresponding track for this record and identifies the sequential position the record should take. Then the sequence-link fields of the new record, and of the record preceding it by sequential key, are adjusted to point to the proper records.

RANDOM RECORD RETRIEVAL

Random retrieval from an indexed-sequential file is performed by the READ macro instruction. In response to the READ instruction, ISAM searches the indexes to locate the track containing the desired record and then searches the track for the record. The block containing the record is

read and the record is made available for processing. For both blocked and unblocked files, only the data portion of the record is read; the key field is not read.

After record processing has been completed, a WRITE macro instruction can be issued to write the record back in its original location. To allow overlap of input and output operations with processing, READ and WRITE do not wait for completion of the operations, but return control to the problem program. The WAITF macro instruction is used at the point in the program where processing must be held up until the I/O operation is complete.

SEQUENTIAL RECORD RETRIEVAL

Sequential retrieval from an indexed-sequential file begins at a location or record specified in a SETL macro instruction. Input blocks are read and each record is presented in sequence in response to the GET macro instruction. When necessary, ISAM reads those records from the overflow area that were displaced from the prime data area by added records. The track index overflow entry is used to indicate when this is necessary. The key field of unblocked records is read along with the data field. With blocked records, however, the key of the block (repeated in the last record of the block) is not read.

After record processing has been completed, a PUT can be issued to write the record back into its original location. If the file is blocked, the entire block is written back after either all records in the block have been processed and a GET is issued for the first record in the next block or an ESETL macro instruction is issued. The PUT macro instruction does not have to be issued for records that have not been changed; a series of GETs can be issued with no intervening PUT. The entire block is written back into the file if, and only if, a PUT is issued for any record in the block.

Once a SETL macro instruction has been issued, GET and PUT are the only I/O operations that can be performed before issuing an ESETL macro instruction. For example, if a WRITE is to be issued to add a record to a file that is being processed sequentially, it must be preceded by an ESETL. After adding the new record, the SETL macro instruction can be reissued, specifying the last record processed as the new starting point.

ROTATIONAL POSITION SENSING (RPS) SUPPORT

RPS is supported in ISAM for all channel programs built by ISMOD. This includes channel programs for all index levels and for both prime and overflow data. The support is provided for LOAD, ADD, and both SEQUENTIAL and RANDOM RETRIEVE modes.

RPS support is provided in LIOCS by dynamically extending the user DTFIS into the virtual area within the user's partition, and by linking the user DTFIS to an RPS version of the logic module in the SVA (Shared Virtual Area). The user must provide sufficient dynamically allocatable space in his partition for the RPS DTFIS extensions, and sufficient space in the SVA to contain the required RPS versions of the logic modules.

The RPS versions of the logic modules in the SVA are reenterable and therefore sharable between partitions. If the linkage to the original module is already coded read-only, the user-supplied save area is not used.

The RPS versions of the logic modules in the SVA are supersets of the functions needed to process the DTFIS being opened. Supersetting of RPS and non-RPS logic modules is not supported.

DTFISs in real partitions or partitions with insufficient allocatable virtual storage are opened without RPS support. If either the device or the system does not support RPS, the DTFIS is opened without RPS support.

The CCB CCW address and the module linkage fields in each DTFIS are modified to point to the DTFIS extension and the RPS version of the logic module in the SVA. Each DTFIS has three RPS indicators set on by OPEN. The first (byte 65, bit 4) indicates that the device containing the prime data being accessed is an RPS device; the second (byte 65, bit 5) indicates that the DTFIS has been extended into partition virtual space; the third (byte 65, bit 7) indicates that the device containing the index being accessed is an RPS device.

The RPS DTFIS extension(see Figure 26) contains CCW build and work areas necessary to construct RPS channel programs. In addition, the extension contains:

- A save area to force reentrancy on all imperative macro calls to the RPS versions of the logic modules.

- Information necessary to reestablish the original DTFIS at close time.
- The RPS error exit routine.

The RPS error exit routine reestablishes addressability to the RPS DTFIS extension and passes control to the ISAM module. This routine gains control when the user returns to the ISAM module via the error exit path.

The original DTFIS is used for all fields except the channel program building areas.

Displacement	Bytes	Contents
0	144	CCW build area
144	5	RPS sector arguments
149	15	RPS work area
164	4	Pointer to RPS error exit routine
168	4	Saved ISMOD register 14 at error exit time
172	4	Saved user register 13
176	4	Saved original CCW address
180	4	Saved original module address
184	72	User register save area
256	44	RPS error exit routine
300	84	Unused area in DTFIS extension

Figure 26. DTFIS Extension for RPS

Note: For an explanation of the Rotational Position Sensing (RPS) feature, refer to the appropriate hardware manual for the device type being used.

DTFIS MACRO

Before an indexed sequential file can be processed, it must be defined by the DTFIS declarative macro. Some of the fields within the DTFIS table generated from this macro instruction are not determined or filled in until the file is opened during execution of the program. Many of the fields in the table are retained with the file in the DASD format-2 label.

In addition to the parameters that describe the file to be processed, the DTFIS macro instruction includes certain parameters identical to those in the ISMOD macro instruction.

The following five DTF tables are generated according to function. They are:

DTFIS LOAD (see Figure 27)
DTFIS ADD (see Figure 28)
DTFIS RETRVE, RANDOM (see Figure 31)
DTFIS RETRVE, SEQNTL (see Figure 32)
DTFIS ADDRTR (see Figure 33)

In addition, the DTF tables for ADD, RETRVE, and ADDRTR are divided into the three parts that appear in the assembly listing. The first part of the DTF table is common to the ADD, RETRVE and ADDRTR functions. The rest of the table is variable and is generated according to the options specified in the DTFIS detail entries.

For a description of the DTFIS header entry and detail entries refer to VSE/Advanced Functions Macro Reference.

Note: The DTFIS may be altered when used by any of the compilers. For further information, refer to the Programmer's Guide for the appropriate compiler.

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename	IJHKCCB	0-15 (0-F)		Command Control Block (CCB).
		16 (10)	0-1	Not used.
			2	1 = COBOL open ignore option.
			3	Not used.
			4	1 = DTF table address constants relocated by OPENR.
			5	Not used.
			6	1 = Data set security.
			7	1 = Wrong block size error during file extension.
		17-19 (11-13)		Address of logic module.
		20 (14)		File type for OPEN/CLOSE (X'24' = LOAD).
	IJHKOPCO	21 (15)		Option byte.
			0	Not used.
			1	Not used.
			2	1 = Cylinder overflow option.
			3	Not used.
			4	1 = Blocked records (used by previous versions).
			5	1 = Verify.
			6	Not used.
			7	1 = Two I/O areas present.
		22-28 (16-1C)		File name.
	IJHKPDDV	29 (1D)		Prime data device type indicator. X'00' = 2311 X'01' = 2314/2319 X'04' = 3330 X'08' = 3340 general X'09' = 3340 35MB X'0A' = 3340 70MB.
&Filename.C	IJHKCCOD	30 (1E)		Status byte.
			0	1 = Uncorrectable DASD error (except WLR error).
			1	1 = WLR error.
			2	1 = Prime data area full.
			3	1 = Cylinder index area not large enough to reference prime data area. Set on only if error detected at SETFL time.
			4	1 = Master index not large enough to reference prime data area. Set on only if error detected at SETFL time.
			5	1 = Duplicate record.
			6	1 = Sequence error.
			7	1 = No EOF record written in prime data area.

Figure 27. DTFIS LOAD Table (Part 1 of 5)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
	IJHKHNDV	31 (1F)		High level index device type indicator. X'00' = 2311 X'01' = 2314/2319 X'04' = 3330 X'08' = 3340 general X'09' = 3340 35MB X'0A' = 3340 70MB.
		32 (20)		Relative position of the DSKXTN (logical unit, cell number) table (in words). This value is the length of the DTF table divided by 4.
		33-34 (21-22)		First prime data track in cylinder (HH).
		35 (23)		First prime data record in cylinder (R).
		36-37 (24-25)		Last prime data track in cylinder (HH).
		38 (26)		High record on master index/cylinder index track (R).
	IJHKNRPD	39 (27)		High record on prime data track (R).
		40 (28)		High record on overflow track (R).
	IJHKNRSH	41 (29)		High record on last track index track in cylinder (whether shared or unshared).
	IJHKNRTI	42 (2A)		High record on track index track other than last in cylinder. If only one track index track in cylinder, it is equal to Byte 41.
	IJHKFLAG	43 (2B)		Condition Code.
			0	1 = WLR checks requested (for extension).
			1	1 = First record in file.
			2	1 = Prime data extent full.
			3	1 = Master index/cylinder index extent too small.
			4	1 = Prime data upper limit has been increased (for extension).
			5	1 = Extension.
			6-7	Not used.
		44-50 (2C-32)		Prime data lower limit (MBBCCHH).
		51-57 (33-39)		Cylinder index lower limit (MBBCCHH).
		58-64 (3A-40)		Master index lower limit (MBBCCHH).
		65 (41)		Switches.
			0-3	Not used.
			4	1 = RPS type device (data).
			5	1 = RPS type DTF.
			6	1 = Master index.
			7	1 = RPS type device (index).

Figure 27. DTFIS LOAD Table (Part 2 of 5)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.H	IJHKLPDR	66-73 (42-49)		Address of last prime data record (MBBCCHHR).
	IJHKLGLN	74-75 (4A-4B)		Logical record length.
		76-77 (4C-4D)		Key length.
	IJHKBKLN	78-79 (4E-4F)		Block length (logical record length times number of records).
		80-81 (50-51)		Overflow record length (logical record length +10).
	IJHKNRCD	82-83 (52-53)		Blocking factor (number of logical records).
		84-85 (54-55)		Index entry length (key length +10).
		86-87 (56-57)		Prime data record length (key length + physical record length).
		88-89 (58-59)		Overflow record length with key (key length + logical record length + 10).
		90-91 (5A-5B)		Prime data record format length (key length + physical record length + 8).
		92-93 (5C-5D)		Overflow record format length (key length + logical record length + 18).
		94-95 (5E-5F)		Key location (in blocked records).
This is the end of the common DTF area. The format of the remainder of the table is variable and is generated according to the parameters specified in the DTFIS macro instruction.				
&Filename.S	IJHKSBKT	96-103 (60-67)		Seek/Search address area (MBBCCHHR).
&Filename.P	IJHKLGCT	104-105 (68-69)		Logical record counter (for blocking).
		106-107 (6A-6B)		Number of bytes for high level index.
		(6C-6F)		
		112 (70)		Status indicators.
			0-1	Not used.
			2	1= File closed.
			3-5	Not used.
			6	1 = Last prime data track full.
			7	1 = Last block full.
	IJHKLTIR	113-117 (71-75)		Last track index normal entry address (CCHHR).
	IJHKLCIR	118-122 (76-7A)		Last cylinder index entry address (CCHHR).
	IJHKLMIR	123-127 (7B-7F)		Last master index entry address (CCHHR).

Figure 27. DTFIS LOAD Table (Part 3 of 5)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.B				CCW build area. See description of SETFL macro, phase 1 - $$BSETFL.
		128-135 (80-87)		Seek CCW.
		136-143 (88-8F)		Search ID equal CCW.
		144-151 (90-97)		TIC CCW.
	IJHKRDWR	152-159 (98-9F)		Read/Write CCW.
		160-167 (A0-A7)		Search ID equal CCW.
		168-175 (A8-AF)		TIC CCW.
		176-183 (B0-B7)		Verify CCW.
&Filename.M	IJHKADCN	184-187 (B8-BB)		Address of IOAREAL.
		188-191 (BC-BF)		Address of data in WORKL. (FIXBLK = address of WORKL; FIXUNB = address of WORKL + key).
		192-195 (C0-C3)		Address of key in WORKL. (FIXBLK = address of WORKL + KEYLOC - 1; FIXUNB = address of WORKL.)
	IJHKBPOS	196-199 (C4-C7)		Block position indicator (address of logical record in IOAREAL).
	IJHKMIXT	200 (C8)	0-2	Master index, extension indicator. Not used.
			3	1 = Extending file, 0 = Creating file.
			4-6	Not used.
			7	1 = Master index being used, 0 = No master index being used.
		201-204 (C9-CC)		Cylinder index upper limit (CCHH).
		205-208 (CD-D0)		Master index upper limit (CCHH).
	IJHKPDUL	209-215 (D1-D7)		Prime data upper limit (old upper limit, if extension) (MBBCCHH).
		216-222 (D8-DE)		Prime data new upper limit (for extension) (MBBCCHH).
	IJHKLTM1	223 (DF)		Last prime data track in cylinder - 1.
	IJHKKLM1	224-225 (E0-E1)		Key length - 1.
	IJHKLLM1	226-227 (E2-E3)		Logical record length - 1.
	IJHKTIDR	228-229 (E4-E5)		Address of track index dummy record (HR).
	IJHKBFDR	230-231 (E6-E7)		Address of record before first prime data record in cylinder (HR).

Figure 27. DTFIS LOAD Table (Part 4 of 5)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
	IJHKNRCM	232 (E8)		Number of records on master index/cylinder index track - 1.
	IJHKCMCT	233-236 (E9-EC)		Master index/cylinder index DASD address control field (CCHH). 2311 = X'00C70009' 2314/2319 = X'00C70013' 3330 = X'01FF0012' 3340 = X'01FF000C'
	IJHKPDCT	237-239 (ED-EF)		Prime data address control field (CCH). 2311 = X'00C700' 2314/2319 = X'00C700' 3330 = X'01FF00' 3340 = X'01FF00'
	IJHKPDBG	240-242 (F0-F2)		Prime data beginning of volume (CCH). 2311 = X'000100' 2314/2319 = X'000100' 3330 = X'000100' 3340 = X'000100'
	IJHKPDEN	243-245 (F3-F5)		Prime data end of volume (CCH). 2311 = X'00C700' 2314/2319 = X'00C700' 3330 = X'019300' 3340 = X'015B00' (35MB) X'02B700' (70MB)
		246-247 (F6-F7)		Used for alignment.
&Filename.E	IJHKXTBL	248-251[1] (F8-FB)		First entry in DSKXTN table (logical unit, cell number).
		256-259[2] (100-103)		X'FFFFFFFF' = End of DSKXTN table.
		260-263 (104-107)		Address of IOAREA2.
		264-267 (108-10B)		Address used to relocate IOAREA2.

[1]Each entry in the DSKXTN table is four bytes long. The minimum number of entries is two. There is one entry per extent.

[2]Location of the end-of-table indicator depends on length of DSKXTN table.

Numbers in parentheses are displacements in hexadecimal notation.

Figure 27. DTFIS LOAD Table (Part 5 of 5)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename	IJHCCCB	0-15 (0-F)		CCB.
		16 (10)	0-1	Not used.
			2	1 = COBOL open ignore option.
			3	1 = Track hold specified.
			4	1 = DTF table address constants relocated by OPENR.
			5	Not used.
			6	1 = Data set security.
			7	1 = Wrong block size error during addition to file.
		17-19 (11-13)		Logic module address.
		20 (14)		File type for OPEN/CLOSE (X'25' = ADD).
	IJHCOPT	21 (15)		Option byte.
			0	Not used.
			1	1 = Prime data in core.
			2	1 = Cylinder overflow.
			3	1 = Cylinder index in core.
			4	1 = Blocked records.
			5	1 = Verify.
			6-7	Not used.
		22-28 (16-1C)		DTF file name.
	IJHCPDDV	29 (1D)		Prime data device type indicator. X'00' = 2311 X'01' = 2314/2319 X'04' = 3330 X'08' = 3340 general X'09' = 3340 35MB X'0A' = 3340 70MB.
&Filename.C	IJHCSTBY	30 (1E)		Status byte.
			0	1 = Uncorrectable DASD error (except WLR).
			1	1 = WLR error.
			2	1 = EOF (sequential).
			3	1 = No record found.
			4	1 = Illegal ID specified.
			5	1 = Duplicate record sensed.
			6	1 = Overflow area full.
			7	1 = Record retrieved from overflow area.
	IJHCHNDV	31 (1F)		Highest level index device type. X'00' = 2311 X'01' = 2314/2319 X'04' = 3330 X'08' = 3340 general X'09' = 3340 35MB X'0A' = 3340 70MB.

Numbers in parentheses are displacements in hexadecimal notation.

Figure 28. DTFIS ADD Table (Part 1 of 6)

DFT Assembler Label	Module DSECT Label	Bytes	Bits	Function
	IJHCPNT	32 (20)		Relative position of the DSKXTN (logical unit, cell number) table (in words). This value is the length of the DTF table divided by 4.
		33-35 (21-23)		First prime data record in cylinder (HHR).
		36-37 (24-25)		Last prime data track in cylinder (HH).
		38 (26)		High record number on master index/ cylinder index track (R).
	IJHCPDH	39 (27)		High record number on prime data track (R).
		40 (28)		High record number on overflow track (R).
	IJHCSTH	41 (29)		High record number on shared track (R).
	IJHCTIH	42 (2A)		High record number on track index (TI) track (R).
	IJHCRTR	43 (2B)		Retrieval byte.
			0	1 = WORKR area specified.
			1	1 = WORKS area specified.
			2	Overflow switch.
			3	1 = Read.
			4	Not used.
			5	1 = Output.
			6	1 = Write key.
			7	1 = PUT macro issued.
		44-50 (2C-32)		Prime data lower limit (MBBCCHH).
	IJHCCIS	51-57 (33-39)		Cylinder index lower limit (MBBCCHH).
	IJHCMIS	58-64 (3A-40)		Master index lower limit (MBBCCHH).
	IJHCILN	65 (41)		Switches.
			0	1 = From WAITF routine.
			1	1 = WAITF seek check bit.
			2-3	Not used.
			4	1 = RPS type device (data).
			5	1 = RPS type DTF.
			6	1 = Master index.
			7	1 = RPS type device (index).
&Filename.H	IJHCCLPA	66-73 (42-49)		Last prime data record address (MBBCCHHR).
	IJHCRESZ	74-75 (4A-4B)		Logical record length (RECSIZE).
	IJHCKYSZ	76-77 (4C-4D)		Key length (KEYLEN).
	IJHCBLSZ	78-79 (4E-4F)		Block size (logical record length times number of records).

Figure 28. DTFIS ADD Table (Part 2 of 6)

DFT Assembler Label	Module DSECT Label	Bytes	Bits	Function
	IJHCRL10	80-81 (50-51)		Overflow record length (logical record length + 10).
	IJHCBFAC	82-83 (52-53)		Blocking factor (number of logical records in block (NRECDS)).
		84-85 (54-55)		Index entry length (key length + 10).
	IJHCABCD	86-87 (56-57)		Prime data record length (key length plus physical record length (block size)).
		88-89 (58-59)		Overflow record length plus key (key length + logical record length + 10).
	IJHCCMAX	90-91 (5A-5B)		Prime data record format length (key length + block size + 8).
		92-93 (5C-5D)		Overflow record format length (key length + logical record length + 18).
	IJHCKYLC	94-95 (5E-5F)		Key location (KEYLOC) for blocked records.
		96-97 (60-61)		Constant = 5.
		98-99 (62-63)		Constant = 10.
	IJHCATB2	100-101 (64-65)		Displacement of Part 2 of the DTFIS table from start of Part 1.
	IJHCATB3	102-103 (66-67)		Displacement of Part 3 of the DTFIS table from start of Part 1.
&Filename.S	IJHCSADR	104-113 (68-71)		Seek/search address area (MBBCCHHRFP).
&Filename.W	IJHCBKCT	114-123 (72-7B)		Random/sequential retrieval work area.
&Filename.P	IJHACPRC	124-127 (7C-7F)		Prime data record count.
	IJHACSTI	128 (80)	0-1	Status indicators. Not used.
			2	1 = File Closed.
			3-5	Not used.
			6	1 = Last prime data track full.
			7	1 = Block complete.
	IJHACLTA	129-133 (81-85)		Last track index normal entry address (CCHHR).
	IJHACLCA	134-138 (86-8A)		Last cylinder index entry address (CCHHR).
	IJHACLMA	139-143 (8B-8F)		Last master index entry address (CCHHR).
	IJHACLOA	144-151 (90-97)		Last independent overflow record address (MBBCCHHR).

Figure 28. DTFIS ADD Table (Part 3 of 6)

DFT Assembler Label	Module DSECT Label	Bytes	Bits	Function
&Filename.I	IJHACCTC	152-153 (98-99)		Number of independent overflow tracks.
&Filename.A	IJHACOFC	154-155 (9A-9B)		Number of full cylinder overflow areas.
&Filename.O	IJHACCRC	156-157 (9C-9D)		Overflow record count.
	IJHACCLL	158-164 (9E-A4)		Independent overflow area lower limit (MBBCCHH).
	IJHACCUP	165-171 (A5-AB)		Independent overflow area upper limit (MBBCCHH).
	IJHAHRAA	172-175 (AC-AF)		A(&Filename.D) - Address of work area for cylinder overflow control record (COCR).
		176-179 (B0-B3)		A(&Filename.D+8) - Address of work area for the current track index normal entry count field.
		180-183 (B4-B7)		A(&Filename.D+16) - Address of work area for current track index overflow entry count field.
		184-187 (B8-BB)		A(&Filename.D+24) - Address of work area for current prime data record count field.
		188-191 (BC-BF)		A(&Filename.D+32) - Address of work area for current overflow record count field.
		192-195 (C0-C3)		A(&Filename.D+40) - Address of work area for track index normal entry data field.
	IJHADLNK	196-199 (C4-C7)		A(&Filename.D+50) - Address of work area for current overflow record linkage field.
	IJHAARAD	200-203 (C8-CB)		A(&IOAREAL) - Address of IOAREAL, the I/O area used for adding records to a file.
	IJHACUSE	204-207 (CC-CF)		A(&WORKL) - Address of WORKL, work area containing user data records to be added to the file.
	IJHADKEY	208-211 (D0-D3)		A(&Filename.K) - Address of the ADD key area.
		212-215 (D4-D7)		A(&IOAREAL+8) - Address of key position in IOAREAL.
	IJHAKLN8	216-219 (D8-DB)		A(&IOAREAL+8+&KEYLEN) - Address of data position in IOAREAL.

Numbers in parentheses are displacements in hexadecimal notation.

Figure 28. DTFIS ADD Table (Part 4 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.2	IJHCASAD	0-3 (0-3)		A(&Filename.S+3) - Address of the seek/search address area+3.
		4 (4)	0	1 = Seek check indicated.
			1-5	Not used.
			6	1 = Over/under seek has occurred.
			7	1 = An error has been found, but a seek check is indicated.
		5-7 (5-7)		A(&Filename.W) - Address of random/sequential retrieval work area.
The following information is generated if the cylinder index in core option is specified.				
	IJHCORST	12-15 (0C-0F)		A(&INDAREA) - Starting address of main storage area specified for cylinder index.
		16-17 (10-11)		AL2(&INDSIZE) - Number of bytes in main storage available for cylinder index.
		18-25 (12-19)		Next cylinder index entry to be read (MBBCCHHR).
		26-30 (1A-1E)		Last cylinder index entry (CCHHR).
	IJHCORBT	31(1F)		Core index byte.
			0	1 = First time through B-transient, $$BINDEX.
			1	1 = End of cylinder index reached.
			2	1 = Index skip option specified.
			3	1 = Suppress in-core option and read cylinder index.
			4-7	Not used.
	IJHCORKY	32-35 (1D-23)		Pointer to key (stored by module).
The following information is generated if the prime data in core add function is specified. This information is aligned on a double word boundary.				
	IJHPSIZE	36-37 (24-25)		Size of IOAREAL.
	IJHPMAX	38-39 (26-27)		Maximum number of prime data records in main storage.
	IJHPDSP1	40-43 (28-2B)		Address of write CCWs.
	IJHPDSP2	44-47 (2C-2F)		Address of read CCWs.
	IJHPSW	48(30)		Switch byte.
			0	1 = EOF.
			1-7	Not used.
		49(31)		Reserved.
	IJHDCWRK	50-51 (32-33)		Work field for I/O module.

Numbers in parentheses are displacements in hexadecimal notation.

Figure 28. DTFIS ADD Table (Part 5 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.B		0-7 (0-7)		CCW X'07', &Filename.S+1, X'40', 6 - Long seek CCW with command chaining.
	IJHCCCW	8-127 (8-7F)		Channel program build area. See Figures 37-58 for a description of the channel program builder.
&Filename.D	IJHACOCR	128-135 (80-87)		Cylinder overflow control record (COCR).
	IJHACTNA	136-143 (88-8F)		Current track index normal entry count field address.
	IJHACTOA	144-151 (90-97)		Current track index overflow entry count field address.
	IJHACRID	152-159 (98-9F)		Current prime data record count field address.
	IJHACFID	160-167 (A0-A7)		Current overflow record count field address.
	IJHACTIN	168-177 (A8-B1)		Track index normal entry data field.
	IJHACINK	178-187 (B2-BB)		Current overflow record sequence link field.
	IJHACTIA	188-197 (BC-C5)		Current track index overflow entry data field.
	IJHAGATE	198 (C6)		X'01' - Add to EOF. X'02' - Add to independent overflow area.
		199-201 (C7-C9)		Overflow control bytes (CCH).
	IJHAOCOH	202-203 (CA-CB)		High HR on overflow track. See Figure 29.
		204-211 (CC-D3)		Volume upper limit for prime data records (MBBCCHHR). See Figure 30.
	IJHAICOM	212-217 (D4-D9)		CLC 0(&KEYLEN,13),0(6) - Unblocked CLC 0(&KEYLEN,13),&KEYLOC-1(6) - Blocked Utility CLC for key.
	IJHAISKY	218-223 (DA-DF)		MVC 0(&KEYLEN,13),0(12) - Unblocked MVC 0(&KEYLEN,13),&KEYLOC-1(12) - Blocked Utility MVC for key.
&Filename.E		224-227[1] (E0-E3)		First entry in DSKXTN table (logical unit, cell number).
		232-235[2] (E8-EB)		4X'FF' - End of DSKXTN table.
&Filename.K		236+ (EC-end)		Key area for ADD only. Number of bytes depends on key length, KEYLEN.

[1] Each entry in the DSKXTN table is four bytes long. The minimum number of entries is two. There is one entry per extent.
[2] Location of the end-of-table indicator depends on length of DSKXTN table.

Numbers in parentheses are displacements in hexadecimal notation.

Figure 28. DTFIS ADD Table (Part 6 of 6)

2311	2314/2319
M = Extent sequence number	M = Extent sequence number
BB = 00	BB = 00
C = 0	C = 0
C = 199	C = 199
H = 0	H = 0
H = 9 - CYLOFL (number of tracks reserved for cylinder overflow)	H = 19 - CYLOFL (number of tracks reserved for cylinder overflow)
R = Number of records that fit on an overflow track	R = Number of records that fit on an overfow track

3330	3340 (35MB)	3340 (70MB)
M = Extent sequence number	M = Extent sequence number	M = Extent sequence number
BB = 0	BB = 0	BB = 0
CC = 403	CC = 347	CC = 695
H = 0	H = 0	H = 0
H = 18 - CYLOFL (number of tracks reserved for cylinder cverflow)	H = 11 - CYLOFL (number of tracks reserved for cylinder overflow)	H = 11 - CYLOFL (number of tracks reserved for cylinder overflow)
R = Number of records that fit on overflow track	R = Number of records that fit on overflow track	R = Number of records that fit on overflow track

Figure 29. Overflow Area Upper Limits (MBBCCHHR)

2311/2314/2319/3330/3340
M = Extent sequence number
BB = 00
CC = 199 for 2311/2314/2319 = 403 for 3330 = 347 for 3340 (35MB) = 695 for 3340 (70MB)
H = 0
H = last prime data track in cylinder
R = Last record on current track

Figure 30. End of Volume Limits for Prime Data Area (MBBCCHHR)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename	IJHCCCB	0-15 (0-F)		Command Control Block (CCB).
		16 (10)	0	Not used.
			1	1 = GET issued.
			2	1 = COBOL open ignore option.
			3	1 = HOLD option specified.
			4	1 = DTF table address constants relocated by OPENR.
			5-6	Not used.
			7	1 = Different blocksize in format-1 label than in DTFIS.
		17-19 (11-13)		Address of logic module.
		20 (14)		File type for OPEN/CLOSE (X'26' = RETRVE).
	IJHCOPT	21 (15)		Option byte.
			0	Not used.
			1	1 = Prime data in core.
			2	1 = Cylinder overflow option.
			3	1 = Cylinder index in core option.
			4	1 = Blocked records.
			5	1 = Verify.
			6-7	Not used.
		22-28 (16-1C)		File name (DTF name).
	IJHCPDDV	29 (1D)		Prime data device type. X'00' = 2311 X'01' = 2314/2319 X'04' = 3330 X'08' = 3340 general X'09' = 3340 35MB X'0A' = 3340 70MB.
&Filename.C	IJHCSTBY	30 (1E)		Status byte.
			0	1 = Uncorrectable DASD error (except WLR error).
			1	1 = WLR error.
			2	1 = EOF (sequential).
			3	1 = No record found.
			4	1 = Illegal ID specified.
			5	1 = Duplicate record sensed.
			6	1 = Overflow area full.
			7	1 = Record retrieved from overflow area.
	IJHCHNDV	31 (1F)		High level index device type. X'00' = 2311 X'01' = 2314/2319 X'04' = 3330 X'08' = 3340 general X'09' = 3340 35MB X'0A' = 3340 70MB.
	IJHCPNT	32 (20)		Relative position of the DSKXTN (logical unit, cell number) table (in words). This value is the length of the DTF table divided by 4.

Figure 31. DTFIS RETRVE, RANDOM Table (Part 1 of 6)

DFT Assembler Label	Module DSECT Label	Bytes	Bits	Function
		33-35 (21-23)		First prime data record in cylinder (HHR).
		36-37 (24-25)		Last prime data track in cylinder (HH).
		38 (26)		High record number on master index/cylinder index track (R).
	IJHCPDH	39 (27)		High record number on prime data track (R).
		40 (28)		High record number on overflow track (R).
	IJHCSTH	41 (29)		High record number on shared track (R).
	IJHCTIH	42 (2A)		High record number on track index track (R).
	IJHCRTR	43 (2B)		Retrieval byte.
			0	1 = WORKR specified.
			1	1 = WORKS specified.
			2	Overflow switch.
			3	1 = Read key.
			4	Not used.
			5	1 = Output.
			6	1 = Write key.
			7	1 = PUT macro issued.
		44-50 (2C-32)		Prime data lower limit (MBBCCHH).
	IJHCCIS	51-57 (33-39)		Cylinder index lower limit (MBBCCHH).
	IJHCMIS	58-64 (3A-40)		Master index lower limit (MBBCCHH).
	IJHCILN	65 (41)		Switches.
			0	1 = From WAITF routine.
			1	1 = Seek check from WAITF.
			2	1 = Data track held.
			3	1 = Index track held.
			4	1 = RPS type device (data).
			5	1 = RPS type DTF.
			6	1 = Master index.
			7	1 = RPS type device (index).
	IJHCCLPA	66-73 (42-49)		Last prime data record address (MBBCCHHR).
	IJHCRESZ	74-75 (4A-4B)		Logical record length.
	IJHCKYSZ	76-77 (4C-4D)		Key length.
	IJHCBLSZ	78-79 (4E-4F)		Block size (logical record length times number of records).
	IJHCRL10	80-81 (50-51)		Overflow record length (logical record length + 10).

Figure 31. DTFIS RETRVE, RANDOM Table (Part 2 of 6)

DFT Assembler Label	Module DSECT Label	Bytes	Bits	Function
	IJHCEFAC	82-83 (52-53)		Blocking factor.
		84-85 (54-55)		Index entry length (key length + 10).
	IJHCABCD	86-87 (56-57)		Prime data record length (key length + physical record length).
		88-89 (58-59)		Overflow record length with key (key length + logical record length + 10).
	IJHCCMAX	90-91 (5A-5B)		Prime data record format length (key length + physical record length + 8).
		92-93 (5C-5D)		Overflow record format length (key length + logical record length + 18).
	IJHCKYLC	94-95 (5E-5F)		Key location (blocked records).
		96-97 (60-61)		Constant = 5.
		98-99 (62-63)		Constant = 10.
	IJHCATB2	100-101 (64-65)		Displacement of Part 2 of the DTFIS table from Part 1.
	IJHCATB3	102-103 (66-67)		Displacement of Part 3 of the DTFIS table from Part 1.
&Filename.S	IJHCSADR	104-113 (68-71)		Seek/search address area (MBBCCHHRFP).
&Filename.W	IJHCBCKT	114-123 (72-7B)		Random/sequential retrieval work area.

Figure 31. DTFIS RETRVE, RANDOM Table (Part 3 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.2	IJHCASAD	0-3 (0-3)		Address of seek/search address area + 3.
		4 (4)	0 1-5 6 7	1 = Seek check indicated. Not used. 1 = Over/under seek has occurred. 1 = An error has been found, but a seek check is indicated.
		5-7 (5-7)		Address of random/sequential retrieval work area.
	IJHSIOAR	8-11 (8-B)		Address of IOAREAS.
	IJHCRARA	12-15 (C-F)		Address of IOAREAR.
	IJHCRKEY	16-19 (10-13)		Address of KEYARG.
	IJHCRWOR	20-23 (14-17)		Address of WORKR.
	IJHSDB1	24-27 (18-1B)		Current sequential I/O area address.
	IJHSLIOR	28-31 (1C-1F)		4-byte NO-OP instruction, or L IOREG,*-4 if IOREG was specified.
	IJHSLMIT	32 (20)		X'00' = No verify, X'40' = Verify.
		33 (21)		X'08' = Unblocked, X'00' = Blocked.
		34 (22)		R = First prime data record on shared track.
		35-39 (23-27)		Upper limit for sequential retrieval (CCHHR).
	IJHSINIT	40-41 (28-29)		H'0' = Blocked records. H'2' = Overflow record. H'8' = Unblocked records.
		42 (2A)		X'C7' = 2311, 2314, or 2319; X'FF' = 3330, 3340.
		43-47 (2B-2F)		Initial values for sequential retrieval.
&Filename.H	IJHSCADR	48-55 (30-37)		Current DASD address for sequential (MBBCCHHR).
	IJHSCOVF	56-63 (38-3F)		Current overflow DASD address for sequential (MBBCCHHR).
	IJHSRCNT	64-65 (40-41)		Sequential record counter.
	IJHSTICU	66-67 (42-43)		Current track index entry for sequential (HR).

Figure 31. DTFIS RETRVE, RANDOM Table (Part 4 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.T		68-69 (44-45)		Number of records tagged for deletion.
	IJHRREGS	70-71 (46-47)		Load IOREG for random retrieval.
&Filename.G	IJHRIDSV	72-79 (48-4F)		DASD address save area (MBBCCHHR).
	IJHRADSV	80-83 (50-53)		Record pointer within I/O area for write operation.
&Filename.R	IJHROVCN	84-87 (54-57)		Nonfirst overflow record count.
The following information is generated when the cylinder index in core option is specified.				
	IJHCORST	92-95 (5C-5F)		A(&INDAREA) - Starting address of main storage area specified for cylinder index.
		96-97 (60-61)		AL2(&INDSIZE) - Number of bytes in main storage available for cylinder index.
		98-105 (62-69)		Next cylinder index entry to be read (MBBCCHHR). (Initialized by $$BINDEX to cylinder index starting address.)
		106-110 (6A-6E)		Last cylinder index entry.
		(6F)	0	1 = First time through transient.
			1	1 = End of index reached.
			2	1 = Index skip option.
			3-7	Not used.
	IJHCORKY	112-115 (70-73)		Pointer to key (stored by the module).
		116-131 (74-83)		Reserved.

Figure 31. DTFIS RETRVE, RANDOM Table (Part 5 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.B		0-7 (0-7)		X'07',&Filename.S+1,X'40',6 - Long seek CCW with command chaining.
	IJHCCCW	8-63 (8-3F)		Area to build CCW string. See Figures 61-67 for a description of the channel program builder for random retrieval.
&Filename.E		64-67[1] (40-43)		First entry in DSKXTN table (logical unit, cell number).
		72-75[2] (48-4B)		4X'FF' End of DSKXTN table.

[1]The length of one entry is the four bytes shown here. The minimum number of entries is 2. There is one entry per extent.

[2]The location of the end-of-table indicator depends on length of DSKXTN table.

Numbers in parentheses are displacements in hexadecimal notation.

Figure 31. DTFIS RETRVE, RANDOM Table (Part 6 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename	IJHCCCB	0-15 (0-F)		Command Control Block (CCB).
		16 (10)	0	Not used.
			1	1 = GET issued.
			2	1 = COBOL open ignore option.
			3	1 = Track hold specified.
			4	1 = DTF table address constants relocated by OPENR.
			5	1 = EOF on sequential retrieve.
			6	1 = Data set security.
			7	1 = Different block size in format-1 label than in DTFIS.
		17-19 (11-13)		Address of logic module.
		20 (14)		File type for OPEN/CLOSE (X'26' = RETRVE).
	IJHCOPT	21 (15)		Option byte.
			0	Not used.
			1	1 = Prime data in core.
			2	1 = Cylinder overflow option.
			3	1 = Cylinder index in core option.
			4	1 = Blocked records.
			5	1 = Verify.
			6	1 = IOAREAS just used, 0 = IOAREA2 just used.
			7	1 = Two I/O areas present.
		22-28 (16-1C)		File name (DTF name).
	IJHCPDDV	29 (1D)		Prime data device type. X'00' = 2311 X'01' = 2314/2319 X'04' = 3330 X'08' = 3340 general X'09' = 3340 35MB X'0A' = 3340 70MB.
&Filename.C	IJHCSTBY	30 (1E)		Status byte.
			0	1 = Uncorrectable DASD error (except WLR error).
			1	1 = WLR error.
			2	1 = EOF (sequential).
			3	1 = No record found.
			4	1 = Illegal ID specified.
			5	1 = Duplicate record sensed.
			6	1 = Overflow area full.
			7	1 = Record retrieved from overflow area.
	IJHCHNDV	31 (1F)		High level index device type. X'00' = 2311 X'01' = 2314/2319 X'04' = 3330 X'08' = 3340 general X'09' = 3340 35MB X'0A' = 3340 70MB.

Figure 32. DTFIS RETRVE, SEQNTL Table (Part 1 of 6)

DFT Assembler Label	Module DSECT Label	Bytes	Bits	Function
	IJHCPNT	32 (20)		Relative position of the DSKXTN (logical unit, cell number) table (in words). This value is the length of the DTF table divided by 4.
		33-35 (21-23)		First prime data record in cylinder (HHR).
		36-37 (24-25)		Last prime data track in cylinder (HH).
		38 (26)		High record number on master index/cylinder index track (R).
	IJHCPDH	39 (27)		High record number on prime data track (R).
		40 (28)		High record number on overflow track (R).
	IJHCSTH	41 (29)		High record number on shared track (R).
	IJHCTIH	42 (2A)		High record number on track index track (R).
	IJHCRTR	43 (2B)		Retrieval byte.
			0	1 = WORKR specified.
			1	1 = WORKS specified.
			2	Overflow switch.
			3	1 = Read key.
			4	1 = First record being processed (after issuing SETL macro).
			5	1 = Output.
			6	1 = Write key.
			7	1 = PUT macro issued.
		44-50 (2C-32)		Prime data lower limit (MBBCCHH).
	IJHCCIS	51-57 (33-39)		Cylinder index lower limit (MBBCCHH).
	IJHCMIS	58-64 (3A-40)		Master index lower limit (MBBCCHH).
	IJHCILN	65 (41)		Switches.
			0	1 = From WAITF routine.
			1	1 = WAITF seek check bit.
			2-3	Not used.
			4	1 = RPS type device (data).
			5	1 = RPS type DTF.
			6	1 = Master index.
			7	1 = RPS type device (index).
	IJHCCLPA	66-73 (42-49)		Last prime data record address (MBBCCHHR).
	IJHCRESZ	74-75 (4A-4B)		Logical record length.
	IJHCKYSZ	76-77 (4C-4D)		Key length.
	IJHCBLSZ	78-79 (4E-4F)		Block size (logical record length times number of records).

Figure 32. DTFIS RETRVE, SEQNTL Table (Part 2 of 6)

DFT Assembler Label	Module DSECT Label	Bytes	Bits	Function
	IJHCRL10	80-81 (50-51)		Overflow record length (logical record length + 10).
	IJHCBFAC	82-83 (52-53)		Blocking factor.
		84-85 (54-55)		Index entry length (key length + 10).
	IJHCABCD	86-87 (56-57)		Prime data record length (key length + physical record length).
		88-89 (58-59)		Overflow record length with key (key length + logical record length + 10).
	IJHCCMAX	90-91 (5A-5B)		Prime data record format length (key length + physical record length + 8).
		92-93 (5C-5D)		Overflow record format length (key length + logical record length + 18).
	IJHCKYLC	94-95 (5E-5F)		Key location (blocked records).
		96-97 (60-61)		Constant = 5.
		98-99 (62-63)		Constant = 10.
	IJHCATB2	100-101 (64-65)		Displacement of Part 2 of the DTFIS table from Part 1.
	IJHCATB3	102-103 (66-67)		Displacement of Part 3 of the DTFIS table from Part 1.
&Filename.S	IJHCSADR	104-113 (68-71)		Seek/search address area (MBBCCHHRFP).
&Filename.W	IJHCBCKT	114-123 (72-7B)		Random/sequential retrieval work area.

Figure 32. DTFIS RETRVE, SEQNTL Table (Part 3 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.2	IJHCASAD	0-3 (0-3)		Address of seek/search address area + 3.
		4 (4)	0	1 = Seek check indicated.
			1-5	Not used.
			6	1 = Over/under seek has occurred.
			7	1 = An error has been found, but a seek check is indicated.
		5-7 (5-7)		Address of random/sequential retrieval work area.
	IJHSICAR	8-11 (8-B)		Address of IOAREAS.
	IJHCRARA	12-15 (C-F)		Address of IOAREA2.
	IJHCRKEY	16-19 (10-13)		Address of KEYARG.
	IJHCRWOR	20-23 (14-17)		Address of WORKR.
	IJHSDB1	24-27 (18-1B)		Current sequential I/O area address.
	IHJSLIOR	28-31 (1C-1F)		L IOREG,*-4 - Load IOREG if IOREG was specified, or a 4-byte NO-OP instruction.
	IJHSLMIT	32 (20)		X'00' = No verify, X'40' = Verify.
		33 (21)		X'08' = Unblocked records, X'00' = Blocked records.
		34 (22)		R = First prime data record on shared track.
		35-39 (23-27)		Upper limit for sequential retrieval (CCHHR).
	IJHSINIT	40-41 (28-29)		H'0' = Blocked records, H'2' = Overflow record, H'8' = Unblocked records.
		42 (2A)		X'C7' = 2311, 2314, or 2319; X'FF' = 3330, 3340.
		43-47 (2B-2F)		Initial values for sequential (CCHHR).
&Filename.H	IJHSCADR	48-55 (30-37)		Current DASD address for sequential retrieval (MBBCCHHR).
	IJHSCOVF	56-63 (38-3F)		Current overflow DASD address (MBBCCHHR).
	IJHSRCNT	64-65 (40-41)		Sequential record counter.
	IJHSTICU	66-67 (42-43)		Current track index entry (HR).

Figure 32. DTFIS RETRVE, SEQNTL Table (Part 4 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.T		68-69 (44-45)		Number of records tagged for deletion.
		70-75 (46-4B)		For boundary alignment.
		76-91 (4C-5B)		Reserved.

Figure 32. DTFIS RETRVE, SEQNTL Table, (Part 5 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.B		0-7 (0-7)		X'07',&Filename.S+1, X'40',6 - Long seek CCW with command chaining.
	IJHCCCW	8-63 (8-3F)		Area to build CCW string. See Figures 68-75 for a description of the channel program builder for sequential retrieval.
&Filename.E		64-67[1] (40-43)		First entry in DSKXTN table (logical unit, cell number).
		72-75[2] (48-4B)		4X'FF' - End of DSKXTN table.

[1]The length of one entry is the four bytes shown here. The minimum number of entries is 2. There is one entry per extent.

[2]The location of the end-of-table indicator depends on length of DSKXTN table.

Number in parentheses are displacements in hexadecimal notation.

Figure 32. DTFIS RETRVE, SEQNTL Table (Part 6 of 6)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename	IJHCCCB	0-15 (0-F)		CCB.
		16 (10)	0	Not used.
			1	1 = GET issued.
			2	1 = COBOL open ignore option.
			3	1 = Track hold option specified.
			4	1 = DTF table address constants relocated by OPENR.
			5	EOF switch.
			6	1 = Data set security.
			7	1 = Wrong block size error during addition to file.
		17-19 (11-13)		Logic module address.
		20 (15)		File type for OPEN/CLOSE (X'27' = ADDRTR).
	IJHCOPT	21 (15)		Option byte.
			0	Not used.
			1	1 = Prime data in core.
			2	1 = Cylinder overflow.
			3	1 = Cylinder index in core.
			4	1 = Blocked records.
			5	1 = Verify.
			6	1 = IOAREAS just used, 0 = IOAREA2 just used.
			7	1 = Two I/O areas present.
		22-28 (16-1C)		DTF file name.
	IJHCPDDV	29 (1D)		Prime data device type indicator. X'00' = 2311 X'01' = 2314/2319 X'04' = 3330 X'08' = 3340 general X'09' = 3340 35MB X'0A' = 3340 70MB.
&Filename.C	IJHCSTBY	30 (1E)		Status byte.
			0	1 = Uncorrectable DASD error (except WLR).
			1	1 = WLR error.
			2	1 = EOF (sequential).
			3	1 = No record found.
			4	1 = Illegal ID specified.
			5	1 = Duplicate record sensed.
			6	1 = Overflow area full.
			7	1 = Record retrieved from overflow area.
	IJHCHNDV	31 (1F)		Highest level index device type. X'00' = 2311 X'01' = 2314/2319 X'04' = 3330 X'08' = 3340 general X'09' = 3340 35MB X'0A' = 3340 70MB.

Figure 33. DTFIS ADDRTR Table (Part 1 of 7)

DFT Assembler Label	Module DSECT Label	Bytes	Bits	Function
	IJHCPNT	32 (20)		Relative position of the DSKXTN (logical unit, cell number) table (in words). This value is the length of the DTF table divided by 4.
		33-35 (21-23)		First prime data record in cylinder (HHR).
		36-37 (24-25)		Last prime data track in cylinder (HH).
		38 (26)		High record number on master index/cylinder index track (R).
	IJHCPDH	39 (27)		High record number on prime data track (R).
		40 (28)		High record number on overflow track (R).
	IJHCSTH	41 (29)		High record number on shared track (R).
	IJHCTIH	42 (2A)		High record number on track index (TI) track (R).
	IJHCRTR	43 (2B)		Retrieval byte.
			0	1 = WORKR area specified.
			1	1 = WORKS area specified.
			2	Overflow switch.
			3	1 = Read.
			4	1 = First record being processed (after issuing SETL macro).
			5	1 = Output.
			6	1 = Write key.
			7	1 = PUT macro issued.
		44-50 (2C-32)		Prime data lower limit (MBBCCHH).
	IJHCCIS	51-57 (33-39)		Cylinder index lower limit (MBBCCHH).
	IJHCMIS	58-64 (3A-40)		Master index lower limit (MBBCCHH).
	IJHCILN	65 (41)		Switches.
			0	1 = From WAITF routine
			1	1 = Seek check from WAITF.
			2	1 = Data track held.
			3	1 = Index track held.
			4	1 = RPS type device (data).
			5	1 = RPS type DTF.
			6	0 = Cylinder index.
				1 = Master Index.
			7	1 = RPS type device (index).
&Filename.H	IJHCCLPA	66-73 (42-49)		Last prime data record address (MBBCCHHR).
	IJHCRESZ	74-75 (4A-4B)		Logical record length (RECSIZE).

Figure 33. DTFIS ADDRTR Table (Part 2 of 7)

DFT Assembler Label	Module DSECT Label	Bytes	Bits	Function
	IJHCKYSZ	76-77 (4C-4D)		Key length (KEYLEN).
	IJHCBLSZ	78-79 (4E-4F)		Block size (logical record length times number of records).
	IJHCRL10	80-81 (50-51)		Overflow record length (logical record length + 10).
	IJHCBFAC	82-83 (52-53)		Blocking factor (number of logical records in block (NRECDS)).
		84-85 (54-55)		Index entry length (key length + 10).
	IJHCABCD	86-87 (56-57)		Prime data record length (key length plus physical record length (block size)).
		88-89 (58-59)		Overflow record length with key (key length + logical record length + 10).
	IJHCCMAX	90-91 (5A-5B)		Prime data record format length (key length + block size + 8).
		92-93 (5C-5D)		Overflow record format length (key length + logical record length + 18).
	IJHCKYLC	94-95 (5E-5F)		Key location (KEYLOC) for blocked records.
		96-97 (60-61)		Constant = 5.
		98-99 (62-63)		Constant = 10.
	IJHCATB2	100-101 (64-65)		Displacement of Part 2 of the DTFIS table from start of Part 1.
	IJHCATB3	102-103 (66-67)		Displacement of Part 3 of the DTFIS table from start of Part 1.
&Filename.S	IJHCSADR	104-113 (68-71)		Seek/search address area.
&Filename.W	IJHCBKCT	114-123 (72-7B)		Random/sequential retrieval work area.
&Filename.P	IJHACPRC	124-127 (7C-7F)		Prime data record count.
	IJHACSTI	128 (80)		Status indicators.
			0-1	Not used.
			2	1 = File closed.
			3-5	Not used.
			6	1 = Last prime data track full.
			7	1 = Block complete.
	IJHACLTA	129-133 (81-85)		Last track index normal entry address (CCHHR).
	IJHACLCA	134-138 (86-8A)		Last cylinder index entry address (CCHHR).

Figure 33. DTFIS ADDRTR Table (Part 3 of 7)

DFT Assembler Label	Module DSECT Label	Bytes	Bits	Function
	IJHACLMA	139-143 (8B-8F)		Last master index entry address (CCHHR).
	IJHACLOA	144-151 (90-97)		Last independent overflow record address (MBBCCHHR).
&Filename.I	IJHACOTC	152-153 (98-99)		Number of independent overflow tracks.
&Filename.A	IJHACOFC	154-155 (9A-9B)		Number of full cylinder overflow areas.
&Filename.C	IJHACORC	156-157 (9C-9D)		Overflow record count.
	IJHACOLL	158-164 (9E-A4)		Independent overflow area lower limit (MBBCCHH).
	IJHACOUP	165-171 (A5-AB)		Independent overflow area upper limit (MBBCCHH).
	IJHAHRAA	172-175 (AC-AF)		A(&Filename.D) - Address of work area for cylinder overflow control record (COCR).
		176-179 (B0-B3)		A(&Filename.D+8) - Address of work area for the current track index normal entry count field.
		180-183 (B4-B7)		A(&Filename.D+16) - Address of work area for current track index overflow entry count field.
		184-187 (B8-BB)		A(&Filename.D+24) - Address of work area for current prime data record count field.
		188-191 (BC-BF)		A(&Filename.D+32) - Address of work area for current overflow record count field.
		192-195 (C0-C3)		A(&Filename.D+40) - Address of work area for track index normal entry data field.
	IJHADLNK	196-199 (C4-C7)		A(&Filename.D+50) - Address of work area for current overflow record sequence-link field.
	IJHAARAD	200-203 (C8-CB)		A(&IOAREAL) - Address of IOAREAL, the I/O area used for adding records to a file.
	IJHACUSE	204-207 (CC-CF)		A(&WORKL) - Address of WORKL, work area containing user data records to be added to the file.
	IJHADKEY	208-211 (D0-D3)		A(&Filename.K) - Address of the ADD key area
		212-215 (D4-D7)		A(&IOAREAL+8) - Address of key position in IOAREAL.
	IJHAKLN8	216-219 (D8-DB)		A(&IOAREAL+8+&KEYLEN) - Address of data position in IOAREAL.

Figure 33. DTFIS ADDRTR Table (Part 4 of 7)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.2	IJHCASAD	0-3 (0-3)		A(&Filename.S+3) - Address of the seek/search address area+3.
		4 (4)	0	1 = Seek check indicated.
			1-5	Not used.
			6	1 = Over/under seek has occurred.
			7	1 = An error has been found, but a seek check is indicated.
		5-7 (5-7)		A(&Filename.W) - Address of the random/ sequential retrieval work area.
	IJHSIOAR	8-11 (8-B)		Address of IOAREAS, I/O area used for sequential retrieval.
	IJHCRARA	12-15 (C-F)		Address of IOAREAR, I/O area used for random retrieval or address of IOAREA2 (if specified) for sequential retrieval.
	IJHCRKEY	16-19 (10-13)		Address of KEYARG, field containing user-supplied key used for random READ/WRITE operations and sequential retrieval initiated by key.
	IJHCRWOR	20-23 (14-17)		Address of WORKR, work area used for random retrieval.
	IJHSDB1	24-27 (18-1B)		Current sequential I/O area address.
	IJHSLIOR	28-31 (1C-1F)		1. L IOREG,*-4 - Load I/O register for sequential or 2. 4-byte NO-OP instruction for random.
	IJHSLMIT	32 (20)		X'00' = No Verify; X'40' = Verify.
		33 (21)		X'00' = Blocked; X'08' = Unblocked.
		34 (22)		R = First prime data record on shared track.
		35-39 (23-27)		Limits for sequential (CCHHR).
	IJHSINIT	40-41 (28-29)		H'0' = Blocked records. H'2' = Overflow record. H'8' = Unblocked records.
		42 (2A)		X'C7' = 2311, 2314, or 2319; X'FF' = 3330, 3340.
		43-47 (2B-2F)		Initial values for sequential.
&Filename.H	IJHSCADR	48-55 (30-37)		Current sequential DASD address (MBBCCHHR).
	IJHSCOVF	56-63 (38-3F)		Current overflow DASD address (MBBCCHHR).
	IJHSRCNT	64-65 (40-41)		Sequential record count.
	IJHSTICU	66-67 (42-43)		Current track index entry for sequential (HR)

Figure 33. DTFIS ADDRTR Table (Part 5 of 7)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.T		68-69 (44-45)		Number of records tagged for deletion.
	IJHRREGS	70-71 (46-47)		LR &IOREG,0 for random (or 2-byte NO-OP for sequential).
&Filename.G	IJHRIDSV	72-79 (48-4F)		DASD address save area for random retrieval (MBBCCHHR).
	IJHRADSV	80-83 (50-53)		Record pointer within I/O area for write (for random retrieval).
&Filename.R	IJHROVCN	84-87 (54-57)		Nonfirst overflow record count.
The following information is generated if the cylinder index in core option is specified. Bytes 88-91 (58-5B) are not used.				
	IJHCORST	92-95 (5C-5F)		A(&INDAREA) - Starting address of main storage area specified for cylinder index.
		96-97 (60-61)		AL2(&INDSIZE) - Number of bytes in main storage available for cylinder index.
		98-105 (62-69)		Next cylinder index entry to be read (MBBCCHHR).
		106-110 (6A-6E)		Last cylinder index entry (CCHHR).
	IJHCORBT	111 (6F)		Core index byte.
			0	1 = First time through $$BINDEX.
			1	1 = End of cylinder index reached.
			2	1 = Index skip option specified.
			3	1 = Suppress index in-core option and read cylinder index.
			4-7	Not used.
	IJHCORKY	112-115 (70-73)		Pointer to key (stored by module).
The following information is generated if the prime data in-core add function is specified. This information is aligned on a double word boundary. If both cylinder index in-core and prime data in-core add functions are specified, the following information is found in bytes 116-131 (74-83).				
	IJHPSIZE	116-117 (74-75)		Size of IOAREAL.
	IJHPMAX	118-119 (76-77)		Maximum number of prime data records in main storage.
	IJHPDSP1	120-123 (78-7B)		Address of write CCWs.
	IJHPDSP2	124-127 (7C-7F)		Address of read CCWs.
	IJHPSW	128(80)		Switch byte.
			0	1 = EOF (Bits 1-7 not used).
		129 (81)		Reserved.
	IJHDCWRK	130-131 (82-83)		Work field for I/O module.

Figure 33. DTFIS ADDRTR Table (Part 6 of 7)

DTF Assembly Label	Module DSECT Label	Bytes	Bits	Function
&Filename.B		0-7		X'07', &Filename.S+1, X'40', 6 - Long seek
		(0-7)		CCW with command chaining.
	IJHCCCW	8-63		Channel program build area. See Figures
		(8-3F)		76-112 for a description of the channel
				program builder.
		64-127		Channel program build area for add function
		(40-7F)		only.
&Filename.D	IJHACOCR	128-135		Cylinder overflow control record (COCR).
		(80-87)		
	IJHACTNA	136-143		Current track index normal entry count
		(88-8F)		field.
	IJHACTOA	144-151		Current track index overflow entry count
		(90-97)		field.
	IJHACRID	152-159		Current prime data record count field.
		(98-9F)		
	IJHACFID	160-167		Current overflow record count field.
		(A0-A7)		
	IJHACTIN	168-177		Track index normal entry data field.
		(A8-B1)		
	IJHACLNK	178-187		Current overflow record sequence-link field.
		(B2-BB)		
	IJHACTIA	188-197		Current track index overflow entry data
		(BC-C5)		field.
	IJHAGATE	198		X'01' - Add to EOF.
		(C6)		X'02' - Add to independent overflow area.
		199-201		Overflow control bytes (CCH).
		(C7-C9)		
	IJHAOCOH	202-203		High HR on overflow track. See Figure 29.
		(CA-CB)		
		204-211		Volume upper limit for prime data records
		(CC-D3)		(MBBCCHHR). See Figure 30.
	IJHAICOM	212-217		CLC 0(&KEYLEN,13),0(6) - Unblocked
		(D4-D9)		CLC 0(&KEYLEN,13),&KEYLOC-1(6) - Blocked
				Utility CLC for key.
	IJHAISKY	218-223		MVC 0(&KEYLEN,13),0(12) - Unblocked
		(DA-DF)		MVC 0(&KEYLEN,13),&KEYLOC-1(12) - Blocked
				Utility MVC for key.
&Filename.E		224-227[1]		First entry in DSKXTN table (logical unit,
		(E0-E3)		cell number).
		232-235[2]		4X'FF' - End of DSKXTN table.
		(E8-EB)		
&Filename.K		236+		Key area for add only. Number of bytes
		(EC-end)		depends on key length, KEYLEN.

[1] Each entry in the DSKXTN table is four bytes long. The minimum number of entries is two. There is one entry per extent.
[2] Location of the end-of-table indicator depends on length of DSKXTN table.

Numbers in parentheses are displacements in hexadecimal notation.

Figure 33. DTFIS ADDRTR Table (Part 7 of 7)

ISMOD MACRO

The ISMOD (Indexed-Sequential Module) macro instruction must be included for each logic module required to support each DTFIS macro in a particular problem program. The logic modules are described by an ISMOD header entry and a series of parameter entries. See VSE/Advanced Functions Macro Reference for an explanation of the parameters.

The following imperative macros use the logic in the ISMOD:

```
ESETL
GET
PUT
READ
WAITF
WRITE, KEY
WRITE, NEWKEY
```

The logic for the other imperative macros used by ISAM (ENDFL, ESETL, SETFL and SETL) is fcund in various B-transient routines. Flowcharts for ISFMS are in alphabetical order by macro within function. The functions appear in the following sequence:

```
LOAD
ADD
RETRVE, RANDOM
RETRVE, SEQNTL
ADDRTR
```

This section does not discuss each of these macros separately but, instead, presents them in the context of a particular function.

REENTERABLE MODULE: A reenterable module is a logic module that can be asynchronously used, or shared, by more than one file. ISMOD is made reenterable by inclusion of the RDONLY=YES parameter in the ISMOD macro instruction. The RDONLY (read-only) parameter assures, regardless of the processing requirements of any file(s) using the module, that the generated logic module is never modified in any way. This feature is implemented through the establishment of unique (one for each task using the module) save areas external to the logic module. Each save area must be 72 bytes and doubleword aligned. The save area for ISMOD contains general registers 2-14, the last overflow record address, the new overflow record address, and a work area for the channel program builder. A task must provide the address of its unique save area in register 13 before an imperative macrc is issued to the file and a logic module is entered by the task.

ERROR OPTION EXTENSIONS: When ERREXT is not specified and an unrecoverable I/O error occurs, ISFMS indicates this error in Filename.C and returns to the problem program. Control is returned to ISFMS only by issuing another macro instruction.

When ERREXT is specified and an unrecoverable I/O error occurs, bit 0 of Filename.C is set on. Also, byte 2, bit 2 of the CCB in the DTF is set on when data transfer has not occurred. The problem program error processing routine should determine if data transfer occurred by checking the data transfer bit (byte 2, bit 2) in the DTF. Information concerning the record being read or written and the operation being performed at the time of the error can be found in the 18-byte parameter list pointed to by register 1. See Figure 34 for a description of this parameter list.

Bytes	Bits	Contents
0-3		DTF address.
4-7		Main storage address of the record in error.
8-15		DASD address of record in error (MBBCCHHR), where M is the extent sequence number and R is the record number. R can be inaccurate if a read error occurred during a read of the highest level index.
16		Record identification:
	0	Data.
	1	Track index.
	2	Cylinder index.
	3	Master index.
		Type of operation:
	4	Not used.
	5	Not used.
	6	Write.
	7	Read.
17		Command code of failing CCW.

Figure 34. ERREXT Parameter List

After checking for errors and taking corrective action if necessary, the problem program error processing routine can return to ISAM via the ERET macro. The ERET IGNORE or ERET SKIP macro returns to ISAM to ignore the error condition and process the record. The ERET RETRY returns to ISAM to make another attempt to read or write the record.

Note: The ERREXT coding is not designed to handle irrecoverable errors that are posted in Filename.C. Examples of irrecoverable errors are No Record Found, Prime Data Area Full, Master Index Full, and so on.

DOUBLE BUFFERING: Double buffering is meaningful only when creating the file or sequentially retrieving from the file. If IOAREA2=YES is specified as an ISMOD macro parameter, and the presence of two I/O

areas is indicated in the DTF table, overlapping of I/O with processing is provided for the load create and sequential retrieve functicns.

ISAM MACRO INSTRUCTIONS TO LOAD OR EXTEND A DASD FILE

The function of originally loading a file of presorted records onto DASD, and the function of extending the file by adding new presorted records beyond the previous high record, are the same. Both are considered a LOAD operation (specified by the DTFIS entry IOROUT), and they both use the same macro instructions in the problem program. However, the type field in the DLBL card must specify ISC for load creation and ISE for load extension.

The areas of the volumes used for the file are specified by job control EXTENT cards. The areas are: the prime data area where the data records are written, a cylinder index area where the user wants ISAM to build the cylinder index, and a master index area if a master index is to be built (specified by the DTFIS entry MSTIND).

During the load operation, ISAM builds the track, cylinder, and master (if specified) indexes.

Three different macro instructions are always required in the problem program to load original or extension records into the logical file on DASD.

The SETFL (set file load mode) macro instruction causes ISAM to set up the file so that the load or extension function can be performed. When loading a file, SETFL preformats the last track of each track index; but when extending the file, SETFL preformats only the last track of the last track index plus each new track index for the extension of the file. This allows prime data on a shared track to be referenced even though no track index entries exist on the shared track.

This macro must be issued whenever the file is to be loaded or extended.

When a WRITE macro instruction with the parameter NEWKEY is issued in the problem program between a SETFL instruction and an ENDFL instructicn (the third macro required for loading), it causes ISAM to load a record onto DASD.

Before issuing the WRITE instruction, the problem program must store the key and data portions of the record in a work area (specified by DTFIS WORKL). The ISAM routines construct the I/O area by moving the data record to the data area, moving the key to the key area, and building the count field. When the I/O area has been filled, ISAM transfers the records to DASD storage and then constructs the count field for the next record. The WAITF macro should not be used when loading or extending an ISAM file.

Before records are transferred, ISAM performs both a sequence check (to ensure that the records are in order by key) and a duplicate-record check.

After each WRITE is issued, ISAM makes the ID of that record or block available to the problem program. The ID is located in an 8-byte field labeled Filename.H.

As records are loaded on DASD, ISAM writes track index entries each time a track is filled, writes a cylinder index entry each time a cylinder is filled, and writes a master index entry (if DTFIS MSTIND is specified) each time a track of the cylinder index is filled.

The ENDFL macro performs an operation (similar to a CLOSE) for the file that has been loaded. It writes the last block of data records, if necessary, and then writes an end-of-file record after the last data record. It writes any index entries that are needed. It also writes inactive track index entries for the unused portion of the prime data extent for the 2311 device type. For DASD types other than 2311, only the remaining portion of the last cylinder containing prime data records has inactive track index entries.

When extending or adding to a file, the user is responsible for checking byte 16, bit 7 of the DTF to determine whether the correct blocksize has been specified.

ISAM MACRO INSTRUCTIONS FOR ADDING RECORDS TO A FILE

After a file has been organized on DASD, new records can be added to the file. Each record is inserted in the proper place sequentially by key. This function is provided by specifying ADD or ADDRTR in the DTFIS entry IOROUT.

The file can contain either blocked or unblocked records, as specified by the DTFIS entry RECFORM. When the file contains blocked records, the user must provide ISAM with the location of the key field that is provided through the DTFIS entry KEYLOC. The records to be inserted are written one record at a time. The records must contain a key field in the same location as the records already in the file. Whenever the addition of records is to follow sequential retrieval (ADDRTR), the macro instruction ESETL must be issued before a record is added.

Two macro instructions, WRITE NEWKEY and WAITF are used in the problem program for adding records to a file.

Before the WRITE macro is issued for unblocked records, the program must store the record (key and data) to be added into a work area specified in the DTFIS entry WORKL. For blocked records, the program must store only the data (the key is assumed to be a part of the data). Before any records are transferred, ISFMS checks for duplicate record keys. If no duplication is found, ISAM inserts the record in the file.

To insert a record into a file, ISAM performs an index search at the highest level index. This search determines if the record to be inserted can be placed within the file, or if it is higher than the last record on the file.

If the record can be inserted within the file, searching of the master index (if available), the cylinder index, and the track index determines the appropriate location to insert the record.

For an entry to an unblocked file, an equal/high search is performed in the prime data area of the track. When a record on the track is found that is equal to or higher than the record to be inserted, the record is read from the track and placed in storage (in the I/O area). The two records are compared to see if a duplicate record is found. If a duplicate record is found, that information is posted to the user in the DTF table at Filename.C. If no duplicate is found, the appropriate record (in the user's work area) is written directly on the track. The record (just displaced from the track) in the I/O area is moved by ISAM to the user's work area. The next record on the track is read into the I/O area.

Then, the record in the work area is written on the track. Succeeding records are shifted until the last record on the track is set up as an overflow record. If the ADD I/O area (IOAREAL) is increased to permit the reading or writing of more than one record on DASD at a time, an equal/high search is performed in the prime data area of the track. When a record on the track is found that is equal to or higher than the record to be inserted, as many records as can fit into the I/O area specified in the DTFIS operand IOAREAL are read from the track and placed in storage (in the I/O area).

The record to be added is compared to existing records in the I/O area. If a duplicate key is found, the condition is posted to the user in the DTF table Filename.C. If no duplicate is found, the records are shifted in storage, leaving the record with the highest key remaining in the user's work area. The other records are rewritten directly onto the track. Any remaining record(s) on the track are then read into the I/O area. The process continues until the last record on the track is set up as an overflow record.

This last record is then written into the appropriate overflow area, and the appropriate track index entries are updated. This is the cylinder overflow area, if CYLOFL has been specified for this file and the area has not been filled.

If the cylinder overflow area is filled, or if only an independent overflow area has been specified by a job control EXTENT card, the end record is transferred to the independent overflow area. If an independent overflow area has not been specified (or is filled) and the cylinder overflow area is filled, there is no room available to store the overflow record. ISAM posts this condition in the DTF table at Filename.C.

In all cases, before any records are written, ISAM determines if room is available.

For an entry to a blocked file, the work area, WORKL, is required in the DTFIS entries. Each record to be added must contain a key field in the same location as the records already in the file. The high-order position of this key field, relative to the leftmost position of the logical record, must be specified to ISAM by the user. The DTFIS entry KEYLOC is used for this specification.

When the WRITE macro is issued in the problem program, ISAM first locates the correct track by referring to the necessary master (if available), cylinder, and track indexes. Then, a search on the key areas of the DASD records on the track is made to locate the desired block of records. The block of records (or as many as will fit into the I/O area if IOAREAL has been increased for reading and writing more than one record on DASD at a time) is read into the I/O area. ISAM then examines the key field within each logical record to find the exact position in which to insert the new record and to check for duplication of records. If duplication of keys exists, the condition is posted in Filename.C. If the key of the record to be inserted (contained in the work area WORKL) is low, it is exchanged with the record presently in the block.

This procedure continues with each succeeding record in the block until the last record is moved into the work area. ISFMS then updates the key area of the DASD record to reflect the highest key in the block. If the IOAREAL has been increased, succeeding blocks in the I/O area are also updated. The block (or blocks) is then written back onto DASD. The remaining blocks on the track are similarly processed

until the last logical record on the track is moved into the work area. This record is then set up as an overflow record with the proper sequence-link field and moved to the overflow area. The indexes are updated and ISAM returns to the problem program for the next record to be added. If the overflow area is filled, the information is posted in Filename.C.

If the proper track for a record is an overflow track (determined by the track index), ISAM searches the overflow chain and checks for duplication. If no duplication is found, ISAM writes the record, preceded by a sequence-link field in the data area of the DASD record, and adjusts the appropriate linkages to maintain sequential order by key. ISAM writes the new record in either the cylinder overflow area or the independent overflow area. If these areas are filled, the user is notified by a bit in Filename.C.

If the new record is higher than all records presently in the file (end-of-file), ISAM checks to determine if the last track containing data records is filled. If it is not, the new record is added, replacing the end-of-file record. The end-of-file record is written in the next record location on the track, or on the next available prime data track. Another track must be available within the file limits. If the end-of-file record is the first record on any track, the new record is written in the appropriate overflow area. After each new record is inserted in its proper location, ISAM adjusts all indexes that are affected by the addition.

The WAITF macro instruction is issued to ensure that the transfer of a record has been completed.

This instruction must be issued before the problem program attempts to process an input record or build another output record for the file concerned. The program does not regain control until the previous transfer of data is complete.

ISAM MACRO INSTRUCTIONS FOR RANDOM RETRIEVAL

When a file has been organized by ISAM, records can be retrieved in random order for processing and/or updating. Retrieval must be specified in the DTFIS entry IOROUT (IOROUT=RETRVE or IOROUT=ADDRTR). Random processing must be specified in the DTFIS entry TYPEFLE=RANDOM.

Because random reference to the file is by record key, the problem program must supply the key of the desired record to ISAM. To do this, the key must be stored in the storage key field specified by the DTFIS entry KEYARG. The specified key designates both the record to be retrieved and the record to be written back into the file in an updating operation. Records added to the file between the READ and the WRITE macro for a particular record to be updated can result in a lost record and a duplicate key.

Three macro instructions (READ KEY, WRITE KEY and WAITF) are available for use in the problem program for retrieving and updating records randomly.

The READ KEY instruction used in conjunction with WAITF macro instruction causes ISAM to retrieve the specified record from the file.

To locate the record, ISAM searches the indexes to determine the track on which the record is stored, and then searches the track for the specific record. When the record is found, ISAM transfers it to the I/O area specified by the DTFIS entry IOAREAR. The ISAM routines also move the record to the specified work area if the DTFIS entry WORKR is included in the file definition.

When records are blocked, ISAM transfers the block that contains the specified record to the I/O area. It makes the individual record available for processing either in the I/O area or the work area (if specified). For processing in the I/O area, ISAM supplies the address of the record in the register specified by DTFIS IOREG. The ID of the record can be referenced by using Filename.G.

The WRITE instruction with the parameter KEY is used in conjunction with the WAITF macro instruction for random updating. It causes ISAM to transfer the specified record from main storage to DASD storage.

ISAM rewrites the record retrieved by the previous read instruction for the same file. The record is updated from the work area, if one is specified; otherwise, from the I/O area. The key need not be specified again ahead of the WRITE instruction.

The WAITF macro instruction is issued to ensure that the transfer of a record has been completed. This instruction must be issued before the problem program attempts to process an input record or build another output record for the file concerned. The program does not regain control until the previous transfer of data is complete.

The WAITF instruction posts any exceptional information in the DTFIS table at Filename.C.

ISAM MACRO INSTRUCTIONS FOR SEQUENTIAL RETRIEVAL

When a file has been organized by ISAM, records can be retrieved in sequential order by key for processing and/or updating. The DTFIS entry IOROUT=RETRVE must be specified. Sequential processing must be specified in the DTFIS entry TYPEFLE=SEQNTL.

Although records are retrieved in order by key, sequential retrieval can start at a record in the file identified either by key or by the ID (identifier in the count field) of a record in the prime data area. Sequential retrieval can also start at the beginning of the logical file. The user specifies, in SETL, the type of reference he will use in the problem program.

Whenever the starting reference is by key and the file contains blocked records (RECFORM=FIXBLK), the user must also provide ISAM with the position of the key field within the records. This is specified in the DTFIS entry KEYLOC. To search for a record, ISAM first locates the correct block by the key in the key area of the DASD record. (The key area contains the key of the highest record in the block.) Then, ISAM examines the key field within each record in the block to find the specified record.

Four macro instructions (SETL, GET, PUT and ESETL) are available for use in the problem program for retrieving and updating records sequentially.

The SETL (set limits) macro instruction initiates the mode for sequential retrieval and initializes the ISAM routines to begin retrieval at the specified starting address. It requires two parameters. The first operand (Filename) specifies the name of the file (specified in the DTFIS header entry) from which records are to retrieved.

The second operand specifies where processing is to begin. If the user is processing by the record ID, the operand Idname or (r) specifies the symbolic name of the main-storage field in which the user supplies the starting (or lowest) reference for ISAM use. The symbolic field contains information as shown in Figure 35. If processing is to begin with a key supplied by the user, the second operand is KEY. The key is to be supplied by the user in the field specified by the DTFIS entry KEYARG. If the specified key is not present in the file, an indication will be given at Filename.C.

The second operand BOF specifies that retrieval is to start at the beginning of the logical file.

Selected groups of records within a file containing identical characters or data in the first locations of each key can be processed by specifying GKEY in the second operand. The GKEY specification allows processing to begin at the first record (or key) within the desired group. The user must supply a key that will identify the significant (high order) bytes of the required group of keys. The remainder (or insignificant bytes) of the key must be padded with blanks, binary zeros, or bytes lower in collating sequence than any of the insignificant bytes in the first key of the group to be processed. The problem program

Byte	Identifier	Contents	Information
0	M	2-245	Extent sequence number of the volume in which the starting record is located.
1-2	B,B	0,0 (for disk)	Always zero for disk.
3-4	C,C	0,1-199 (for 2311/2314/2319) 0-403 (for 3330) 0-347 (for 3340 with 35MB) 0-695 (for 3340 with 70MB)	Cylinder number for disk.
5-6	H,H	0,0-9 (for 2311) 0,0-19 (for 2314/2319) 0,0-18 (for 3330) with 70MB)	Head position for 2311, 2314, 2319, 3330, and 3340 disks.
7	R	1-254	Record location.

Figure 35. Pointer to First Record to be Processed by Sequential Retrieval

must determine when the generic group is completed. Otherwise, ISAM continues through the remainder of the group.

This method also allows starting at a key equal to or greater than the one specified in the DTFIS entry KEYARG without getting an error indication in Filename.C.

The GET macro instruction causes ISAM to retrieve the next record in sequence from the file. It can be written in either of two forms, depending on where the record is to be processed.

The first form is used if records are to be processed in the I/O area (specified by DTFIS IOAREAS). It requires only one parameter, which is the name of the file from which the record is to be retrieved. ISFMS transfers the record from this file to the I/O area, and the record is available for the execution of the next instruction in the problem program. The key is located at the beginning of IOAREAS and the register (IOREG) points to the data. If blocked records are specified, ISAM makes each record available by supplying its address in the register specified by the DTFIS entry IOREG. The key is contained in the record.

The second form of the GET instruction is used if records are to be processed in a work area (DTFIS specifies WORKS). It requires two parameters both of which can be specified as symbols or in register notation. The first is the name of the file, and the second is the name of the work area. When register notation is used, workname should not be preloaded into register 1. The record is available for the execution of the next program instruction.

If blocked records are specified in the file definition, each GET that transfers a block of records to main storage will, if necessary, also write the preceding block back into the file in its previous block location. GET writes the preceding block if a PUT instruction has been issued for at least one of the records in the block. If no PUT instructions have been issued, updating is not required for this block, and GET does not cause the block to be rewritten. Whenever an unblocked record is retrieved from the prime data area, ISAM supplies the ID of that record in the field addressed by Filename.H. If blocked records are specified, ISAM supplies the ID of the block. The PUT macro instruction is used for sequential updating of a file, and causes ISAM to transfer records to the file in sequential order. PUT returns a record that was obtained by a GET. It can be written in either of two forms, depending on where records are processed.

The first form is used if records are processed in the I/O area (specified by DTFIS IOAREAS). It requires only the name of the file to which the records are to be transferred.

The second form of the PUT instruction is used if records are processed in a work area. It requires two parameters, both of which can be specified either as a symbol or in register notation. The first is the name of the file, and the second is the name of the work area. When register notation is used, workname should not be loaded into register 1. The work area name may be the same as that specified in the preceding GET for this file, but this is not required. ISAM moves the record from the work area specified in the PUT instruction to the I/O area specified for the file in the DTFIS entry IOAREAS.

When unblocked records are specified, each PUT writes a record back onto the file in the same location from which it was retrieved by the preceding GET for this file. Thus, each PUT updates the last record that was retrieved from the file. If some records do not require updating, a series of GET instructions can be issued without intervening PUT instructions. Therefore, it is not necessary to rewrite unchanged records.

When blocked records are specified, PUT instructions do not transfer records to the file. Instead, each PUT indicates that the block is to be written after all the records in the block have been processed. When processing for the block is complete and a GET is issued to read the next block into main storage, that GET also writes the completed block back into the file in its previous location. If a PUT instruction is not issued for any record in the block, GET does not write the completed block. The ESETL macro instruction writes the last block processed, if necessary, before the end of file. The ESETL (end set limit) macro instruction ends the sequential mode initiated by the SETL macro. If blocked records are specified, ESETL writes the last block back if a PUT was issued.

Note: If ADDRTR and/or RANSEQ are specified in the same DTF, ESETL should be issued before issuing a READ or WRITE. Another SETL can be issued to restart sequential retrieval.

ISAM LOAD: ENDFL Macro, Phase 1 - $$BENDFL, Charts DA-DB

Objective: To validate IOAREAL address limits and DTFIS table limits. To reset error indicators in DTF table. To pad key field and write partially filled block if present. To write EOF record. To write TI (track index) entries, CI (cylinder index entry, and MI (master index) entry if needed. To compute number of bytes used in the highest level index used. To write track index inactive entries if needed.

Entry: From the ENDFL macro expansion.

Exits: To the second phase of the ENDFL macro, $$BENDFF.

Method: This phase first validates the address limits of IOAREAL and the DTFIS table via an SVC 26. It then resets error indicators in the DTF table for prime data area full, duplicate record, and sequence error. It checks for a partially filled blocked record. If one is present, it pads the key field with all X'F's and writes the partially filled block.

A series of tests is made to determine the location of the last prime data record written. If the record was not the last record on the track, the last track full indicator is set off. If the record was the last record on the track, the address in the ID field of IOAREAI is modified. If space is available in the prime data area, the EOF record is written. If enough space is not available for the EOF record, this condition is posted at Filename.C in the DTF table.

A test is made to determine if the last prime data track was full. If not, this routine writes the track index normal entry and the track index overflow entry. It also writes the cylinder index normal entry, and (if the master index is being used) the master index normal entry. If the last prime data track was full, and the last track index record number was not 0, this routine writes a cylinder index normal entry and a master index normal entry, if the master index is being used.

If the last track index record number was zero, but the track was not 0, a cylinder index normal entry is written. If the last track index track was 0, and the cylinder index record is not the last record on the track, a master index normal entry is written (if the master index is being used). Otherwise, pointers are set to the lower limit of the highest index level being used.

The routine then computes the total number of bytes used in the normal entries of the highest level index being used.

When the total number of bytes in the highest level index has been determined, this phase formats the track index inactive entries and then tests to determine if there are more track index records on the cylinder. If so, track index inactive entries are written until there are no more records on the cylinder. A test is made for the device type. If the device is a 2311, this routine continues to write track index inactive entries until the end of the prime data extent is reached, keeping a count of the number of cylinders containing track index inactive entries. For a DASD other than 2311, this phase does not format any more track index inactive entries.

When there are no more track index inactive entries to be written, the address of the DTF is saved for the next phase, and this phase exits to phase $$BENDFF.

ISAM LOAD: ENDFL Macro, Phase 2 - $$BENDFF, Charts DC-DD

Objective: To write cylinder and master index inactive entries for any unused cylinders. To write cylinder index and master index dummy end entries. To write cylinder index and master index dummy chained entries.

Entry: From the first phase of the ENDFL macro, $$BENDFL.

Exits: To the problem program via an SVC 11.

Method: This transient routine first formats the cylinder index inactive entry. Using the count of the number of cylinders containing track index inactive entries, as determined by $$BENDFL, this routine writes cylinder index inactive entries for the unused cylinders. While the inactive entries are being written, a count is kept of the number of tracks containing cylinder index inactive entries. After the last cylinder index inactive entry is written, a cylinder index dummy end entry is written.

The routine then tests to determine if the master index is being used. If it is, master index inactive entries are written using the count of cylinder index tracks containing inactive entries. One master index inactive entry is written for each track of cylinder index inactive entries. After the last master index inactive entry is written, this routine writes a master index dummy end entry.

At the end of each master index or cylinder index cylinder, there is a master index or cylinder index dummy chained entry that points to the next master index or cylinder index cylinder. In other words, the last record on the last track of a cylinder of cylinder index records points to record 0 on track 0 of the following cylinder if it is also in the cylinder index extent. After the dummy end entries have been written, this routine writes dummy chained entries, if any, for the cylinder index, and for the master index, if it is being used.

When the dummy chained entries have been written, this phase exits to the problem program via an SVC 11.

ISAM LOAD: SETFL Macro, Phase 1 - $$BSETFL, Charts DF-DF

Objective: To validate DTFIS table limits and IOAREAL limits. To test for prime data on data cell and disk devices. To determine the number of cylinders in the prime data extent, the maximum number of cylinder index entries in the cylinder index extent, and to check if the cylinder index extent is too small. To check if the master index extent (if present) is too small. To build the basic CCW string for use by the LOAD module and ENDFL transients. To move last prime data, track index, cylinder index, and master index record addresses to the DTF table.

Entry: From the SETFL macro expansion.

Exits: To the $$BSETFF for normal exit. To problem program (via SVC 11) if cylinder index or master index extents are too small.

Method: This B-transient first validates the address limits of IOAREAL and the DTFIS table. It then tests whether the prime data is on a data cell or disk device, and moves the address limits to the prime data control field of the DTF table.

This phase then calculates the number of prime data extents minus one. The total number of cylinders minus one is then calculated.

This phase next calculates the number of active records in the cylinder index, and compares this number with the total number of prime data cylinders minus 1. If the number of cylinders is greater than or equal to the number of cylinder index records, the cylinder index extent is too small and flags are set to indicate this condition.

A test is then made to determine whether the master index is being used. If it is, the total number of cylinder index records referenced by the master index is used to determine the number of cylinder index cylinders referenced by the master index. One dummy record per cylinder index cylinder is subtracted from the total number of cylinder index records. The result of this subtraction (number of cylinder index records referenced by the master index) is compared to the total number of prime data cylinders minus one. If the number of prime data cylinders is greater than or equal to the number of cylinder index records, the master index is too small, and flags are set to indicate this condition. If the master index is not being used, this check is bypassed.

The phase then checks to determine if either the cylinder index or master index extent is too small. If so, this phase returns to the problem program via an SVC 11. If the extents are large enough, the record number of the track index dummy record is calculated, and the logical transient proceeds to build the CCW string shown in Figure 36.

When the CCW string has been completed, the seek/search address is set up. The lower limit address of the prime data area is moved to the seek/search address area, and a test is made to see if the file is to be extended. If so, the extension indicator in the DTF table is set on, and the address of the last prime data record is saved for $$BSETFF. LTIRA (last track index record address) is initialized to the address of the last track index overflow entry. A test is made to see whether the upper limit address of the prime data extent has been increased. If so, an indicator is set for $$BSETFF, and the old prime data upper limit is moved to the seek/search address area. The upper limit address is initialized with the new prime data upper limit.

A test is then made to see if the last prime data track is full. If it is not full, the last two track index entries must be rewritten during the load operation (since the highest key on the prime data track increases when new records are written). Therefore, the LTIRA is decreased to point to the track index entries for the previous track. The LCIRA (last cylinder index record number) and the LMIRA (last master index record number) are also decreased by one. (LMIRA is only decreased if the master index is being used.) $$BSETFF is then fetched.

If the last prime data track is full, and it is not the end of the cylinder, the LPDRA (last prime data record address) is increased to record zero on the next track.

CCW Built	Function
X'07', Address of Prime Data Lower Limit, Command Chaining, 6.	Long seek.
X'23', Address of Sector Argument, Command Chaining, 1.	Set Sector for beginning of the prime data area.
X'31', Address of Prime Data Lower Limit, Command Chaining, 5.	Search identifier equal for the beginning of the prime data area.
X'08', Pointer to *-8, Command Chaining, -.	TIC to *-8.
X'1D', Address of IOAREAL, Suppress Length Indicator, 16384.	Write count, key and data in prime data area If the verify option is specified, the command chaining bit is set on in the flag field.
X'22', Address of Sector Argument, Command Chaining, 1.	Read Sector for last record written.
If the verify option is specified, the following CCWs are built in addition to those above.	
X'23', Address of Sector Argument, Command Chaining, 1.	Set Sector for beginning of the prime data area.
X'31', Address of Prime Data Lower Limit, Command Chaining, 5.	Search identifier equal.
X'08', Pointer to *-8, Command Chaining, -.	TIC to *-8.
X'1E', 0, Suppress Length Indicator and Data Transfer, 1.	Read count, key and data to verify prime data record written. No data is transferred to main storage.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 36. CCW Chain Built by $$BSETFL to Write Prime Data Records

A test is made to see whether the last track index entry address is the last record on the track index track. If so, the ITIRA is set to record zero on the next track. Then the LCIRA and the LMIRA are decreased by one, and $$BSETFF is fetched.

If the last prime data track is full, and the cylinder is full, the LPDRA and the LTIRA are updated to the start of the next cylinder of the prime data extent. Since the last prime data cylinder is full, the last LCIRA is nct changed. If the last cylinder index track is not full, the LMIRA is decreased by one. If the last cylinder index track is full, LMIRA is not changed.

When all processing has been completed, this phase exits to phase $$BSETFF to initialize the CCW chain and I/O areas.

ISAM LOAD: SETFL Macro, Phase 2 - $$BSETFF, Chart DG

Objective: To initialize the CCW chain and I/O areas required to write the last track index track in each cylinder of the prime data area and the COCR (cylinder overflow control record), if the cylinder overflow option has been specified in the DTFIS table.

Entry: From the first phase of the SETFL macro, $$BSETFL.

Exit: To the third phase of the SETFL macro, $$BSETFG.

Method: This phase first gets the key length from the DTFIS table and stores it in the count fields associated with the write count, key and data CCWs . These CCWs are built to write up to 41 track index records with one EXCP.

A track index dummy record is built in the user's IOAREAL and the number of cylinder overflow tracks in each cylinder is calculated and saved in the COCR data field. This phase then determines the correct Write Count, Key and Data CCW to write the track index dummy record. When the correct CCW is found, the address of IOAREAL is moved to its data address field.

A test is made to determine if the

cylinder overflow option has been specified. If it has, Record Zero (R0) in the track index contains a COCR. The COCR is found in the data area of R0. The COCR contains the address of the last overflow record on the cylinder and the number of tracks remaining in the cylinder overflow area. This phase initializes the COCR with the address of the first cylinder overflow track and the number of cylinder overflow tracks.

It tests to determine if the track index records to be formatted are on track 0 of each cylinder. If so, the seek command code is changed to a NO-OP, the flag bits are set to indicate command chaining, the file protect indicator is reset and $$BSETFG is fetched for execution. All writing is done on track 0 with one EXCP. If the track index records are not on track 0, a test for DASD file protect is made. If the DASD file protect feature is not present, flag bits of the seek CCW are set to indicate command chaining, the file protect indicator is reset and $$BSETFG is fetched for execution. All writing is done with one EXCP. If the file protect feature is present, the file protect indicator is left on and $$BSETFG is fetched. All writing is done with two EXCPs. If the cylinder overflow option has not been specified, the file protect indicator is reset. All writing will be done with one EXCP.

ISAM LOAD: SETFL Macro, Phase 3 - $$BSETFG, Chart DH

Objective: To format last track index track in each prime data cylinder. To write COCR data if the cylinder overflow option has been specified.

Entry: From the second phase of the SETFL macro, $$BSETFF.

Exit: To problem program via an SVC 11 or to phase 4 of the SETFL macro, $$BSETFH, via an SVC 2, or to phase 3A of the SETFL macro, $$BSETFI, if RPS is supported.

Method: This logical transient phase first checks to determine whether the file has been extended, but the prime data area upper limit has remained the same. If so, no formatting is required, and a branch is taken to read the last prime data record. If the file is being created or being extended with increased upper limit, this phase formats the last track index track in each prime data cylinder (1 track per EXCP through use of an extended CCW chain), and writes the COCR data for each cylinder if the cylinder overflow option is specified.

ISAM LOAD: SETFL Macro, Phase 3A - $$BSETFI, Chart DK

Objective: To build an RPS CCW chain for the load function if RPS is supported, and to initialize the remaining DTF fields.

Entry: From the third phase of the SETFL macro, $$BSETFG.

Exit: To the problem program via an SVC 11 or to phase 4 of the SETFL macro, $$BSETFH, via an SVC 2.

Method: If the DTF is an RPS type DTF, the Load CCW chain is built in the DTF extension with embedded RPS CCWs. Then this phase tests to determine whether the file is being extended or created. If the file is being created, the seek/search address is set up, the count field of the IOAREAL is initialized with the address, key length and data length of the last prime data record. The CCB is initialized with the address of the CCW chain.

If the file is being created, the block position address is set with the current logical record address. The logical record counter is saved in the DTF table, the CCB is initialized with the address of the CCW chain, and this phase exits to the problem program via an SVC 11.

If the file is being extended, $$BSETFH is fetched via an SVC 2.

ISAM LOAD: SETFL Macro, Phase 4 - $$BSETFH, Chart DJ

Objective: For extension of file, to read the last prime data record so that keys may be compared by the ISMOD macro.

Entry: From the third phase of the SETFL macro, $$BSETFG.

Exit: To problem via an SVC 11.

Method: For extension of file, this phase reads the last prime data record (the address was saved by the first phase, $$BSETFL). This provides keys for a comparison in the load operation. If the records are blocked and the last block was not filled by a previous load operation, this phase finds the padded record and sets the block position address to load the next prime data record at the location of the padded record. If the records are blocked and the last block is full, this phase reads the last cylinder index entry to obtain the highest key of the load file and sets the block position address to load the next prime data record at the location of the data in IOAREAL.

ISAM LCAD: WRITE Macro, NEWKEY, Charts DL-DP

Objective: To ensure that keys are in ascending sequence. To write prime data record in correct location. To write track index entries, cylinder index entry and master index entry, if necessary.

Entry: From the WRITE, NEWKEY macro expansion.

Exit: To problem program via return register 14.

Method: This routine first tests switches in the DTF table to determine if the prime data area is full or if the cylinder/master index is too small. If either condition exists, this routine exits to the problem program via linkage register 14. A test is then made to determine if IOAREA2=YES is specified as an ISMOD macro parameter option. If IOAREA2 is specified and the presence of two I/O areas is indicated in the DTF table to allow overlapping of I/O with processing while creating the file, this routine gets the addresses of IOAREAL and IOAREA2 and determines if the ENDFL macro was issued. If it has been issued, a wait for I/O completion and a test for ERREXT=YES are made. If ERREXT is specified, additional error conditions can be returned to the problem program, thus giving the user greater flexibility in attempting to continue processing.

If IOAREA2 is not specified or if the ENDFL macro was not issued, a test is made to determine if the current record is the first record in the file. If it is the first record in the file, a test for IOAREA2=YES is made. If IOAREA2 is specified and there are two I/O areas, the traffic bit in the CCB is turned on and the IOAREA2 address constant is relocated. If IOAREA2 is not specified or if the current record is not the first record in the file, the current key is moved to the I/O area and a test is made to determine if the previous key is lower than the current key. If the previous key is not lower, a test for duplicate keys is made. If the keys are equal, a duplicate record indicator is set at Filename.C. If the current key is lower than the previous key, an out-of-sequence indicator is set at Filename.C. Control then returns to the problem program.

If the previous key is lower than the current key, the current key and data are moved to the I/O area, the prime data record count and logical record count are updated by 1 and a test is made to determine if this is the first logical record in the block. If it is the first logical record, the record number in the count field of the I/O area is updated. If IOAREA2 is specified, a test for a full block is made. If the block is not full, the logical record count and the block position address are saved and control returns to the problem program.

If the block is full, the logical record count is reset to 0 and a test for IOAREA2=YES is made. If IOAREA2 is specified, the addresses of the two I/O areas are interchanged and saved in the DTF table. The I/O area data address is saved as the block position address, and a test for ERREXT=YES is made. If ERREXT is specified, the record type in the parameter list is set to indicate data. A test for IOAREA2=YES is made. If IOAREA2 is specified and there are two I/O areas present, the prime data record ID is updated again and a prime data record from the second I/O area is written. If IOAREA2 is not specified, a prime data record from IOAREAL is written.

Note: The preceding process works for both blocked and unblocked records.

A check is then made to determine if the data record was written on a shared track. If it was written on a shared track, and it was not the last record of a shared track, control returns to the problem program. If the record was the last on a shared track, a test for IOAREA2=YES is made. If IOAREA2 is specified, a wait for I/O completion is made. The track index normal entry is then initialized to indicate a shared track index entry.

If the record was not written on a shared track and it was not the last record on the prime data track, control returns to the problem program. If the end of the prime data has been reached, a test for IOAREA2=YES is made. If IOAREA2 is specified, a wait for I/O completion is made and a test for ERREXT=YES is made. If ERREXT is specified, the record type in the parameter list is set to indicate track index. The track index normal and overflow entries are written and the last track index record address is saved.

Tests are then made for end of cylinder. If the last prime data record written was not the next-to-last or last record on the cylinder, the current prime data track number is updated by 1 and control returns to the problem program. If the record was the next to last record in the cylinder, a test is made for the end of the prime data extent. If the end of the extent has not been reached, the prime data track number is increased and control returns to the problem program. If it is the end of the extent, the end-of-extent indicators in the DTF table are set and control returns to the problem program.

If the record was the last prime data record on the cylinder, a test for ERREXT=YES is made. If ERREXT is specified, the record type in the parameter list is set to indicate cylinder index. A cylinder index entry is then written and

the last cylinder record address is updated. If a master index is being used, a master index entry is written if:

1. The cylinder index entry is the next to last track in the cylinder.

2. The cylinder index is the last record on the track. Before the master index entry is written, a test for ERREXT=YES is made. If ERREXT is specified, the record type in the parameter list is set to indicate master index.

Next, this routine tests for the end of the prime data volume. If end of volume has been reached, the extent sequence number is updated by 1 and the seek/search address is modified to the beginning address of the volume. If it is not the end of volume, the address in the seek/search area is updated, and the last track index record address and the address in the count field of the I/O area are modified. (This modification also occurs for end of prime data volume.) This routine then exits to the problem program via return register 14.

ISAM ADD: WAITF Macro, Charts EA-ED

Objective: To add a record to an indexed sequential file, adjusting the indexes and other records as necessary.

Entry: From the WAITF macro expansion.

Exit: To the prcblem program via linkage register 14.

Method: This routine first tests for ERREXT=YES. If ERREXT is specified, additional error conditions can be returned to the problem program, thus giving the user greater flexibility in attempting to continue processing. After waiting for the completion of the I/O operation, this routine determines the type of add function to be performed. The three types of add functions are:

- Normal add to the prime data area
- Add to the overflow area
- EOF add.

Normal Add to the Prime Data Area: If a normal add to the prime data area is required, this rcutine determines if the record is to be added to the last prime data track. If it is and the last prime data track is full, the overflow record address is calculated, and EXCP is issued to search and read the prime data track to determine the pcint of insertion and a wait for I/O completion is made. Figures 37-58 give a description of the channel program builder for the ADD function. If the addition is not on the last prime data track, the overflow record address is calculated and the prime data track is searched to determine the point of insertion for the record to be added to the file. When an equal/high key is found during the search, the count and data fields of that location are read into a save area in the DTF table and IOAREAL respectively.

A test is made to determine if the prime data in core option has been specified as an ISMOD macro parameter. If it has been specified, as many records as can fit into the I/O area specified in the DTFIS operand IOAREAL are read from the prime data track into main storage. The key of the record to be added is compared to the keys of the existing records in the I/O area. If a duplicate key is found, the condition is indicated to the user in the DTF table entry labeled Filename.C. If no duplicate key is found, the records are shifted in main storage leaving the record with the highest key remaining in the user's work area, WORKL. The other records are rewritten directly onto the track. Any remaining records on the track are then read into the I/O area. The process continues until the last record on the track is set up as an overflow record. When the last prime data record on the track has been rewritten, the new overflow record is written in the overflow area, the track index normal and overflow entries and the COCR are written and control returns to the problem program.

If the prime data in core option has not been specified as an ISMOD macro parameter, a test for blocked records is made. If the file is unblocked, the record previously found on the search key equal/high is reread to get the key field. If it is a duplicate key, a switch is set on in the DTFIS table indicating a duplicate key has been sensed, and a return to the problem program is made. If there are no duplicate keys, the user's key and data are written from the work area, WORKL, onto the DASD file. The record in the I/O area, IOAREAL, replaces the user's record in the work area. The next record on the track replaces the one in the I/O area. This process is repeated until the end of track is reached.

If the end-of-file (EOF) record is read during the process of shifting the records over one record position, this routine writes the last record over the EOF record, and then writes a new EOF record (see Figures 39, 47, 48).

If the file contains blocked records, this routine reads the block of records (or as many as fit in the I/O area if IOAREAL has been increased for reading and writing more than one record at a time) into IOAREAL. The key field within each logical record is analyzed to determine the correct position in which to insert the new record. If there is duplication of keys, a switch is set on in the DTFIS table and control

returns to the problem program.

If the key of the record to be inserted (contained in WORKL) is low, it is exchanged with the record presently in the block. This prccedure continues with each succeeding record in the block until the last record is moved into the work area. The key field of the DASD record is then updated to reflect the highest key in the block. If the size of IOAREAL has been increased, succeeding blocks in the I/O area are also updated. The block (or blocks) is then written back onto DASD. The remaining blccks on the track are similarly prccessed until the last logical record on the track is moved into WORKL. This record is then set up as an overflow record with the correct sequence-link field added and written in the overflow area. The sequence-link field for the new overflow record is taken from the track index overflow entry. The indexes are updated and control returns to the problem program for the next record to be added. If the overflow area is full, this information is indicated to the user in the DTF table entry labeled Filename.C.

The track index normal entry key field is updated to the key of the new last record, the track index overflow entry data field is updated to the address of the new overflow entry (that entry has the lowest key for the overflcw for that track) and the COCR is updated. These records are written on the DASD file before control returns to the problem program.

If the last block in the prime data area is padded, the last record to be shifted is included in that block. If the EOF record is read during the process of shifting the records one reccrd positicn, the last record is written as a new block and a new EOF record is written before returning control to the problem prcgram.

Add to the Overflow Area: This routine computes the new overflow record address and reads the overflow chain to get the address of the record with the next highest key. This address is stored in the sequence-link field of the new record. The new overflow record is then written in either the cylinder overflow area or independent overflow area. If these areas are full, this condition is indicated to the user in the DTFIS table entry labeled Filename.C. Each time an overflow record is added to the independent overflow area, an EOF record is written to maintain the integrity of the indexed sequential file (see Figure 49). The next overflow record followed by an EOF record overlays the previous EOF record.

If the new overflow record has the lowest key in the overflow chain, its address is used to build a new track index overflow entry. The new overflow entry is then written on the DASD file (see Figure 46) and control returns to the problem program. If a cylinder overflow condition occurs, the updated COCR (cylinder overflow control record) is written on DASD before control is returned to the problem program (see Figure 43).

If the new overflow record does not have the lowest key, the sequence-link field of the record with the next lower key is updated to contain the address of the new overflow record. This overflow record is then rewritten on DASD and the COCR is updated. Control returns to the problem program.

EOF Add: This routine first determines if the last prime data track is full. If the last prime data track is not full, the new record is inserted on it. If the file is blocked, the block is read and the new record is inserted.

If the file is not blocked or if it is blocked and the last block is full, a new last prime data record address is stored and the new record is written at that address. A new EOF record is then written (see Figure 39).

If the last prime data track is full, the new record is inserted in the overflow area. The new overflow record address is computed and the record is written in the overflow area.

If an overflow chain is present, the next lower record in the chain is found and the address of the new record is moved to the sequence-link field of the next lower record.

If no overflow chain is present, the address of the new overflow record is moved to the track index overflow entry. The track index overflow entry is then written with the new high key. The master index (if present) and the cylinder index are updated with the new high key. A test for the cylinder index in core option is then made. If it has not been specified, control is returned to the problem program. If the cylinder index in core option has been specified, the new key is inserted into the appropriate index in core entry before returning control to the problem program.

ISAM ADD: WRITE Macro, NEWKEY, Charts EE-EF

Objective: To perform the necessary initialization to add a record to a file.

Entry: From the WRITE, NEWKEY macro expansion.

Exit: To the problem program via linkage register 14.

Method: After initializing the pointers to the three parts of the DTFIS table, this routine gets the starting address of the highest level index, builds a CCW chain to search the highest level index (see Figure 37) executes the channel program and tests for ERREXT=YES. If ERREXT is specified, additional error conditions can be returned to the problem program, thus giving the user greater flexibility in attempting to continue processing. The channel program is executed and a wait for I/O completion is made. The routine then tests the F code of the index level pointer to determine if the next search is of the cylinder or track index. The F code refers to the index level just searched. If it was the master index, the next search is on the cylinder index. See Figure 20 for a description of the F code.

If the F code indicates a dummy chained entry, the search of the master, cylinder or track index continues. If the index level pointer did not indicate a dummy chained entry, a test for an inactive or dummy end entry is made. If an inactive or dummy end entry is indicated, the EOF add indicator is set on in the DTFIS table, a CCW chain is built to read the last track index entries (see Figure 50), the channel program to bypass the last of the track index entries is executed, a wait for the I/O operation to be completed is made, and control returns to the problem program. Processing continues with the record following the last key.

If an inactive or dummy end entry is not indicated, a test for the presence of a master index is made. If the master index is not present, indicating the cylinder index was just searched, a search of the track index is performed, and a return to the problem program is made.

If the master index is present, a test is made to determine if the cylinder index in-core option was specified as an ISMOD macro parameter. If it was not specified, an EXCP is issued to search the cylinder index, followed by a wait for I/O completion, an EXCP to search the track index, a wait for I/O completion, and a return to the problem program. If the cylinder index in-core option was specified, a search of the track index is performed, and a return to the problem program is made. When HOLD=YES is specified in the DTF, any held data tracks and index tracks are freed before control returns to the problem program.

CCW Builder Control Code[1]	CCW Built	Function
7961	X'69', &KEYARG, CC and SLI, Key Length	Search key equal or high the master/ cylinder index. Key supplied by user in the DTFIS table.
0C6B	X'08', Pointer to *+16, CC and SLI, 5	TIC to *+16.
D17B	X'1A', [2]&Filename.D+8, CC, SLI and SKIP, 5	Read home address into work area for the current track index normal entry count field in the DTFIS table.
516C	X'92', [2]&Filename.D+8, CC and SLI, 10	Read count (multiple track) into work area for the current track index normal entry count field in the DTFIS table.
7961	X'69', &KEYARG, CC and SII, Key Length	Search key equal or high the master/ cylinder index. Key supplied by user in the DTFIS table.
046B	X'08', Pointer to *-16, CC and SLI, 5	TIC to *-16.
150C	X'06', [2]&Filename.D+40, 00, 10	Read data (next 10-byte index level pointer) into work area for track index normal entry data field in DTFIS table.

[1-2] See notes 1 and 2 in Figure 58.

Figure 37. Channel program builder for ADD -- CCW chain built to search master cylinder index.

CCW Builder Contrcl Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the track index using the pointer (CCHHR) in the common seek/search area.
066B	X'08', Pointer to *-8,CC and SLI, 5	TIC to *-8.
106C	X'06', [2]&Filename.D, CC and SLI, 10	Read data (COCR record) into the cylinder overflow control record (COCR) area.
516C	X'92', [2]&Filename.D+8, CC and SLI, 10	Read count (multiple track) into work area for the current track index normal entry count field in the DTFIS table.
7941	X'69', &KEYARG, CC, Key Length	Search key equal or high the track index. Key supplied by user in the DTFIS table.
046B	X'08', Pointer to *-16, CC and SLI, 5	TIC to *-16.
156C	X'06', [2]&Filename.D+40, CC and SLI, 10	Read data (next 10-byte pointer to prime data record) into work area for track index normal entry data field in DTFIS table.
526C	X'92', [2]&Filename.D+16, CC and SLI, 10	Read count (multiple track) into work area for current track index overflow entry count field in DTFIS table.
1D0C	X'06', [2]&Filename.W, 00 10	Read data (10-byte overflow entry) into random/sequential retrieval work area.

[1-2] See notes 1 and 2 in Figure 58.

Figure 38. Channel program builder for ADD -- CCW chain built to search track index.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for the last prime data record address using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
342C	X'10', [2]&Filename.D+32, SLI, 10	Write count, key and data of EOF record located in current overflow record count field in DTFIS table.

[1-2] See notes 1 and 2 in Figure 58.

Figure 39. Channel program builder for ADD -- CCW chain built to write new EOF record.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the prime data track using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
436C	X'12', [2]&Filename.D+24, CC and SLI, 10	Read count field for current prime data record.
7941	X'69', &KEYARG, CC, Key Length	Search key equal or high the prime data track. Key supplied by user in DTFIS table.
046B	X'08', Pointer to *-16, CC and SLI, 5	TIC to *-16.
1B02	X'06', Address of IOAREAL+8 +KEYLEN, 00, Block Size	Read data (prime data block) into IOAREAL+8+Key Length.

[1-2] See notes 1 and 2 in Figure 58.

Figure 40. Charnel program builder for ADD -- CCW chain built to find prime data record.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the track index using pointer, CCHHR, in common seek/search address area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
B06C	X'05', [2]&Filename.D, CC and SLI, 10	Rewrite COCR located in cylinder overflow control record work area in DTFIS table.
E14B	X'B1', [2]&Filename.D+8, CC, 5	Search identifier equal (multiple track) for the pointer, CCHHR, in the normal entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
2A45	X'0D', Address of IOAREAL+8, CC, Key Length + 10	Rewrite track index normal entry located at IOAREAL+8.
E24B	X'B1', [2]&Filename.D+16, CC, 5	Search identifier equal (multiple track) for the pointer, CCHHR, in the overflow entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
BDCC	X'05', [2]&Filename.W, CC and DC, 10	Rewrite overflow entry located in random/sequential retrieval work area.
824B	X'31', [2]&Filename.D+16, CC, 5	Search identifier equal for the pointer, CCHHR, in the overflow entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
1D3C	X'06', [2]&Filename.W, SLI and SKIP, 10	Read data to verify record just written. Information is not transferred to main storage.

[1-2] See notes 1 and 2 in Figure 58.

Figure 41. Channel program builder for ADD -- CCW chain built to rewrite track index entry.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'13', [2]&Filename.S+3, CC, 5	Search identifier equal for R0 using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
B06C	X'05', [2]&Filename.D, CC and SLI, 10	Write data (updated COCR) from the cylinder overflow control record (COCR) area in the DTFIS table.
E14B	X'B1', [2]&Filename.D+8, CC, 5	Search identifier equal (multiple track) the track index using the pointer, CCHHR, in the work area for the current track index normal entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
BDCC	X'05', [2]&Filename.W, CC and DC, 10	Write data (track index overflow entry) from the random/sequential retrieval work area.
814B	X'31', [2]&Filename.D+8, CC, 5	Search identifier equal the track index using the pointer, CCHHR, in the work area for the current track index normal entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
1D3C	X'06', [2]&Filename.W, SLI and SKIP, 10	Read data to verify record just written. Information is not transferred to main storage.

[1-2] See notes 1 and 2 in Figure 58.

Figure 42. Channel program builder for ADD -- CCW chain built to write track index entry.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for R0 using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
B02C	X'05', [2]&Filename.D, SLI, 10	Write data (updated COCR) from the cylinder overflow control record area in the DTFIS table.

[1-2] See notes 1 and 2 in Figure 58.

Figure 43. Channel program builder for ADD -- CCW chain built to write COCR.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
AA07	X'0E', Address of IOAREAL+8, 00, Key Length + Record Length + 10	Read key and data of previously low overflow record into IOAREAL+8.

[1-2] See notes 1 and 2 in Figure 58.

Figure 44. Channel program builder for ADD -- CCW chain built to read previous overflow reccrd.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
2A47	X'0D', Address of IOAREAL+8, CC, Key Length + Record Length + 10	Write key and data of previously low overflow record located at IOAREAL+8.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
AA37	X'0E', Address of IOAREAL+8, SLI and SKIP, Key Length + Record Length + 10	Read key and data to verify record just written. Information is not transferred to main storage.

[1-2] See notes 1 and 2 in Figure 58.

Figure 45. Channel program builder for ADD -- CCW chain built to write previous overflow record.

CCW Builder Contrcl Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for last overflow record address using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
37C9	X'1D', Address of IOAREAL, CC and DC, Key Length + Record Length + 18	Write count, key and data of new overflow record located at IOAREAL.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for last overflow record address using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08' Pointer to *-8, CC and SLI, 5	TIC to *-8.
C739	X'1E', Address of IOAREAL, SLI and SKIP, Key Length + Record Length + 18	Read count, key and data to verify record record just written. Information is not transferred to main storage.

[1-2] See notes 1 and 2 in Figure 58.

Figure 46. Channel program builder for ADD -- CCW chain built to write new overflow record.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for present EOF record address minus 1 using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
37C8	X'1D', Address of IOAREAL, CC and DC, Key Length + Block Size + 8	Write count key and data of new record to be added located at IOAREAL.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for present EOF record address minus 1 using pointer CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
C738	X'1E', Address of IOAREAL, SLI and SKIP, Key Length + Block Size + 8	Read count, key and data to verify record just written. Information is not transferred to main storage.

[1-2] See notes 1 and 2 in Figure 58.

Figure 47. Channel program builder for ADD -- CCW chain built to write over EOF record (blocked records).

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for present EOF record address minus 1 using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
37C8	X'1D', Address of IOAREAL, CC and DC, Key Length + Block Size + 8	Write count, key and data of new record to be added, located at IOAREAL.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for present EOF record address minus 1 using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
C738	X'1E', Address of IOAREAL, SLI and SKIP, Key Length + Block Size + 8	Read count, key and data to verify record just written. Information is not transferred to main storage.

[1-2] See notes 1 and 2 in Figure 58.

Figure 48. Channel program builder for ADD -- CCW chain built to write over EOF record (unblocked records).

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for present EOF record address minus one using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
37AC	X'1D', Address of IOAREAL, DC and SLI, 10	Write count, key and data of EOF record located at IOAREAL.

[1-2] See notes 1 and 2 in Figure 58.

Figure 49. Channel program builder for ADD -- CCW chain built to write EOF in independent overflow area.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the track index using the pointer (CCHHR) in the common seek/search area.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
106C	X'06', [2]&Filename.D, CC and SLI, 10	Read data (COCR record) into the cylinder overflow control record (COCR) area.
E14B	X'B1', [2]&Filename.D+8, CC, 5	Search identifier equal (multiple track) the track index for the last normal entry using information in the work area for the current track index normal entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
154C	C'06', [2]&Filename.D+40, CC, 10	Read data (last track index normal entry) into work area for track index normal entry data field.
526C	X'92', [2]&Filename.D+16, CC and SLI, 10	Read count (multiple-track) of last track index overflow entry into work area for the current track index overflow entry count field.
1D0C	X'06', Filename.W, 00, 10	Read data (last track index overflow entry) into random/sequential retrieval work area.

[1-2] See notes 1 and 2 in Figure 58.

Figure 50. Channel program builder for ADD -- CCW chain built to read last track index entry.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the overflow chain using the pointer (CCHHR) in the common seek/search area.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
AA07	X'0E', Address of IOAREAL+8, 00, Key Length + Record Length + 10	Read key and data of overflow record into IOAREAL+8.

[1-2] See notes 1 and 2 in Figure 58.

Figure 51. Channel program builder for ADD -- CCW chain built to read overflow record.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for last prime data record address using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
1B02	X'06', Address of IOAREAL+ 8+KEYLEN, 00, Block Size	Read block into IOAREAL + 8 + KEYLEN.

[1-2] See notes 1 and 2 in Figure 58.

Figure 52. Channel program builder for ADD -- CCW chain built to read last prime data record.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 3	Search identifier equal for last prime data record address using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
2AC6	X'0D', Address of IOAREAL+8, CC and DC, Key Length + Block Size	Write key and data of prime data block located at IOAREAL+8.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for last prime data record address using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
AA36	X'0E', Address of IOAREAL+8, SLI and SKIP, Key Length + Block Size	Read key and data to verify record just written. Information is not transferred to main storage.

[1-2] See notes 1 and 2 in Figure 58.

Figure 53. Channel program builder for ADD -- CCW chain built to write block of prime data records and verify.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for last track index address using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
2A45	X'0D', Address of IOAREAL+8, CC, Key Length + 10	Write key and data of track index normal entry located at IOAREAL+8.
E24B	X'B1', [2]&Filename.D+16, CC, 5	Search identifier equal (multiple track) the track index for the last over flow entry using the count for the current track index overflow entry.
066B	C'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
2845	X'0D', Address of WORKL, CC, Key Length + 10	Write key and data of track index overflow entry located at WORKL.
824B	X'31', [2]&Filename.D+16, CC, 5	Search identifier equal the track index for the last overflow entry using the count for the current track index overflow entry.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
A835	X'0E', Address of WORKL, SLI and SKIP, Key Length + 10	Read key and data to verify record just written. Information is not transferred to main storage.

[1-2] See notes 1 and 2 in Figure 58.

Figure 54. Channel program builder for ADD -- CCW chain built to write track index entry.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the master/ cylinder index using the pointer, CCHHR, in the common seek/search area in the DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
150C	X'06', [2]&Filename.D+40, 00, 10	Read data (index entry) into work area for track index normal entry data field.

[1-2] See notes 1 and 2 in Figure 58.

Figure 55. Channel program builder for ADD -- CCW chain built to read index entry.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the master/cylinder index using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
2A45	X'0D', Address of IOAREAL+8, CC, Key Length + 10	Write key and data of master/cylinder index entry located at IOAREAL+8.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the master/cylinder index using pointer, CCHHR, in common seek/search area in DTFIS table.
AA35	X'0E', Address of IOAREAL+8, SLI and SKIP, Key Length + 10	Read key and data to verify record just written. No information is transferred to main storage.

[1-2] See notes 1 and 2 in Figure 58.

Figure 56. Channel program builder for ADD -- CCW chain built to write index entry.

CCW Builder Control Code[1]	CCW Built	Function
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the track index using the pointer, CCHHR, in the common seek/search area in the DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
B06C	X'05', [2]&Filename.D, CC and SLI, 10	Write data (COCR) from the cylinder overflow control record work area in DTFIS table.
E24B	X'B1', [2]&Filename.D+16, CC, 5	Search identifier equal (multiple-track) the track index using the pointer, CCHHR, in the work area for current track index overflow entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
AA35	X'0E', Address of IOAREAL+8, SLI and SKIP, Key Length + 10	Read key and data to verify record just written. Information is not transferred to main storage.

[1-2] See notes 1 and 2 in Figure 58.

Figure 57. Channel program builder for ADD -- CCW chain built to write track index overflow entry.

Note 1:

The first character of the control code references an operation code at IJHCSTRI.
The second character of the control code references a data area at IJHAHRAA.
The third character of the control code references the following information:

Control Character	CCW Flag Field	Meaning
0	X'00'	End of CCW Chain
2	X'20'	SLI (Suppress Length Indicator)
3	X'30'	SLI and SKIP (Suppress Data Transfer)
4	X'40'	CC (Command Chaining)
6	X'60'	CC and SLI
7	X'70'	CC, SLI, and SKIP
A	X'A0'	SLI and DC (Data Chaining)
C	X'C0'	CC and DC

The fourth character of the control code references a byte count (length) field at IJHCRESZ.

Note 2:

&Filename = DTF name supplied by user.
&Filename.X = X is suffix supplied by DTFIS for unique DTF labels.

Figure 58. Channel program builder for ADD -- notes.

$$BINDEX Read Cylinder Index Into Storage, Charts FA-FB

Objective: To read all or part of the cylinder index into main storage.

Entry: From the indexed-sequential logic module (ISMOD).

Exit: To the prcblem program.

Method: This phase determines the number of cylinder index entries that can be read into main storage at one time. Each cylinder index entry consists of a key area and a data area. The key area contains the highest key asscciated with the cylinder, and its length is the same as that specified for lcgical data records in the DTFIS entry KEYLEN. The data area is ten bytes lcng and contains the pointer to the track index for that cylinder. See Figure 20 for the format of this 10-byte pointer. When this phase reads the cylinder index entry into main storage, only six bytes of the 10-byte pointer are retained. The last four bytes of the pointer to the track index are the same for all entries in the cylinder index. Therefore, only the first six bytes of the pointer are required for processing.

If it is the first time through this B-transient phase, the key of the first core index entry is set to 0. If it is not the first time through this phase, the key of highest entry minus 1 that was previously read into main storage is moved to the key area of the first core index entry. A test is made to determine if the index skip opticn was specified in the DTF entry. If the index skip option was specified, any cylinder index entries preceding the one needed to process a given key are not read into main storage. In order to skip the cylinder index entries preceding the one needed to process a given key, a CCB to read the cylinder index is built along with a string of CCWs. Figure 59 gives a description of the CCW string.

This transient then executes the channel program and determines if the address of the first cylinder index entry read is the address of the required entry (SKEH, TIC, NO-OP). If it is, there are no cylinder index entries to be skipped and the cylinder index is then read into main storage from that point. If the addresses are not the same (RDID, SKEH, TIC, RD), a check is made to determine if this is a dummy chained entry.

If it is a dummy chained entry (indicating the end of the cylinder), its address points to the first track of the next cylinder containing the cylinder index. This phase subtracts 1 from the record number of the dummy chained entry to get the preceding cylinder index entry, moves the chain address to the next cylinder index entry to be read (in the DTFIS table), and reads the cylinder index into main storage starting with the entry preceding the dummy chained entry.

If it is not a dummy chained entry, a test is made to determine if the required entry is the first record on the first track of the cylinder. If it is, this phase sets up to read the cylinder index into main storage starting with entry preceding the dummy chained entry for the previous cylinder. If the required entry

is not the first record on the first track of the cylinder, a test is made to determine if the record number of the cylinder index entry is 1.

If it is 1, the track number is decreased by 1 and the record number is updated to the maximum record number for the cylinder index track. The cylinder index is read into main storage starting with the last record on the preceding track. If the required cylinder index record number is greater than 1, the record number is decreased by 1, and the cylinder index is read into main storage starting with the preceding entry on the track. Each time a cylinder index entry is read, the number of available index entries in main storage is decreased by 1.

If the index skip option was not specified in the DTF, this phase decreases the number available core index entries by 2. These two core index entries contain dummy entries. The first dummy entry at the beginning of the cylinder index storage area contains either a key of all zeros (if this is first time the cylinder index has been read into main storage) or it contains the key of the last cylinder index entry read into main storage. The second dummy entry is located at the end of cylinder index storage area and has a key of all X'F's.

Before a part or all of the cylinder index is read into main storage, a test is made to determine how many cylinder index records can fit in the area available. A CCB and a CCW chain are built to perform the actual read operation. Figure 59 gives a description of the CCW chain. The channel program is executed and the number of core index entries is decreased by the number of records read. The cylinder index is read into main storage until either the end of the cylinder index is reached or there are no more core index entry positions.

CCW Built	Function
X'07', Address of cylinder index entry, Command Chaining, 6	Long seek.
X'31', Address of cylinder index entry, Ccmmand Chaining, 6	Search identifier equal (SIDE) the cylinder index.
X'08', Pointer to *-8, -, -.	TIC to *-8.
X'69', &KEYARG, Command Chaining and Suppress Length Indicator, KEYLEN (Key Length).	Search key equal or high (SKEH) the cylinder index. Key supplied by user in the DTF table.
X'08', Pointer to *+16, -, -.	TIC to *+16.
X'03', -, Suppress Length Indicator, 1.	NO-OP.
X'92', IDOFHIT, Command Chaining, 8.	Read count (multiple-track) (RIDM) into IDOFHIT, 8-byte area for record found on SIDE.
X'69' &KEYARG, Command Chaining and Suppress Length Indicator, KEYLEN (Key Length).	Search key equal or high (SKEH) the cylinder index.
X'08', Pointer to *-16, -, -.	TIC to *-16.
X'06', POINTER, End of Chain, 10.	Read data portion of cylinder index entry (RD) into POINTER, 10-byte area for pointer to track index.

Figure 59. CCW chain built by $$BINDEX to skip cylinder index entries preceding the one tc process a given key.

CCW Built	Function
X'07', Address of cylinder index entry, Command Chaining, 6.	Long Seek.
X'31', Address of cylinder index entry, Command Chaining, 6.	Search identifier equal (SIDE) the cylinder index.
X'08', Pointer to *-8, -, -.	TIC to *-8.
X'0E', Address of cylinder index entry in main storage (multiple of key length + 6), Command Chaining, Key Length + 10.	Read key and data (RKD) of cylinder index entry into storage.

Figure 60. CCW chain built by $$BINDEX to read the cylinder index into storage.

ISAM RETRVE, RANDOM: READ Macro, KEY, Chart FC

Objective: To perform the random retrieval function for an indexed sequential file by searching the indexes to determine the track on which the desired record is stored.

Entry: From the READ, KEY macro expansion.

Exit: To the problem program via linkage register 14.

Method: This routine first initializes pointers and status bits in the DTFIS table. It then constructs the CCW chain to search the master or cylinder index (see Figure 61). It determines the highest level index (master or cylinder) being used, and tests for ERREXT=YES. If ERREXT is specified, additional error conditions can be returned to the problem program, thus giving the user greater flexibility in attempting to continue processing. This routine then searches the highest level index to get the address of the next index to be searched.

A test of the F code from the index level pointer is then made to determine if the next search is of the track index (see Figure 20). The F code refers to the index level just searched. If the master index was just searched, the next search is on the cylinder index. If the next search is not on the track index, the routine gets the index entry type and determines the routine to process that type.

If the entry type is a normal entry, the routine returns to search the next index. If the entry type is a dummy end entry or an inactive entry, the routine branches to an error routine to set a no-record-found flag in the DTF table and to return to the problem program. If the entry is a dummy chained entry, the routine returns to search the index using the address supplied by the 10-byte index level pointer.

When the next search is found to be on the track index, a test is made to determine if the track index takes up one track or more. If the track index does not require a full track, the routine builds a new CCW chain to search the track index (see Figure 62). This routine then issues the EXCP and SVC7 (WAIT) to search the track index. If the over/under seek routine is not needed, it returns to the problem program.

ISAM RETRVE, RANDOM: WAITF Macro, Charts FD-FG

Objective: To ensure that the last EXCP issued has been completed and that the condition is normal. If the operation is a read, to locate the specified record and complete the transfer of data to the I/O area specified by the DTFIS entry IOAREAR, and to the specified work area if the DTFIS entry WORKR is included in the file definition. If the operation is a write, the objective is to return control to the problem program.

Entry: From the WAITF macro expansion.

Exit: To the problem program via linkage register 14.

Method: This routine first tests for ERREXT=YES. If ERREXT is specified, additional error conditions can be returned to the problem program, thus giving the user greater flexibility in attempting to continue processing. After initializing pointers to the DTFIS table, this routine tests the status byte in the DTF table, to determine if the condition so far is normal. If an abnormal condition exists, control returns to the problem program.

If the condition is normal, the routine issues a WAIT to determine if the EXCP issued by the READ or WRITE routines has been completed, and also tests for errors. Then, if the operation is a WRITE, this routine returns control to the problem program.

If the operation is a READ, this routine

must complete the read operation by moving the data to the I/O area. If first moves the address of the track in which the desired record is stored to the seek/search area, and initializes pointers to KEYARG and the I/O area. It also gets the relative key location and key length.

The routine then gets the index entry type (F code) from the search address and determines the routine to process that type. If the entry is a normal entry on an unshared track, a new CCW chain is built to find the record in the prime data area. (see Figure 63). If blocked records are specified, the CCW command code is modified to search high or equal. An EXCP and WAIT are issued to find the record and read the block into the I/O area. If records are unblocked, the record is moved into WORKR, if specified, and control returns to the problem program. If records are blocked, this routine tests to determine if KEYARG is less than the key in the first logical record. If it is, the record has not been found, and the corresponding bit is set on in the DTF table. Otherwise, the corresponding key is found within the block and the routine moves the block to WORKR, if specified. Control then returns to the problem program.

For a normal entry on a shared track, the routine decreases the record number in the search address by 1, and builds a new CCW chain to find records on a shared track (see Figure 64). Processing continues as in the routine to process a normal entry on an unshared track.

If the entry is an overflow end entry or an overflow chained entry, this routine first constructs a CCW chain to search the overflow chain (see Figure 65). An EXCP and WAIT are issued to locate the record in the overflow chain. A test is made to determine if the desired record has been found. If it has not been found, the routine tests for an overflow end entry. If it is an overflow end entry, the no-record-found bit is set on in the DTF table. If it is not an overflow end entry, the sequence link field is inserted in the seek/search address, and the overflow chain is searched again.

If the record has been found, overflow bits are set on in the DTF table, and the non-first overflow record count is increased by 1. The logical record is moved to WORKR, if specified. Control returns to the problem program via register 14.

If the entry is a dummy end entry or an inactive entry, the routine sets a no-record-found bit on in the DTF table, and returns control to the problem program.

ISAM RETRVE, RANDOM: WRITE Macro, KEY, Chart FH

Objective: To perform random retrieval output for an indexed sequential file.

Entry: From WRITE, KEY macro expansion.

Exit: To the problem program via register 14.

Method: This routine first sets the write bit on in the DTFIS table. It then tests for an uncorrectable DASD error, wrong length record error, or no record found error. If any of these errors exist, the no-record-found bit is set on in the DTF table, and control returns to the problem program.

If there are no errors, the status byte in the DTF table is reset, and pointers to the DTF table are initialized. This routine then gets the count field of the record as saved by the READ routine, the address of WORKR, and the address of the logical record within the I/O area. The record, or block of records, is moved to the I/O area from WORKR, if specified. The CCW chain to write records is then built (see Figure 66).

If the entry to be written is not an overflow entry, the byte count field in the write and verify CCWs is modified to the block length from the DTF table. This routine then issues the EXCP to write the record, and returns control to the problem program without issuing a WAIT. The WAIT function is left to the WAITF macro, which must be issued before the user can continue processing.

ISAM RETRVE, RANDOM: FREE Macro, Chart FK

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

Entry: From the FREE macro expansion.

Exit: To the problem program via linkage register 14.

Method: This routine determines whether the track hold option has been specified in the DTF. If so, both the held data track and the applicable held index track are released. All tracks are released by SVC 36 and control returns to the problem program.

CCW Builder Control Code[1]	CCW Built	Function
7461	X'69', &KEYARG, CC and SLI, Key Length	Search key equal or high the master/ cylinder index. Key supplied by user in the DTFIS table.
0C6B	X'08', Pointer to *+16, CC and SLI, 5	TIC to *+16.
D17B	X'1A', [2]&Filename.W, CC, SLI, and SKIP, 5	Read home address into random/sequential retrieval work area in DTFIS table.
536C	X'92', &IOAREAR, CC and SLI, 10	Read count (multiple track) into IOAREAR.
7461	X'69', KEYARG, CC and SLI, Key Length	Search key equal or high the master/ cylinder index. Key supplied by user in DTFIS table.
046B	X'08', Pointer to *-16, CC and SLI, 5	TIC to *-16.
110C	X'06', [2]&Filename.W, 00, 10	Read data (10-byte index level pointer) into random/sequential retrieval area in DTFIS table.

[1-2] See notes 1 and 2 in Figure 67.

Figure 61. Channel program builder for random retrieval -- CCW chain built to search master cylinder index.

CCW Builder Contrcl Code[1]	CCW Built	Function
D36B	X'1A', &IOAREAR, CC and SLI, 5	Read home address into IOAREAR.
9461	X'E9', &KEYARG, CC and SLI, Key Length	Search key equal or high (multiple-track) track index. Key supplied by user in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
110C	X'06', [2]&Filename.W, 00, 10	Read data (10-byte index level pointer) into random/sequential retrieval area in DTFIS table.

[1-2] See notes 1 and 2 in Figure 67.

Figure 62. Channel program builder for random retrieval -- CCW chain built to search track index.

CCW Builder Control Code[1]	CCW Built	Function
7461	X'69', &KEYARG, CC and SLI, Key Length	Search key equal or high in prime data area. Key supplied by user in DTFIS table.
0C4B	X'08', Pointer to *+16, CC, 5	TIC to *+16.
D17B	X'1A', [2]&Filename.W, CC, SLI, and SKIP, 5	Read home address into random/sequential retrieval work area in DTFIS table.
406C	X'12', [2]&Filename.S+3, CC and SLI, 10	Read count into common seek/search area in DTFIS table.
6461 or 7461	X'29' or X'69', &KEYARG, CC and SLI, Key Length	If records are unblocked, search key equal the prime data area. If records are blocked, search key equal or high in prime data area. Key supplied by user in DTFIS table.
044B	X'08', Pointer to *-16, CC, 5	TIC to *-16.
1302	X'06', &IOAREAR, 00, Block Length	Read data (block) containing starting record into IOAREAR.

[1-2] See notes 1 and 2 in Figure 67.

Figure 63. Channel program builder for random retrieval -- CCW chain built to find reccrd in prime data area (unshared track).

CCW Builder Control Code[1]	CCW Built	Function
804B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the prime data area using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
064B	X'08', Pointer to *-8, CC, 5	TIC to *-8.
406C	X'12', [2]&Filename.S+3, CC and SLI, 10	Read count into common seek/search area in DTFIS table.
6461 or 7461	X'29' or X'69', &KEYARG, CC and SLI, Key Length	If records are unblocked, search key equal the prime data area. If records are blocked, search key high or equal the prime data area. Key supplied by user in DTFIS table.
044B	X'08', Pointer to *-16, CC, 5	TIC to *-16.
1302	X'06', &IOAREAR, 00, Block Length	Read data (block) containing record into IOAREAR.

[1-2] See notes 1 and 2 in Figure 67.

Figure 64. Channel program builder for random retrieval -- CCW chain built to find record in prime data area (shared track).

CCW Builder Control Code[1]	CCW Built	Function
804B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the overflow chain using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
064B	X'08', Pointer to *-8, CC, 5	TIC to *-8.
6461	X'29', &KEYARG, CC and SLI, Key Length	Search key equal the overflow chain. Key supplied by user in DTFIS table.
116C	X'06', [2]&Filename.W, SLI, 10	Read data (10-byte sequence-link field) into random/sequential retrieval area in DTFIS table. This CCW is executed when the required overflow record is not found in the overflow chain.
1303	X'06', &IOAREAR, 00, Record Length + 10	Read data (sequence-link field plus logical record) into IOAREAR. This CCW is executed when the matching key is found in the overflow chain.

[1-2] See notes 1 and 2 in Figure 67.

Figure 65. Channel program builder for random retrieval -- CCW chain built to find record in overflow chain.

CCW Builder Control Code[1]	CCW Built	Function
804B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal prime data area using pointer (CCHHR) in common seek/ search area in DTFIS table.
064B	X'08', Pointer to *-8, CC, 5	TIC to *-8.
B3C3	X'05', &IOAREAR, CC, Record Length + 10	Write data from IOAREAR.
804B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the prime data area, using pointer (CCHHR) in common seek/search area in DTFIS table.
064B	X'08', Pointer to *-8, CC, 5	TIC to *-8.
1333	X'06', &IOAREAR, SLI and SKIP, Record Length + 10	Read data to verify record just written. Information is not transferred to main storage.

[1-2] See notes 1 and 2 in Figure 67.

Figure 66. Channel program builder for random retrieval -- CCW chain built to write record.

Note 1:

The first character of the control code references an operation code at IJHCSTRI.
The second character of the control code references a data area at IJHCASAD.
The third character of the control code references the following information:

Control Character	CCW Flag Field	Meaning
0	X'00'	End of CCW chain
2	X'20'	SLI (Suppress Length Indicator)
3	X'30'	SLI and SKIP (Suppress Data Transfer)
4	X'40'	CC (Command Chaining)
6	X'60'	CC and SLI
7	X'70'	CC and SLI and SKIP
C	X'C0'	CC and DC (Data Chaining)

The fourth character of the control code references a byte count (length) field at IJHCRESZ.

Note 2:

&Filename = DTF name supplied by user.
&Filename.X = X is suffix supplied by DTFIS for unique DTF labels.

Figure 67. Channel program builder for random retrieval -- notes.

ISAM RETRVE, SEQNTL: ESETL Macro, Chart GA

Objective: To write the last record if necessary, and reset the status byte in the DTFIS table.

Entry: From the ESETL macro expansion.

Exit: To the problem program via linkage register 14.

Method: After initializing pointers to the DTFIS table, this routine sets the status byte in the DTFIS table tc 0. A test is then made to determine if the PUT issued bit is on in the retrieval byte of the DTF table. If it is on, the last block of records is written. A test for IOAREA2=YES is then made. If IOAREA2 is specified as an ISMOD macro parameter option to allow overlapping of I/O with processing, a bit is set in the DTF table to indicate the first record is being processed and a wait for I/O completicn is made. If HOLD=YES is specified, an SVC 36 releases any held tracks. Control then returns to the problem program.

ISAM RETRVE, SEQNTL: GET Macro, Charts GB-GE

Objective: To perform the sequential retrieval input function for an indexed-sequential file.

Entry: From the GET macro expansion.

Exit: To the prcblem program via linkage register 14.

Method: This routine initializes pointers to the DTFIS table, and then tests for IOAREA2=YES. If IOAREA2 is not specified as an ISMOD macro parameter option, a test is made to determine if the last record read was in the overflow area. If the last record read was in the overflow area, the contents of the sequence-link field is moved to the seek/search area and a test is made to determine if the end of the overflow chain has been reached. If the last record read was in the overflow area and HOLD=YES has been specified, the track is released, then the overflow record is read and addresses are saved. If HOLD=YES is specified, the index track and data track are held during update procedure. The index track is then released. The record is then moved to WORKS, if specified, and control returns to the problem program.

If the end of the chain has been reached, the current DASD address is updated to the next track and the next record is read. The record is moved to the work area if specified, addresses are saved, and control returns to the problem program.

If the last record read was not in the overflow area, the routine determines if all records in the block have been processed. If all the records in the block have not been processed, the I/O area pointer is updated to the next logical record. If that record is not a padding record, it is moved to the work area, if specified, addresses are saved, and control returns to the problem program. If it is a padding record, the EOF indicator is set in the DTFIS table, and control returns to the

problem program.

If all records in the block have been processed, a check is made to determine if the PUT macro has been issued. If it has, the record is written.

A test is made to determine if the end of the track has been reached. If it has been reached, the track index is searched to find the next track index entry. The current track index record number in the DTF table is then updated, and a test is made to determine if there is any overflow record indicated in the track index entry just read. If the track index entry indicates an overflow record is present, that overflow record is read, and moved to the work area if specified. Control returns to the problem program.

If there is no overflow record indicated in the track index entry, the current address is updated by 1, and the record is read and moved into the work area, if specified. Control returns to the problem program.

If IOAREA2 has been specified to allow overlapping of I/O with processing, a test is made to determine if the last record read was in the overflow area. If the last record read was in the overflow area, a test is made to determine if the record being processed by the user is an overflow record. If it is an overflow record, a wait for I/O completion is made, the next available I/O area address is obtained, the current record address is saved, and a test is made to determine if the first record is being processed.

If the first record is being processed, the overflow record is read, its address is saved in the DTF table and the record is moved to the work area if specified. Control then returns to the problem program.

If the first record is not being processed, the address of the next overflow record is moved to the seek/search address and a test is made to determine if the end of the overflow chain has been reached. If the end has not been reached, the overflow record is read, and addresses are saved. If HOLD=YES is specified, an SVC 36 releases any held tracks. Control then returns to the problem program.

If the end of the overflow chain has been reached, the current disk address is updated to the next track and the next record is read. The record is moved to the work area if specified, addresses are saved, and control returns to the problem program.

If the last record was not an overflow record and the current record is not an overflow record, this routine determines if all the records in the block have been processed. If all records in the block have not been processed, the I/O area pointer is updated to the next logical record and the record number is updated by 1. If the next logical record is not a padding record, it is moved to the work area, if specified, addresses are saved and control returns to the problem program. If it is a padding record, the EOF indicator is set in the DTF table and control returns to the problem program.

When all the records in the block have been processed, a check is made to determine if the PUT macro has been issued. If the PUT macro has been issued, the record is written. A test for the presence of two I/O areas is then made. If the presence of two I/O areas is indicated in the DTF table, a test is made to determine if the first record is being processed. If the first record is not being processed, a wait for I/O completion is made and the record address is saved in the DTF table.

A test is made to determine if the end of the track has been reached. If the end of the track has been reached, the track index is searched to find the next track index entry. The current track index record number in the DTF table is then updated, and a test is made to determine if there is any overflow record indicated in the track index entry just read. If the presence of an overflow entry is indicated, and there are two I/O areas present, this routine tests to determine if this is the first record. If it is not the first record, the record counter is set to 1. The overflow record address is moved to the seek/search address, and the overflow record is read and moved to the work area, if specified. Control then returns to the problem program.

If there is no overflow record indicated in the track index entry, the current address is updated by 1, the next record is read and moved to the work area, if specified, and control returns to the problem program.

ISAM RETRVE, SEQNTL: PUT Macro, Chart GF

Objective: To perform the sequential retrieval output function for an indexed sequential file.

Entry: From the PUT macro expansion.

Exit: To the problem program via linkage register 14.

Method: After initializing pointers to the DTFIS table, this routine tests whether a GET has been issued. If a GET has not been issued, there is an SVC 50 (error). Otherwise, the GET issued switch is turned off and the output bit in the retrieval byte in the DTFIS table is turned on. The

record is moved from the work area to the I/O area, and the output bit in the retrieval byte is set off.

If the record is unblocked, or is in the overflow area, this routine writes the record and returns control to the problem program. If the records are blocked, this routine sets the bit in the retrieval byte to indicate that the PUT macro has been issued for this record, and returns control to the problem program. The GET macro routine causes the block to be written on DASD when it determines that all records in the block have been processed.

ISAM RETRVE, SEQNTL: SETI Macro, $$BSETL, Charts GG-GL

Objective: To initialize for sequential retrieval based on information supplied by the user in the SETL macro.

Entry: From the SETL macro expansion.

Exit: To the problem program.

Method: This logical transient first validates the limits of the DTFIS table and IOAREAS to ensure that they lie within the partition. If HOLD=YES has been specified in the DTF, $$BSETL1 is fetched to perform the SETL macro functions. If track hold has not been specified, $$BSETL then initializes for sequential retrieval based on the information supplied by the user in the SETI macro. The SETL macro specifies the type of reference used to identify the first record to be processed. The types of reference are:

- KEY. Key of starting record in the file.
- GKEY. Location of starting record in the file, identified by a record key within a desired group. The user supplies a key that identifies the high order bytes of the required group of keys. For example, a GKEY specification of D6430000 would permit file processing to start at the first key containing D643XXXX, regardless of the characters represented by the Xs.
- BOF. Beginning of the logical file.
- ID (MBBCCHHR). Location of starting record in the prime data area.

If sequential retrieval is to begin with a record associated with a particular key (KEY), the key of the beginning record must be placed in the field defined by the DTFIS entry KEYARG before issuing the SETL macro. This phase searches the master index (if present), cylinder index and track index until it finds the track index entry associated with the specified key. It determines whether the record with the desired key is on a shared or unshared prime data track or in the overflow area.

If the record is on a shared track, the search is initialized to begin after the remainder of the track index has been bypassed. If the record is on a prime data track and records are unblocked, the track index overflow entry address associated with the desired record is calculated and stored in the DTFIS table, and the track is searched equal for the desired record. If the record is not found, a no-record-found indicator is set in the DTFIS table (Filename.C).

If the file contains blocked records, the track is searched equal/high for the desired block. The user must supply (in the DTFIS entry KEYLOC) the position of the key field in the data record. The block is then searched. When the record with the matching key is found, its address is saved in the DTFIS table and control returns to the problem program. If the record is not found, the no-record-found indicator is set in the DTFIS table (Filename.C).

If the record with the desired key is in the overflow area, the track index normal and overflow entry addresses are stored in the DTFIS table, and the overflow chain is searched for the desired record. When the desired record is found, its address is saved in the DTFIS table. If the desired record is not found in the overflow chain, a no-record-found indicator is set in the DTFIS table (Filename.C) and control returns to the problem program.

If GKEY was specified in the SETL macro, the CCW chain to read the desired record is modified to search key equal or high. The search for the desired record then proceeds im the same manner as if KEY were specified in the SETL macro. However, in this case (GKEY), a no-record-found condition should not occur unless the key specified is higher than the existing highest key in the file.

If BOF was specified in the SETL macro, the address of the first prime data record in the file is saved in the sequential retrieve section of the DTFIS table, and the track index overflow entry address associated with the desired record is calculated and stored in the DTFIS table. Control then returns to the problem program.

If the starting record address is referenced by a symbolic name in the SETL macro, this phase analyzes the 8-byte DASD address (MBBCCHHR) in the field specified by the symbolic name for validity. If the address is invalid, an illegal ID indicator is set in the DTFIS table (Filename.C) and control returns to the problem program. If the starting address is valid, this phase saves the address in the DTFIS table, calculates the track index overflow entry

address associated with the desired record, stores it in the DTFIS table and returns control to the problem program.

In order to perform a search of the master, cylinder or track indexes, prime data area and overflow area for the starting record, a CCW string is built to search the required areas. Figures 68-71 give a description of the channel program built to perform the necessary search.

ISAM RETRVE, SEQNTL: SETL Macro, $$BSETL1, Charts GM-GR

Objective: To initialize for sequential retrieval when HOLD=YES, based on information supplied by the user in the SETL macro.

Entry: From the SETL macro expansion.

Exit: To the problem program.

Method: This logical transient first validates the limits of the DTFIS table and IOAREAS to ensure that they lie within the partition. It then initializes for sequential retrieval based on the information supplied by the user in the SETL macro. The SETL macro specifies the type of reference used to identify the first record to be processed. The types of reference are:

- KEY. Key of starting record in the file.
- GKEY. Location of starting record in the file, identified by a record key within a desired group. The user supplies a key that identifies the high order bytes of the required group of keys. For example, a GKEY specification of D6430000 would permit file processing to start at the first key containing D643XXXX, regardless of the characters represented by the Xs.
- BOF. Beginning of the logical file.
- ID (MBBCCHHR). Location of starting record in the prime data area.

If sequential retrieval is to begin with a record associated with a particular key (KEY), the key of the beginning record must be placed in the field defined by the DTFIS entry KEYARG before the SETL macro is issued. This phase searches the master index (if present), cylinder index, and track index until it finds the track index entry associated with the specified key. It determines whether the record with the desired key is on a shared or unshared prime data track or in the overflow area.

If the record is on a shared track, the search is initialized to begin after the remainder of the track index has been bypassed. If the record is on a prime data track and records are unblocked, the track index overflow entry address associated with the desired record is calculated and stored in the DTFIS table, and the track is searched equal for the desired record. If the record is not found, a no-record-found indicator is set in the DTFIS table (Filename.C).

If the file contains blocked records, the track is searched equal/high for the desired block. The user must supply (in the DTFIS entry KEYLOC) the position of the key field in the data record. The block is searched. When the record with the matching key is found, its address is saved in the DTFIS table, and control returns to the problem program. If the record is not found, the no-record-found indicator is set in the DTFIS table (Filename.C).

If the record with the desired key is in the overflow area, the track index normal and overflow entry addresses are stored in the DTFIS table, the appropriate index and data tracks are held, and the overflow chain is searched for the desired record. When the desired record is found, its address is saved in the sequential retrieval section of the DTFIS table. If the desired record is not found in the overflow chain, a no-record-found indicator is set in the DTFIS table (Filename.C), the held index and data tracks are released, and control returns to the problem program.

If GKEY was specified in the SETL macro, the CCW chain to read the desired record is modified to search key equal or high. The search for the desired record proceeds in the same manner as if KEY were specified in the SETL macro. However, in this case (GKEY), a no-record-found condition should not occur unless the key specified is higher than the existing highest key in the file.

If BOF was specified in the SETL macro, the address of the first prime data record in the file is saved in the sequential retrieval section of the DTFIS table, and the track index overflow entry address associated with the desired record is calculated and stored in the DTFIS table. Control returns to the problem program.

If the starting record address is referenced by a symbolic name in the SETL macro, this phase checks the validity of the 8-byte DASD address (MBBCCHHR) in the field specified by the symbolic name. If the address is invalid, an illegal ID indicator is set in the DTFIS table (Filename.C), and control returns to the problem program. If the starting address is valid, this phase saves the address in the DTFIS table, calculates the track index overflow entry address associated with the desired record, stores it in the DTFIS table, holds the required index and data tracks, and returns control to the problem

program.

Before I/O is performed, a switch is tested to see if track hold has been specified. If it has, the data track(s) and the corresponding index track(s) are held until I/O is complete. If any error conditions occur (for example, no record found, wrong length record, or a DASD read error), any held tracks are freed before returning to the user.

In order to perform a search of the master, cylinder, or track indexes, prime data area, and overflow area for the starting record, a CCW string is built to search the required area. Figures 68 - 71 give a description of the channel program built to perform the necessary search.

Label	CCW Builder Control Code[1]	CCW Built	Function
	7441	X'69', &KEYARG, CC, Key Length	Search key equal or high the master/ cylinder index. Key supplied by user in DTFIS table.
	0C40	X'08', Pointer to *+16, CC, Record Length	TIC to *+16.
	B04B	X'1A', [2]&Filename.S+3, CC, 5	Read home address into common seek/search area in DTFIS table.
	506B	X'92', [2]&Filename.S+3, CC and SLI, 5	Read count (multiple-track) - CCHHR - into common seek/search area in DTFIS table.
	7441	X'69', &KEYARG, CC, Key Length	Search key equal or high the master/ cylinder index. Key supplied by user.
	0440	X'08', Pointer to *-16, CC, Record Length	TIC to *-16.
	110C	X'06', [2]&Filename.W, 00, 10	Read data (10-byte index level pointer) into random/sequential retrieval area in DTFIS table. The data field is then moved from the random/sequential retrieval area to the common seek/search area for the next search.

[1--6] See notes 1 through 6 in Figure 75.

Figure 68. Channel program builder for sequential retrieval -- CCW chain built to search master cylinder index.

Label	CCW Builder Control Code[1]	CCW Built	Function
	806C	X'31', [2]&Filename.S+3, CC and SLI, 10	Search identifier equal the track index using the 10-byte pointer in the common seek/search area.
	0640	X'08', Pointer *-8, CC, Record Length	TIC to *-8.
	126C	X'06', &IOAREAS, CC and SLI, 10	Read data (10-byte track index pointer) into IOAREAS, input/output area for sequential retrieval supplied by user.
	506B	X'92', [2]&Filename.S+3, CC and SLI, 5	Read count (multiple-track) - CCHHR - into common seek/search area in DTFIS table.
	7441	X'69', &KEYARG, CC, Key Length	Search key equal or high the track index. Key supplied by user.
	0240	X'08', Pointer to *-24, CC, Record Length	TIC to *-24.
	110C	X'06', [2]&Filename.W, 00, 10	Read data (10-byte pointer) into random/sequential retrieval area in DTFIS table. The data field is then moved from the random/sequential retrieval area for the next search.

[1]--[6] See notes 1 through 6 in Figure 75.

Figure 69. Channel program builder for sequential retrieval -- CCW chain built to search track index.

Label	CCW Builder Control Code[1]	CCW Built	Function
STRI1	084B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the prime data area using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
	066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
	416C	X'12', [2]&Filename.W, CC and SLI, 10	Read count into common seek/search area in the DTFIS table.
	6441 or 7441	X'29' or X'69', &KEYARG, CC, Key Length	If KEY is specified in the SETL macro and/or records are unblocked, this CCW searches key equal the prime data area. If GKEY is specified in the SETL macro and/or records are blocked, this CCW searches key equal or high the prime data area for the starting record.
	046B	X'08', Pointer to *-16, CC and SLI, 5	TIC to *-16.
	1202	X'06', &IOAREAS, 00, Block Size	Read data (block containing starting record) into IOAREAS.

[1]--[6] See notes 1 through 6 in Figure 75.

Figure 70. Channel program builder for sequential retrieval -- CCW chain built to find starting record in prime data area.

Label	CCW Builder Control Code[1]	CCW Built	Function
STRI3	804B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the overflow chain using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
	0640	X'08', Pointer to *-8, CC, Record Length	TIC to *-8.
	6441 or 7441	X'29' or X'69', &KEYARG, CC, Key Length	If KEY is specified in the SETL macro, this CCW searches key equal the overflow chain for the starting record. If GKEY is specified in the SETL macro, this CCW searches key equal or high the overflow chain for the starting record.
	112C	X'06', [2]&Filename.W, SLI, 10	Reads data (10-byte sequence link field) into random/sequential retrieval area in DTFIS table. This CCW is executed when the required overflow record is not found in the overflow chain.
	1203	X'06', &IOAREAS, 00, Record Length +10	Read data (sequence link field plus starting record) into IOAREAS. This CCW is executed when the matching key is found in the overflow chain.

[1]--[6] See notes 1 through 6 in Figure 75.

Figure 71. Channel program builder for sequential retrieval -- CCW chain built to find starting record in overflow chain.

CCW Parameter[3] Control Code	CCW Built	Function
0540	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the prime data area using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
	X'0', Pointer to *-8, CC, 0	TIC to *-8.
	X'05',[4] &IOAREAS, CC,[5] Block Length[6]	Write data (block) onto prime data area.
	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal to verify write operation.
	X'08', Pointer to *-8, CC, 0	TIC to *-8.
	X'06',[4] &IOAREAS, SKIP, Block Length[6]	Read data to verify write operation.

[1]--[6] See notes 1 through 6 in Figure 75.

Figure 72. Channel program builder for sequential retrieval -- CCW chain built to write records.

CCW Parameter[3] Control Code	CCW Built	Function
0601	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the track index area using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
	X'08', Pointer to *-8, CC, 0	TIC to *-8.
	X'06',[4] [2]&Filename.W, CC,[5]10	Read data (10-byte index level pointer) into random/sequential retrieval area in the DTFIS table.
	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal to verify read operation.
	X'08', Pointer to *-8, CC, 0	TIC to *-8.
	X'06', &IOAREAS, SKIP, Block Size	Read data to verify read operation.

[1]--[6] See notes 1 through 6 in Figure 75.

Figure 73. Channel program builder for sequential retrieval -- CCW chain built to search track index.

CCW Parameter[3] Control Code	CCW Built	Function
0600	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the prime data area using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
	X'08', Pointer to *-8 ,CC, 0	TIC to *-8.
	X'06',[4] &IOAREAS, CC,[5] Block Length[3]	Read data into IOAREAS.
	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal to verify read operation.
	X'08', Pointer to *-8 ,CC, 0	TIC to *-8.
	X'06',[4] &IOAREAS, SKIP, Block Length[4]	Read data to verify read operation.

[1]--[6] See notes 1 through 6 in Figure 75.

Figure 74. Channel program builder for sequential retrieval -- CCW chain built to read records.

Note 1: The CCW chains are built by the B-transients, $$BSETL and $$BSETL1. The CCW builder control code references information in the $$BSETL and $$BSETL1 assemblies.

The first character of the control code references an operation code at IJHROP.
The second character of the control code references a data area at IJHARA.
The third character of the control code references the following information:

Control Character	CCW Flag Field	Meaning
0	X'00'	End of CCW chain
2	X'20'	SLI (Suppress Length Indicator)
4	X'40'	CC (Command Chaining)
6	X'60'	CC and SLI

The fourth character of the control code references a byte count (length) field at RELNT.

Note 2:

&Filename = DTF name supplied by user.
&Filename.X = X is suffix supplied by DTFIS for unique DTF labels.

Note 3:

The CCW parameter is found in the ISMOD assembly.

The first byte of the parameter is the command code.

The second byte of the parameter contains flags with the exception of the chain to search the track index. In this case, the second byte is an indicator to the channel program builder that the CCW chain is to search the track index.

Note 4: If the file contains unblocked records, the command code is modified to either Read Key and Data, or Write Key and Data.

Note 5: If the verify option has not been specified, the command chaining bit is not set on.

Note 6: If the file contains unblocked records, the byte count field contains the physical record length plus Key Length.

Figure 75. Channel program builder for sequential retrieval -- notes.

ISAM ADDRTR: ESETL Macro, Chart JA

Objective: To write the last record, if necessary, and to reset the status byte in the DTFIS table.

Entry: From the ESETL macro expansion.

Exit: To the problem program via linkage register 14.

Method: After initializing pointers to the DTFIS table, this routine sets the status byte in the DTFIS table to 0. A test is then made to determine if the PUT issued bit is on in the retrieval byte of the DTF table. If it is on, the last block of records is written. A test for IOAREA2=YES is then made. If IOAREA2 is specified as an ISMOD macro parameter option to allow overlapping of I/O with processing, a bit is set in the DTF table to indicate the first record is being processed and a wait for I/O completion is made. Control then returns to the problem program.

ISAM ADDRTR: GET Macro, Charts JB-JE

Objective: To perform sequential retrieval input for an indexed sequential file.

Entry: From the GET macro expansion.

Exit: To the problem program via linkage register 14.

Method: This routine initializes pointers to the DTFIS table, and then tests for IOAREA2=YES. If IOAREA2 is not specified as an ISMOD macro parameter option, a test is made to determine if the last record read was in the overflow area. If the last record read was in the overflow area, the contents of the sequence-link field is moved to the seek/search area, and a test is made to determine if the end of the overflow chain has been reached. If it has not been reached, the overflow record is read, addresses are saved, and the record is moved to WORKS, if specified. Control then returns to the problem program.

If the end of the chain has been reached, the current disk address is updated to the next track and the next record is read. The record is moved to the work area, if specified, addresses are saved, and control returns to the problem program.

If the last record read was not in the overflow area, the routine determines if all records in the block have been processed. If all the records in the block have not been processed, the I/O area pointer is updated to the next logical record. If that record is not a padding record, it is moved to the work area, if specified, addresses are saved, and control returns to the problem program. If it is a padding record, the EOF indicator is set in the DTFIS table, and control passes to the problem program.

If all records in the block have been processed, a check is made to determine if the PUT macro has been issued. If it has, the record is written.

A test is made to determine if the end of the track has been reached. If it has been reached, the track index is searched to find the next track index entry. The current track index record number in the DTF table is then updated, and a test is made to determine if there is any overflow record indicated in the track index entry just read. If the track index entry indicates an overflow record is present, that overflow record is read, and moved to the work area, if specified. Control returns to the problem program.

If there is no overflow record indicated in the track index entry, the current address is updated by 1, and the record is read and moved into the work area, if specified. Control returns to the problem program.

If IOAREA2 has been specified to allow overlapping of I/O with processing, a test is made to determine if the last record read was in the overflow area. If the last record read was in the overflow area, a test is made to determine if the record being processed by the user is an overflow record. If it is an overflow record, a wait for I/O completion is made, the next available I/O area address is obtained, the current record address is saved, and a test is made to determine if the first record is being processed.

If the first record is being processed, and if track hold has been specified, the appropriate index and data tracks are held, the overflow record is read, addresses are saved, and the record is moved to a workarea, if specified. Any held tracks are released, and control returns to the problem program.

If the first record is not being processed, the address of the next overflow record is moved to the seek/search address and a test is made to determine if the end of the overflow chain has been reached. If the end has not been reached, and if track hold has been specified, the appropriate index and data tracks are held, the overflow record is read, addresses are saved and the record is moved to the work area, if specified. Any held tracks are released and control returns to the problem program.

If the end of the overflow chain has been reached, the current disk address is updated to the next track and the next record is read. The record is moved to the work area, if specified, addresses are

saved, and control returns to the problem program.

If the last record was not an overflow record and the current record is not an overflow record, this routine determines if all the records in the block have been processed. If all records in the block have not been processed, the I/O area pointer is updated to the next logical record and the record counter is updated by 1. If the next logical record is not a padding record, it is moved to the work area, is specified, addresses are saved, and control returns to the problem program. If it is a padding record, the EOF indicator is set in the DTF table and control returns to the problem program.

When all the records in the block have been processed, a check is made to determine if the PUT macro has been issued. If the PUT macro has been issued, the record is written. A test for the presence of two I/O areas is then made. If the presence of two I/O areas is indicated in the DTF table, a test is made to determine if the first record is being processed. If the first record is not being processed, a wait for I/O completion is made and the record address is saved in the DTF table.

A test is made to determine if the end of the track has been reached. If the end of the track has been reached, the track index is searched to find the next track index entry. The current track index record number in the DTF table is updated. If HOLD is specified, the index track is held while a test is made to determine if there is any overflow record indicated in the track index entry just read. If the presence of an overflow entry is indicated, and there are two I/O areas present, this routine tests to determine if this is the first record. If it is not the first record, the record counter is set to 1. The overflow record address is moved to the seek/search address, the overflow record is read, held if HOLD=YES is specified, and moved to the work area, if specified. Control then returns to the problem program.

If there is no overflow record indicated in the track index entry, the current address is updated by 1, the next record is read and moved to the work area, if specified, and control returns to the problem program.

ISAM ADDRTR: PUT Macro, Chart JF

Objective: To perform sequential retrieval output for an indexed sequential file.

Entry: From the PUT macro expansion.

Exit: To the problem program via linkage register 14.

Method: After initializing pointers to the DTFIS table, this routine tests whether a GET has been issued. If a GET has not been issued, there is an SVC 50 (error). Otherwise, the GET issued switch is turned off and the output bit in the retrieval byte in the DTFIS table is turned on. The record is moved from the work area to the I/O area, and the output bit in the retrieval byte is set off.

If the record is unblocked, or is in the overflow area, this routine writes out the record and returns control to the problem program. If the records are blocked, this routine sets the bit in the retrieval byte to indicate that the PUT macro has been issued for this record, and returns control to the problem program. The GET macro routine causes the block to be written on DASD when it determines that all records in the block have been processed.

ISAM ADDRTR: READ Macro, KEY, Chart JG

Objective: To perform the random retrieval input function for an indexed-sequential file by searching the indexes to determine the track on which the desired record is stored.

Entry: From the READ, KEY macro.

Exit: To the problem program via linkage register 14.

Method: This routine first initializes pointers and status bits in the DTFIS table. It then constructs the CCW chain to search the master or cylinder index (see Figure 76). It then determines the highest level index (master or cylinder) being used, and tests for ERREXT=YES. If ERREXT is specified, additional error conditions can be returned to the problem program. This allows the user greater flexibility in attempting to continue processing. This routine then searches the highest level index to get the address of the next index to be searched.

A test of the F code from the index level pointer is made to determine if the next search is of the track index. The F code refers to the index level just searched. If the master index was just searched, the next search is on the cylinder index. If the next search is not on the track index, the routine gets the index entry type and determines the routine to process that type.

If the entry type is a normal entry, the routine returns to search the next index. If the entry type is a dummy end entry or an inactive entry, the routine branches to an error routine to set a no-record-found

flag in the DTF table and return to the problem program. If the entry is a dummy chained entry, the routine returns to search the index by using the address supplied by the 10-byte index level pointer.

When the next search is found to be on the track index, a test is made to determine if the track index takes up one track or more. If the track index does not require a full track, the routine builds a new CCW chain to search the track index (see Figure 77). The routine issues the EXCP to search the track index, and returns control to the problem program without issuing a WAIT. The WAIT function is left to the WAITF macro, which must be issued before the user can process the record.

ISAM ADDRTR: SETL Macro, $$BSETL, Charts JH-JK

Objective: To initialize for sequential retrieval based on information supplied by the user in the SETL macro (BOF or ID).

Entry: From the SETL macro expansion.

Exit: To the problem program or to $$BSETL1.

Method: This logical transient first validates the limits of the DTFIS table and IOAREAS to ensure that they lie within the partition. $$BSETL then initializes for sequential retrieval based on the information supplied by the user in the SETL macro. The SETL macro specifies the type of reference used to identify the first record to be processed. The types of reference are:

1. KEY. Key of starting record in the file.
2. GKEY. Location of starting record in the file, identified by a record key within a desired group. The user supplies a key that identifies the high order bytes of the required group of keys. For example, a GKEY specification of D6430000 would permit file processing to start at the first key containing D643XXXX, regardless of the characters represented by the Xs.
3. BOF. Beginning of logical file.
4. ID (MBBCCHHR). Location of starting record in the prime data area.

Note: References 1 and 2 are processed by $$BSETL1.

If sequential retrieval is to begin with a record associated with a particular key (KEY), the key of the beginning record must be placed in the field defined by the DTFIS entry KEYARG before the SETL macro is issued. This phase searches the master index (if present), cylinder index and track index until it finds the track index entry associated with the specified key. It determines whether the record with the desired key is on a shared or unshared prime data track or in the overflow area.

If the record is on a shared track, the search is initialized to begin after the remainder of the track index has been bypassed. If the record is on a prime data track and records are unblocked, the track index overflow entry address associated with the desired record is calculated and stored in the DTFIS table, and the track is searched equal for the desired record. If the record is not found, a no-record-found indicator is set in the DTFIS table (Filename.C).

If the file contains blocked records, the track is searched equal/high for the desired block. The user must supply (in the DTFIS entry KEYLOC) the position of the key field in the data record. The block is searched. When the record with the matching key is found, its address is saved in the DTFIS table, and control returns to the problem program. If the record is not found, the no-record-found indicator is set in the DTFIS table (Filename.C).

If the record with the desired key is in the overflow area, the track index normal and overflow entry addresses are stored in the DTFIS table, and the overflow chain is searched for the desired record. When the desired record is found, its address is saved in the sequential retrieval section of the DTFIS table. If the desired record is not found in the overflow chain, a no-record-found indicator is set in the DTFIS table (Filename.C), and control returns to the problem program.

If GKEY was specified in the SETL macro, the CCW chain to read the desired record is modified to search key equal or high. The search for the desired record proceeds in the same manner as if KEY were specified in the SETL macro. However, in this case (GKEY), a no-record-found condition should not occur unless the key specified is higher than the existing highest key in the file.

If BOF was specified in the SETL macro, the address of the first prime data record in the file is saved in the sequential retrieval section of the DTFIS table, and the track index overflow entry address associated with the desired record is calculated and stored in the DTFIS table. Control returns to the problem program.

If the starting record address is referenced by a symbolic name in the SETL macro, this phase analyzes the 8-byte DASD address (MBBCCHHR) in the field specified by the symbolic name for validity. If the

address is invalid, an illegal ID indicator is set in the DTFIS table (Filename.C), and control returns to the problem program. If the starting address is valid, this phase saves the address in the DTFIS table, calculates the track index overflow entry address associated with the desired record, stores it in the DTFIS table, and returns control to the problem program.

In order to perform a search of the master, cylinder or track indexes, prime data area, and overflow area for the starting record, a CCW string is built to search the required areas. Figures 91-94 give a description of the channel program built to perform the necessary search.

ISAM ADDRTR: SETL Macro, $$BSETL1, Charts JL-JQ

Objective: To initialize for sequential retrieval based on information supplied by the user in the SETL macro (KEY or GKEY).

Entry: From $$BSETL.

Exit: To the problem program or to $$BSETL2.

Method: This logical transient initializes for sequential retrieval based on the information supplied by the user in the SETL macro. The SETL macro specifies the type of reference used to identify the first record to be processed. The types of reference are:

1. KEY. Key of starting record in the file.

2. GKEY. Location of starting record in the file, identified by a record key within a desired group. The user supplies a key that identifies the high order bytes of the required group of keys. For example, a GKEY specification of D6430000 would permit file processing to start at the first key containing D643XXXX, regardless of the characters represented by the Xs.

3. BOF. Beginning of the logical file.

4. ID (MBBCCHHR). Location of starting record in the prime data area.

Note: References 3 and 4 are processed by $$BSETL.

If sequential retrieval is to begin with a record associated with a particular key (KEY), the key of the beginning record must be placed in the field defined by the DTFIS entry KEYARG before the SETL macro is issued. This phase searches the master index (if present), cylinder index, and track index until it finds the track index entry associated with the specified key. It determines whether the record with the desired key is on a shared or unshared prime data track or in the overflow area.

If the record is on a shared track, the search is initialized to begin after the remainder of the track index has been bypassed. If the record is on a prime data track and records are unblocked, the track index overflow entry address associated with the desired record is calculated and stored in the DTFIS table, and the track is searched equal for the desired record. If the record is not found, a no-record-found indicator is set in the DTFIS table (Filename.C).

If the file contains blocked records, the track is searched equal/high for the desired block. The user must supply (in the DTFIS entry KEYLOC) the position of the key field in the data record. The block is searched. When the record with the matching key is found, its address is saved in the DTFIS table, and control returns to the problem program. If the record is not found, the no-record-found indicator is set in the DTFIS table (Filename.C).

If the record with the desired key is in the overflow area, the track index normal and overflow entry addresses are stored in the DTFIS table, the appropriate index and data tracks are held, and the overflow chain is searched for the desired record. When the desired record is found, its address is saved in the sequential retrieval section of the DTFIS table. If the desired record is not found in the overflow chain, a no-record-found indicator is set in the DTFIS table (Filename.C), the held index and data tracks are released, and control returns to the problem program.

If GKEY was specified in the SETL macro, the CCW chain to read the desired record is modified to search key equal or high. The search for the desired record proceeds in the same manner as if KEY were specified in the SETL macro. However, in this case (GKEY), a no-record-found condition should not occur unless the key specified is higher than the existing highest key in the file.

If BOF was specified in the SETL macro, the address of the first prime data record in the file is saved in the sequential retrieval section of the DTFIS table, and the track index overflow entry address associated with the desired record is calculated and stored in the DTFIS table. Control returns to the problem program.

If the starting record address is referenced by a symbolic name in the SETL macro, this phase checks the validity of the 8-byte DASD address (MBBCCHHR) in the field specified by the symbolic name. If the address is invalid, an illegal ID indicator is set in the DTFIS table (Filename.C), and control returns to the

problem program. If the starting address is valid, this phase saves the address in the DTFIS table, calculates the track index overflow entry address associated with the desired record, stores it in the DTFIS table, holds the required index and data tracks, and returns control to the problem program.

Before I/O is performed, a switch is tested to see if track hold has been specified. If it has, the data track(s) and the corresponding index track(s) are held until I/O is complete. If any error conditions occur (for example, no record found, wrong length record, or a DASD read error), any held tracks are freed before returning to the user.

In order to perform a search of the master, cylinder, or track indexes, prime data area, and overflow area for the starting record, a CCW string is built to search the required area. Figures 91-94 give a description of the channel program built to perform the necessary search.

ISAM ADDRTR: SETL Macro, $$BSETL2

Objective: To complete the initialization of the DTF, to free or hold tracks as required, and to return control to the user.

Entry: From $$BSETL1.

Exit: To the problem program.

Method: This phase is called by $$BSETL1 if no I/O errors have been detected. The initialization of the DTF is completed for sequential retrieval starting with the specified key or low key in a group of keys. If HOLD=YES is specified in the DTF, the index track and the data track are held until exit, when the index track is freed. The data track is freed by the module. If an error occurs, the proper error bit is posted in Filename.C and exit to the user is made after freeing all held tracks.

ISAM ADDRTR: WAITF Macro, Charts KA-KE

Objective:

1. To ensure that the last EXCP issued has been completed and that the condition is normal.
2. If the operation is a READ, to locate the specified record and to complete the transfer of data to the I/O area specified by the DTFIS entry IOAREAR, and to the specified work area if the DTFIS entry WORKR is included in the file definition.
3. If the operation is a WRITE (NEWKEY) to complete the addition of the record to an indexed sequential file, adjusting the indexes and other records as necessary.
4. If the operation is a WRITE (KEY), to return control to the problem program.

Entry: From the WAITF macro expansion.

Exit: To the problem program via linkage register 14.

Method: This routine first tests for ERREXT=YES. If ERREXT is specified, additional error conditions can be returned to the problem program. This allows the user greater flexibility in attempting to continue processing. After initializing pointers to the three sections of the DTFIS table, this routine tests the status byte in the DTF table to determine if the condition so far is normal. If not, control returns to the problem program.

If the condition is normal, the routine issues a WAIT to determine if the EXCP issued by the READ or WRITE routines has been completed, and also tests for errors. Then, if the operation is a WRITE (KEY), this routine returns control to the problem program.

If the operation is a READ, this routine must complete the READ operation by moving the data to the I/O area. If first sets the address of the track on which the desired record is stored in the seek/search area, and initializes pointers to KEYARG and the I/O area. It also gets the relative key location and key length.

The routine picks up the index entry type (F code) from the search address and determines the routine to process that type. If the entry is a normal entry on an unshared track, a new CCW chain is built to find the record in the prime data area (see Figure 78). If blocked records are specified, the CCW command code is modified to search high or equal. An EXCP and WAIT are issued to find the record and read the block into the I/O area. If records are unblocked, the record is moved into WORKR (if specified), and control returns to the problem program. If records are blocked, this routine tests to determine if the key specified in KEYARG is less than the key in the first logical record. If so, the record has not been found, and the corresponding bit is set on in the DTF table. Otherwise, the corresponding key is found within the block and the routine moves the block to WORKR, if specified. Control is then returned to the problem program.

For a normal entry on a shared track, the routine decreases the record number in the search address by 1, and builds a new CCW chain to find records on a shared track

(see Figure 79). Processing continues as in the routine to process a normal entry on an unshared track.

If the entry is an overflow end entry or an overflow chained entry, this routine first constructs a CCW chain to search the overflow chain. An EXCP and a WAIT are issued to locate the record in the overflow chain. A test is made to determine if the desired record has been found. If not, the routine tests for a overflow end entry. If so, the no-record-found bit is set on in the DTF table. If the entry is not an overflow end entry, the sequence-link field is inserted in the seek/search address, and the overflow chain is searched again.

If the record has been found, overflow bits are set on in the DTF table, and the first nonoverflow record count is increased by 1. The logical record is moved to WORKR, if specified. Control returns to the problem program via register 14.

If the entry is a dummy end entry or an inactive entry, the routine sets a no-record-found bit on in the DTF table, and returns control to the problem program.

If the operation is a WRITE (NEWKEY), this routine determines the type of add function to be performed. The three types of add functions are:

- Normal add to the prime data area.
- Add to the overflow area.
- EOF add.

Normal Add to the Prime Data Area: This routine first determines if the record is to be added to the last prime data track. If it is and the last prime data track is full, the overflow record address is calculated, an EXCP (see Figure 85) is issued to search and read the prime data track to determine the point of insertion, and a wait for I/O completion is made. Figures 76 through 111 describe the channel program builder for the ADDRTR function. If the addition is not on the last prime data track, the overflow record address is calculated and the prime data track is searched to determine the point of insertion for the record to be added to the file. When an equal/high key is found during the search, the count and data fields of that location are read into a save area in the DTF table and IOAREAL respectively.

A test is made to determine if the prime data in core option was specified as an ISMOD macro parameter. If it was specified, as many records as can fit into the I/O area specified in the DTFIS operand IOAREAL are read from the prime data track into main storage. The key of the record to be added is compared to the keys of the existing records in the I/O area. If a duplicate key is found, the condition is indicated to the user in the DTF table entry labeled Filename.C. If no duplicate key is found, the records are shifted in main storage leaving the record with the highest key remaining in the user's work area, WORKL. The other records are rewritten directly onto the track. Any remaining records on the track are then read into the I/O area. The process continues until the last record on the track is set up as an overflow record. When the last prime data record on the track has been rewritten, the new overflow record is written in the overflow area, the track index normal and overflow entries and the COCR are written on DASD, and control returns to the problem program.

If the prime data in core option has not been specified as an ISMOD macro parameter, a test for blocked records is made. If the file contains unblocked records, the record previously found on the search key equal/high is reread to get the key field. If it is a duplicate key, a switch is set on in the DTFIS table indicating a duplicate key has been sensed and a return to the problem program is made. If there are no duplicate keys, the user's key and data are written from the work area, WORKL, onto the DASD file. The record in the I/O area, IOAREAL, replaces the user's record in the work area. The next record on the track replaces the one in the I/O area. This process is repeated until the end of track is reached.

If the end-of-file (EOF) record is read during the process of shifting the records over one record position, this routine writes the last record over the EOF record and then writes a new EOF record (see Figures 84, 92, 93).

If the file contains blocked records, this routine reads the block of records (or as many as fit in the I/O area if IOAREAL was increased for reading and writing more than one record at a time) into IOAREAL. The key field within each logical record is analyzed to determine the correct position in which to insert the new record. If there is duplication of keys, a switch is set on in the DTFIS table and control returns to the problem program.

If the key of the record to be inserted (contained in WORKL) is low, it is exchanged with the record presently in the block. This procedure continues with each succeeding record in the block until the last record is moved into the work area. The key field of the DASD record is then updated to reflect the highest key in the block. If the size of IOAREAL has been increased, succeeding blocks in the I/O area are also updated. The block (or blocks) is then written back onto DASD. The remaining blocks on the track are similarly processed until the last logical

record on the track is moved into WORKL. This record is then set up as an overflow record with the correct sequence-link field added and written in the overflow area. The sequence-link field for the new overflow record is taken from the track index overflow entry. The indexes are updated, and cortrcl returns to the problem program for the next record to be added. If the overflow area is full, this information is indicated to the user in the DTF table entry labeled Filename.C.

The track index normal entry key field is updated to the key of the new last record, the track index overflow entry data field is updated tc the address of the new overflow entry (that entry has the lowest key for the overflow for that track), and the COCR is updated. These records are written on the DASD file before control returns to the problem program.

If the last block in the prime data area is padded, the last record to be shifted is included in that block. If the EOF record is read during the process of shifting the records one record position, the last record is written as a new block and a new EOF record is written before returning control to the problem program.

Add to the Overflow Area: This routine computes the new overflow record address and reads the overflow chain to get the address of the record with the next highest key. This address is stored in the sequence-link field of the new record. The new overflow reccrd is then written in either the cylinder overflow area or independent overflcw area (see Figure 91). If these areas are full, this condition is indicated to the user in the DTFIS table entry labeled Filename.C. Each time an overflow record is added to the independent overflow area, an EOF record is written to maintain the integrity of the indexed sequential file (see Figure 94). The next overflow record followed by an EOF record overlays the previous EOF record.

If the new overflow record has the lowest key in the overflow chain, its address is used to build a new track index overflow entry. The new overflow entry is then written on the DASD file, and control returns to the problem program. If a cylinder overflcw condition occurs, the updated COCR (cylinder overflow control record) is written on DASD before control returns to the problem program.

If the new overflow record does not have the lowest key, the sequence-link field of the record with the next lower key is updated to contain the address of the new overflow record. This overflow record is then rewritten cn DASD and the COCR is updated (see Figures 88-90). Control returns to the problem program.

EOF Add: This routine first determines if the last prime data track is full. If the last prime data track is not full, the new record is inserted on it. If the file contains blocked records and the record can fit in the last block, the block is read and the new record is inserted.

If the file is not blocked, or if it is blocked and the last block is full, a new last prime data record address is stored and the new record is written at that address. A new EOF record is then written.

If the last prime data track is full, the new record is inserted in the overflow area. The new overflow record address is computed and the record is written in the overflow area.

If an overflow chain is present, the next lower record in the chain is found and the address of the new record is moved to the sequence-link field of the next lower record.

If no overflow chain is present, the address of the new overflow record is moved to the track index overflow entry. The track index overflow entry is then written with the new high key. The master index (if present) and the cylinder index are updated with the new high key. A test for the cylinder index in core option is then made. If it has not been specified, control returns to the problem program. If the cylinder index in core option has been specified, the new key is inserted into the appropriate index in core entry before returning control to the problem program.

ISAM ADDRTR: WRITE Macro, KEY, Chart KF

Objective: To perform the random retrieval output function for an indexed sequential file.

Entry: From WRITE, KEY macro expansion.

Exit: To the problem program via register 14.

Method: This routine first sets the write bit in the DTFIS table. If then tests for an uncorrectable DASD error, wrong length record error, or no record found error. If any of these errors exist, the no-record-found bit is set on in the DTF table, and control returns to the problem program.

If there are no errors, the status byte in the DTF table is reset, and pointers to the DTF table are initialized. This routine then picks up the count field of the record as saved by the read routine, the address of WORKR, and the address of the logical record within the I/O area. The record, or block of records, is moved to the I/O area from WORKR (if specified).

The CCW chain to write records is then built (see Figure 81).

If the entry to be written is not an overflow entry, the byte count field in the write and verify CCWs is modified to the block length from the DTF table. This routine then issues the EXCP to write the record, and returns control to the problem program without issuing a WAIT. The WAIT function is left to the WAITF macro, which must be issued before the user can continue processing.

ISAM ADDRTR: WRITE Macro, NEWKEY, Charts EE-EF

Objective: To perform the necessary initialization to add a record to a file.

Entry: From the WRITE, NEWKEY macro expansion.

Exit: To the problem program via linkage register 14.

Method: After initializing the pointers to the three parts of the DTFIS table, this routine gets the starting address of the highest level index, builds a CCW chain to search the highest level index (see Figure 82) and tests for ERREXT=YES. If ERREXT is specified, additional error conditions can be returned to the problem program. This allows the user greater flexibility in attempting to continue processing. The channel program is executed and a wait for I/O completion is made. It then tests the F code of the index level pointer to determine if the next search is of the cylinder or track index. The F code refers to the index level just searched. If the master index was just searched, the next search is on the cylinder index. See Figure 20 for a description of the F code.

If the F code indicates a dummy chained entry, the search of the master, cylinder or track index continues. If the index level pointer did not indicate a dummy chained entry, a test for an inactive or dummy end entry is made. If an inactive or dummy end entry is indicated, the EOF add indicator is set on in the DTFIS table, a CCW chain is built to read the last track index entries, the channel program to bypass the last of the track index entries is executed, a wait for the I/O operation to be completed is made, and control returns to the problem program. Processing continues with the record following the last key.

If an inactive or dummy end entry is not indicated, a test for the presence of a master index is made. If the master index is not present, indicating the cylinder index was just searched, a search of the track index is performed, and a return to the problem program is made.

If the master index is present, a test is made to determine if the cylinder index in core option was specified as an ISMOD macro parameter. If it was not specified, an EXCP is issued to search the cylinder index, followed by a wait for I/O completion, an EXCP to search the track index, a wait for I/O completion, and a return to the problem program. If the cylinder index in core option was specified, a search of the track index is performed, and a return to the problem program is made.

Any tracks which have been held during update are released before control returns to the user.

CCW Builder Control Code[1]	CCW Built	Function
7461	X'69', &KEYARG, CC and SLI, Key Length	Search key equal or high the master/ cylinder index. Key supplied by user in the DTFIS table.
0C6B 0E6B	X'08', Pointer to *+16, CC X'08', Pointer to *+24, CC and SLI, 5	TIC to *+16 for non-RPS. TIC to *+24 for RPS.
D17B	X'1A', [2]&Filename.W, CC, SLI, and SKIP, 5	Read home address into random/sequential retrieval work area in DTFIS table.
F081 0E6B	X'23', SECARG=0, CC, 1[8] X'08'	Set Sector for start of track. Set when CYLINDEX is a non-RPS device.
536C	X'92', &IOAREAR, CC and SLI, 10	Read count (multiple-track) into IOAREAR.
7461	X'69', &KEYARG, CC and SLI, Key Length	Search key equal or high the master/ cylinder index. Key supplied by user in DTFIS table.
046B	X'08', Pointer to *-16, CC and SLI, 5	TIC to *-16.
110C	X'06', [2]&Filename.W, 00, 10	Read data (10-byte index level pointer) into random/sequential retrieval area in DTFIS table.

[1--8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 76. Channel program builder for ADDRTR -- CCW chain built to search master-cylinder index for random retrieve function.

CCW Builder Control Code[1]	CCW Built	Function
D36B	X'1A', &IOAREAR, CC and SLI, 5	Read home address into IOAREAR.
F041	X'23', SECARG=0, CC, 1[8]	Set Sector for start of track.
9461	X'E9', &KEYARG, CC and SLI, Key Length	Search key equal or high (multiple-track) track index. Key supplied by user in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
110C	X'06', [2]&Filename.W, 00, 10	Read data (10-byte index level pointer) into random/sequential retrieval area in DTFIS table.

[1--8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 77. Channel program builder for ADDRTR -- CCW chain built to search track index for random retrieve function.

CCW Bfilder Contrcl Code[1]	CCW Built	Function
7461	X'69', &KEYARG, CC and SLI, Key Length	Search key equal or high in prime data area. Key supplied by user in DTFIS table.
0C4B	X'08', Pointer to *+16, CC, 5	TIC to *+16.
F001	X'23', SECARG=0, CC, 1[8]	Set Sector for start of track.
D17B	X'1A', [2]&Filename.W, CC, SLI, and SKIP, 5	Read hcme address into random/sequential retrieval area in DTFIS table.
406C	X'12', [2]&Filename.S+3, CC and SLI, 10	Read count into common seek/search area in DTFIS table.
6461 or 7461	X'29' or X'69', &KEYARG, CC and SLI, Key Length	If records are unblocked, search key equal in prime data area. If records are blocked, search key equal or high in prime data area. Key supplied by user in DTFIS table.
044B	X'08', Pointer to *-16, CC, 5	TIC to *-16.
1302	X'06', &IOAREAR, 00, Block Length	Read data (block) containing starting record into IOAREAR.

[1--8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 78. Channel program builder for ADDRTR -- CCW chain built to find record in prime data area (unshared track) for random retrieve function.

CCW Builder Control Code[1]	CCW Built	Function
F101	X'23', SECARG=1, CC, 1[8]	Set Sector for prime data area.
804B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the prime data area using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
064B	X'08', Pointer to *-8 CC, 5	TIC to *-8.
406C	X'12', [2]&Filename.S+3, CC and SLI, 10	Read count into common seek/search area in DTFIS table.
6461 or 7461	X'29' or X'69', &KEYARG, CC and SLI, Key Length	If records are unblocked, search key equal the prime data area. If records are blocked, search key high or equal the prime data are. Key supplied by user in DTFIS table.
044B	X'08', Pointer to *-16, CC, 5	
1302	X'06', &IOAREAR, 00, Block Length	Read data (block) containing record into IOAREAR.

[1--8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 79. Channel program builder for ADDRTR -- CCW chain built to find record in prime data area (shared track) for random retrieve function.

CCW Builder Control Code[1]	CCW Built	Function
F121	X'23', SECARG=1, CC, 1[8]	Set Sector for overflow chain.
804B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the overflow chain using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
064B	X'08', Pointer to *-8, CC, 5	TIC to *-8.
6461	X'29', &KEYARG, CC and SLI, Key Length	Search key equal the overflow chain. Key supplied by user in DTFIS table.
116C	X'06', [2]&Filename.W, SLI, 10	Read data (10-byte sequence link field) into random/sequential retrieval area in DTFIS table. This CCW is executed when the required overflow record is not found in the overflow chain.
1303	X'06', &IOAREAR, 00, Record Length + 10	Read data (sequence link field plus logical record) into IOAREAR. This CCW is executed when the matching key is found in the overflow chain.

[1--8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 80. Channel program builder for ADDRTR -- CCW chain built to find record in overflow chain for random retrieve function.

CCW Builder Control Code[1]	CCW Built	Function
F121	X'23', SECARG=1, CC, 1[8]	Set Sector for prime data area.
804B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal prime data area using pointer (CCHHR) in common seek/ search area in DTFIS table.
064B	X'08', Pointer to *-8, CC, 5	TIC to *-8.
B3C3	X'05', &IOAREAR, CC, Record Length + 10	Write data from IOAREAR.
F131	X'23', SECARG=1, CC, 1[8]	Set Sector for prime data area.
804B	X'31, [2]&Filename.S+3, CC, 5	Search identifier equal the prime data area, using pointer (CCHHR) in common seek/search area in DTFIS table.
064B	X'08', Pointer to *-8, CC, 5	TIC to *-8
1333	X'06', &IOAREAR, SLI and SKIP, Record Length + 10	Read data to verify record just written. Information is not transferred to main storage.

[1--8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 81. Channel program builder for ADDRTR -- CCW chain built to write record for random retrieve function.

CCW Builder Control Code[1]	CCW Built	Function
7961	X'69', &KEYARG, CC and SLI, Key Length	Search key equal or high the master/ cylinder index. Key supplied by user in the DTFIS table.
0C6B	X'08', Pointer to *+16, CC and SLI, 5	TIC to *+16.
F08[1]	X'23', SECARG=0, CC, 1[8]	Set Sector for start of track.
D17B	X'1A', [2]&Filename.D+8, CC, SLI and SKIP, 5	Read home address into work area for the current track index normal entry count field in the DTFIS table.
516C	X'92', [2]&Filename.D+8, CC and SLI, 10	Read count (multiple-track) into work area for the current track index normal entry count field in the DTFIS table.
7961	X'69', &KEYARG, CC and SLI, Key Length	Search key equal or high the master/ cylinder index. Key supplied by user in the DTFIS table.
046B	X'08', Pointer to *-16, CC and SLI, 5	TIC to *-16.
150C	X'06', [2]&Filename.D+40, 00, 10	Read data (next 10-byte index level pointer) into work area for track index normal entry data field in DTFIS table.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 82. Channel program builder for ADDRTR -- CCW chain built to search master cylinder index for add function.

CCW Builder Control Code[1]	CCW Built	Function
F041	X'23', SECARG=0, CC, 1[8]	Set Sector for start of track.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the track index seek/search area.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
106C	X'06', [2]&Filename.D, CC and SLI, 10	Read data (COCR record) into the cylinder overflow record (COCR) area.
516C	X'92', [2]&Filename.D+8, CC and SLI, 10	Read count (multiple-track) into work area for the current track index normal entry count field in the DTFIS table.
7941	X'69', &KEYARG,CC, Key Length	Search key equal or high the track index. Key supplied by user in the DTFIS table.
046B	X'08', Pointer to *-16, CC and SLI, 5	TIC to *-16.
156C	X'06', [2]&Filename.D+40, CC and SLI, 10	Read data (next 10-byte pointer to prime data record) into work area for track index normal entry data field in DTFIS table.
526C	X'92', [2]&Filename.D+16, CC and SLI, 10	Read count (multiple-track) into work area for current track index overflow entry count field in DTFIS table.
1D0C	X'06', [2]&Filename.W, 00, 10	Read data (10-byte overflow entry) into random/sequential retrieval work area.

[1] -- [8] See nctes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 83. Channel program builder for ADDRTR -- CCW chain built to search track index for add function.

CCW Builder Control Code[1]	CCW Built	Function
F101	X'23', SECARG=1, CC 1[8]	Set Sector for last data record or start of new track.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for the last prime data record address using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
342C	X'10', [2]&Filename.D+32, SLI, 10	Write count, key and data of EOF record located in current overflow record count field in DTFIS table.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 84. Channel program builder for ADDRTR -- CCW chain built to write new EOF record for add function.

CCW Builder Control Code[1]	CCW Built	Function
F101	X'23', SECARG=1, CC, 1[8]	Set Sector for current prime data record.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier the prime data area using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
436C	X'12', [2]&Filename.D+24, CC and SLI, 10	Read count for current prime data record.
7941	X'69', &KEYARG, CC, Key Length	Search key equal or high the prime data area. Key supplied by user in DTFIS table.
046B	X'08', Pointer to *-16, CC and SLI, 5	TIC to *-16.
1B02	X'06', Address of IOAREAL+8 +KEYLEN, 00, Block Size	Read data (prime data block) into IOAREAL+8+Key Length.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 85. Channel program builder for ADDRTR -- CCW chain built to find prime data record for add function.

CCW Builder Control Code[1]	CCW Built	Function
F041	X'23', SECARG=0, CC, 1[8]	Set Sector for start of track.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the track index using pointer, CCHHR, in common seek/ search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
B06C	X'05', [2]&Filename.D, CC and SLI, 10	Rewrite COCR located in cylinder overflow control record work area in DTFIS table.
F141	X'23' SECARG=1, CC, 1[8]	Set Sector for the track index normal entry required.
E14B	X'B1', [2]&Filename.D+8, CC, 5	Search identifier equal (multiple-track) for the pointer, CCHHR, in the normal entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
2A45	X'0D', Address of IOAREAL+8, CC, Key Length + 10	Rewrite track index normal entry located at IOAREAL+8.
E24B	X'B1', [2]&Filename.D+16, CC, 5	Search identifier equal (multiple-track) for the pointer, CCHHR, in the overflow entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
BDCC	X'05', [2]&Filename.W, CC and DC, 10	Rewrite overflow entry located in random/ sequential retrieval work area.
F251	X'23', SECARG=2, CC, 1[8]	Set Sector for the track index overflow entry required.
F240	X'22', SECARG=2, CC, 1[8]	Read Sector for the next track index overflow entry.
824B	X'31', [2]&Filename.D+16, CC, 5	Search identifier equal for the pointer, CCHHR, in the overflow entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
1D3C	X'06', [2]&Filename.W, SLI and SKIP, 10	Read data to verify record just written. Information is not transferred to main storage.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 86. Channel program builder for ADDRTR -- CCW chain built to rewrite track index entry for add function.

CCW Builder Control Code[1]	CCW Built	Function
F041	X'23', SECARG=0, CC, 1[8]	Set Sector for start of track.
8C4B	X'13', [2]&Filename.S+3, CC, 5	Search identifier equal for R0 using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
B06C	X'05', [2]&Filename.D, CC and SLI, 10	Write data (updated COCR) from the cylinder overflow control record (COCR) area in the DTFIS table.
F141	X'23', SECARG=1, CC, 1[8]	Set Sector for current track index normal entry.
E14B	X'B1', [2]&Filename.D+8, CC, 5	Search identifier equal (multiple-track) the track index using the pointer, CCHHR, in the work area for the current track index normal entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
BDCC	X'05', [2]&Filename.W, CC and DC, 10	Write data (track index overflow entry) from the random/sequential retrieval work area.
F151	X'23', SECARG=1, CC, 1[8]	Set Sector for current track index normal entry.
814B	X'31', [2]&Filename.D+8, CC, 5	Search identifier equal the track index using the pointer, CCHHR, in the work area for the current track index normal entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
1D3C	X'06', [2]&Filename.W, SLI and SKIP, 10	Read data to verify record just written. Information is not transferred to main storage.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 87. Channel program builder for ADDRTR -- CCW chain built to write track index entry for add function.

CCW Builder Control Code[1]	CCW Built	Function
F041	X'23', SECARG=0, CC, 1[8]	Set Sector for start of track.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for R0 using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
B02C	X'05', [2]&Filename.D, SLI, 10	Write data (updated COCR) from the cylinder overflow control record area in the DTFIS table.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 88. Channel program builder for ADDRTR -- CCW chain built to write COCR for add function.

CCW Builder Control Code[1]	CCW Built	Function
F121	X'23', SECARG=1, CC, 1[8]	Set Sector for previously low overflow record.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
AA07	X'0E', Address of IOAREAL+8, 00, Key Length + Record Length + 10	Read key and data of previously low overflow record into IOAREAL+8.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 89. Channel program builder for ADDRTR -- CCW chain built to read previous overflow record for add function.

CCW Builder Control Code[1]	CCW Built	Function
F121	X'23', SECARG=1, CC, 1[8]	Set Sector for previously low overflow record.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
2A47	X'0D', Address of IOAREAL+8, CC, Key Length + Record Length + 10	Write key and data of previously low overflow record located at IOAREAL+8.
F131	X'23', SECARG=1, CC, 1[8]	Set Sector for previously low overflow record.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
AA37	X'0E', Address of IOAREAL+8, SLI and SKIP, Key Length + Record Length + 10	Read key and data to verify record just written. Information is not transferred to main storage.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 90. Channel program builder for ADDRTR -- CCW chain built to write previous overflow record for add function.

CCW Builder Control Code[1]	CCW Built	Function
F121	X'23', SECARG=1, CC, 1[8]	Set Sector for last overflow record.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for last overflow record address using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
37C9	X'1D', Address of IOAREAL, CC and DC, Key Length + Record Length + 18	Write count, key and data of new overflow record located at IOAREAL.
F131	X'23', SECARG=1, CC, 1[8]	Set Sector for last overflow record.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for last overflow record using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
C739	X'1E', Address of IOAREAL, SLI and SKIP, Key Length + Record Length + 18	Read count, key and data to verify record just written. Information is not transferred to main storage.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 91. Channel program builder for ADDRTR -- CCW chain built to write new overflow record for add function.

CCW Builder Control Code[1]	CCW Built	Function
F701	X'23', SECARG=7, CC, 1[9]	Set Sector for last overflow record.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for present EOF record address minus one using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
37C8	X'1D', Address of IOAREAL, CC and DC, Key Length + Block Size + 8	Write count key and data of new record to be added located at IOAREAL.
F111	X'23', SECARG=1, CC, 1[9]	Set Sector for last overflow record.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for present EOF record address minus one using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
C738	X'1E', Address of IOAREAL, SLI and SKIP, Key Length + Block Size + 8	Read count, key and data to verify record just written. Information is not transferred to main storage.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 92. Channel program builder for ADDRTR -- CCW chain built to write over EOF record (blocked records) for add function.

CCW Builder Control Code[1]	CCW Built	Function
F101	X'23', SECARG=1, CC, 1[8]	Set Sector for present EOF record address minus one.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for present EOF record address minus one using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
37C8	X'1D', Address of IOAREAL, CC and DC, Key Length + Block Size + 8	Write count, key and data of new record to be added, located at IOAREAL.
F111	X'23', SECARG=1, CC, 1[8]	Set Sector for present EOF record address minus one.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for present EOF record address minus one using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
C738	X'1E', Address of IOAREAL, SLI and SKIP, Key Length + Block Size + 8	Read count, key and data to verify record just written. Information is transferred to main storage.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 93. Channel program builder for ADDRTR -- CCW chain built to write over EOF record (unblocked records) for add function.

CCW Builder Control Code[1]	CCW Built	Function
F121	X'23', SECARG=1, CC, 1[8]	Set Sector for present EOF record address minus one.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for present EOF record address minus one using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
37AC	X'1D', Address of IOAREAL, DC and SLI, 10	Write count, key and data of EOF record located at IOAREAL.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 94. Channel program builder for ADDRTR -- CCW chain built to write EOF record in independent overflow area for add function.

CCW Builder Control Code[1]	CCW Built	Function
F041	X'23', SECARG=0, CC, 1[8]	Set Sector for track start.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the track index using the pointer (CCHHR) in the common seek/search area.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
106C	X'06', [2]&Filename.D, CC and SLI, 10	Read data (COCR record) into the cylinder overflow control record (COCR) area.
F141	X'23', SECARG=1, CC, 1[8]	Set Sector for last normal track index record.
E14B	X'B1', [2]&Filename.D+8, CC, 5	Search identifier equal (multiple-track) the track index for the last normal entry using information in the work area for the current track index normal entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
154C	X'06', [2]&Filename.D+40, CC, 10	Read data (last track index normal entry) into work area for track index normal entry data field.
526C	X'92', [2]&Filename.D+16, CC and SLI, 10	Read count (multiple-track) of last track index overflow entry into work area for the current track index overflow entry count field.
1D0C	X'06', [2]&Filename.W, 00, 10	Read data (last track index overflow entry) into random/sequential retrieval work area.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 95. Channel program builder for ADDRTR -- CCW chain built to read last track index entry for add function.

CCW Builder Control Code[1]	CCW Built	Function
F121	X'23', SECARG=1, CC, 1[8]	Set Sector for overflow chain.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the overflow chain using the pointer (CCHHR) in the common seek/search area.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
AA07	X'0E', Address of IOAREAL+8, 00, Key Length + Record Length + 10	Read key and data of overflow record into IOAREAL+8.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 96. Channel program builder for ADDRTR -- CCW chain built to read overflow record for add function.

CCW Builder Control Code[1]	CCW Built	Function
F101	X'23', SECARG=1, CC, 1[6]	Set Sector for last prime data record.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for last prime data record using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
1B02	X'06', Address of IOAREAL+8+KEYLEN, 00, Block Size	Read block into IOAREAL + 8 + KEYLEN.

[1] -- [6] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 97. Channel program builder for ADDRTR -- CCW chain built to read last prime data record for add function.

CCW Builder Control Code[1]	CCW Built	Function
F101	X'23', SECARG=1, CC, 1[6]	Set Sector for last prime data record.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for last prime data record using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
2AC6	X'0D', Address of IOAREAL+8, CC and DC, Key Length + Block Size	Write key and data of prime data block located at IOAREAL+8.
F111	X'23', SECARG=1, CC, 1[6]	Set Sector for last prime data record.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for last prime data record using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
AA36	X'0E', Address of IOAREAL+8, SLI and SKIP, Key Length + Block Size	Read key and data to verify record just written. Information is not transferred to main storage.

[1] -- [6] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 98. Channel program builder for ADDRTR -- CCW chain built to write block of prime data records and verify for add function.

CCW Builder Control Code[1]	CCW Built	Function
F141	X'23', SECARG=1, CC, 1[8]	Set Sector for last track index.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal for last track index entry using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
2A45	X'0D', Address of IOAREAL+8, CC, Key Length + 10	Write key and data of track index normal entry located at IOAREAL+8.
E24B	X'B1', [2]&Filename.D+16, CC, 5	Search identifier equal (multiple-track) the track index for the last overflow entry using the count for the current track index overflow entry.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
2845	X'0D', Address of WORKL, CC, Key Length + 10	Write key and data of track index overflow entry located at WORKL.
F240	X'22', SECARG=2, CC, 1[8]	Obtain Sector for last record processed.
F251	X'23', SECARG=2, CC, 1[8]	Set Sector for last record processed.
824B	X'31', [2]&Filename.D+16, CC, 5	Search identifier equal the track index for the last overflow entry using the count for the current track index overflow entry.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
A835	X'0E', Address of WORKL, SLI and SKIP, Key Length + 10	Read key and data to verify record just written. Information is not transferred to main storage.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 99. Channel program builder for ADDRTR -- CCW chain built to write track index entry for add function.

CCW Builder Control Code[1]	CCW Built	Function
F181	X'23', SECARG=1, CC, 1[8]	Set Sector for master/cylinder index.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the master/cylinder index using the pointer, CCHHR, in the common seek/search area in the DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
150C	X'06', [2]&Filename.D+40, 00, 10	Read data (index entry) into work area for track index normal entry data field.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 100. Channel program builder for ADDRTR -- CCW chain built to read index entry for add function.

CCW Builder Control Code[1]	CCW Built	Function
F181	X'23', SECARG=1, CC, 1[8]	Set Sector for master/cylinder index.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the master/cylinder index using pointer, CCHHR, in common seek/search area in DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
2A45	X'0D', Address of IOAREAL+8, CC, Key Length + 10	Write key and data of master/cylinder index entry located at IOAREAL+8.
F191	X'23', SECARG=1, CC, 1[8]	Set Sector for master/cylinder index.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the master/cylinder index using pointer, CCHHR, in common seek/search area in DTFIS table.
AA35	X'0E', Address of IOAREAL+8, SLI and SKIP, Key Length + 10	Read key and data to verify record just written. No information is transferred to main storage.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 101. Channel program builder for ADDRTR -- CCW chain built to write index entry for add function.

CCW Builder Control Code[1]	CCW Built	Function
F041	X'23', SECARG=0, CC, 1[8]	Set Sector for track start.
8C4B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the track index using the pointer, CCHHR, in the common seek/search area in the DTFIS table.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
B06C	X'05', [2]&Filename.D, CC and SLI, 10	Write data (COCR) from the cylinder overflow control record work area in DTFIS table.
F141	X'23', SECARG=1, CC, 1[8]	Set Sector for the current track index overflow entry.
E24B	X'B1', [2]&Filename.D+16, CC, 5	Search identifier equal (multiple-track) the track index using the pointer, CCHHR, in the work area for current track index overflow entry count field.
066B	X'08', Pointer to *-8, CC and SLI, 5	TIC to *-8.
AA35	X'0E', Address of IOAREAL+8, SLI and SKIP, Key Length + 10	Read key and data to verify record just written. Information is not transferred to main storage.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 102. Channel program builder for ADDRTR -- CCW chain built to write track index overflow entry for add function.

CCW Parameter[3]	CCW Built	Function
	X'23', SECARG=1, CC, 1[8]	Set Sector for prime data area.
0540	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the prime data area using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
	X'08', Pointer to *-8, CC, 0	TIC to *-8.
	X'05',[4] &IOAREAS, CC,[5] Block Length[6]	Write data (block) onto prime data area.
	X'03', SECARG=4, CC, 1	NOOP
	X'23', SECARG=1, CC, 1[8]	Set Sector to verify record written.
	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal to verify write operation.
	X'08', Pointer to *-8, CC, 0	TIC to *-8.
	X'06',[4] &IOAREAS, SKIP, Block Length[6]	Read data to verify write operation.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 103. Channel program builder for ADDRTR -- CCW chain built to write records for sequential retrieve function.

CCW Parameter[3]	CCW Built	Function
	X'23', SECARG=1, CC, 1[8]	Set Sector for track index area.
0601	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the track index area, using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
	X'08', Pointer to *-8, CC, 0	TIC to *-8.
	X'06', [2]&Filename.W, CC,[5] 10	Read data (10-byte index level pointer) into random/sequential retrieval area in DTFIS table.
	X'22', SECARG=3, CC, 1[8]	Read Sector for next track index area.
	X'23', SECARG=1, CC, 1[8]	Set Sector to verify record read.
	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal to verify read operation.
	X'08', Pointer to *-8, CC, 0	TIC to *-8.
	X'06', &IOAREAS, SKIP, Block Size	Read data to verify read operation.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 104. Channel program builder for ADDRTR -- CCW chain built to search track index for sequential retrieve function.

CCW Parameter[3]	CCW Built	Function
	X'23', SECARG=1, CC, 1[8]	Set Sector for prime data area.
0600	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the prime data area using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
	X'08', Pointer to *-8, CC, 0	TIC to *-8.
	X'06',[4] &IOAREAS, CC,[5] Block Length[6]	Read data into IOAREAS.
	X'22', SECARG=2, CC, 1[8]	Read Sector for next prime data area.
	X'23', SECARG=1, CC, 1[8]	Set Sector to verify record read.
	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal to verify read operation.
	X'08', Pointer to *-8, CC, 0	TIC to *-8.
	X'06', &IOAREAS, SKIP, Block Length[6]	Read data to verify read operation.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Note: The shaded areas indicate CCWs built for RPS only.

Figure 105. Channel program builder for ADDRTR -- CCW chain built to read record for sequential retrieve function.

Label	CCW Builder[7] Control Code	CCW Built	Function
	7441	X'69', &KEYARG, CC, Key Length	Search key equal or high the master/ cylinder index. Key supplied by user in DTFIS table.
	0C40	X'08', Pointer to *+16, CC, Record Length	TIC to *+16.
	B04B	X'1A', [2]&Filename.S+3, CC, 5	Read home address into common seek/ search area in DTFIS table.
	506B	X'92', [2]&Filename.S+3, CC and SLI, 5	Read count (multiple-track) - CCHHR - into common seek/search area in DTFIS table.
	7441	X'69', &KEYARG, CC, Key Length	Search key equal or high the master/ cylinder index. Key supplied by user.
	0440	X'08', Pointer to *-16, CC, Record Length	TIC to *-16.
	110C	X'06', [2]&Filename.W, 00, 10	Read data (10-byte index level pointer) into random/sequential retrieval area in DTFIS table. The data field is then moved from the random/sequential retrieval area to the common seek/search area for the next search.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Figure 106. Channel program builder for ADDRTR -- CCW chain built by $$BSETL (1) to search MI for sequential retrieve function.

Label	CCW Builder[7] Control Code	CCW Built	Function
	806C	X'31', [2]&Filename.S+3, CC and SLI, 10	Search identifier equal the track index using the 10-byte pointer in the common seek/search area.
	0640	X'08', Pointer *-8, CC, Record Length	TIC to *-8.
	126C	X'06', &IOAREAS, CC and SLI, 10	Read data (10-byte track index pointer) into IOAREAS, input/output area for sequential retrieval supplied by user.
	506B	X'92', [2]&Filename.S+3, CC and SLI, 5	Read count (multiple-track) - CCHHR - into common seek/search area in DTFIS table.
	7441	X'69', &KEYARG, CC, Key Length	Search key equal or high the track index. Key supplied by user.
	0240	X'08', Pointer to *-24, CC, Record Length	TIC to *-24.
	110C	X'06', [2]&Filename.W, 00, 10	Read data (10-byte pointer) into random/sequential retrieval area in DTFIS table. The data field is then moved from the random/sequential retrieval area to the common seek/ search area for the next search.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Figure 107. Channel program builder for ADDRTR -- CCW chain built by $$BSETL (1) to search TI for sequential retrieve function.

Label	CCW Builder[7] Control Code	CCW Built	Function
STRI1	804B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the prime data area using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
	066B	X'08', Pointer to *-8 ,CC and SLI, 5	TIC to *-8.
	416C	X'12', [2]&Filename.W, CC and SLI, 10	Read count into common seek/search area in the DTFIS table.
	6441 or 7441	X'29' or X'69', &KEYARG, CC, Key Length	If KEY is specified in the SETL macro and/or records are unblocked, this CCW searches key equal the prime data area. If GKEY is specified in the SETL macro and/or records are blocked, this CCW searches key equal or high the prime data area for the starting record.
	046B	X'08', Pointer to *-16, CC and SLI, 5	TIC to *-16.
	1202	X'06', &IOAREAS, 00, Block Size	Read data (block) containing starting record into IOAREAS.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Figure 108. Channel program builder for ADDRTR -- CCW chain built by $$BSETL (1) to find first record in prime data area for sequential retrieve function.

Label	CCW Builder Control Code[7]	CCW Built	Function
STRI3	804B	X'31', [2]&Filename.S+3, CC, 5	Search identifier equal the overflow chain using the pointer (CCHHR) in the common seek/search area in the DTFIS table.
	0640	X'08', Pointer to *-8, CC, Record Length	TIC to *-8.
	6441 or 7441	X'29' or X'69', &KEYARG, CC, Key Length	If KEY is specified in the SETL macro, this CCW searches key equal the overflow chain for the starting record. If GKEY is specified in the SETL macro, this CCW searches key equal or high the overflow chain for the starting record.
	112C	X'06', [2]&Filename.W, SLI, 10	Reads data (10-byte sequence link field) into random/sequential retrieval area in DTFIS table. This CCW is executed when the required overflow record is not found in the overflow chain.
	1203	X'06', &IOAREAS, 00, Record Length +10	Read data (sequence link field plus starting record) into IOAREAS. This CCW is executed when the matching key is found in the overflow chain.

[1] -- [8] See notes 1 through 8 in Figures 110 and 111.

Figure 109. Channel program builder for ADDRTR -- CCW chain built by $$BSETL (1) to find first record in overflow chain for sequential retrieve function.

Note 1:

The CCW builder control code references information in the DTF DSECT section of the ISMOD assembly.

The first character of the control code references an operation code at IJHCSTRI.

The second character of the control code references a data area at either IJHCASAD for random retrieve function or IJHAHRAA for add function.

The third character of the control code references the following information:

Control Character	CCW Flag Field	Meaning
0	X'00'	End of CCW Chain
2	X'20'	SLI (Suppress Length Indicator)
3	X'30'	SLI and SKIP (Suppress data transfer)
4	X'40'	CC (Command Chaining)
6	X'60'	CC and SLI
7	X'70'	CC, SLI, and SKIP
A	X'A0'	SLI and DC (Data Chaining)
C	X'C0'	CC and DC

The fourth character of the control code references a byte count (length) field at IJHCRESZ.

Note 2:

&Filename = DTF name supplied by user.

&Filename.X = X is suffix supplied by DTFIS for unique DTF labels.

Note 3:

The CCW parameter is located in the ISMOD assembly.

The first byte of the parameter is the command code.

The second byte of the parameter contains flags with the exception of the chain to search the track index. In this case, the second byte is an indicator to the channel program builder that the CCW chain is to search the track index.

Note 4: If the file contains unblocked records, the command code is modified to Read Key and Data, cr Write Key and Data.

Note 5: If the verify option has not been specified, the command chaining bit is not set on.

Note 6: If the file contains unblocked records, the byte count field contains the physical record length plus key length.

Figure 110. Charnel program builder for ADDRTR -- notes 1-6.

Note 7:

The CCW chains are built by the B-transients, $$BSETL and $$BSETL1. The CCW builder control code references information in the $$BSETL and $$BSETL1 assemblies.

The first character of the control code references an operation code at IJHROP.

The second character of the control code references a data area at IJHARA.

The third character of the control code references the following information:

Control Character	CCW Flag Field	Meaning
0	X'00'	End of CCW chain
2	X'20'	SLI (Suppress Length Indicator)
4	X'40'	CC (Command Chaining)
6	X'60'	CC and SLI

The fourth character of the control code references a byte count (length) field at RELNT.

Note 8:

If the first character of the CCW builder control code is an "F", it indicates that this is a sector control type of CCW.

The second character of the control code is a displacement into the sector arguments fields in the DTFIS extension to indicate which sector argument field is to be used with this CCW.

The third character of the control code references the following information:

Control Character	Meaning
1	Suppress sector calculation.
2	Prime data overflow CCW chain.
4	Track index CCW chain.
8	Cylinder index/master index CCW chain.

The fourth character of the control code references the following information:

Control Character	CCW Command Code	Meaning
0	X'22'	Read Sector CCW
1	X'23'	Set Sector CCW

When building a sector control type of CCW, the command chaining flag bit is always turned on in the CCW, and the CCW byte count field is always set to 1.

Figure 111. Channel program builder for ADDRTR -- notes 7-8.

ISAM INITIALIZATION AND TERMINATION PROCEDURES

When files are opened for indexed sequential (DTFIS) and the file is on more than one volume, all volumes must be opened, before processing of the file begins. All labels are checked/written at the initial file open.

Job control accepts label information previously supplied on VOL, DLAB, and XTENT statements (not valid for the 3330 family) as well as information on the simplified DLBL and EXTENT statements. Job control reads the DASD label information supplied on these statements, and stores the information in the label information area. The open monitor logical transient, $$BOPEN2, reads the DASD label information into the label save area in main storage for use by the ISAM open/close logical transients.

The extents in the DASD label information record are checked for overlap on each other. If overlap exists, a message is issued, and the job is canceled. Checks are made to determine if all the correct packs are mounted, if serial numbers match and if any extent limits overlap the VTOC. The extents for the master index (if specified) and cylinder index are checked to determine if they are contiguous, and the limits are saved. The

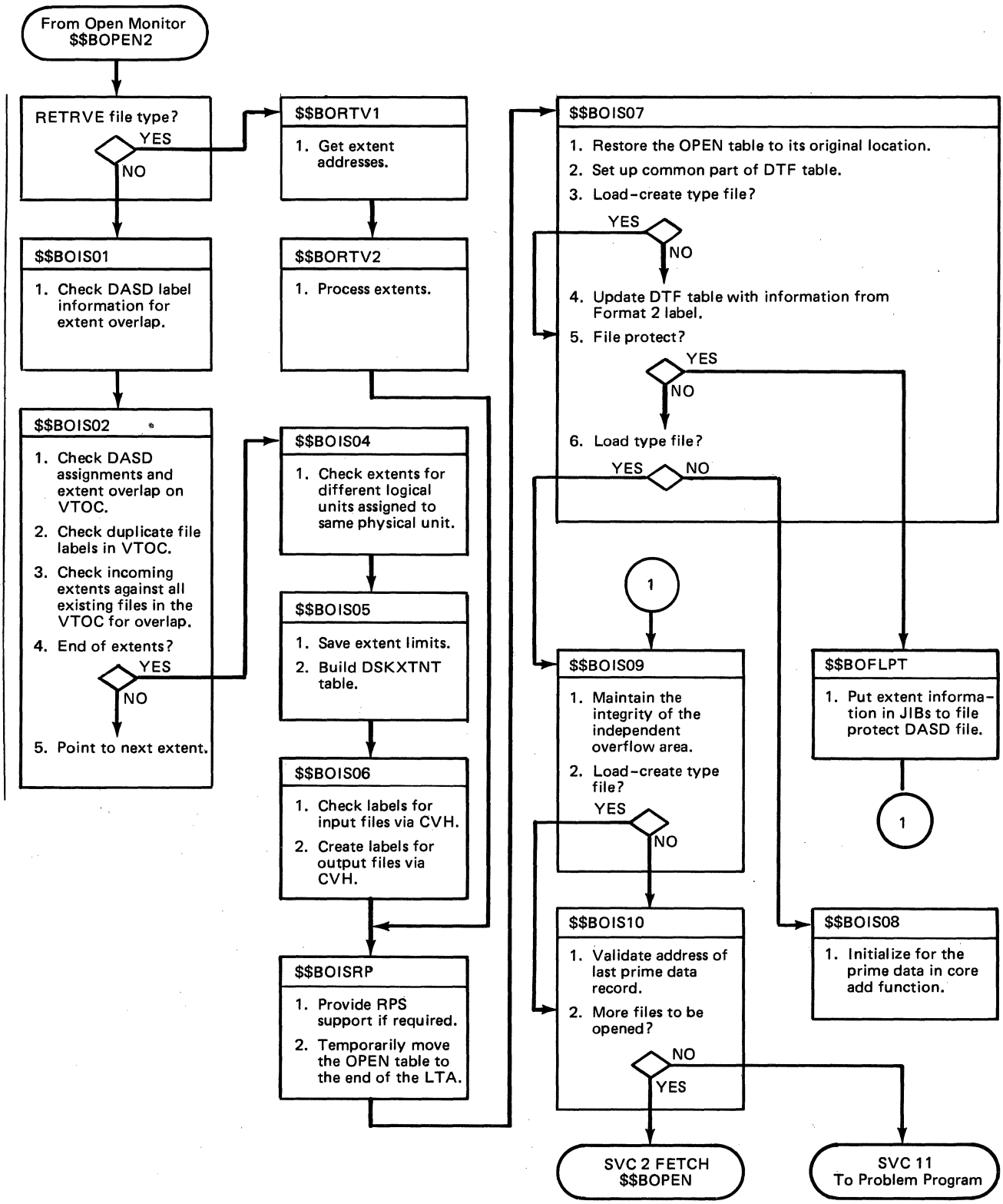
routine checks the prime data extents for continuity, and a check is made for the last prime data extent. The overflow extents are checked and saved for future reference.

For output, file labels are created and written in their appropriate locations and sequence, and the extent information is inserted in the labels. The format-2 label for the file is read, and the DTF table is updated for each function. A DSKXTN table is created, which is located at the end of the DTF table. It contains the logical unit and cell number for each extent. This table is used by logical IOCS to reference the extent information in the DTF table.

When the file is closed, the format-1 and format-2 labels are updated from the DTF tables. Then the open switch is set off, and control returns to the close monitcr or to the user.

For a more detailed description of label handling see VSE/Advanced Functions DASD Labels.

Chart 02. ISAM Open



For input and output files, the initial steps to open a file are the same. The SYSRES DASD label information is set up, and the extents are checked for overlap on each other and the VTOC. A check was made previously to determine if all the packs for the file have been mounted by checking all volume labels against the SYSRES DASD label informatien in the label save area in main storage.

All extents for each volume are checked against the VTOC limits and for overlap with each other. This checking is done by the Common VTOC Handler (CVH) residing in the SVA. If dcne for an output file, the filename is alsc checked for duplication. The extents are checked against the SYSRES DASD label information, the extent limit groups are saved, and the DSKXTN (logical unit and cell number) table at the end of the DTF table is built. The labels are checked, and the extent limits are inserted for input files. For output, the labels are created and written. The DTF tables are updated, and the routine returns contrcl to the problem program or to the open monitor.

See VSE/Advanced Functions Diagnosis Reference: IICCS Volume 4 for details of the Ccmmon VTOC Handler.

For input and output files the steps to close a file are the same. The format-1 and format-2 labels are updated and written back in the VTOC via CVH. Control returns to the problem program or to the close monitor.

$$BOIS01: ISAM Open, Phase 1, Charts LA-LB

Objective: To check the DASD label information.

Entry: From the open monitor, $$BOPEN.

Exit: To $$BOIS02 or to $$BOMSG1 (if an error condition occurs).

Method: This phase determines the address of the first and last extent in the SYSRES DASD label information in the label save area. It determines if this is a creation of a file, and turns on an indicator if it is. The phase then clears the reserved field in the SYSRES DASD label information record.

If it is an ADD or ADDRTR type file, this phase computes the number of tracks of the independent overflow extent limits on either the 2311, 2314, or 2319 devices and stores the result. It then fetches $$BOIS02.

$$BOIS02: ISAM Open, Phase 2, Chart LC

Objective: To determine if a DASD has been assigned to the file, if the format-5 label indicator is on, and if the extent limits overlap the VTOC (via CVH).

Entry: From $$BOIS01.

Exits: To $$BOIS04, or to $$BOMSG1 (if an error condition occurs).

Method: This phase determines if the extents for each DASD volume overlap the VTOC or themselves. The Common VTOC Handler (CVH) is invoked and a return code is passed back. RC=0 indicates there was no overlap. Depending on the overlap, the appropriate message is issued. After all extents have been checked, this phase fetches $$BOIS04 to continue processing extents.

$$BOIS04: ISAM Open, Phase 4, Chart LD

Objective: To check extents for different logical units assigned to the same physical units.

Entry: From $$BOIS02.

Exit: To $$BOIS05.

Method: This B-transient gets the logical and physical unit assignments for the index, prime data and independent overflow type extents. It checks the extents for different logical units assigned to the same physical unit. If this condition exists, the logical unit assignment of the extent in the SYSRES DASD label information located in the label save area is modified to correct this condition. This process continues until all extents have been checked.

$$BOIS05: ISAM Open, Phase 5, Charts LE-LG

Objective: To check the extents for valid ISAM format, to save extent limits and to build the DSKXTN table containing the logical unit and cell number of each extent.

Entry: From $$BOIS04.

Exit: To $$BOIS06 or to $$BOMSG1 when an error condition occurs.

Method: This phase first determines the extent type (overflow, index, or prime data), checks the type for validity, and branches to the appropriate routine to process the extent. If the extent type is not for an independent overflow, index, or prime data area, an error message is

initialized, and the message writer, $$BOMSG1, is fetched to write the message on SYSLOG.

If an index type extent is indicated, this phase determines if a master index has been specified along with a cylinder index. If there is a master index and a cylinder index, they must be assigned to the same physical unit and they must be contiguous. If both conditions are satisfied, the limits for the master and cylinder indexes are saved and the next extent is processed. If either condition is not met, an error message is initialized and the message writer is fetched.

If an overflow type extent is indicated, this phase saves the extent limits and builds an entry in the DSKXTN table. The DSKXTN table is located at the end of the DTFIS table and is used by ISAM to reference the extent information in the DTF tables. It contains the logical unit and cell number for each extent.

If a prime data type extent is indicated, it checks the first prime data extent upper limit to determine if more prime data extents are allowed. If more prime data extents are allowed, the phase checks the remaining prime data extent limits for continuity, and checks for the last prime data extent.

After all extents have been processed, this phase checks the index extent sequence numbers and fetches $$BOIS06 for execution.

$$BOIS06: ISAM Open, Phase 6, Charts LH-LI

Objective: For input files, checks format-1 labels and stores extent information in them. For output files, creates format-1 and format-2 labels and stores extent information in format-1 labels.

Entry: From $$BCIS05 or $$BODSMW.

Exit: To $$BOISRP, $$BODSMW, or $$BOMSG1 if an error condition occurs.

Method: If this is an input file, this phase sets up the 44-byte file name from the SYSRES DASD label information as the key, and reads the matching format-1 label from the VTOC (via CVH). It checks the format-1 label, moves the extent limits into the label, writes the updated format-1 label in the VTOC, and increases the volume sequence number by one. It continues processing until there are no more extents.

If this is an output file, this phase creates a format-1 label and then a format-2 label. For a description of the format-1 label and the format-2 label refer VSE/Advanced Functions DASD Labels.

After creating the format-1 and format-2 labels, the phase increases the volume sequence number and continues processing until there are no more extents.

If this is a mixed input/output file, the phase updates the format-1 label and writes it back in the VTOC.

For an extend file, the format-1 label is checked for the data security indicator. If it is on, and the data security message has not been issued, it is issued via a fetch of $$BODSMW.

For a create file, the format-1 label is built with the data security indicator on if data security has been requested.

All communication with the VTOC in this phase is done by the CVH.

$$BOISRP: ISAM Open, RPS Phase, Chart LJ

Objective: To provide RPS support if the device containing the prime data and the supervisor support RPS.

Entry: From $$BOIS06 or $$BORTV2.

Exit: To $$BOIS07.

Method: The PUBs for the devices containing the index and the prime data are checked to see if they support RPS. If either or both do, the appropriate bits are set on in the DTFIS. Also, a check is made to see if the supervisor supports RPS. If either the supervisor or the device containing the prime data does not support RPS, control is passed to $$BOIS07. Otherwise, the following operations are performed:

1. The RPS switch in the DTFIS is turned on.
2. A 384-byte area is obtained in the user partition GETVIS area for the RPS DTFIS extension (see Figure 112).
3. The name of the required RPS ISMOD superset is determined and the superset is loaded into the SVA.
4. The DTFIS and the RPS DTFIS extension are initialized for RPS support.
5. Control is passed to $$BOIS07. (The OPEN table is moved to the end of the LTA so that $$BOIS07 can be loaded as one contiguous phase. After $$BOIS07 gets control it moves the OPEN table back to its original location.)

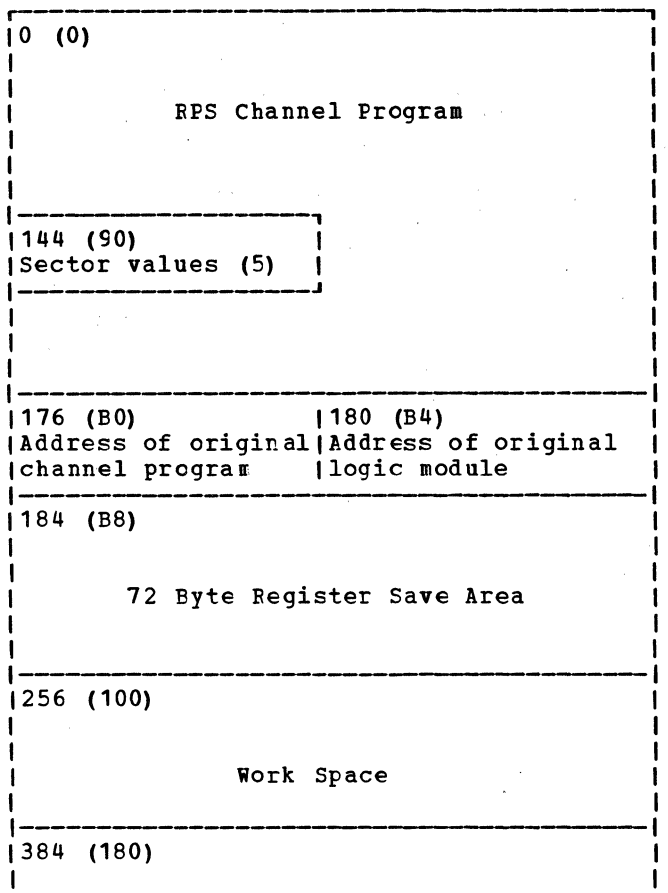


Figure 112. RPS DTF Extension Work Area

$$BOIS07: ISAM Open, Phase 7, Charts MA-MD

Objective: To restore the OPEN table to its original location, to read the Format 1 and Format 2 labels for this file, and update the DTFIS table for each function.

Entry: From $$BOISRP.

Exit: To $$BOIS08 or $$BOIS09 for load-create files. To $$BOFLPT for file protection. To $$BOMSG1 if an error condition occurs.

Method: For a load-create file, this phase moves the prime data upper limit from the extent save area to the DTF table. For load-extension, add, retrieve, and add-retrieve files, it reads the format-1 and format-2 labels. It then checks for file protection, and whether the file is a load file. If there is no file protection, $$BOFLPT is fetched. If there is file protection and it is a load file, $$BOIS09 is fetched to write an EOF record at the beginning of the independent overflow area (if one is specified). If there is no file protection and it is not a load file, $$BOIS08 is fetched. For add and add-retrieve files that are already opened, a check is made for the track hold specification. if HOLD=YES is specified, $$BOMSG1 is fetched to print out error message 4269I FILE IS OPEN FOR ADD.

$$BOIS08: ISAM Open, Phase 8, Charts ME-MF

Objective: To build CCWs in the high order bytes of IOAREAL, and to update the prime data in-core add section of the DTF table.

Entry: From $$BOIS07 or $$BOFLPT.

Exit: To $$BOIS09 or to $$BOMSG1 if an error condition occurs.

Method: This phase determines if the prime data in-core add function is specified in the DTFIS table. If it is specified, this phase increases the size of IOAREAL (output area used for loading or adding records to a file) to permit the writing and reading of more than one physical record on or from a DASD (Direct Access Storage Device) per EXCP (Execute Channel Program).

The phase first calculates the maximum number of prime data records that can be read into or written from main storage at one time. It then calculates the starting address of the CCW build area in IOAREAL and aligns this address on a doubleword boundary. It builds the following CCWs to write and read more than one physical record per EXCP.

1. For a write:

A.	CCW	X'31', Pointer to IOAREAL, Command Chaining, 5	Search identifier equal the prime data records in IOAREAL.
B.	CCW	X'08', Pointer to *-8	TIC to *-8.
C.	CCW	X'08', Pointer to Read CCWs (see 2.)	TIC to address of read CCWs in DTF table.
D.	CCW	X'0D', Pointer to Key Field of IOAREAL, Command Chaining, Key Length + Block Size	Write key and data of prime data record in IOAREAL.

2. For a read:

A.	CCW	X'31', Pointer to Seek/Search Address Area, Command Chaining, 5	Search identifier equal the prime data area using the pointer, CCHHR, in the seek/search address area in the DTF table.
B.	CCW	X'08', Pointer to *-8	TIC to *-8.
C.	CCW	X'1E', Pointer to IOAREAL, Command Chaining, 8 + Key Length + Block Size	Read count, key, and data of prime data record into IOAREAL.

This phase continues to build the CCWs for a read or a write until the count for the maximum number of prime records in main storage reaches 0. This count is reduced by 1 each time the CCWs for a prime data record are built.

The CCWs built for a read or a write are preceded by a long seek CCW and TIC to either 1 or 2 depending on the operation to be performed. When the last read CCW is built, its flag field is set to 0, indicating the end of the CCW chain, and $$BOIS09 is fetched to search the independent overflow area (if specified) for the EOF record.

$$BOIS09: ISAM Open, Integrity Phase 1, Charts MG-MI

Objective:

- For a load type file, to write an EOF record at the beginning of the independent overflow area.
- For an add type file, to search the independent overflow area for the EOF record.

Entry: From $$BOIS07 or $$BOIS08.

Exit: To the TES processor, $$BOPEN, to $$BOIS10, or to the problem program.

Method: This B-transient first determines if the file has an independent overflow area. If there is no independent overflow area specified, a test for a load-create type file is made. If it is a load-create type file, a test for more files to be opened is made. If more files are to be opened, $$BOPEN is fetched for execution. If no more files are to be opened, control is returned to the problem program. If it is not a load-create type file, $$BOIS10 is fetched for execution.

If an independent overflow area is specified, a test for file type is made. If it is a retrieve or load-extend type file, control is returned to either the TES processor ($$BOPEN) if more files are to be opened or to the problem program.

If it is a load-create type file, an EOF record is written at the beginning of the independent overflow area and a test is made to determine if a new independent overflow extent has been specified. If it has not been specified, the number of tracks in the independent overflow area is calculated and stored in the DTF table and control is returned to either the TES processor ($$BOPEN) if more files are to be opened or to the problem program. If a new overflow extent has been specified, control is returned to either the open monitor or to the problem program.

If it is an add type file, the independent overflow area is searched for the EOF record. When the EOF record is found, the number of tracks in the independent overflow area is calculated and stored in the DTF table, the last prime updated, and $$BOIS10 is fetched to validate the last prime data record address.

$$BOIS10: ISAM Open, Integrity Phase 2, Charts MJ-MK

Objective: To validate the last prime data record address by scanning the prime data area for the end-of-file (EOF) record.

Entry: From $$BOIS09.

Exit: To the TES processor, $$BOPEN, or to the problem program.

Method: This routine searches for the EOF record in the prime data area. When the EOF record is read, the last prime data record address is saved in the DTF table and control is returned to either the TES processor ($$BOPEN) if more files are to be opened or to the problem program.

$$BCISOA: ISAM Close, Charts NA-NC

Objective: To close the file by updating the format-1 and format-2 labels with information from the DTFIS table.

Entry: From the close monitor, $$BCLOSE.

Exit: To the problem program or to the close monitor, $$BCLOSE, if more files are to be closed.

Method: If a load-create type file is to be closed, this phase updates the format-1 label with information from the DTFIS table, and writes the updated label back in the VTOC. It then updates the format-2 label with information from the DTFIS table, and writes the updated format-2 label back in the VTOC.

For all other type files, this phase updates only the format-2 label with information from the DTFIS table. It then writes the updated format-2 label back in the VTOC. If more files are to be closed, the close monitor, $$BCLOSE, is fetched. If no more files are to be closed, control returns to the problem program.

All communication with the VTOC in this phase is done by the CVH.

$$BORTV1: ISAM RETRVE Open, Phase 1, Charts ND-NF

Objective: To open the RETRVE part of the DTFIS table by reading the format-1 label to get information needed for the table.

Entry:

- From the open monitor, $$BOPEN2.
- From the data security message writer, $$BODSMW.
- From $$BORTV2 if a new extent on a different logical unit has been found.

Exits: To $$BORTV2 upon normal completion. To $$BOFLPT for DASD file protection. To $$BOMSG1, if messages are to be written on SYSLOG. To $$BCDSMW to write the data security message.

Method: A test is made to determine if entry was made to this phase from a phase other then $$BOPEN2. If so, control branches to a predetermined location within this phase. If not, processing continues inline. Next, this phase computes the length of the incoming DASD label information and gets the address of the first extent. The first-time switch is set on, and a check is made for the expiration date in the DASD label information. If the expiration date is not present, this routine gets the retention period for the DASD labels and finds the expiration date.

The format-1 label is then read and checked for the data security indicator. If the file has not been opened, $$BODSMW is fetched to print the data security message. After determining the number of extents in format-1 label and getting the address of the first extent, this phase fetches $$BORTV2 for execution.

$$BORTV2: ISAM RETRVE Open, Phase 2, Charts NG-NH

Objectives: To open the RETRVE part of the DTFIS table by reading the format-2 label to get information needed for the table. To insert entries in the DSKXTN table located at the end of the DTF table.

Entry: From $$BORTV1.

Exits: To $$BOISRP upon normal completion. To $$BOFLPT for DASD file protection. To $$BOMSG1 if messages are to be written on SYSLOG. To $$BORTV1 if a new extent on a different logical unit is found.

Method: A test is made to determine if this phase was entered from a phase other than $$BORTV1. If it was, control branches to a predetermined location within this phase. If it was not, processing continues inline.

This phase then gets the address of the first extent in the format-1 label and tests to determine if an extent is present. If an extent is not present, control branches to read the format-2 label. If an extent is present, a test is made to determine if the first prime data extent switch is on. If it is on, an entry is made in the DSKXTN table. If it is not on, a test is made to determine if the extent is a prime data extent. If it is a prime data extent, the extent sequence number and the extent lower limit are saved and the first prime data extent switch is set.

The routine then makes an entry in the DSKXTN table for each extent. Only three extents are possible per volume for an indexed sequential file: the master/cylinder index area, the prime data area, and the independent overflow area. Each entry in the DSKXTN table is four bytes, containing two bytes of the logical unit and two bytes of the cell number. The location of the entry within the table is determined by multiplying the extent sequence number by 4, and adding the result to the starting address of the table minus 4. If the DSKXTN table is full, an error condition exists and a message is issued to that effect.

After the DSKXTN entry is made, a test is made to determine if the file-protect option has been specified. If it has, the routine sets up the extent for file protect and fetches $$BOFLPT.

If the file-protect option has not been specified or if this phase was entered from $$BOFLPT, this routine gets the address of the next extent in the format-1 label, and checks to determine if all extents in the label have been processed. If they have not been processed, control returns to process the next extent.

When all extents have been processed, the first time through B-transient switch

is checked. If on, the switch is set off and a test is made to determine whether this is the first volume. If it is not the first volume, an error message is issued. If it is the first volume, a test is made to determine if the format-2 pointer is present in the format-1 label. If it is not present, an error message is issued. If the format-2 pointer is present, the format-2 label is read, and the master index and cylinder index lower limits from the label are saved.

A check is made to determine if all the extents have been processed. If they have not been processed, this routine scans the SYSRES DASD label information in the label save area to find the next extent on a different logical unit, or the end of the extents. If a new extent on a different logical unit is found, control returns to $$BORTV1 to read the volume label and process the extents for that logical unit.

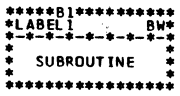
When all extents have been processed, this phase sets the file-protect option indicator off (if it is on) and exits to phase $$BOISRP.

EXPLANATION OF FLOWCHART SYMBOLS

DESCRIPTION | **EXAMPLE**

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

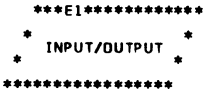
*B2
IF ANY ADDITIONAL EXPLANATION IS REQUIRED, ITS LOCATION ON THE CHART IS IDENTIFIED BY AN ASTERISK AND THE BLOCK ID.



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

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

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



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

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

The beginning or end of a program or routine.

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

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

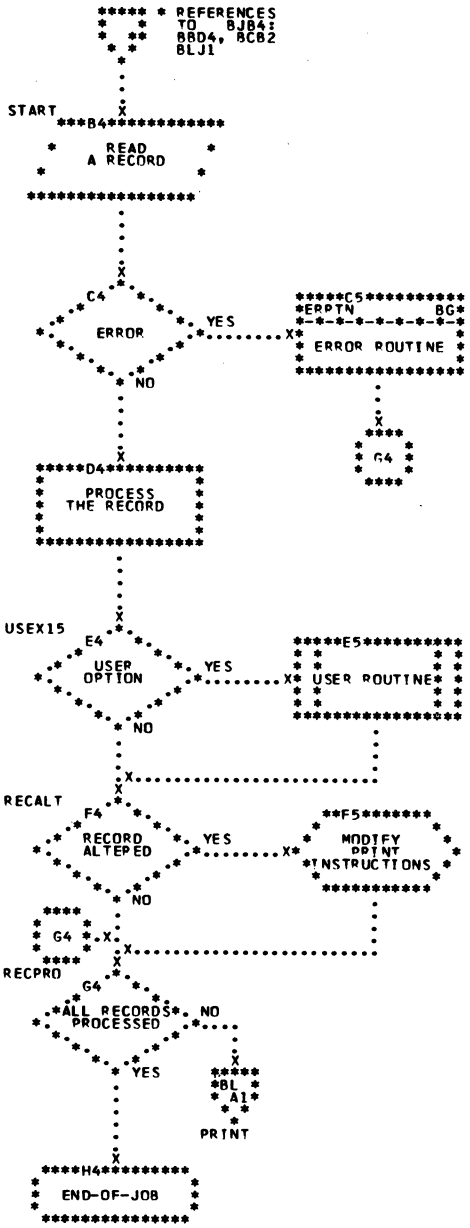


Chart AA. DAMOD: Input/Output Macros

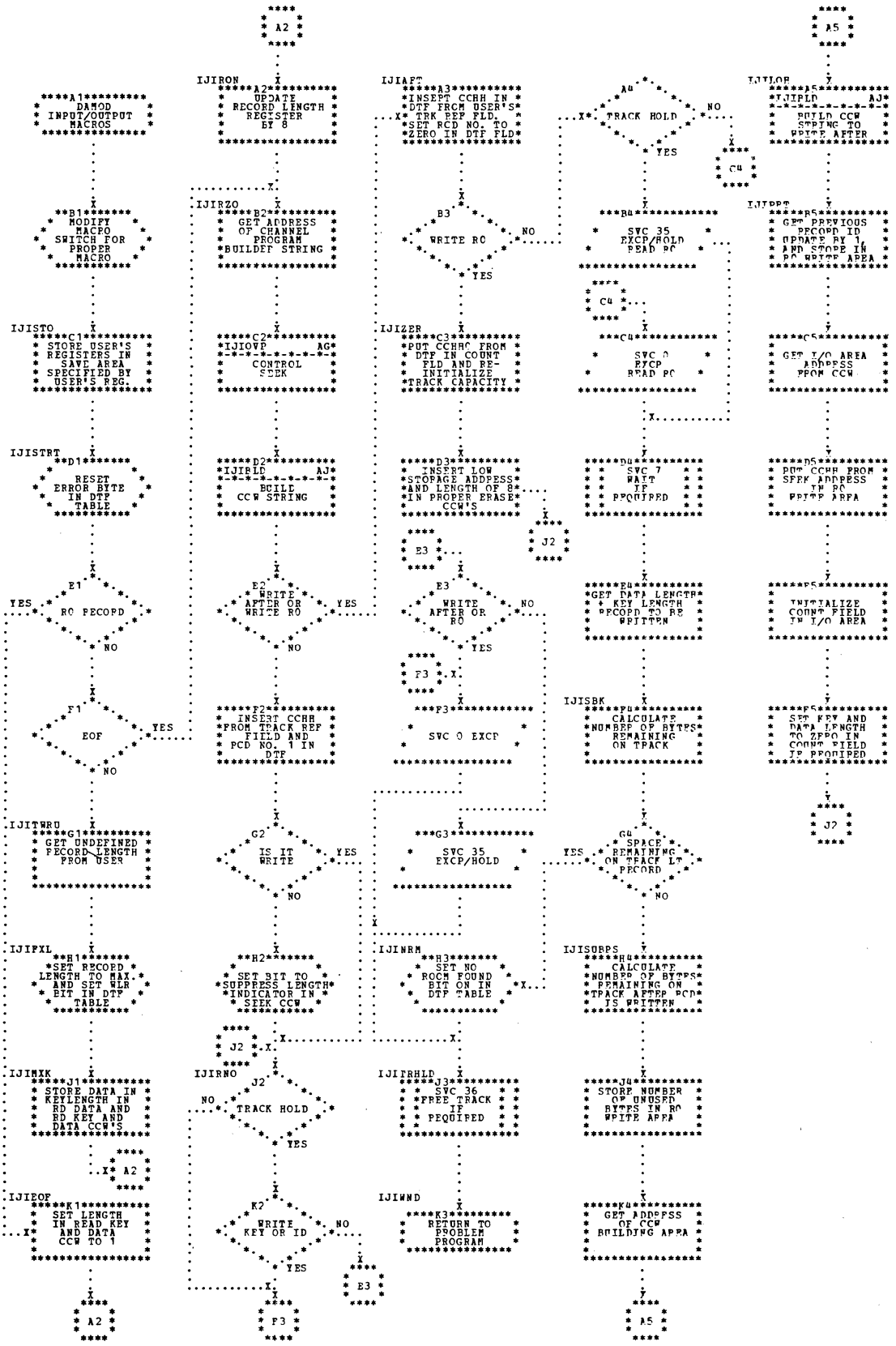


Chart AB. DAMOD: WAITF Macro (Part 1 of 4)

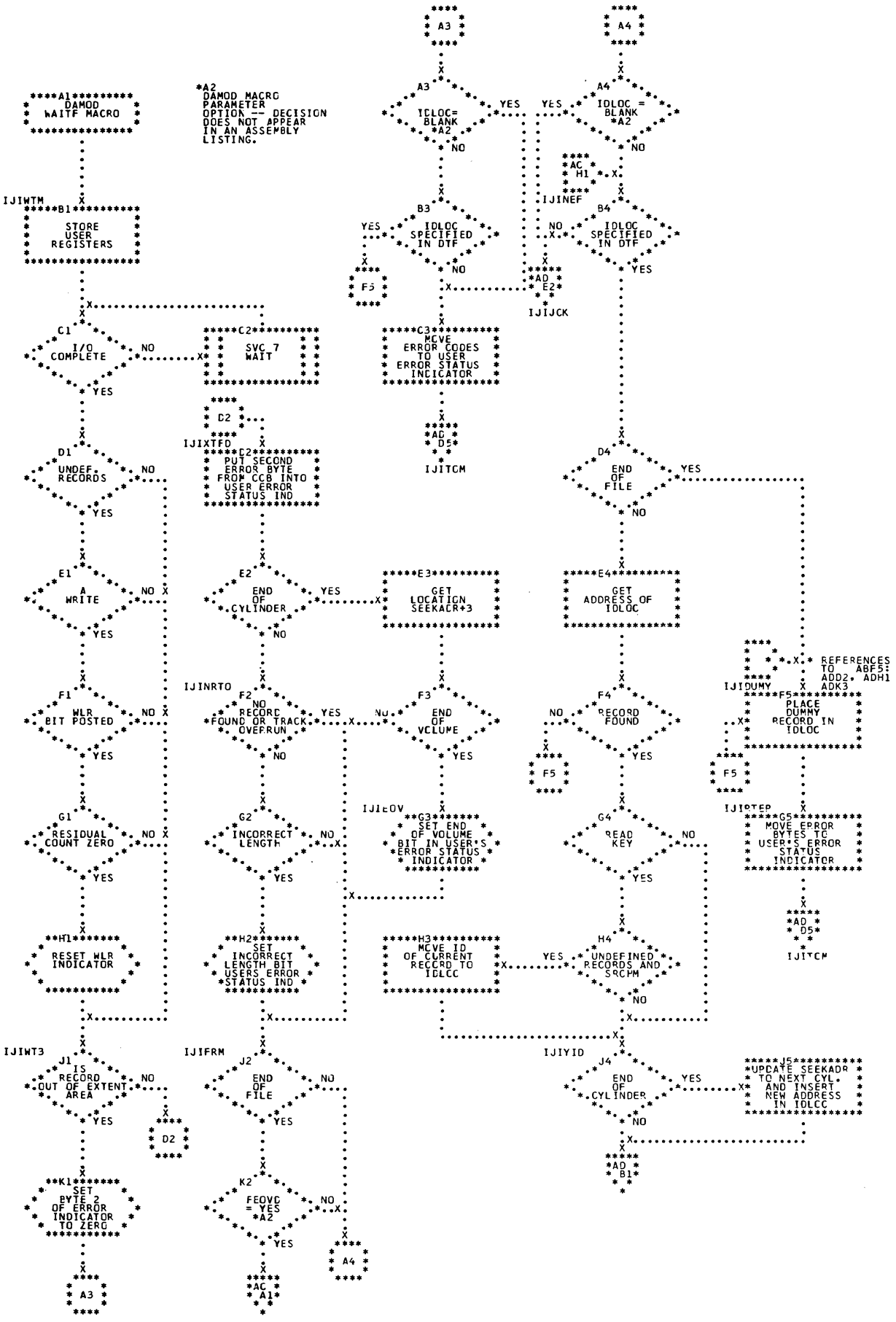


Chart AC. DAMOD: WAITF Macro (Part 2 of 4)

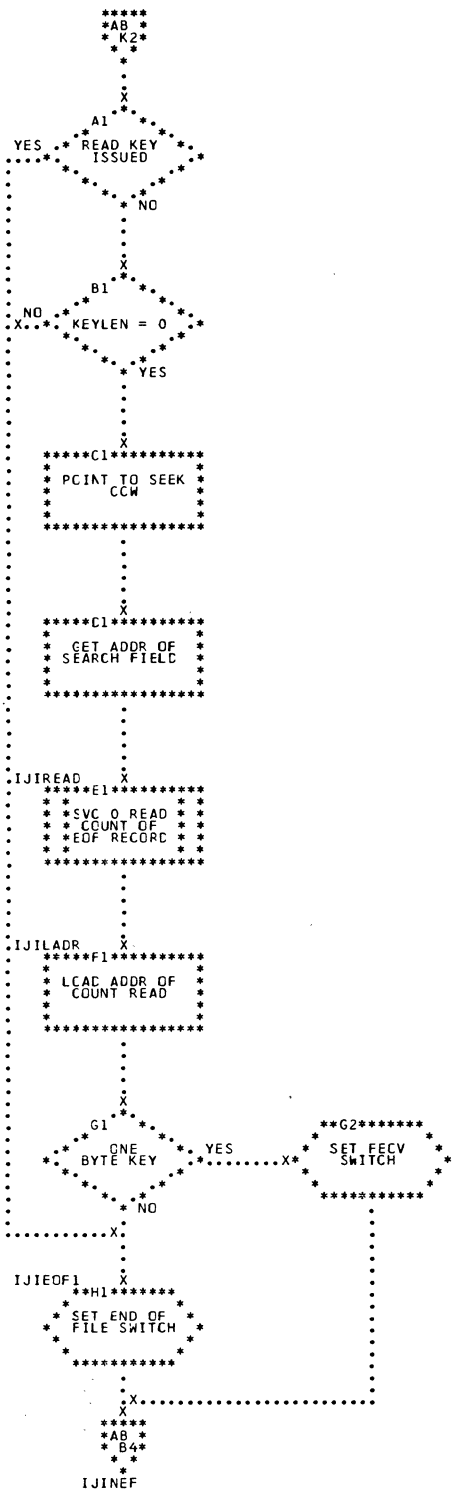


Chart AD. DAMOD: WAITF Macro (Part 3 of 4)

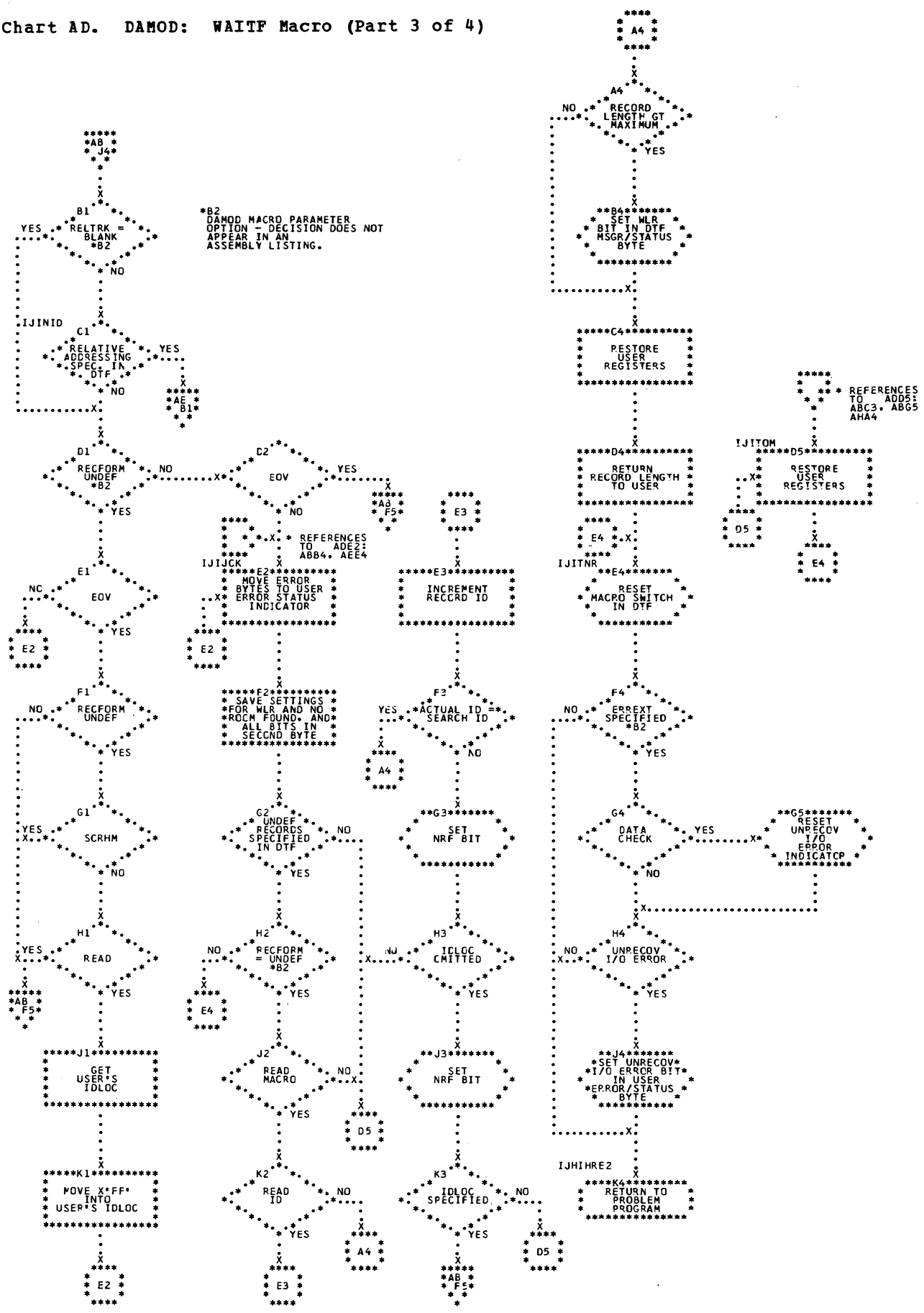


Chart AE. DAMOD: WAITF Macro (Part 4 of 4)

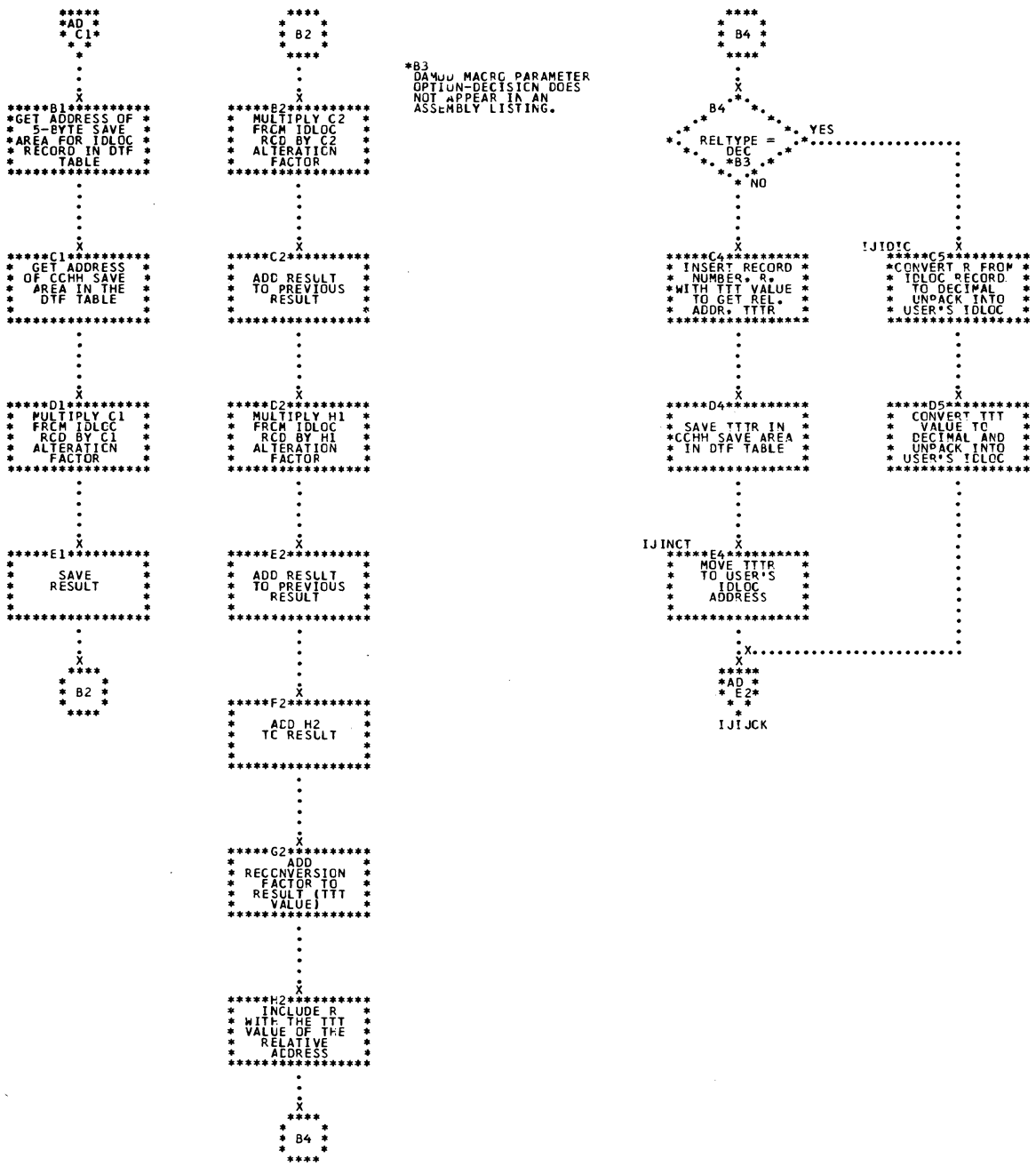


Chart AF. DAMOD and DAMODV: CNTRL and FREE Macros

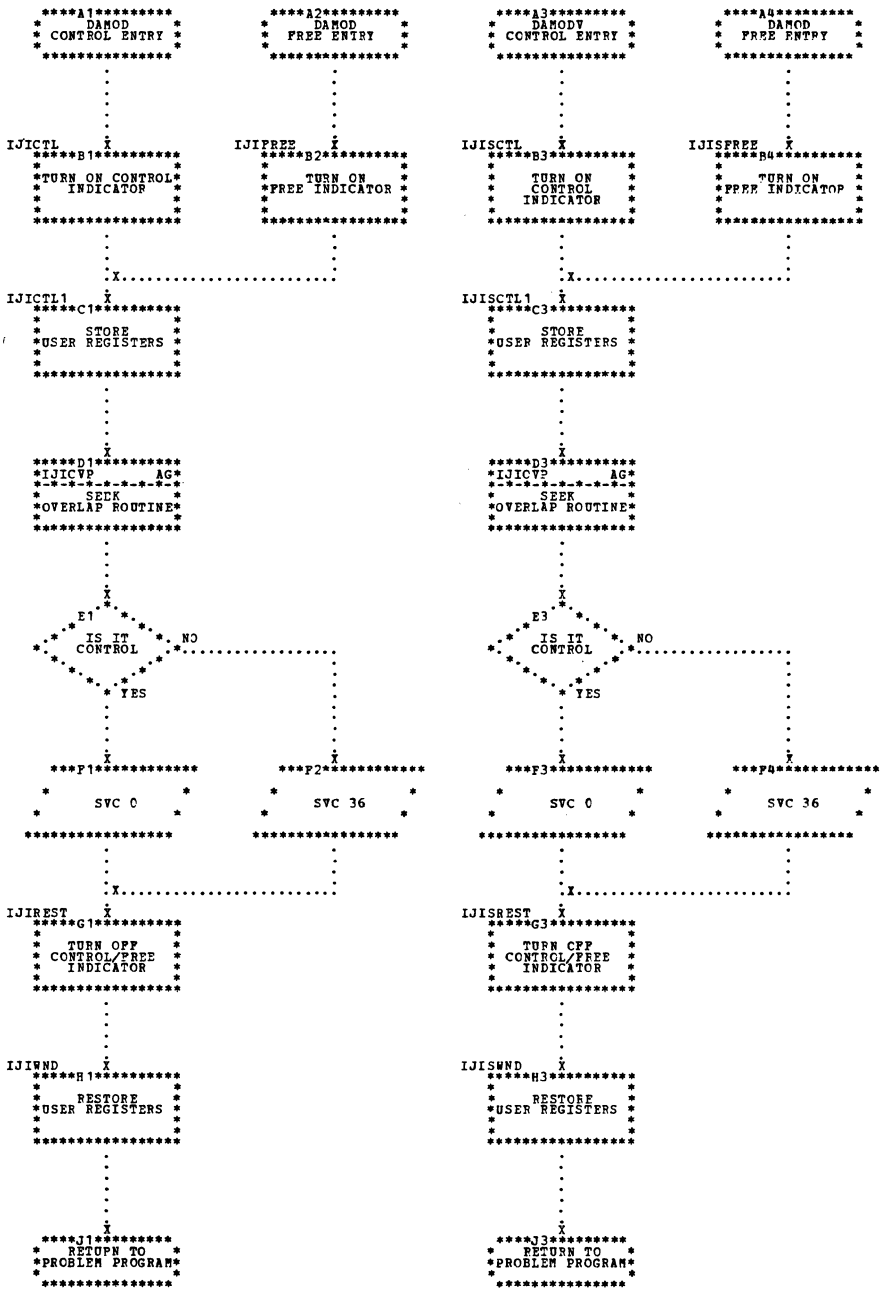


Chart AG. DAMOD: Seek Overlap Subroutine (Part 1 of 2)

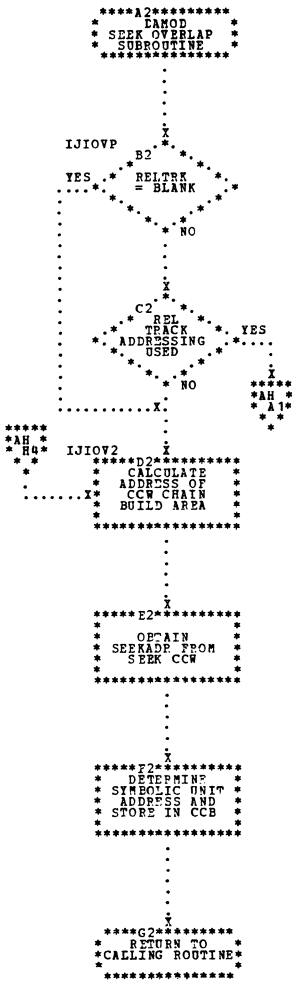


Chart AH: DAMOD: Seek Overlap Subroutine (Part 2 of 2)

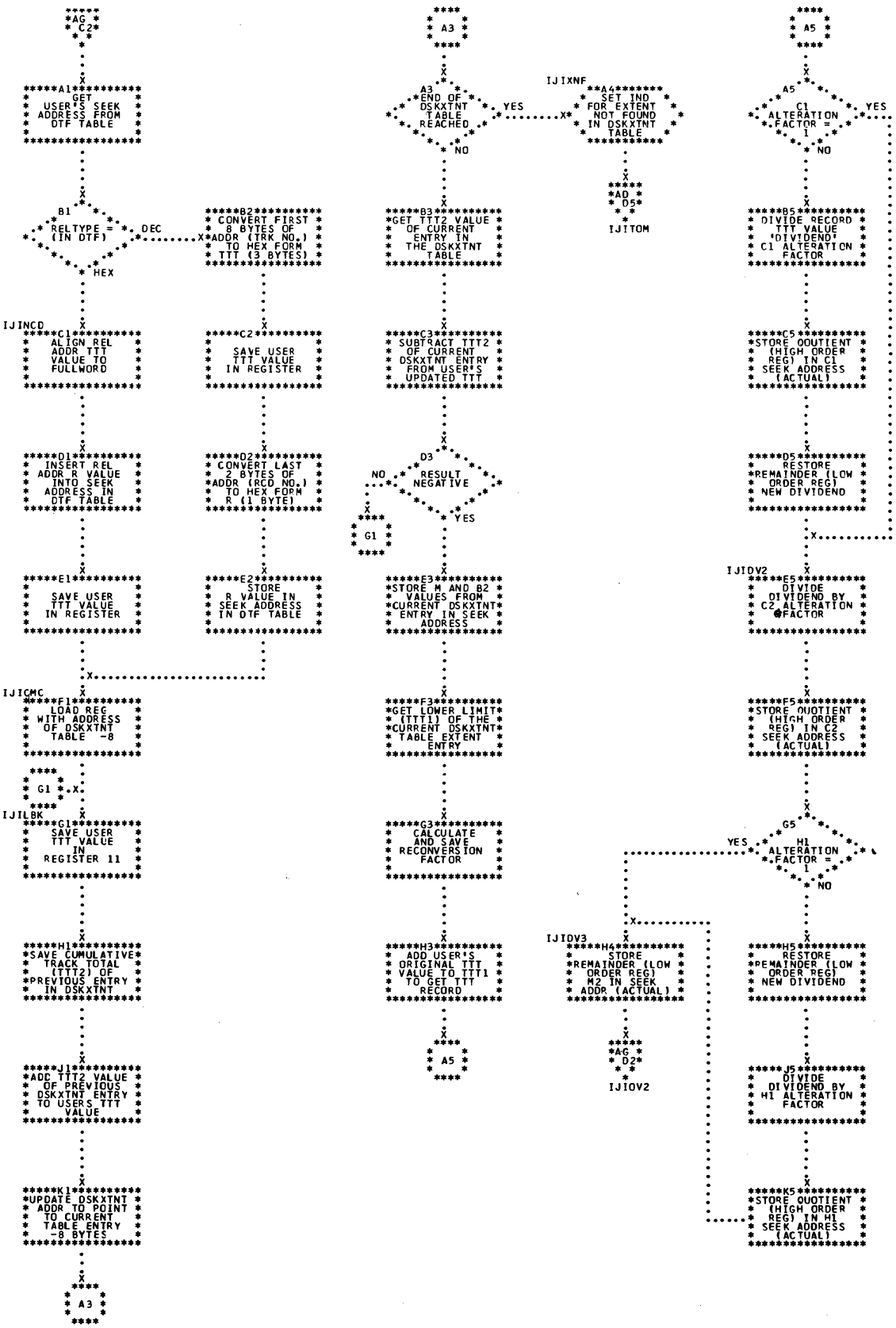


Chart AJ. DAMOD and DAMODV: Channel Program Builder Subroutine

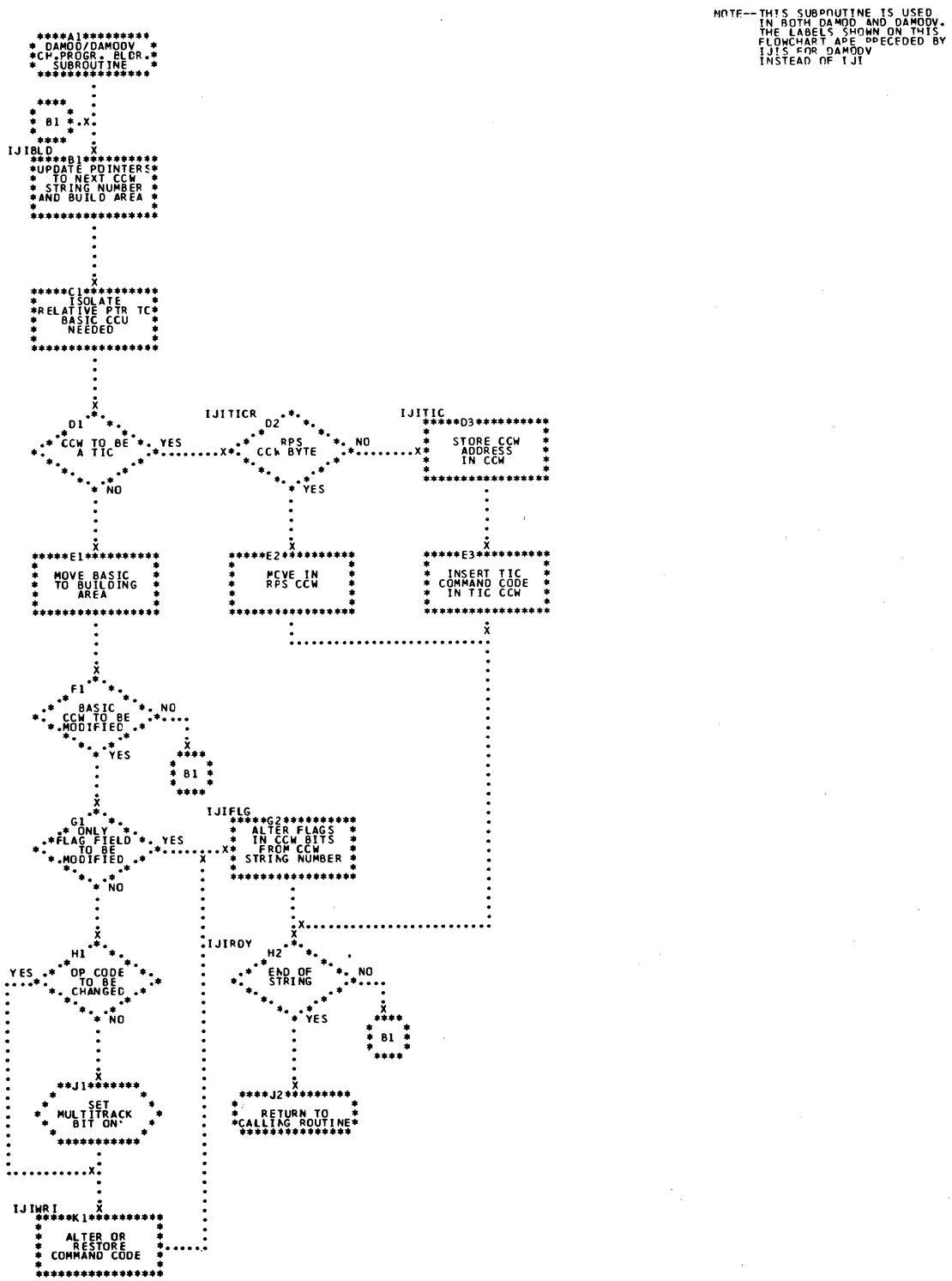


Chart AK. DAMODV: Input/Output Macros (Part 1 of 5)

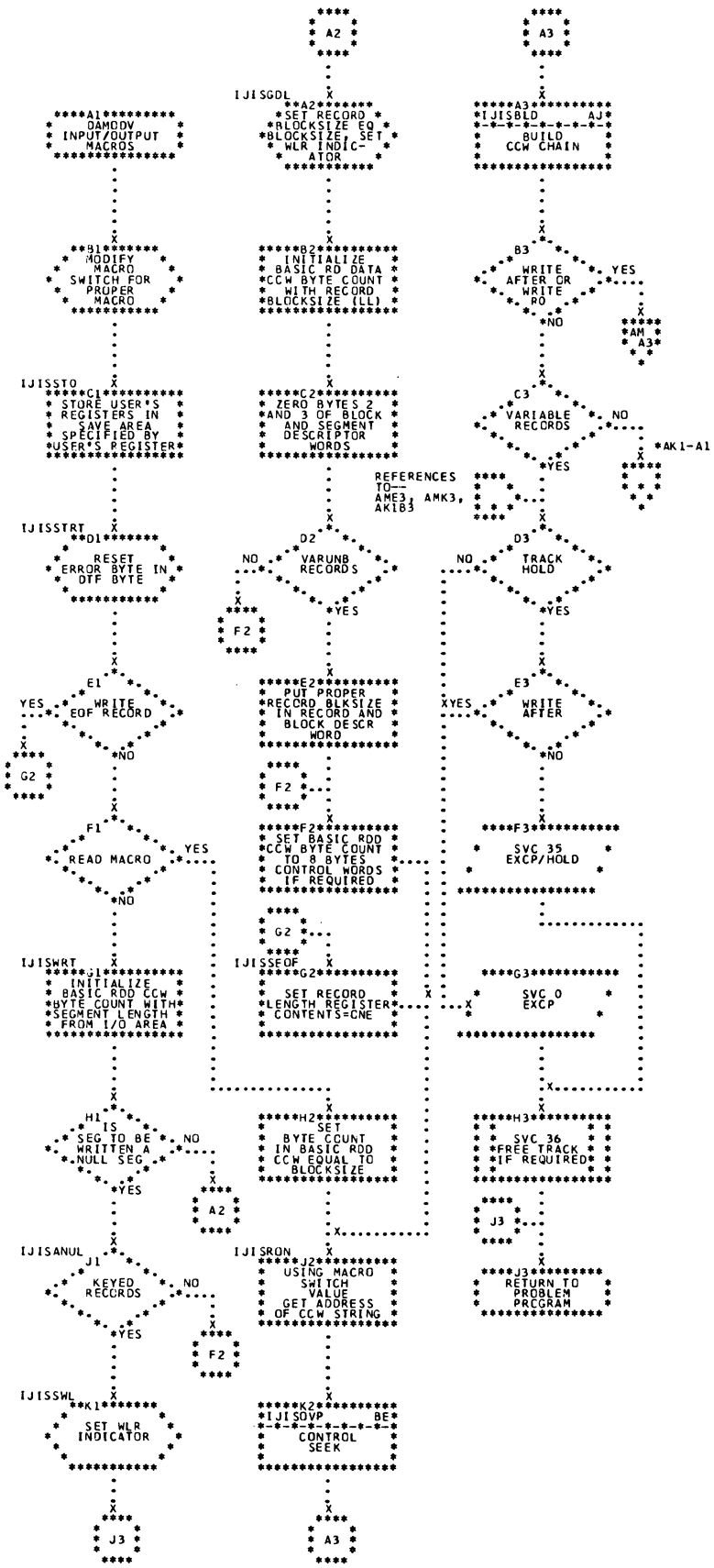


Chart AK1. DAMODV: Input/Output Macros (Part 2 of 5)

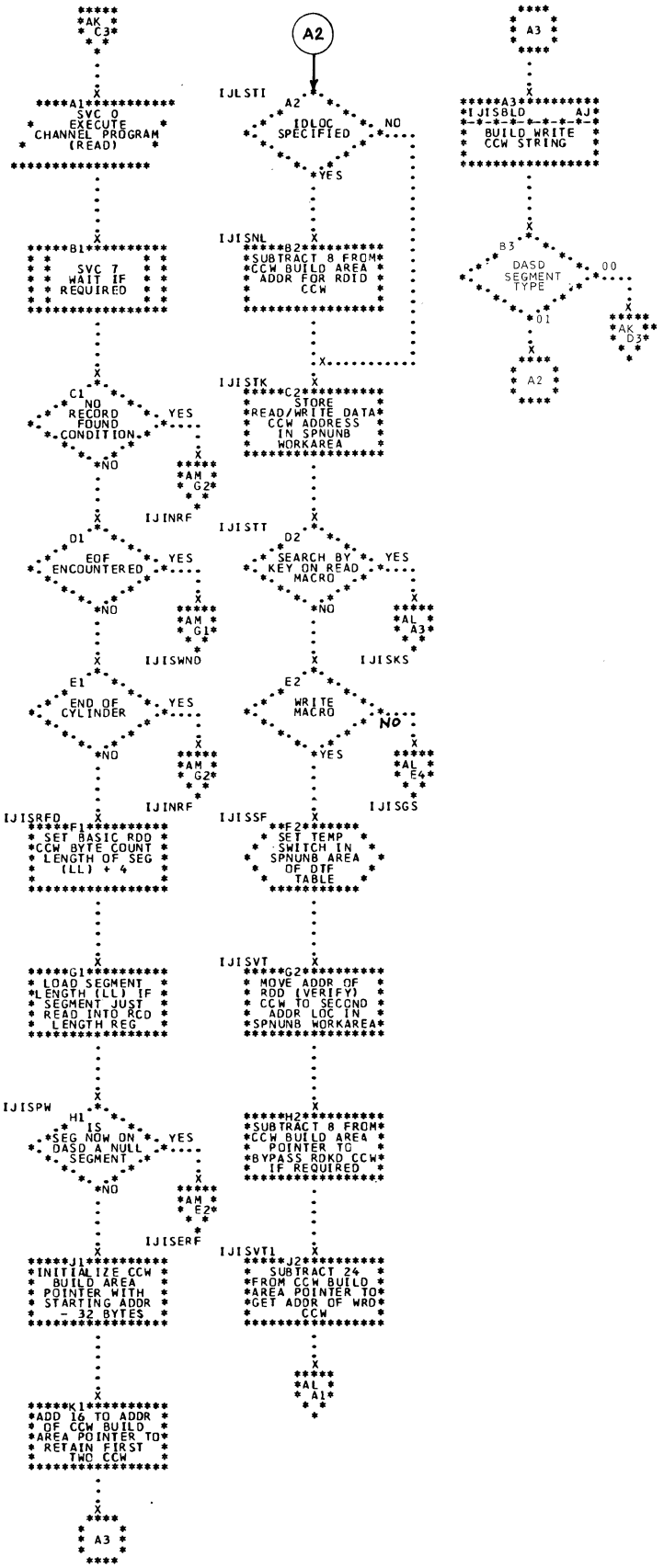


Chart AL. DAMODV: Input/Output Macros (Part 3 of 5)

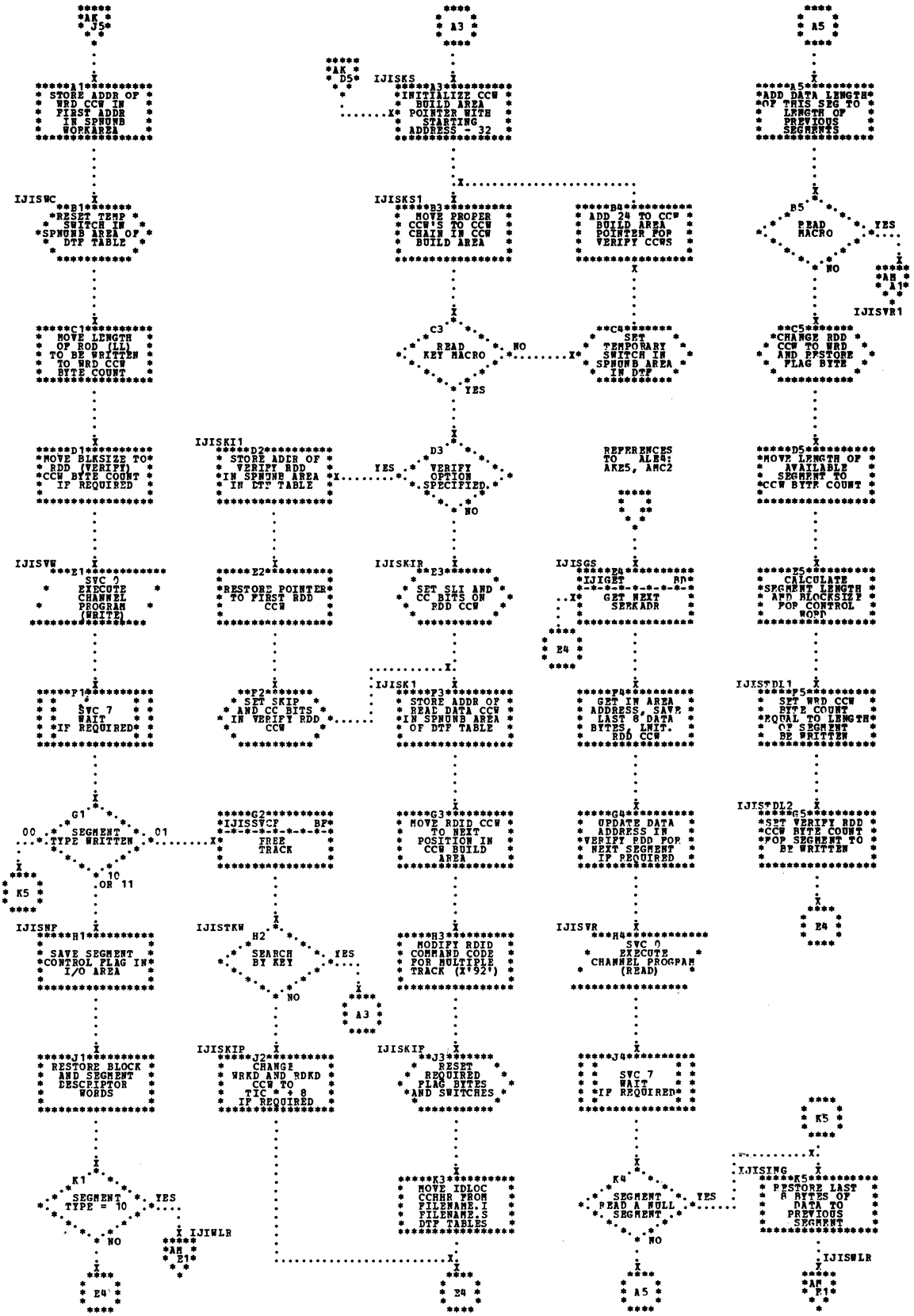


Chart AM. DAMODV: Input/Output Macros (Part 4 of 5)

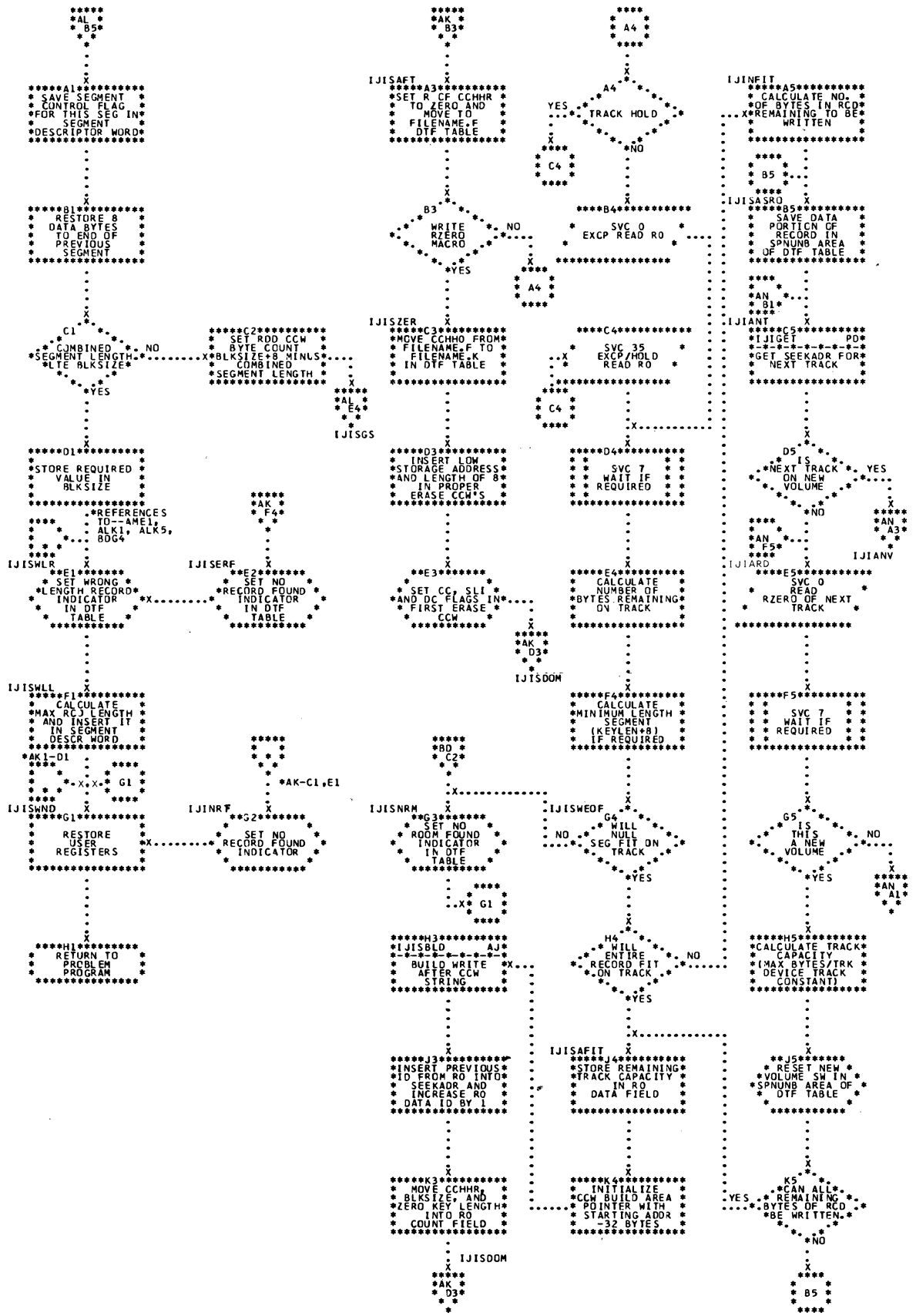


Chart AN. IAMODV: Input/Output Macros (Part 5 of 5)

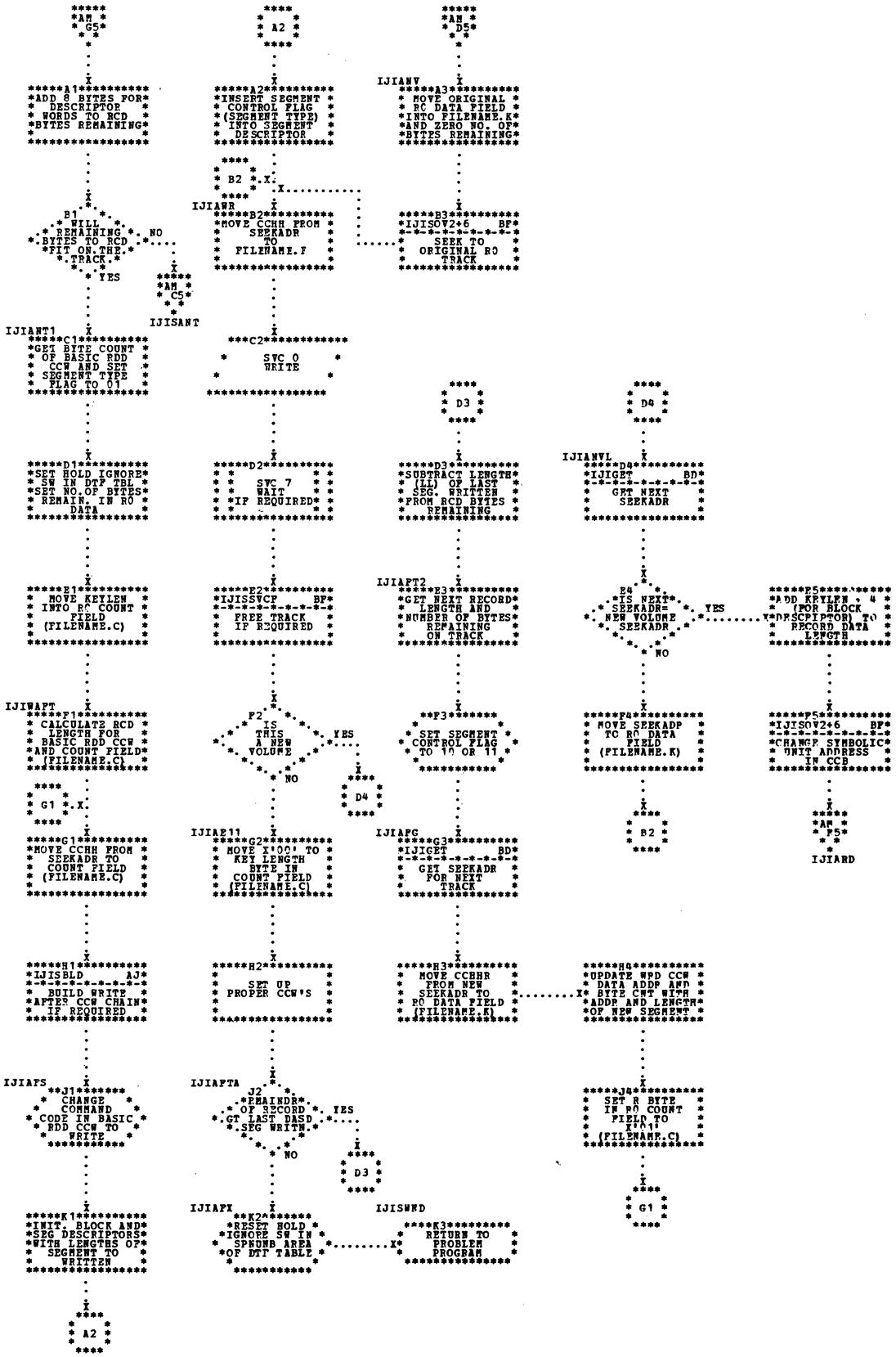


Chart BA. DAMODV WAITF Macro (Part 1 of 3)

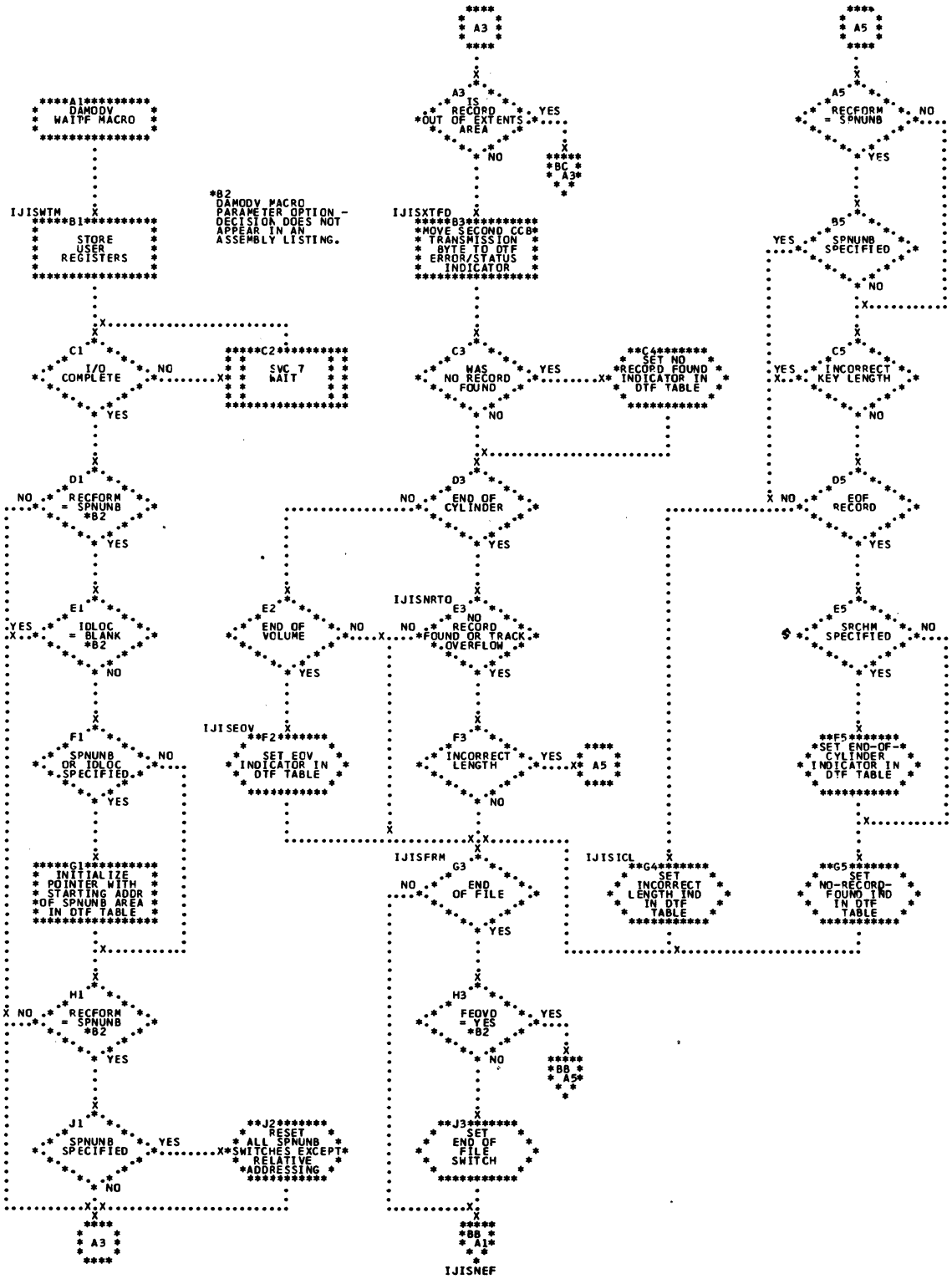


Chart BB. DAMODV WAITF Macro (Part 2 of 3)

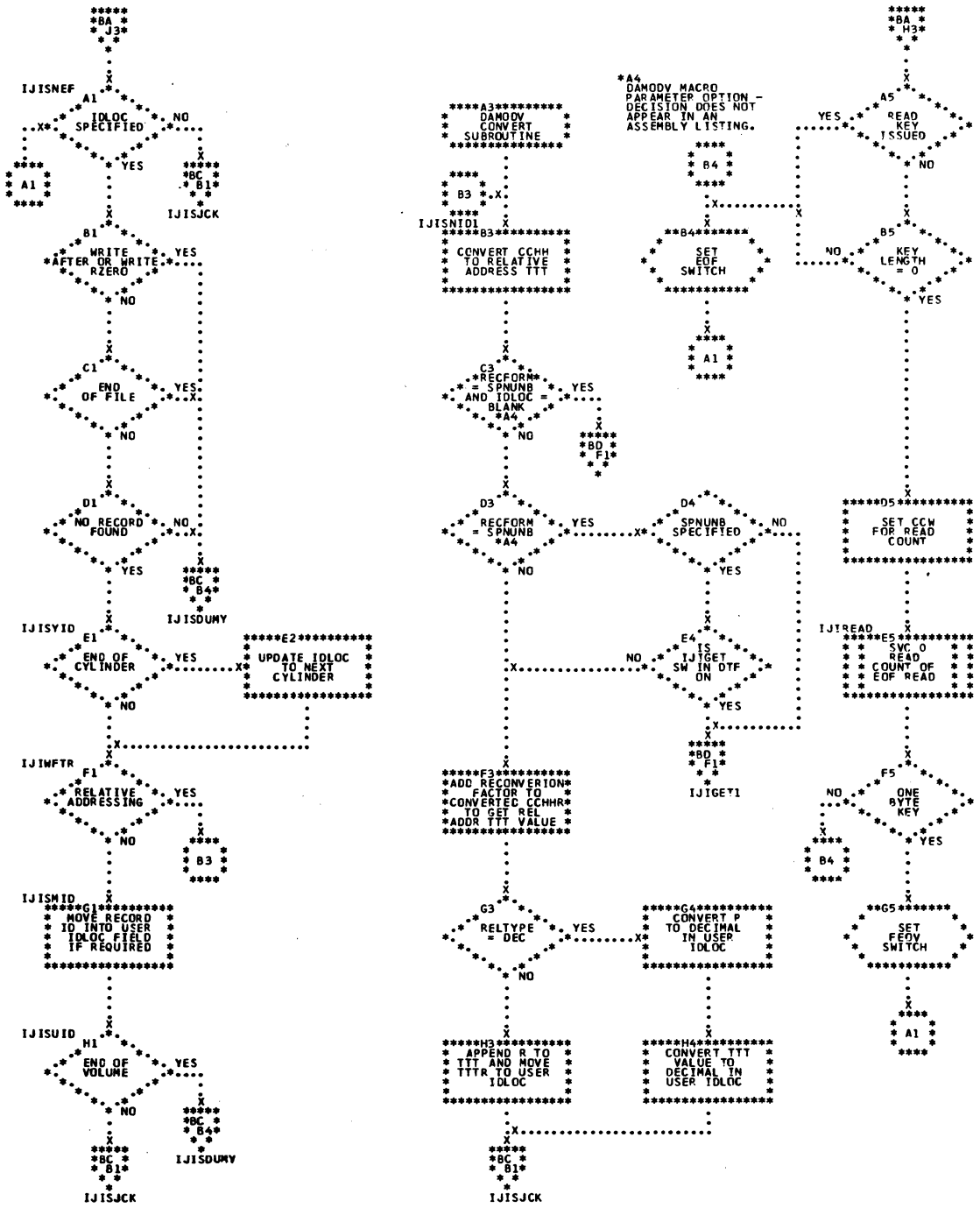


Chart BC. DAMODV WAITF Macro (Part 3 of 3)

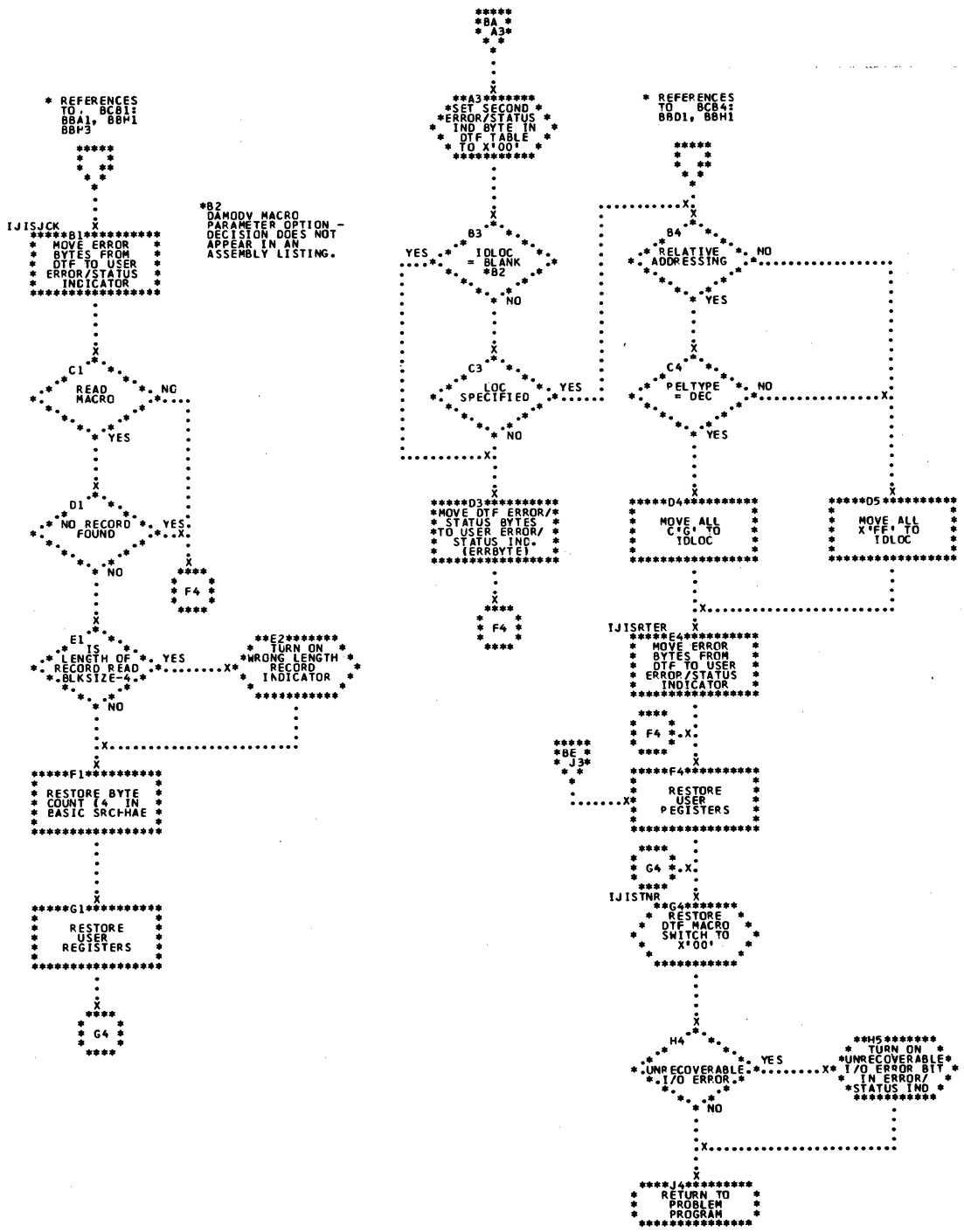


Chart BD. DAMODV: Get Subroutine

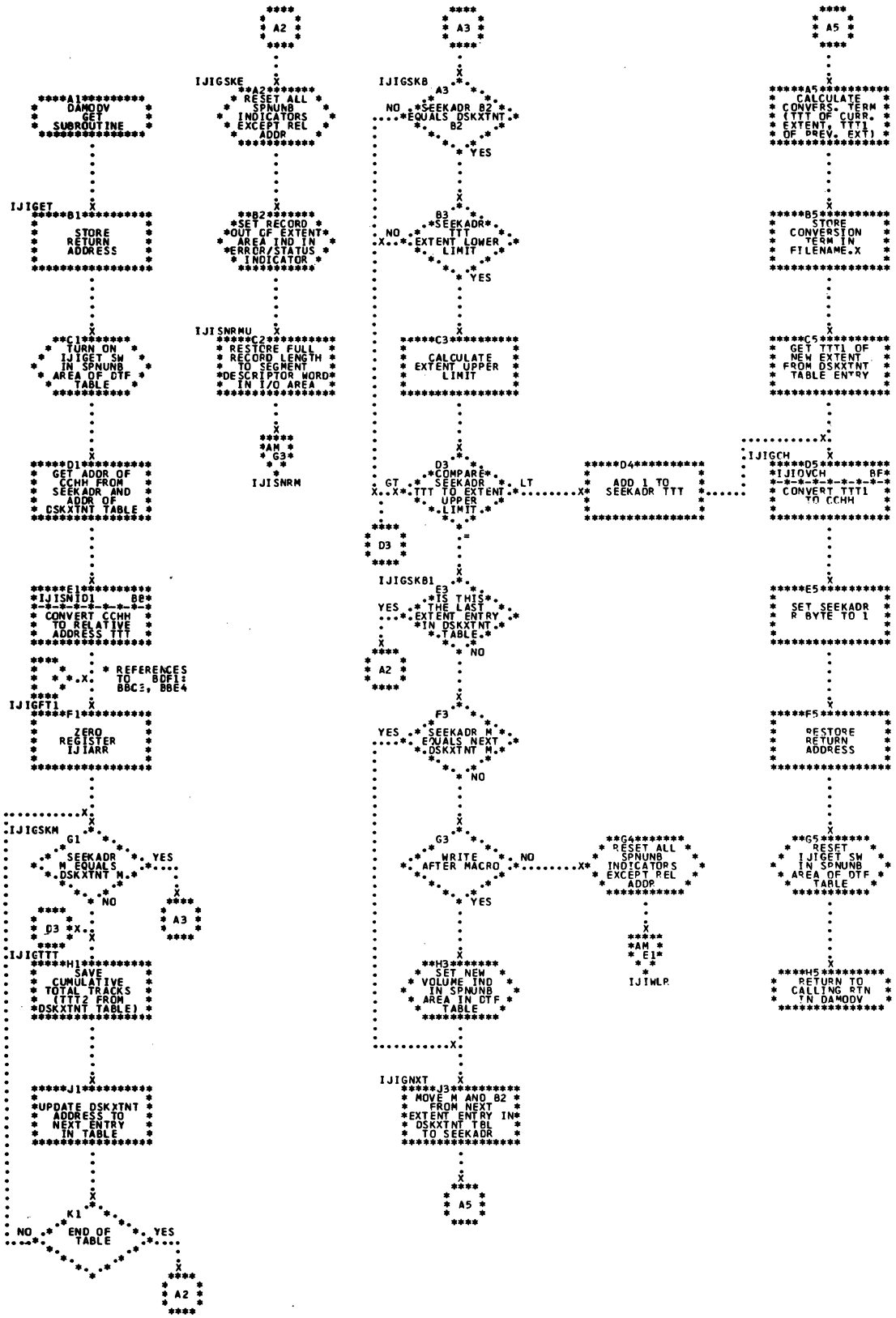


Chart BE. DAMODV: Seek Overlap Subroutine (Part 1 of 2)

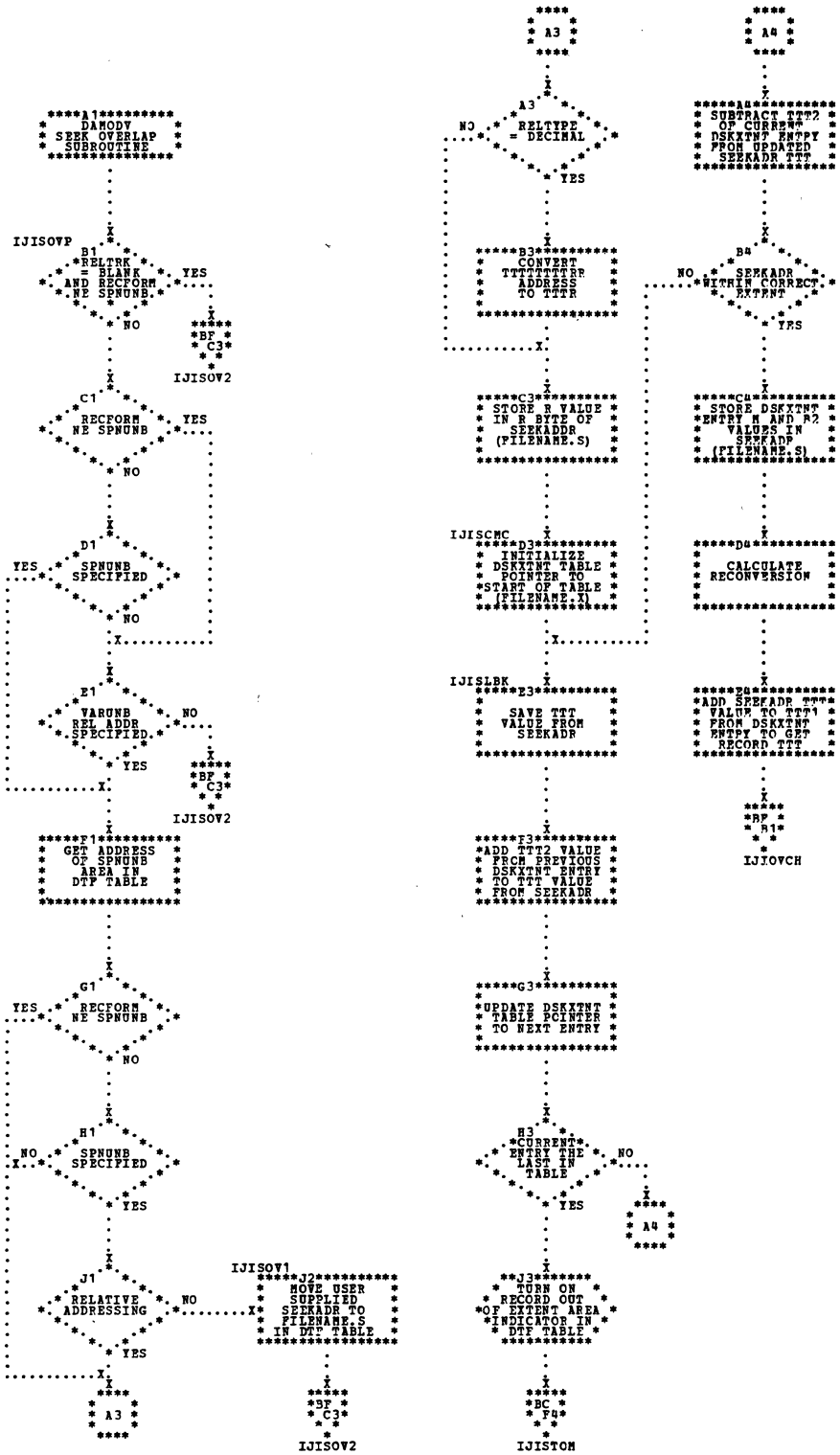


Chart BF. DAMODV: Seek Overlap Subroutine (Part 2 of 2)

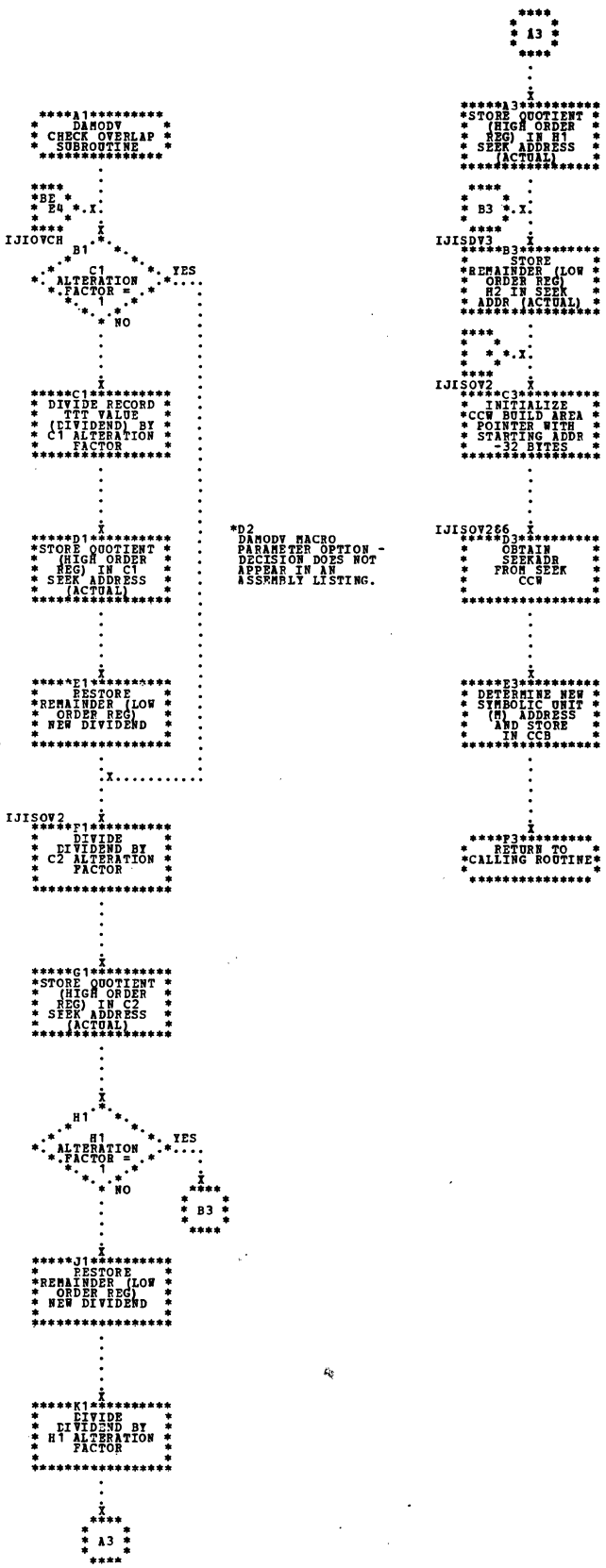


Chart CA. $$BODACL: DA Close Input/Output (Part 1 of 3)

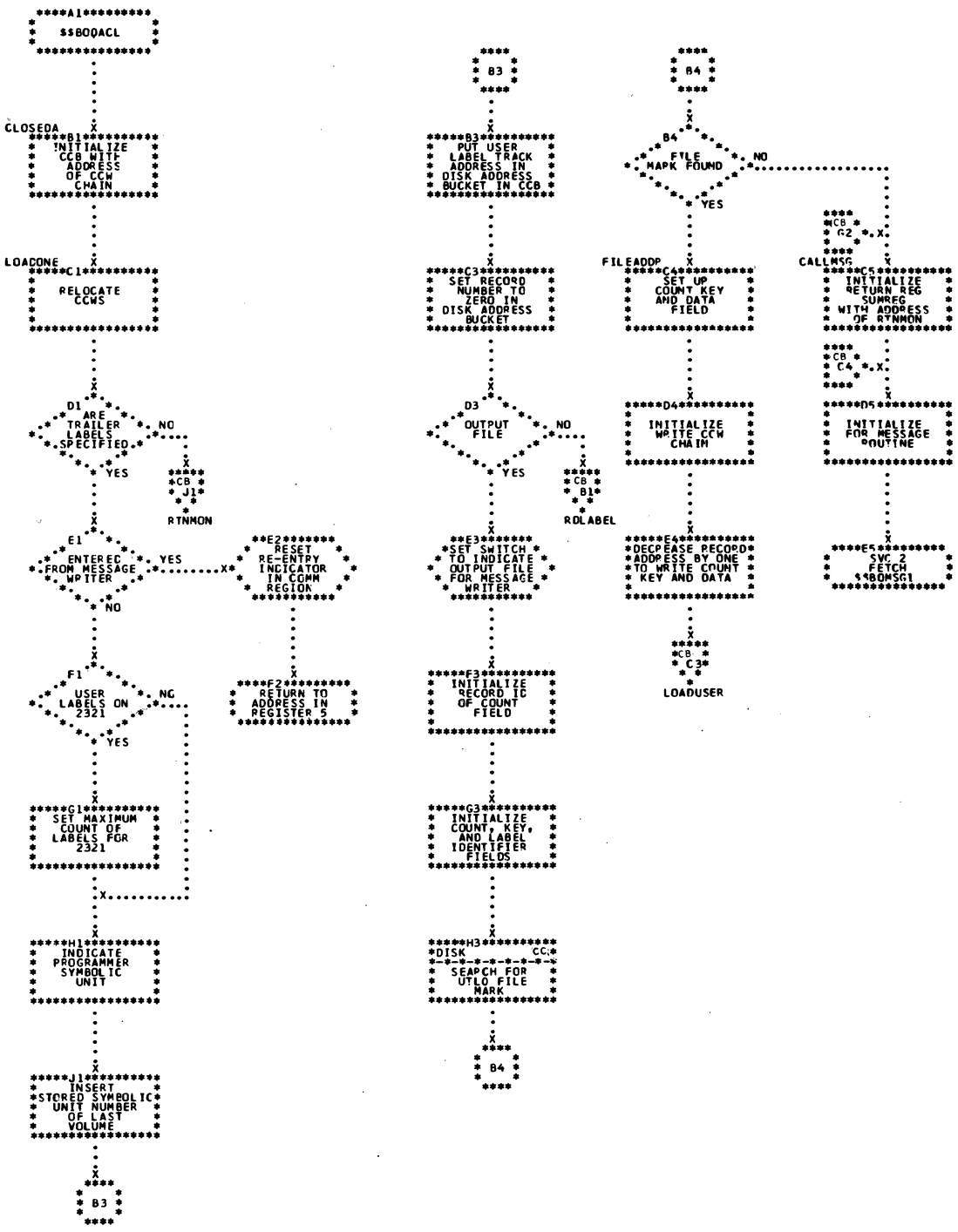


Chart CB. $$BODACL: DA Close Input/Output (Part 2 of 3)

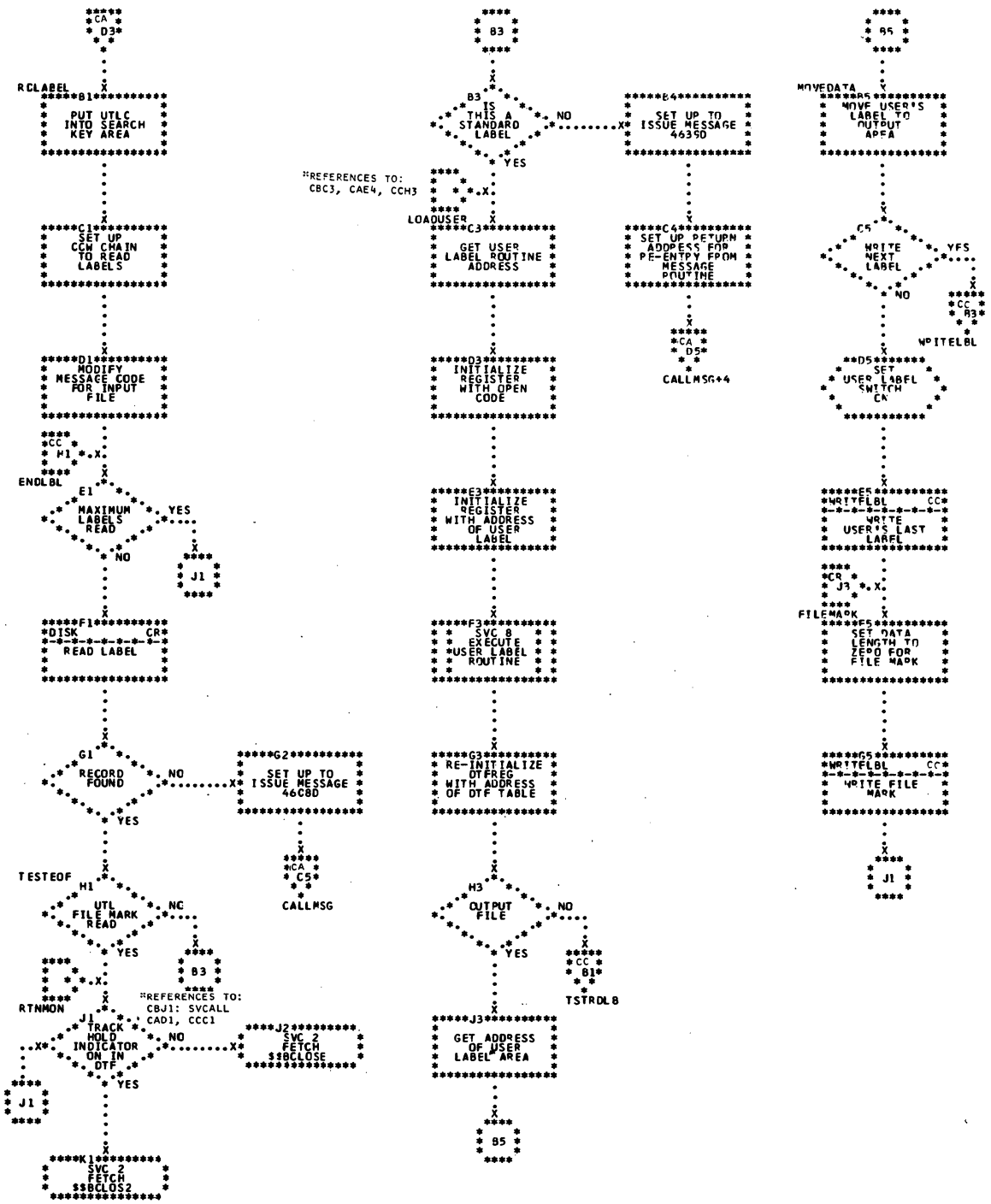
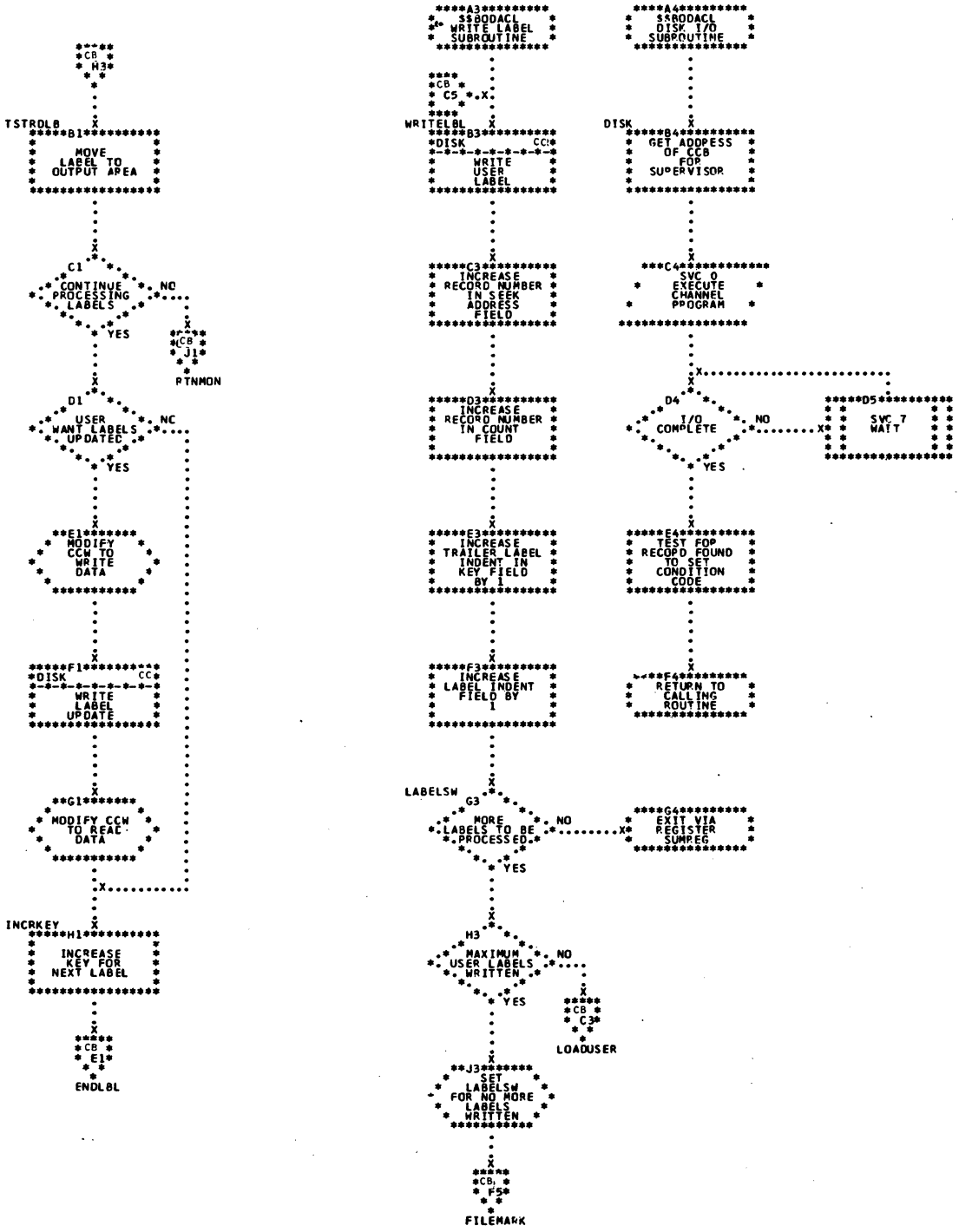


Chart CC. $$BODACL: DA Close Input/Output (Part 3 of 3)



ISAM CHARTS

Chart DA. ISAM LOAD: ENDFL Macro, Phase 1, $$BENDFL (Part 1 of 2)

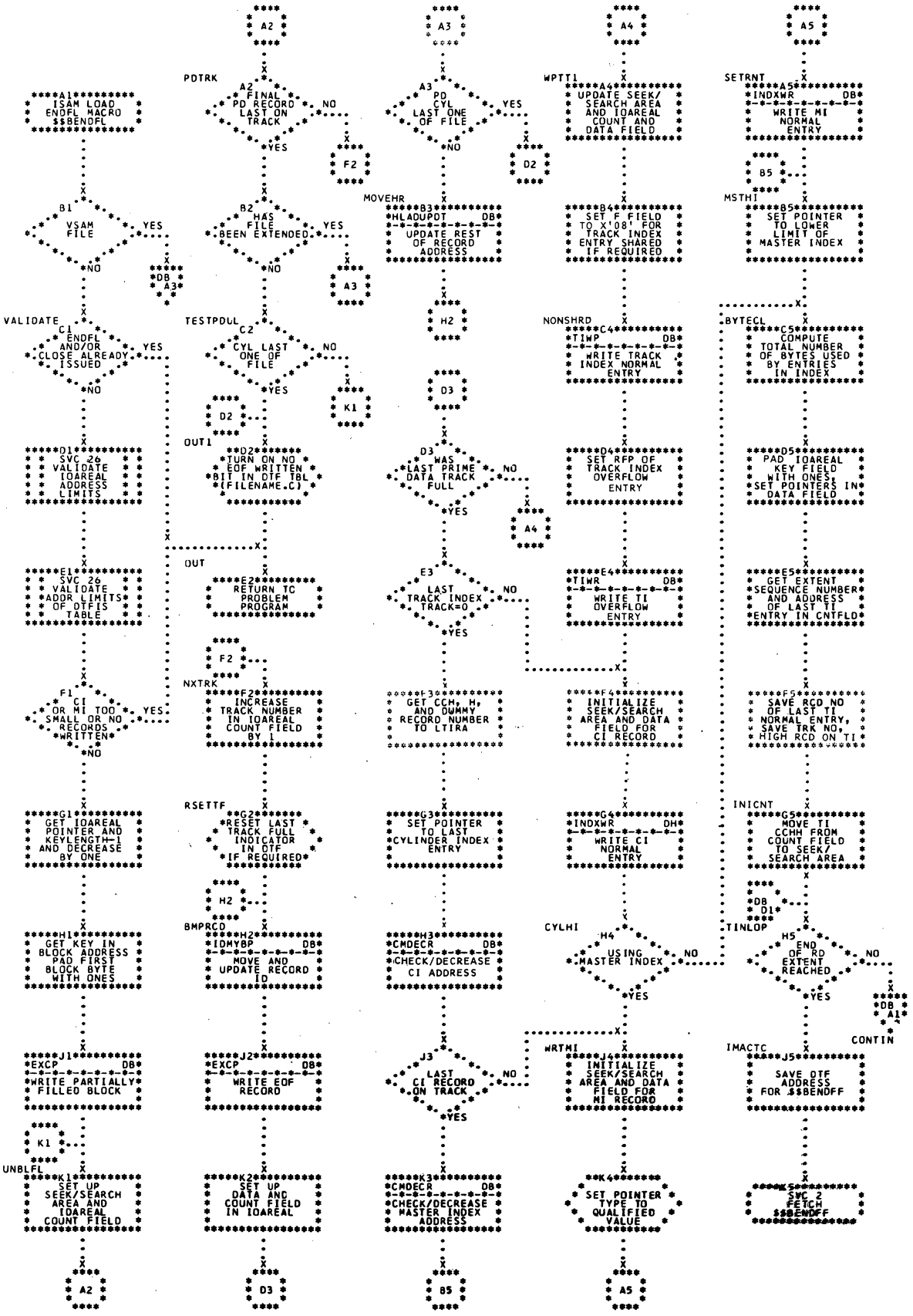


Chart DB. ISAM LOAD: ENDFL Macro, Phase 1, $$BENDFL (Part 2 of 2)

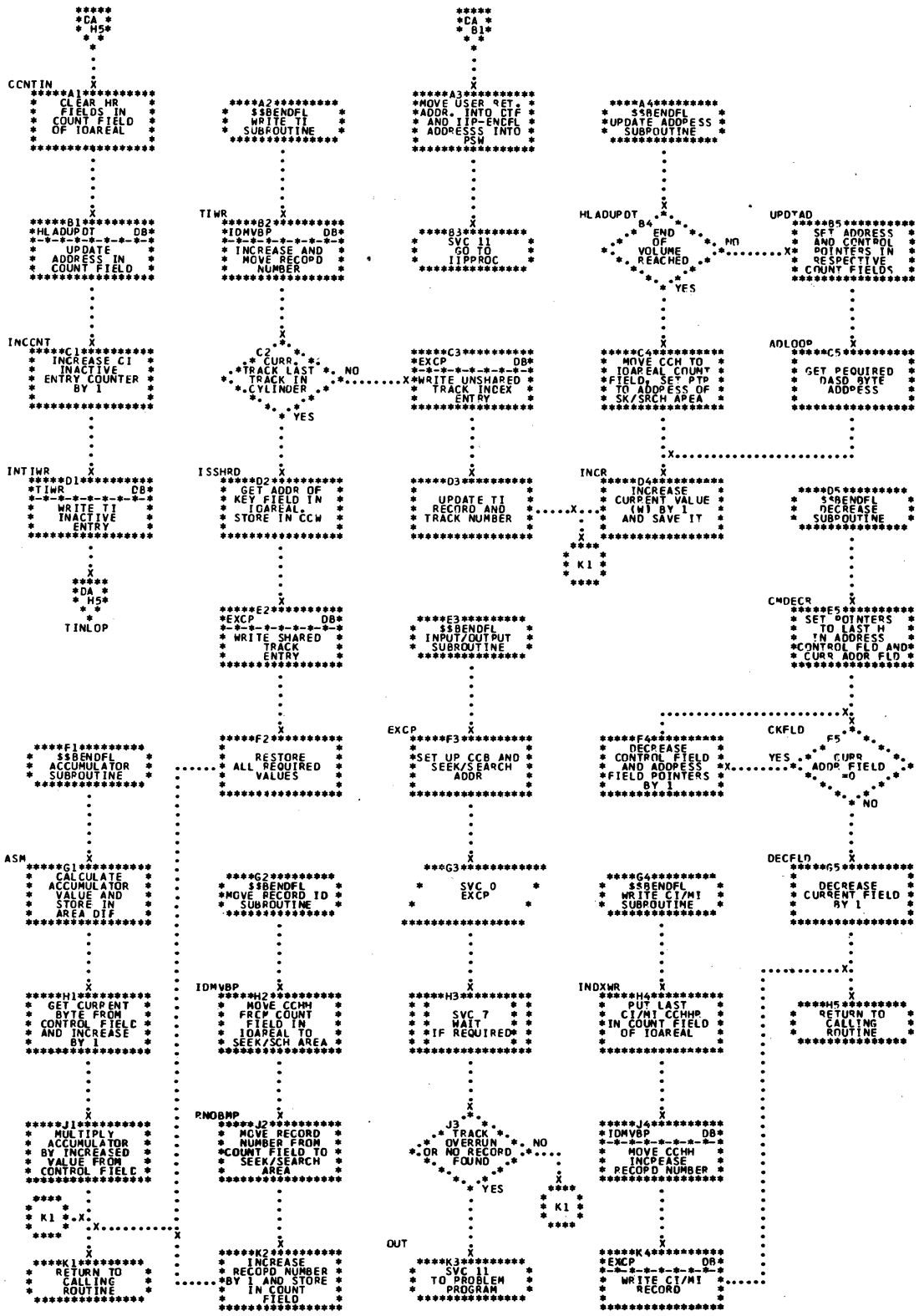


Chart DC. ISAM LOAD: ENDFL Macro, Phase 2, $$BENDFF (Part 1 of 2)

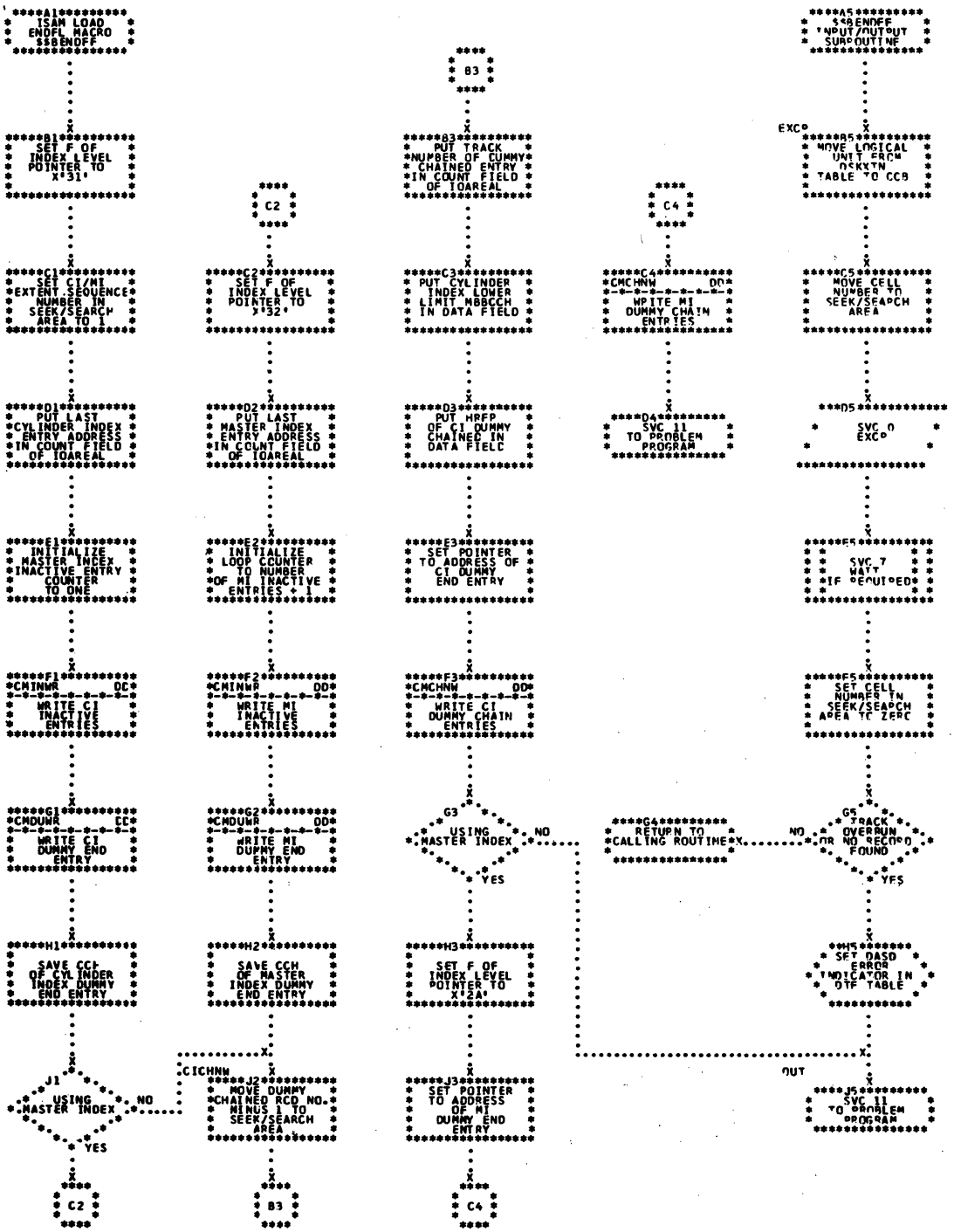


Chart DD. ISAM LOAD: ENDFL Macro, Phase 2, $$BENDFF (Part 2 of 2)

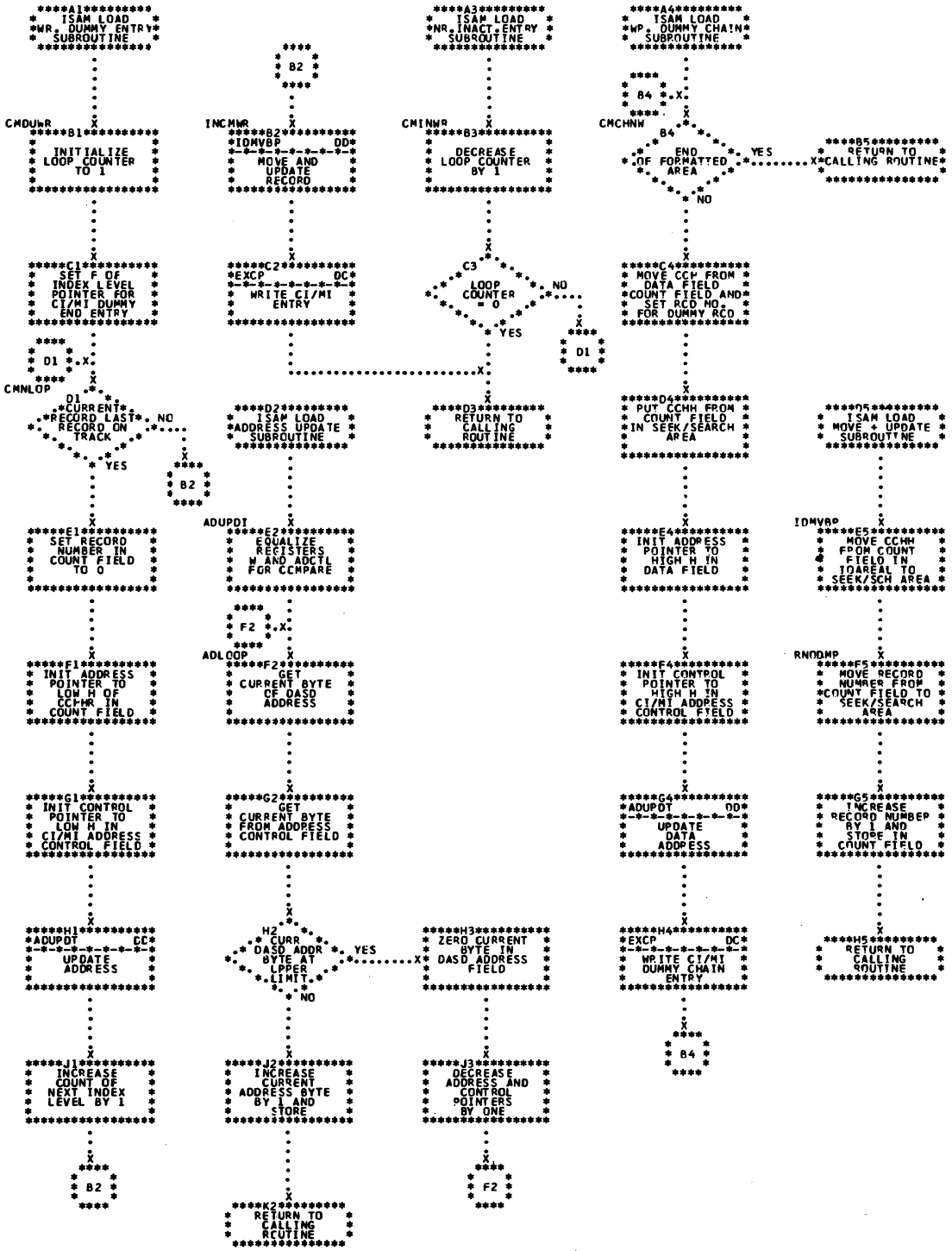


Chart DE. ISAM LOAD: SETFL Macro, Phase 1, $$BSETFL (Part 1 of 2)

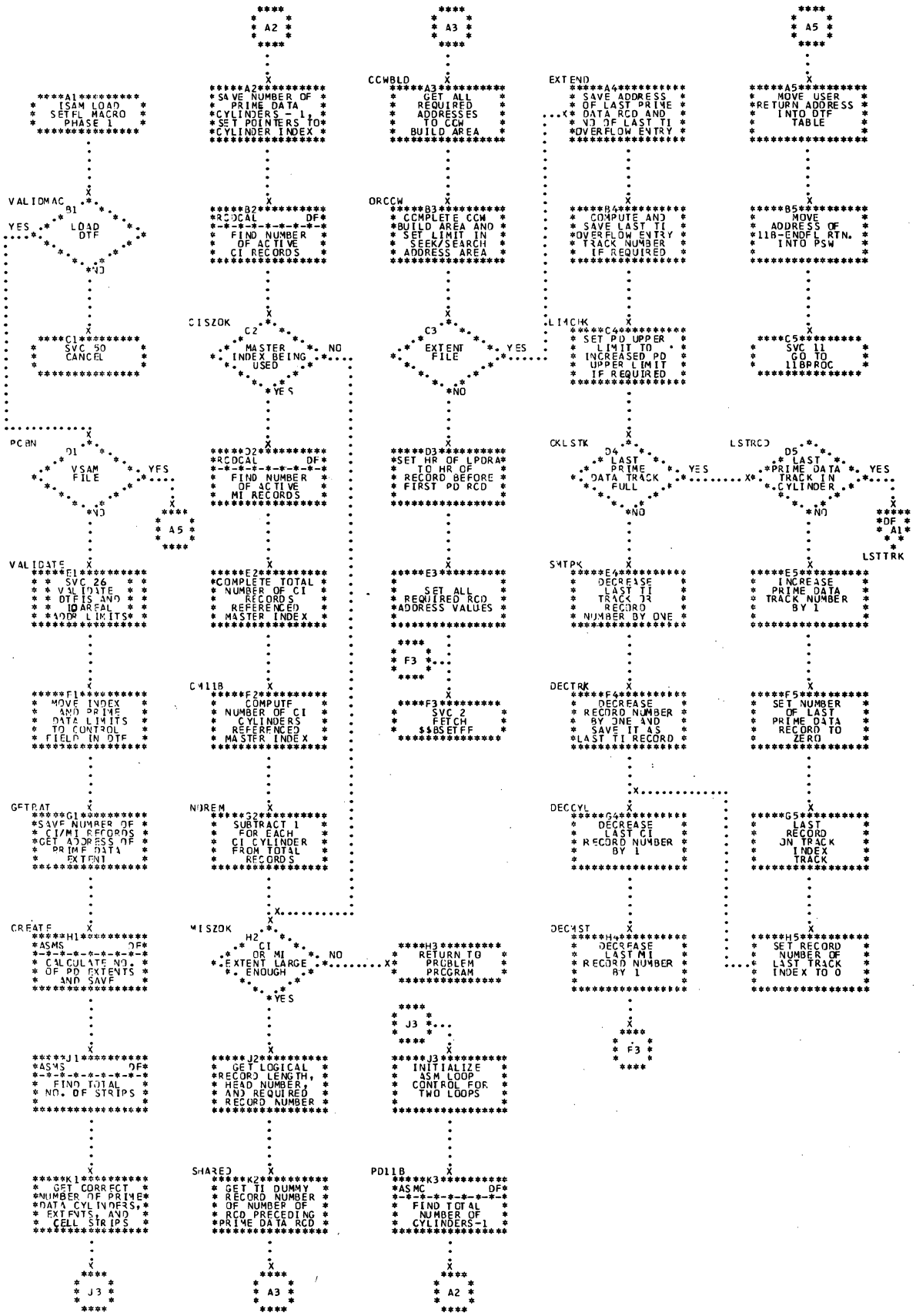


Chart DF. ISAM LOAD: SETFL Macro, Phase 1, $$BSETFL (Part 2 of 2)

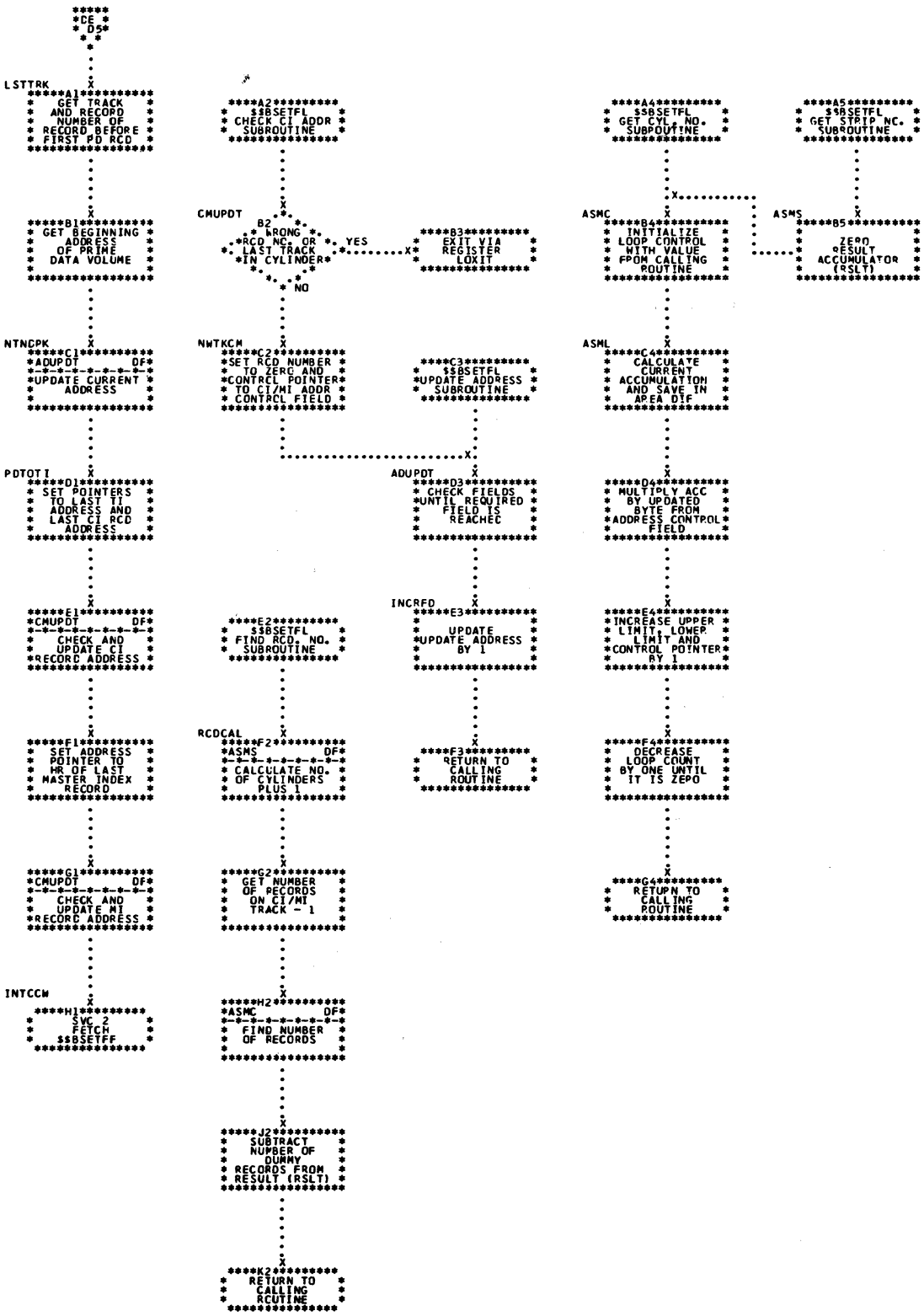


Chart DG. ISAM LOAD: SETFL Macro, Phase 2, $$BSETFF

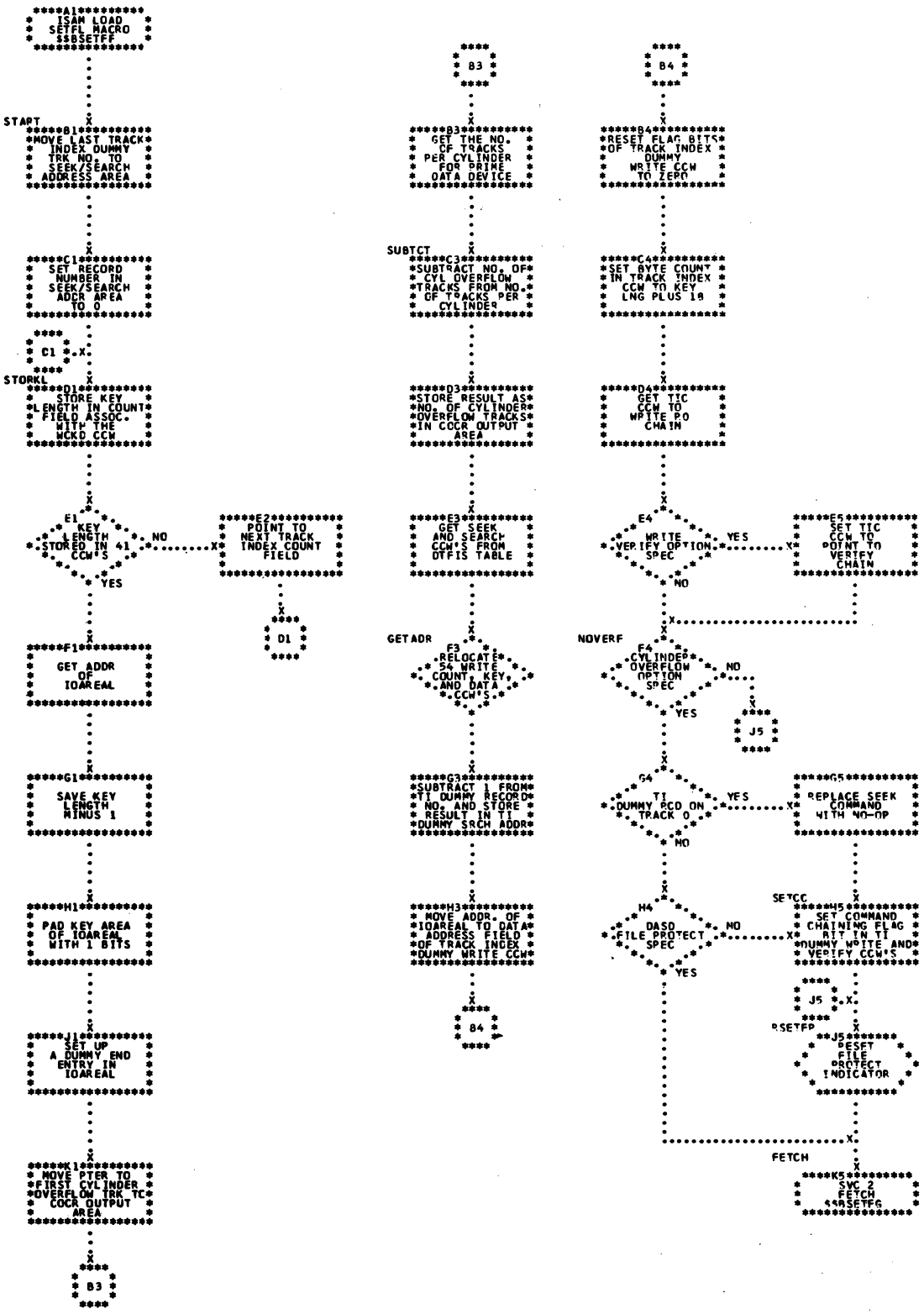


Chart DH. ISAM LOAD: SETFL Macro, Phase 3, $$BSETFG

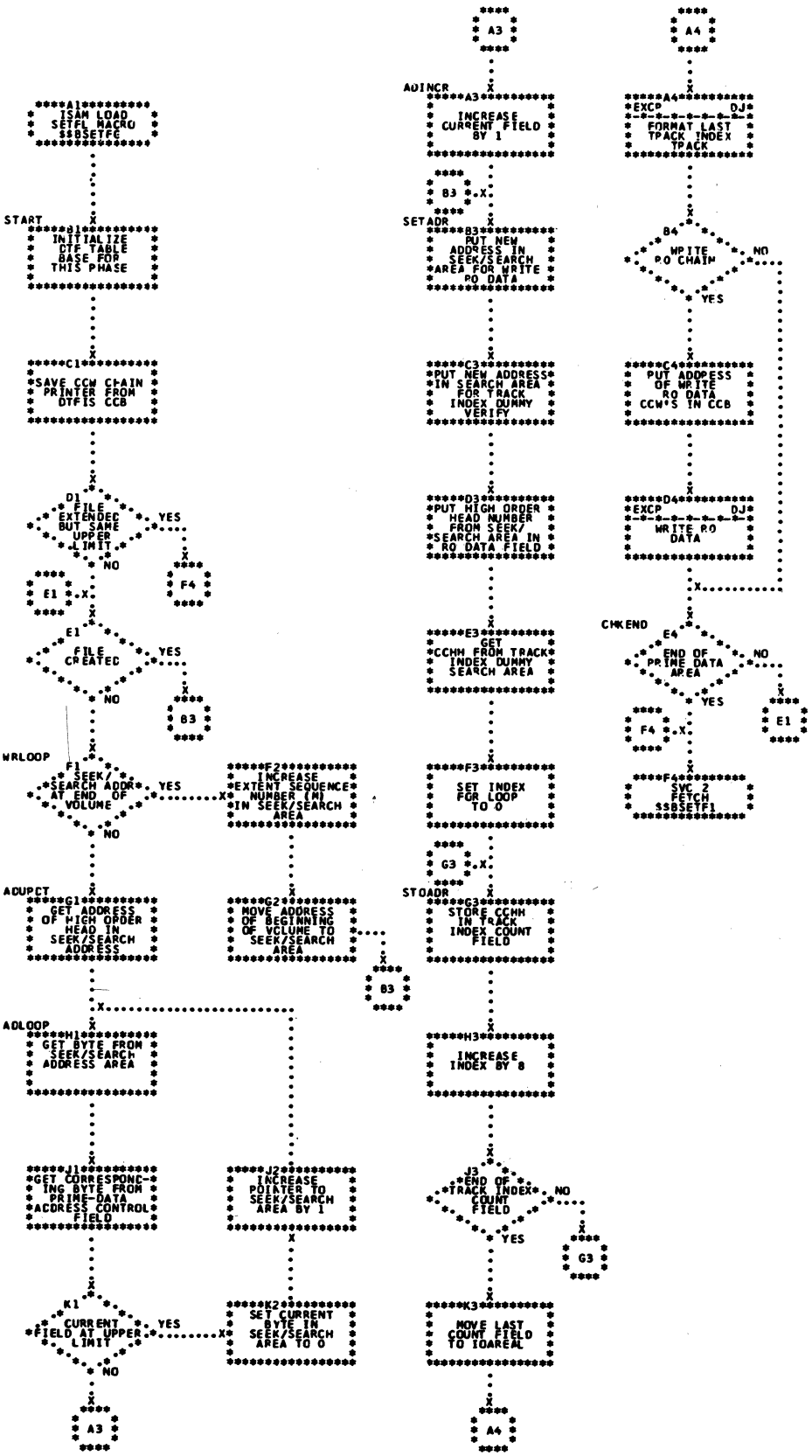


Chart DJ1 ISAM LOAD: SETFL Macro, Phase 4, $$BSETFH

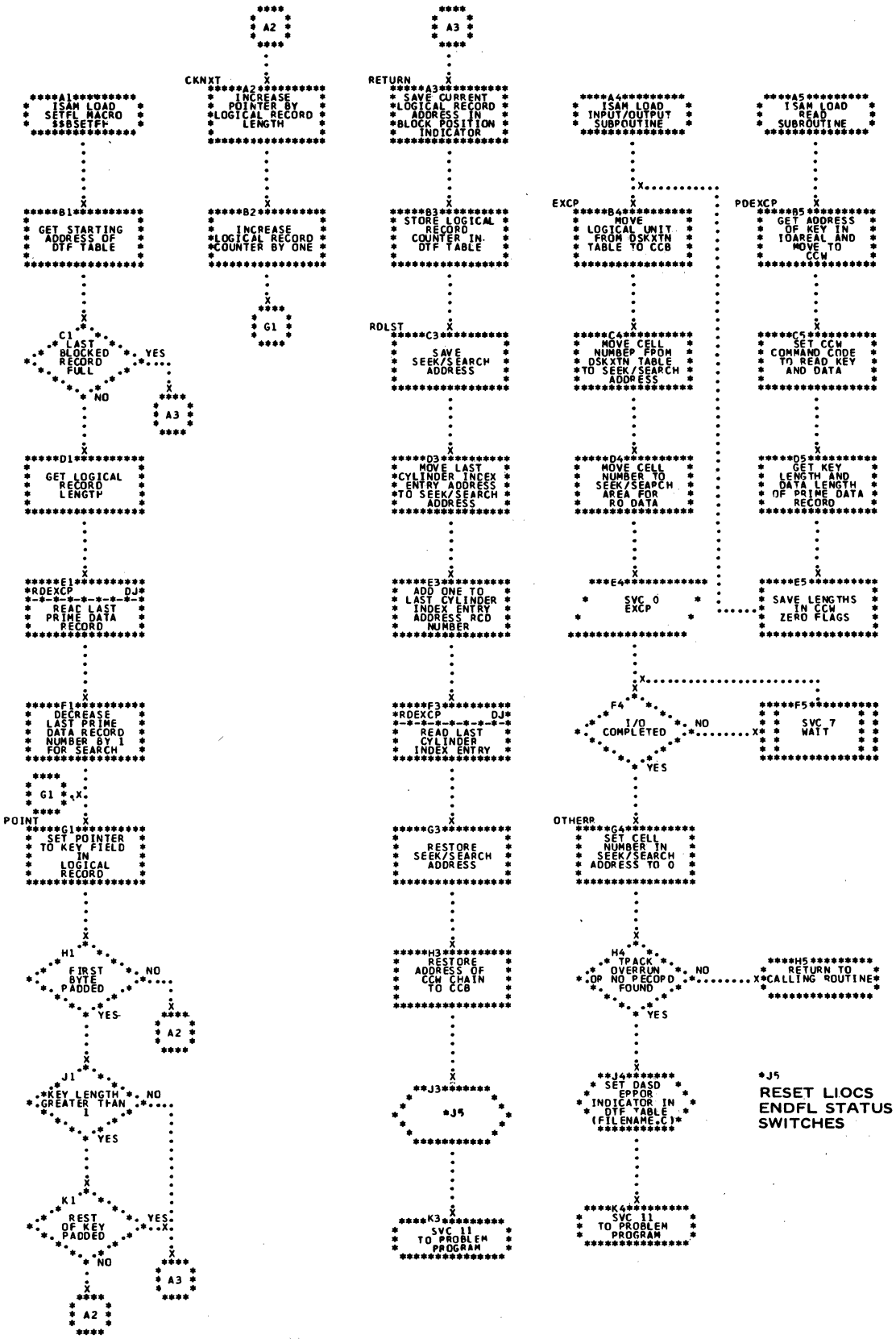


Chart DK. ISAM LOAD: SETFL Macro, Phase 3A, $$BSETFI

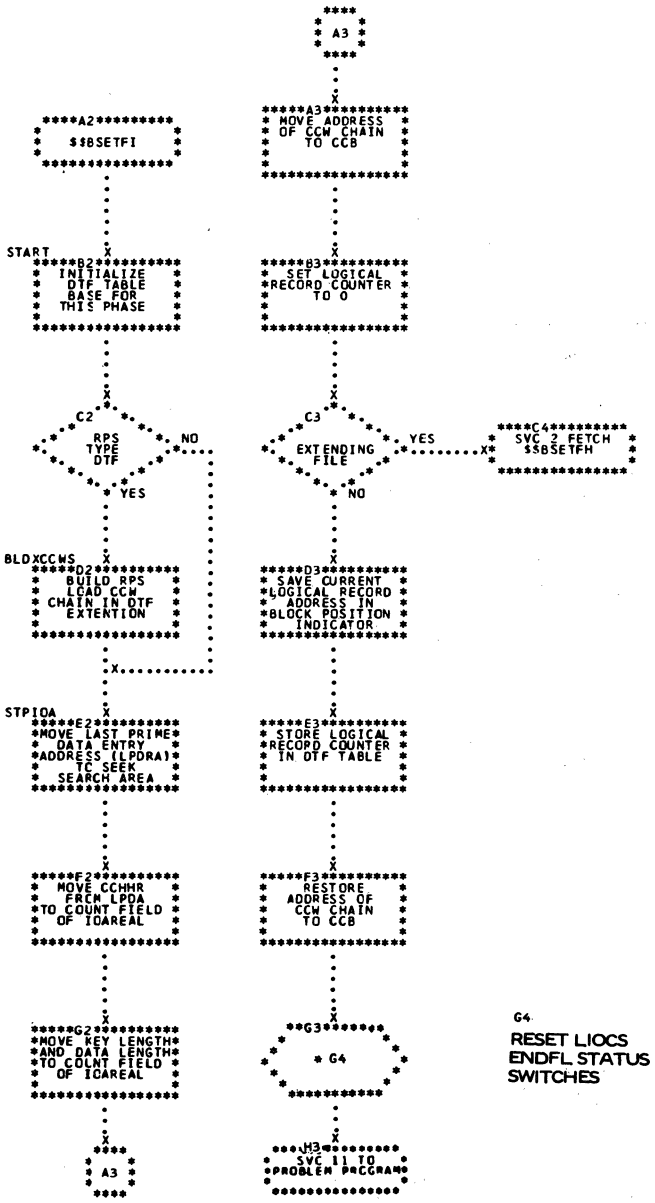


Chart DL. ISAM LOAD: WRITE Macro, NEWKEY (Part 1 of 4)

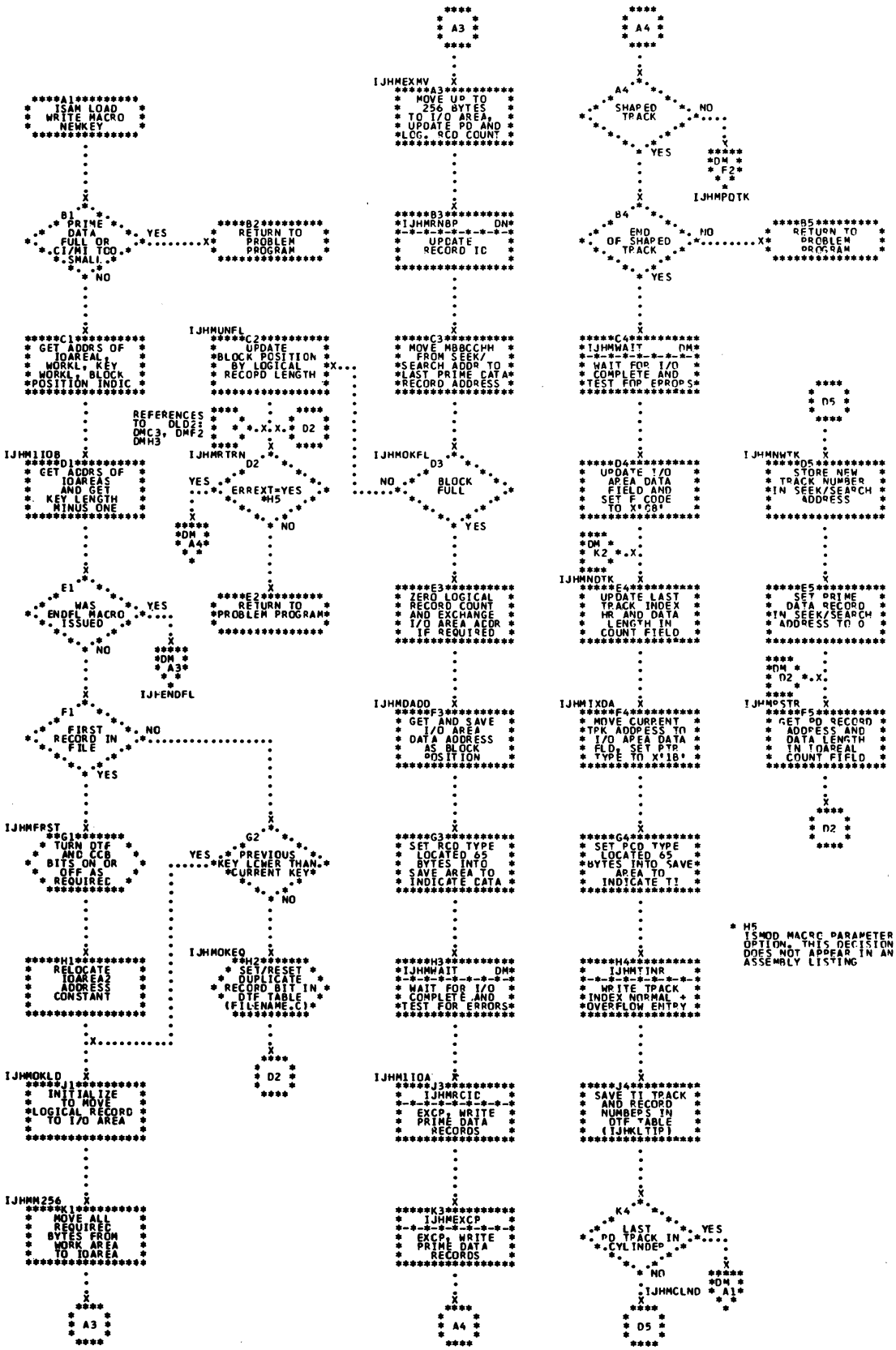


Chart DM. ISAM LOAD: WRITE Macro, NEWKEY (Part 2 of 4)

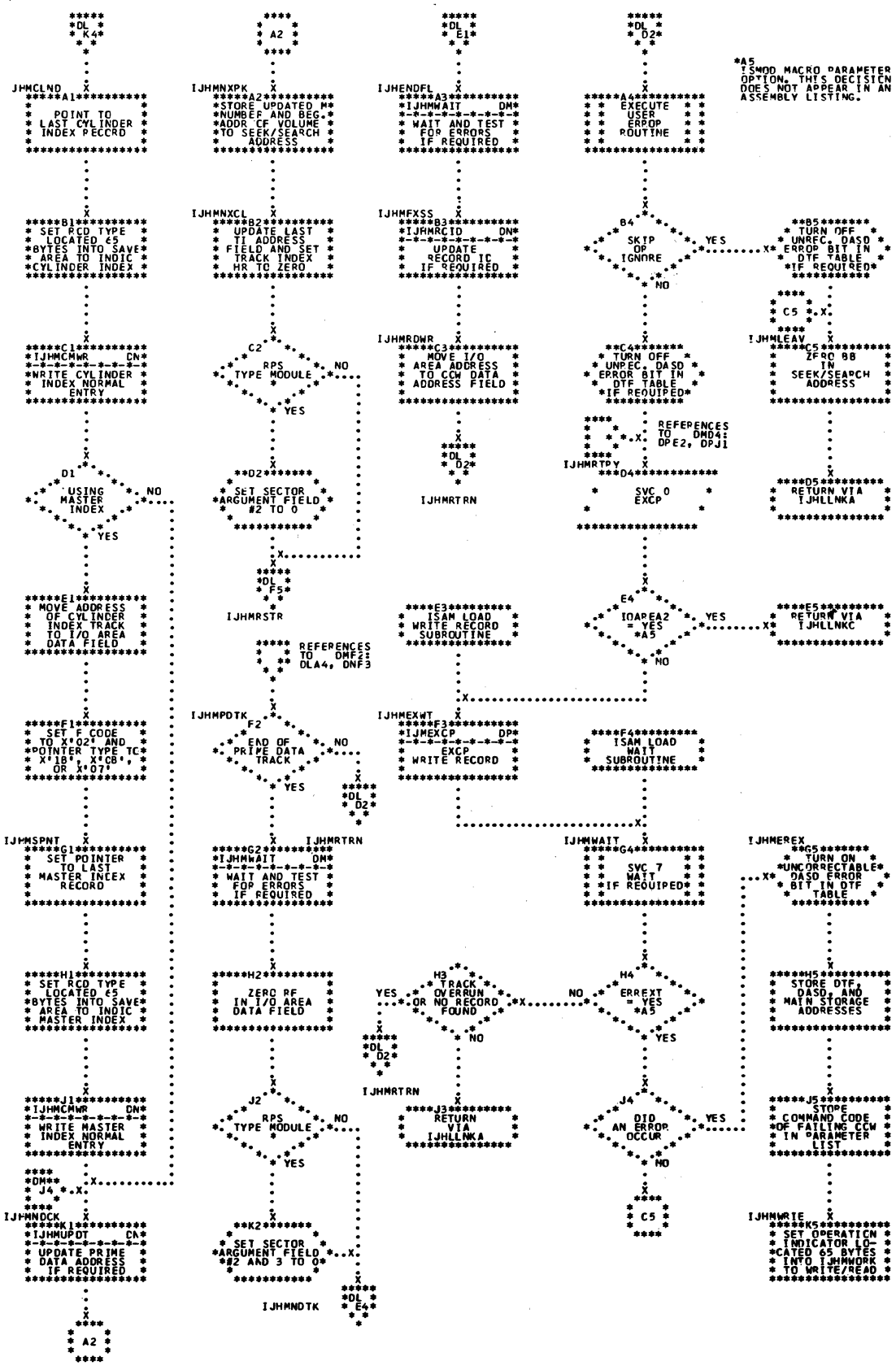


Chart DN. ISAM LOAD: WRITE Macro, NEWKEY (Part 3 of 4)

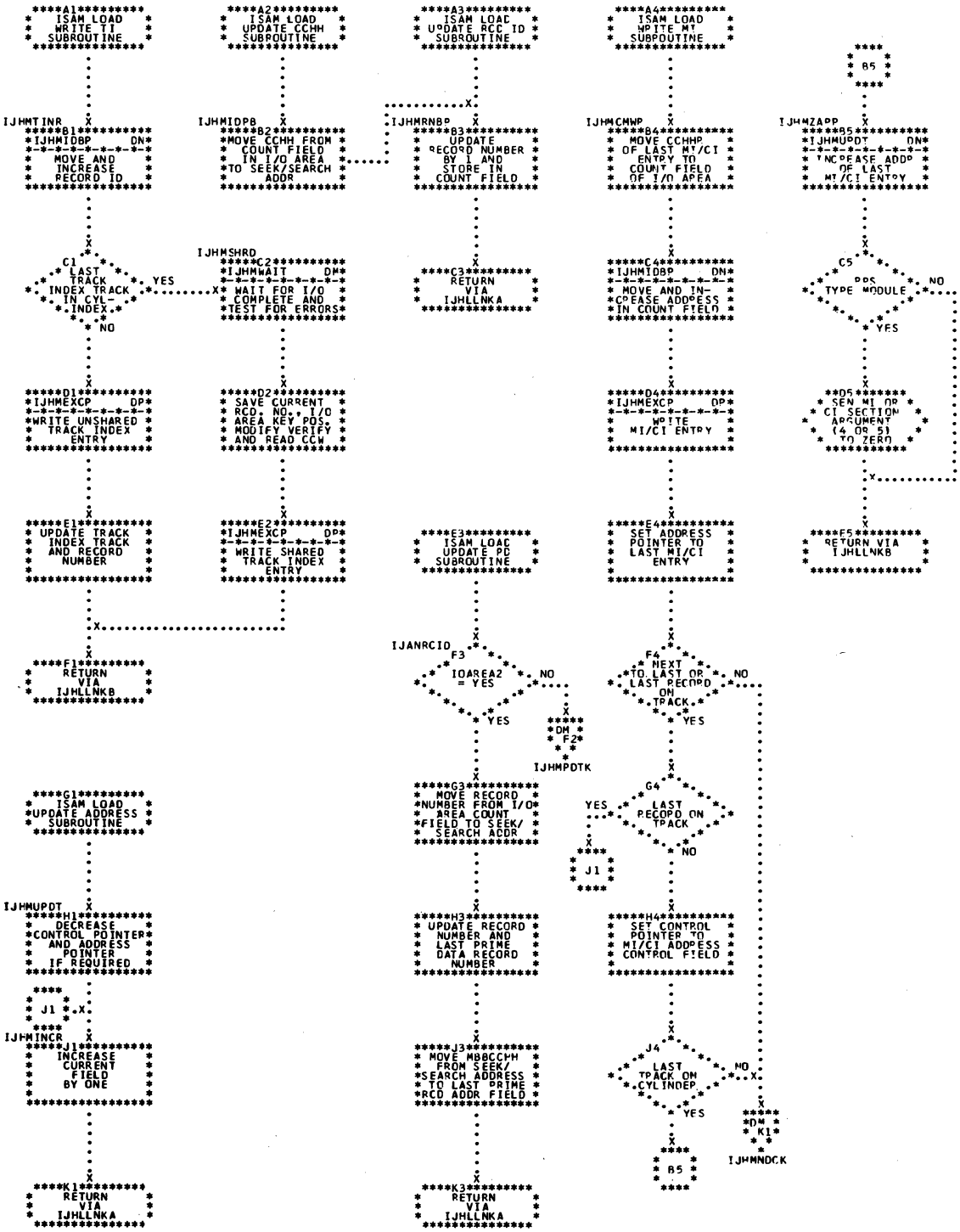


Chart DP. ISAM LOAD: WRITE Macro, NEWKEY (Part 4 of 4)

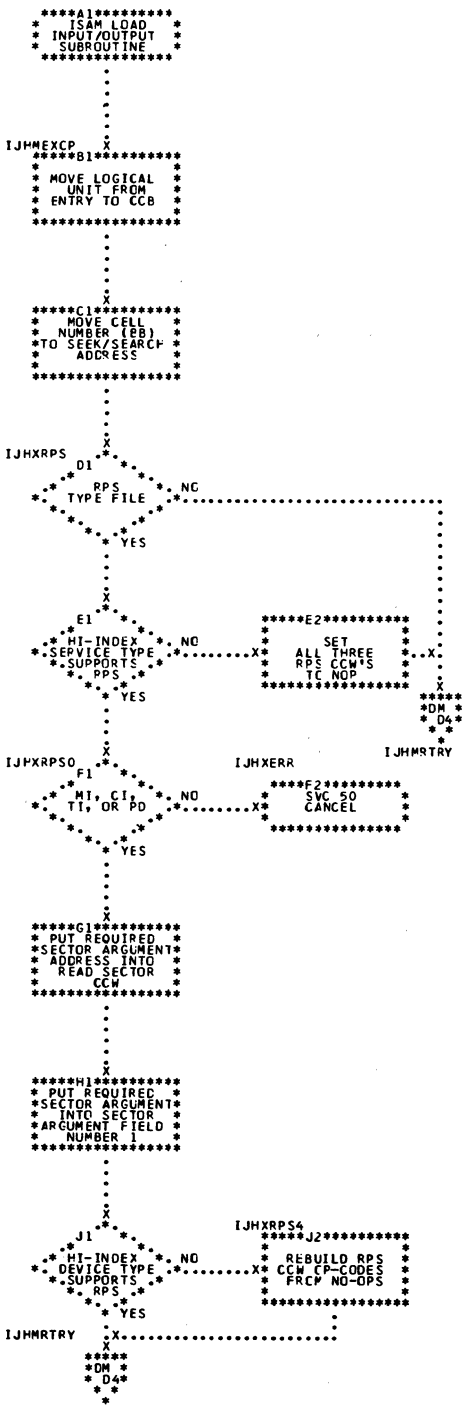


Chart EA. ISAM ADD: WAITF Macro (Part 1 of 4)

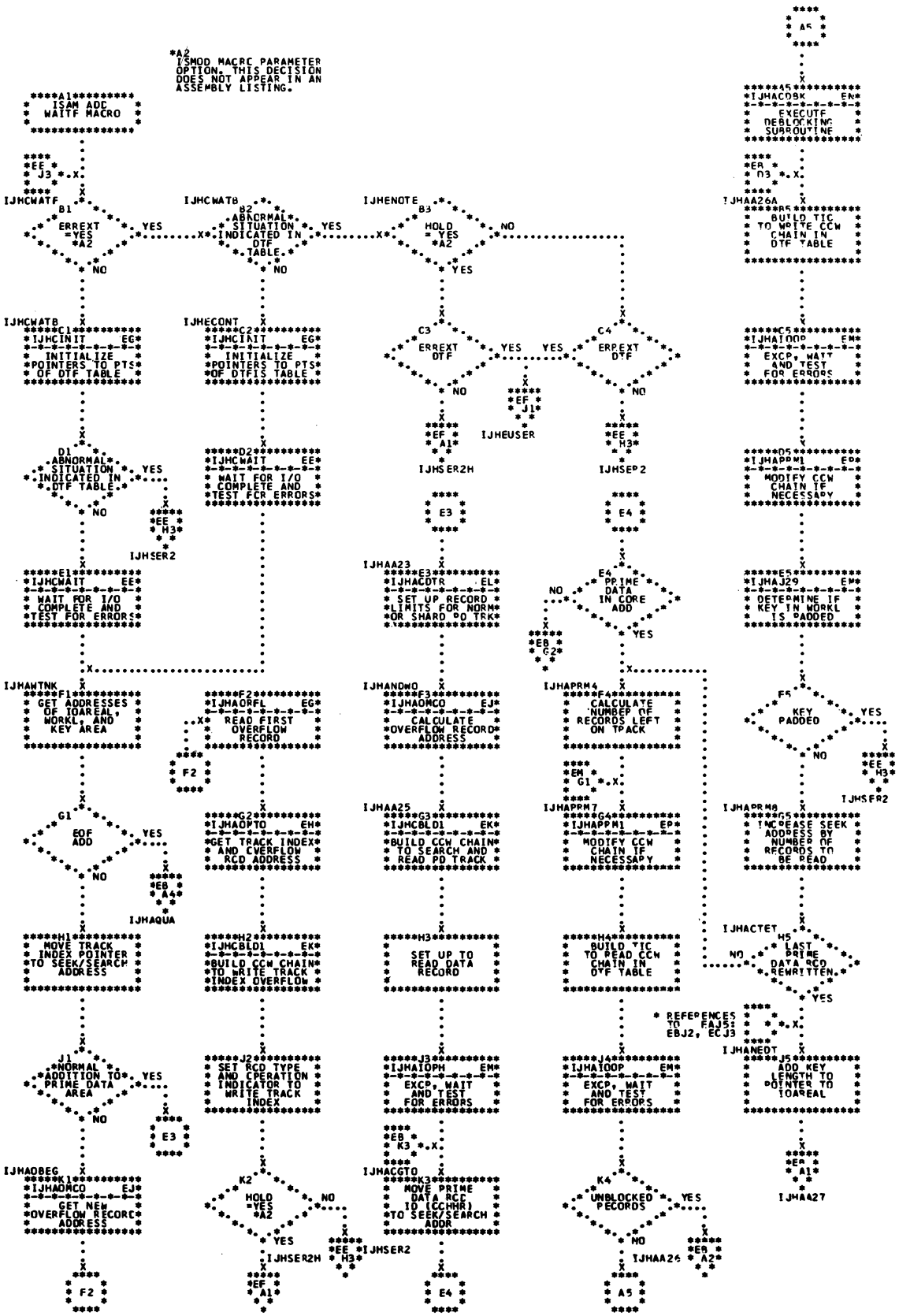


Chart EB. ISAM ADD: WAITF Macro (Part 2 of 4)

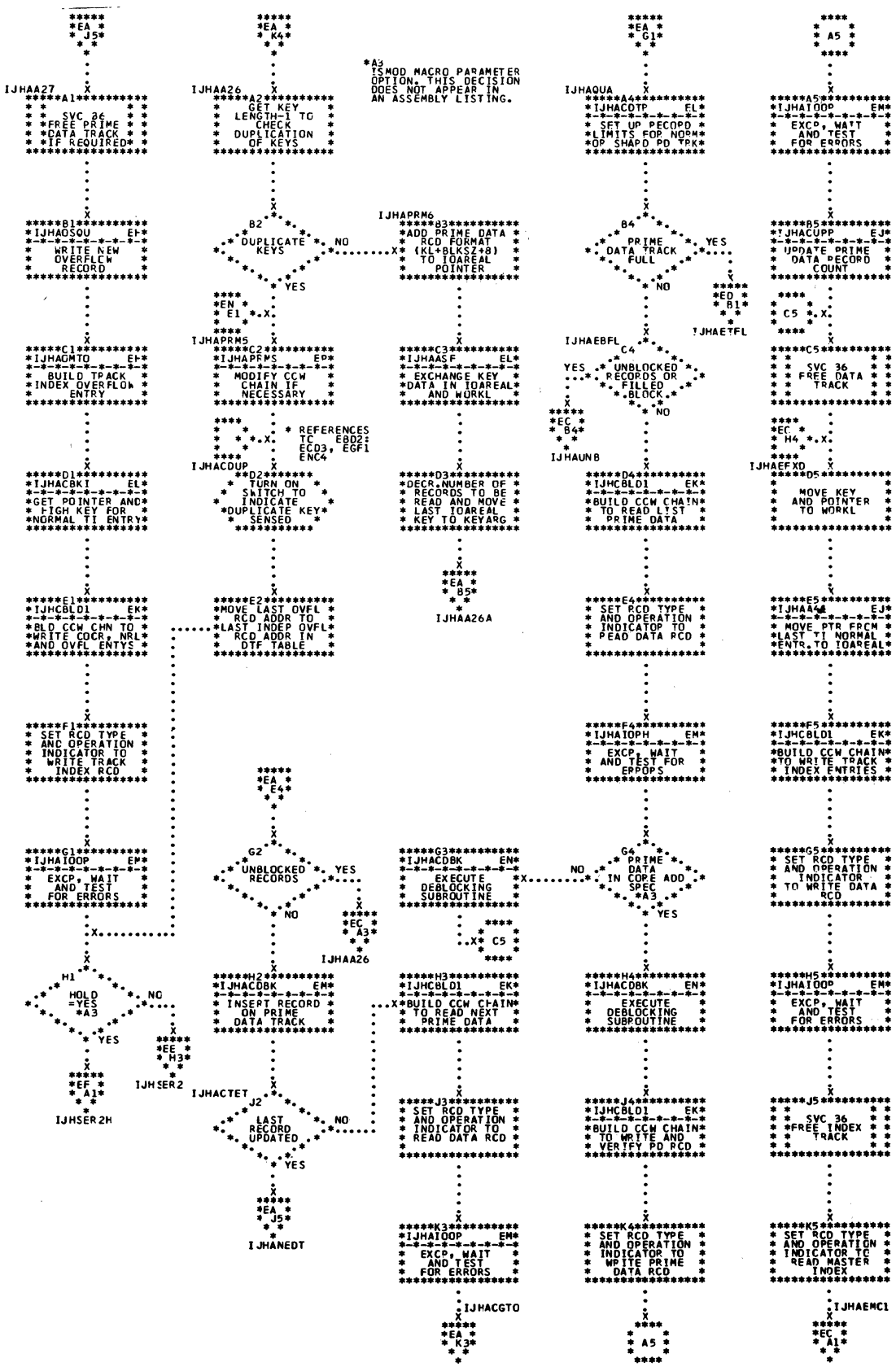


Chart EC. ISAM ADD: WAITF Macro (Part 3 of 4)

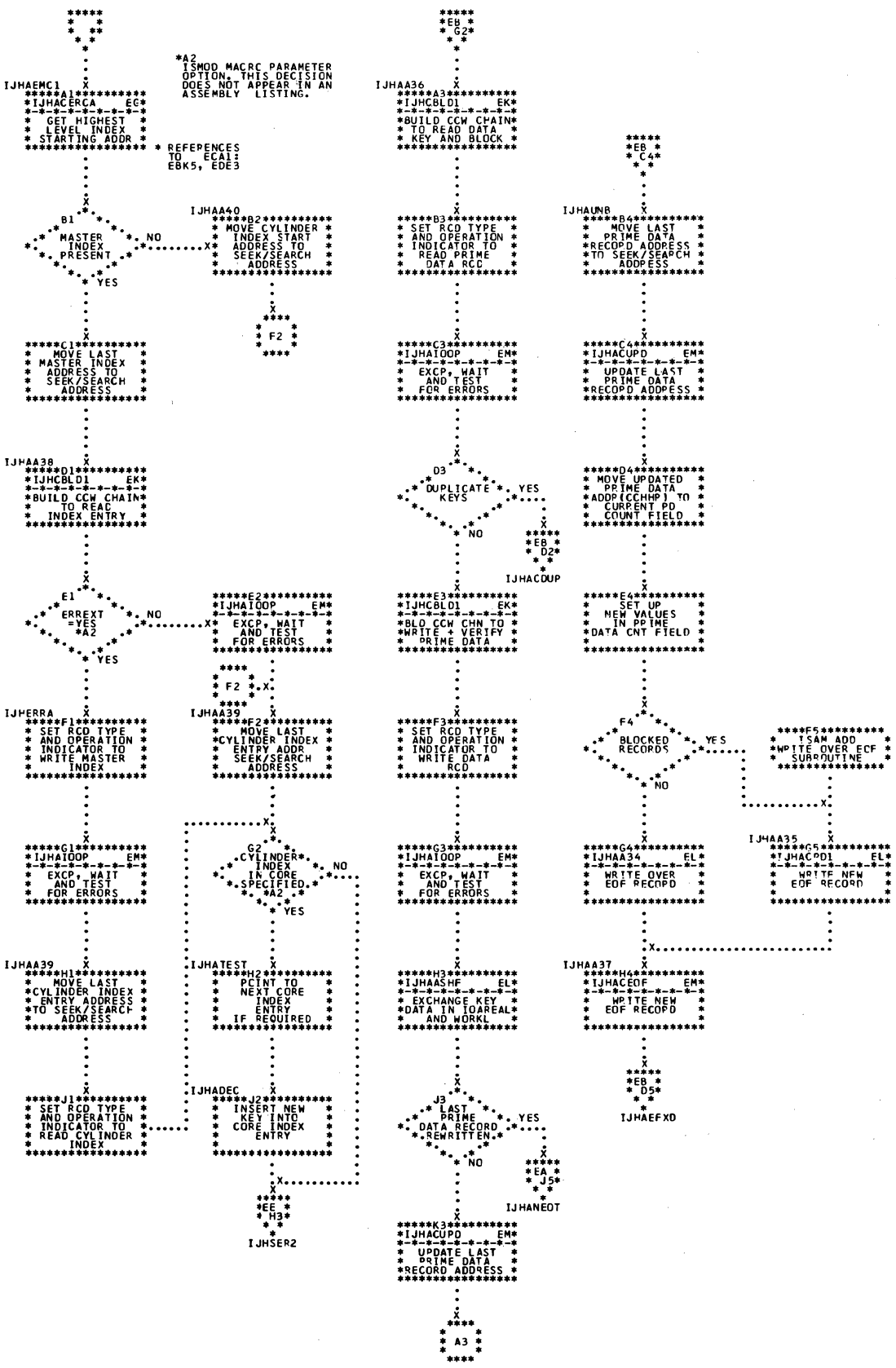


Chart ED. ISAM ADD: WAITF Macro (Part 4 of 4)

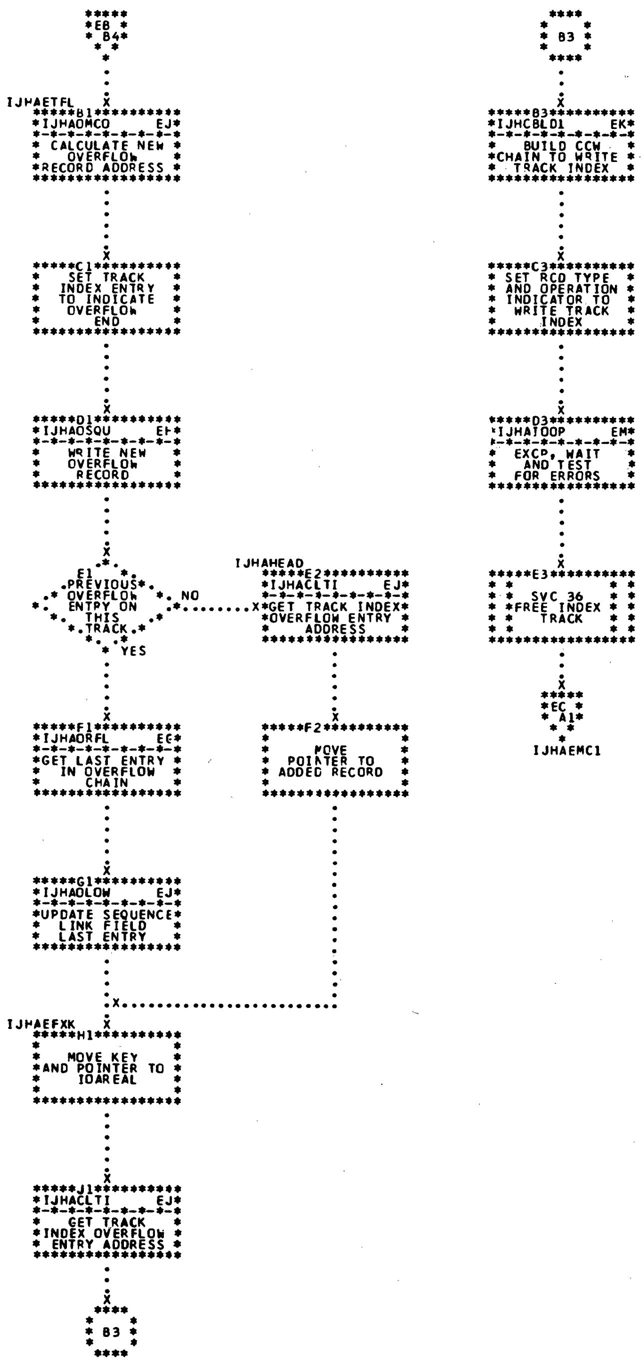


Chart EE. ISAM ADD and ADDRTR: WRITE Macro, NEWKEY (Part 1 of 2)

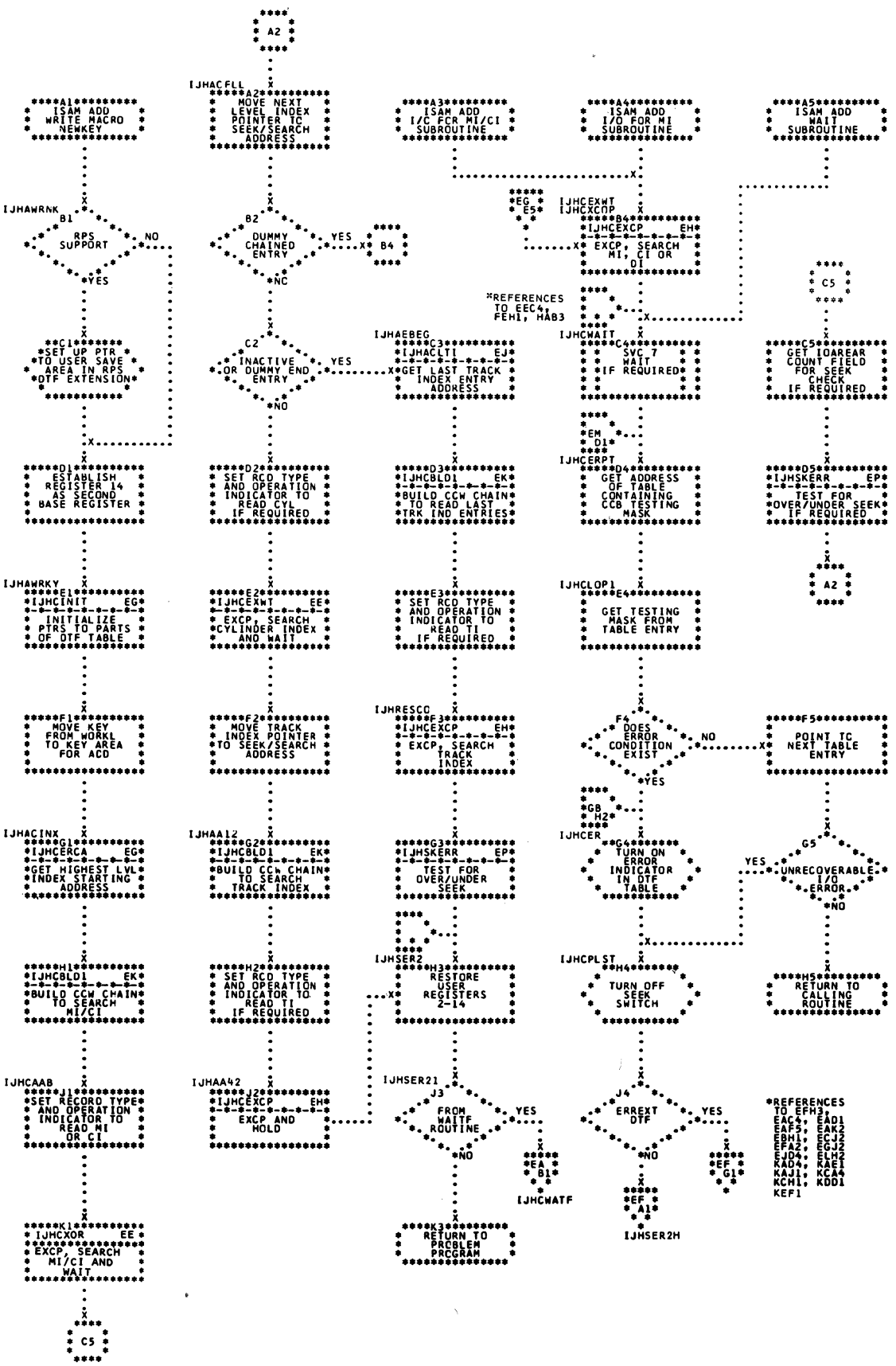


Chart EF. ISAM ADD and ADDRTR: WRITE Macro, NEWKEY (Part 2 of 2)

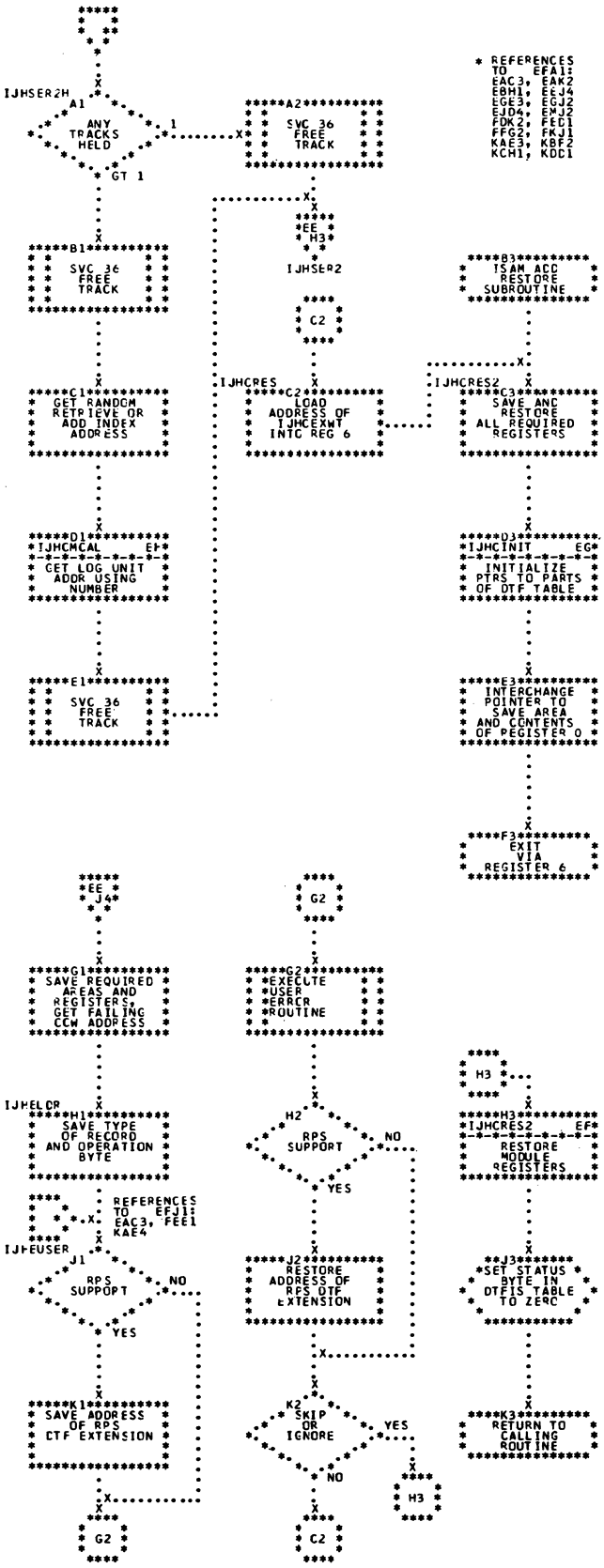


Chart EG. ISAM ADD, RETRVE, and ADDRTR: Subroutines (Part 1 of 8)

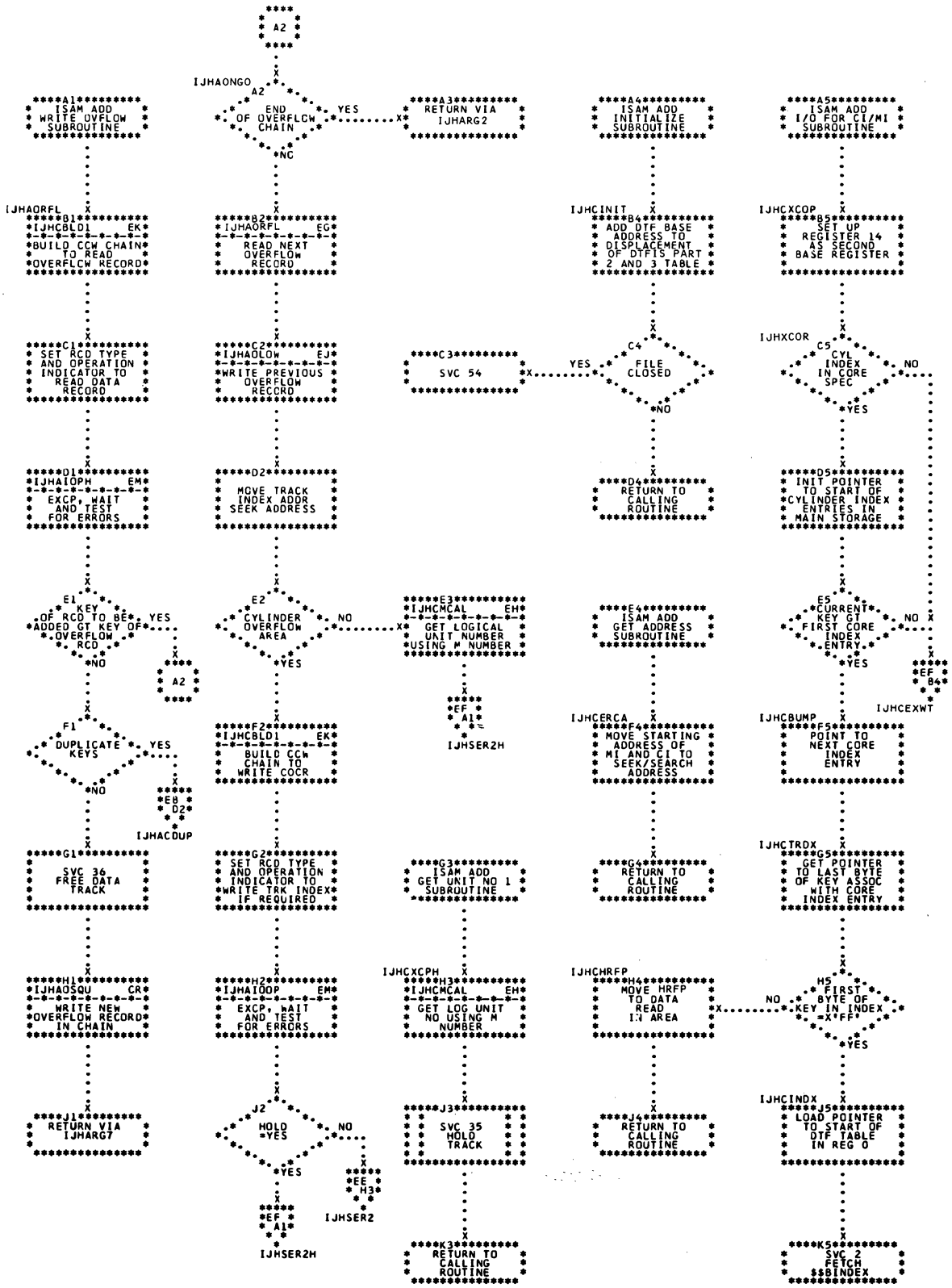


Chart EH. ISAM ADD, RETRVE, and ADDRTR: Subroutines (Part 2 of 8)

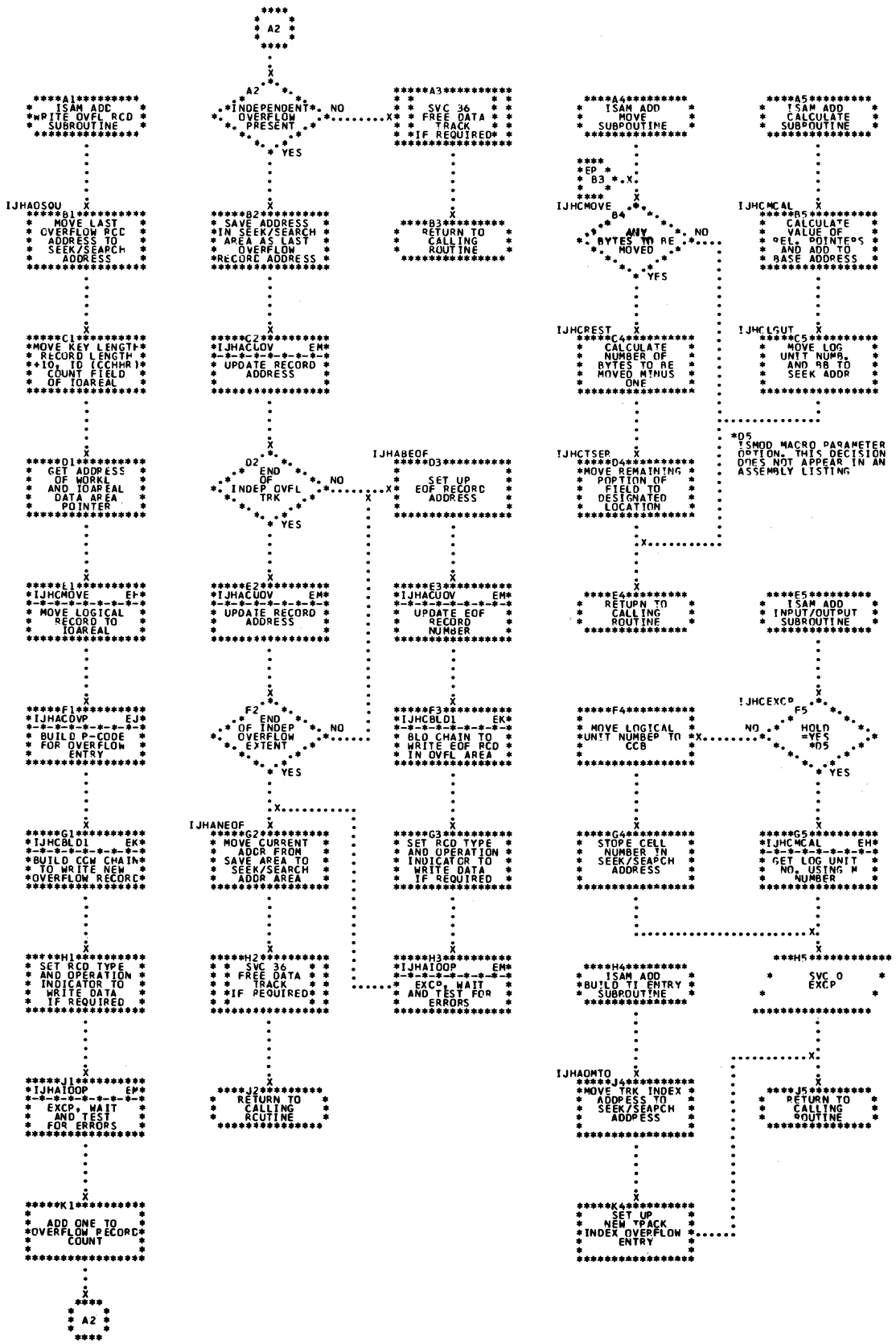


Chart EJ. ISAM ADD, RETRVE, and ADDRTR: Subroutines (Part 3 of 8)

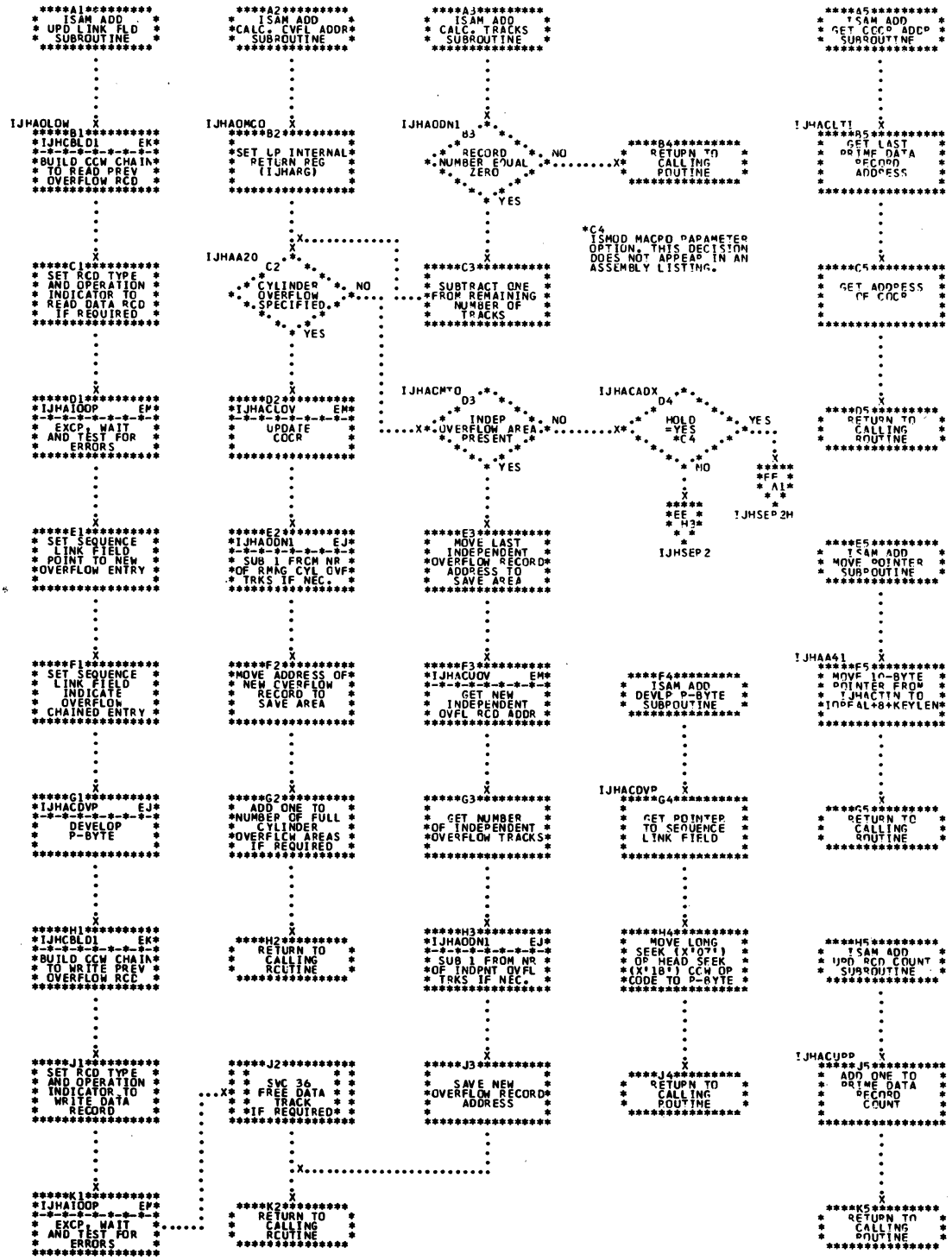


Chart EK. ISAM ADD, RETRVE, and ADDRTR: Subroutines (Part 4 of 8)

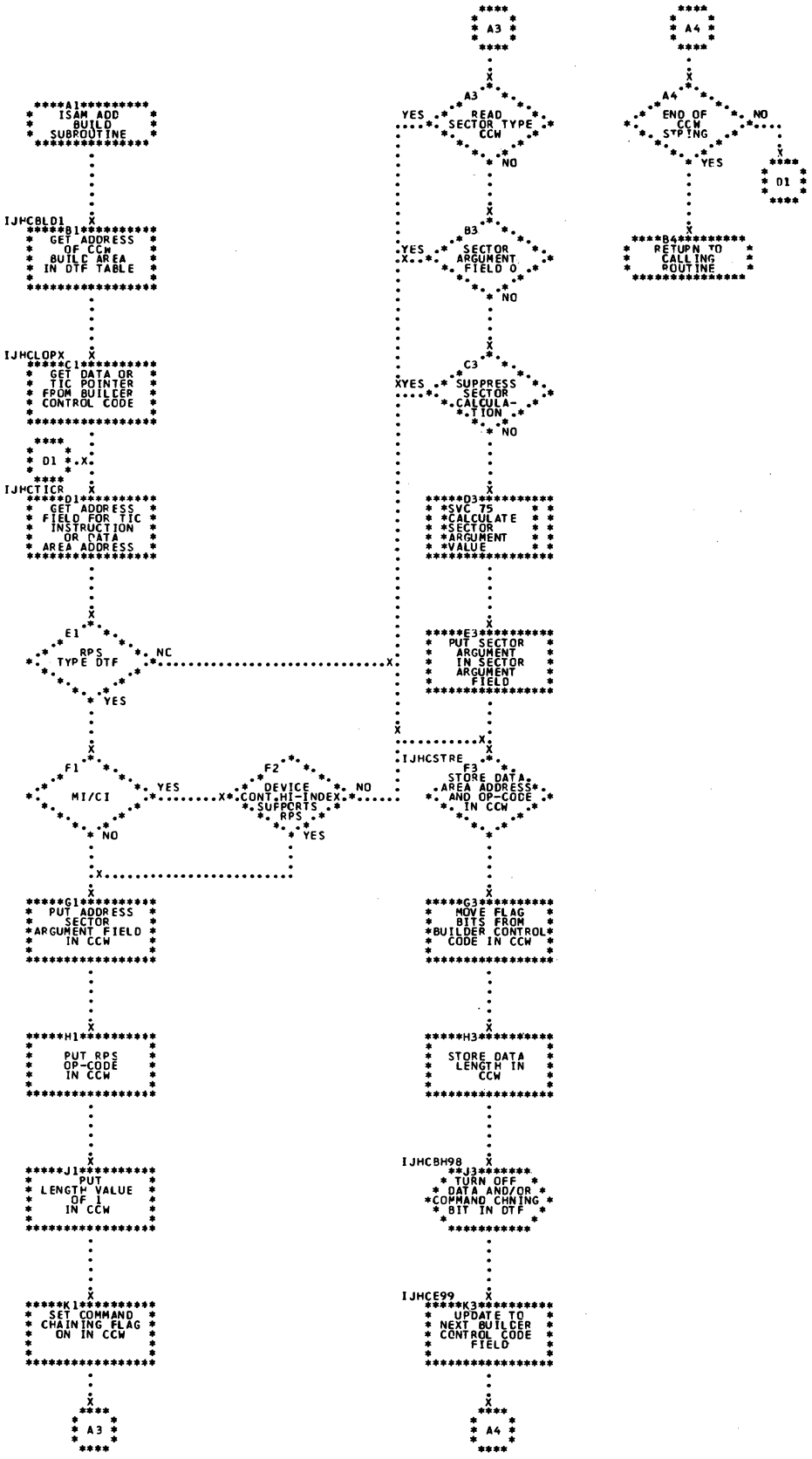


Chart EL. ISAM ADD, RETRVE, and ADDRTR: Subroutines (Part 5 of 8)

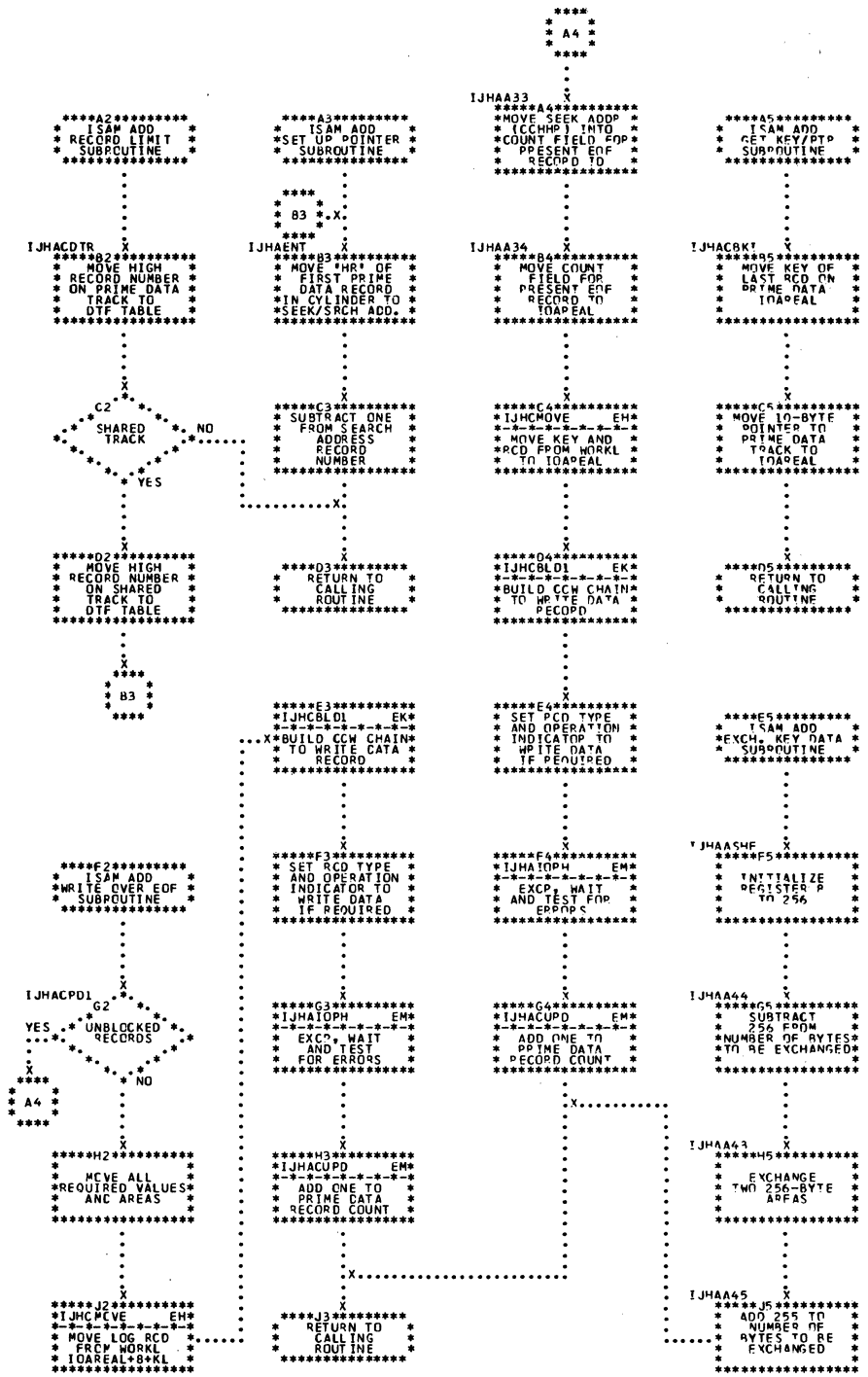


Chart EM. ISAM ADD, RETRVE, and ADDRTR: Subroutines (Part 6 of 8)

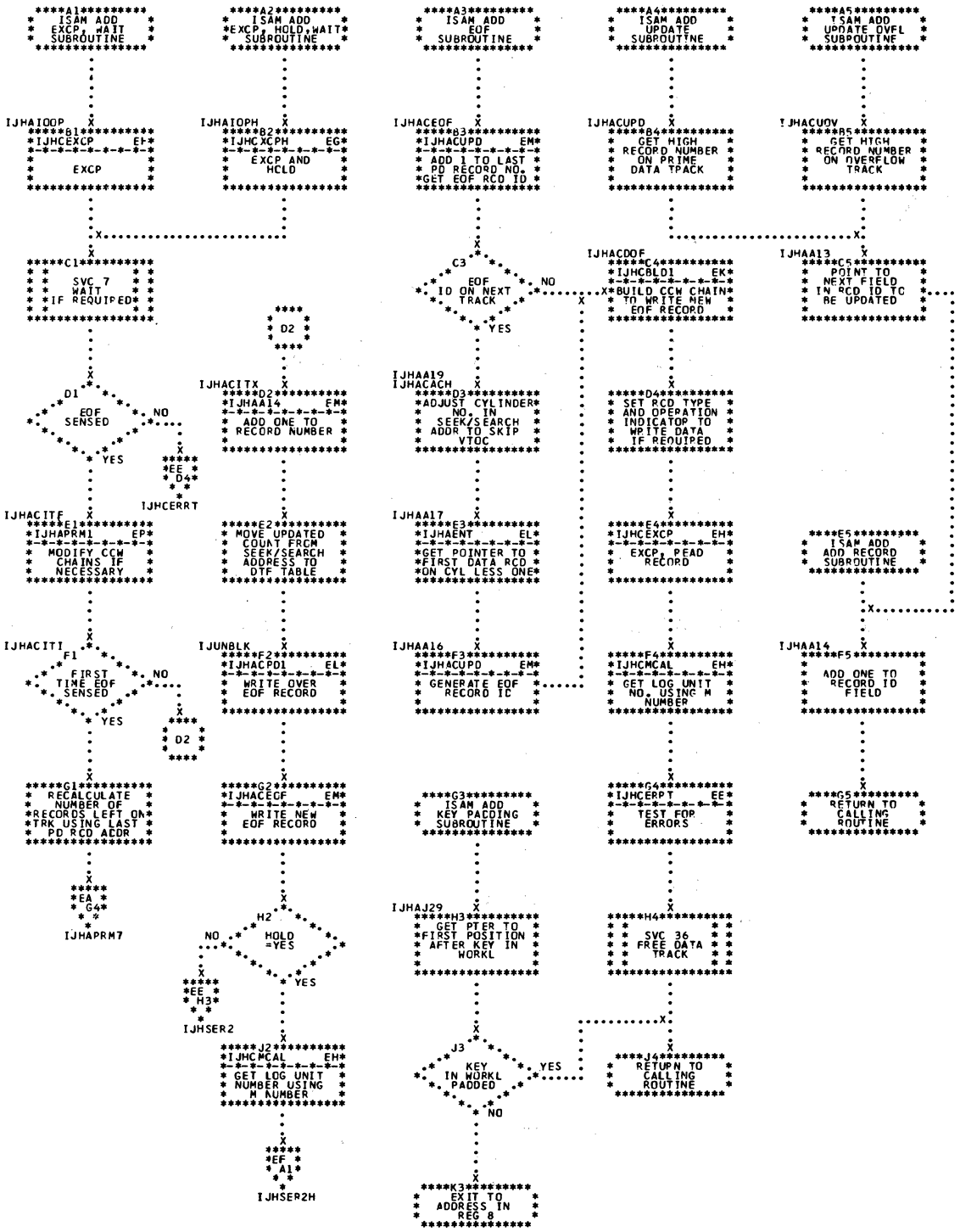


Chart EN. ISAM ADD, RETRVE, and ADDRTR: Subroutines (Part 7 of 8)

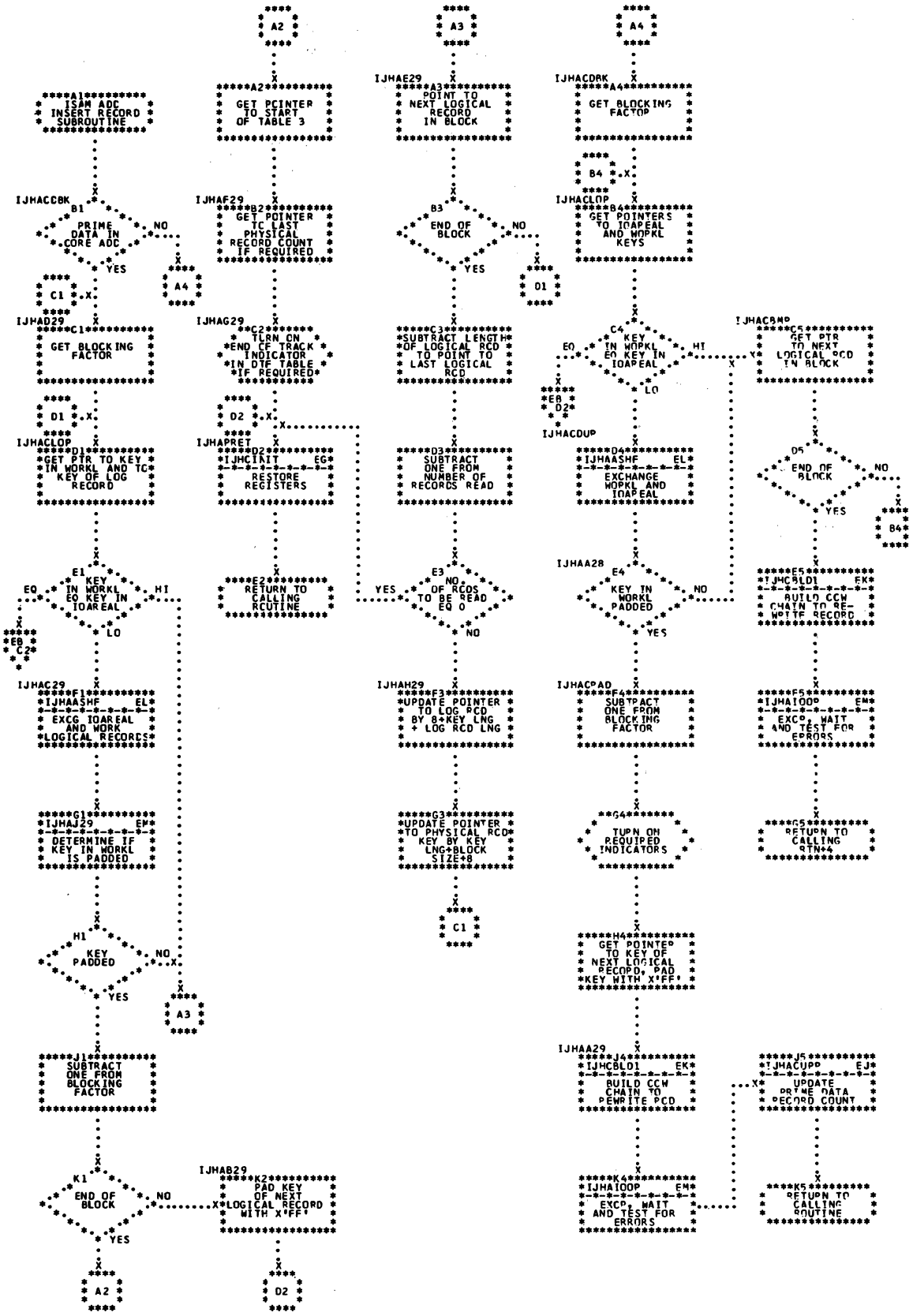


Chart EP4. ISAM ADD, RETRVE, and ADDRTR: Subroutines (Part 8 of 8)

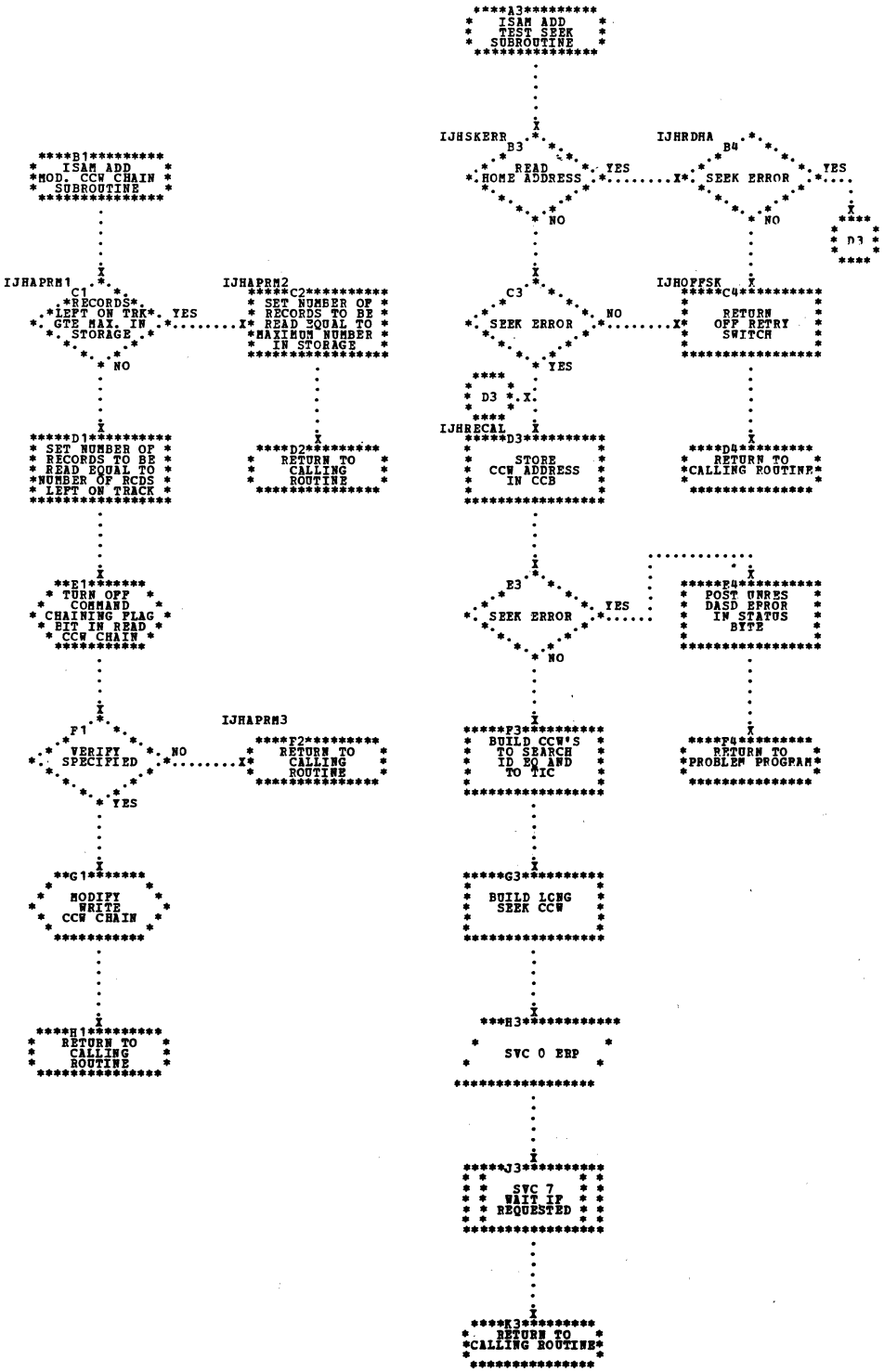


Chart FA. $$BINDEX: Read Cylinder Index into Storage (Part 1 of 2)

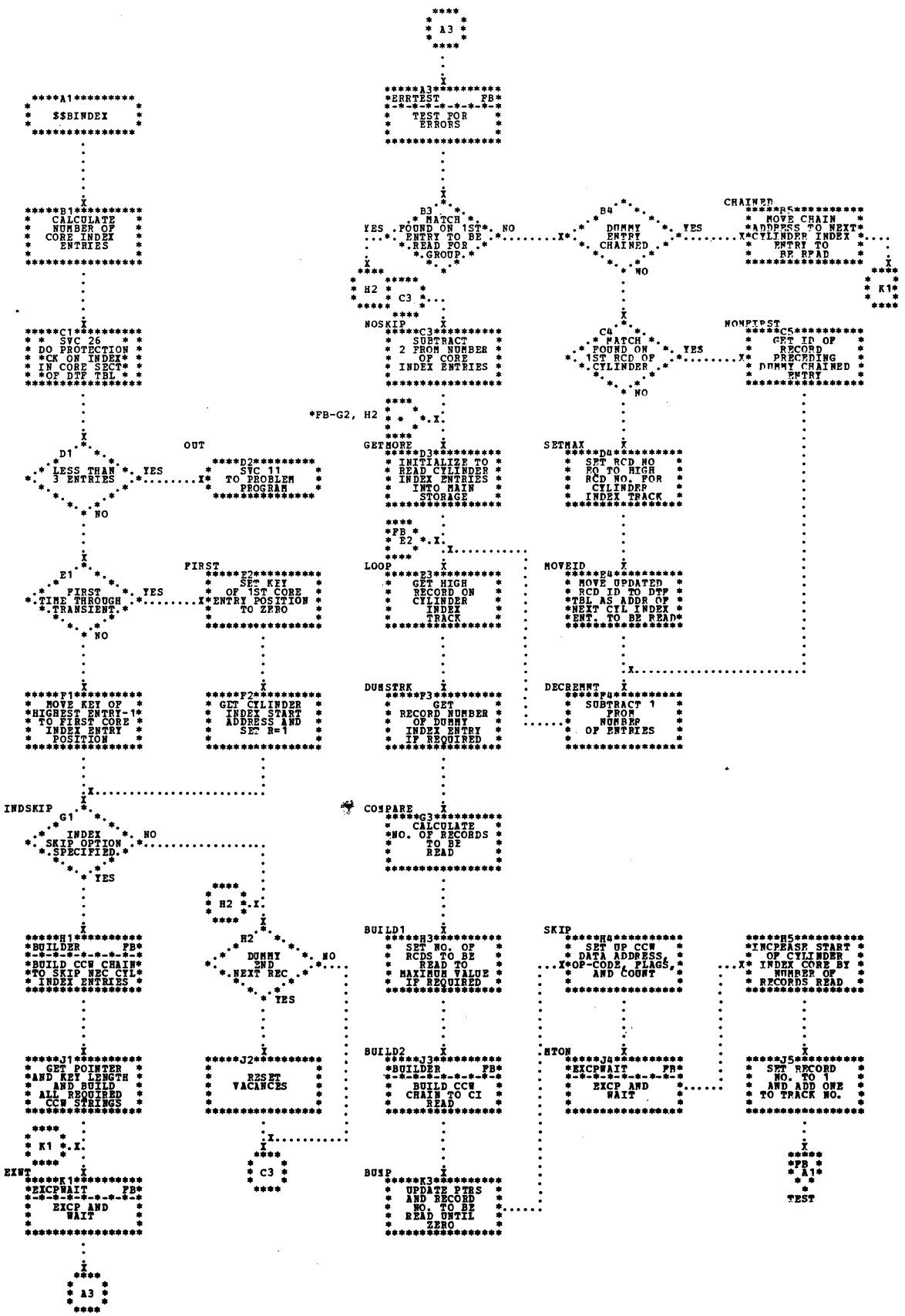


Chart FB. $$BINDEX: Read Cylinder Index into Storage (Part 2 of 2)

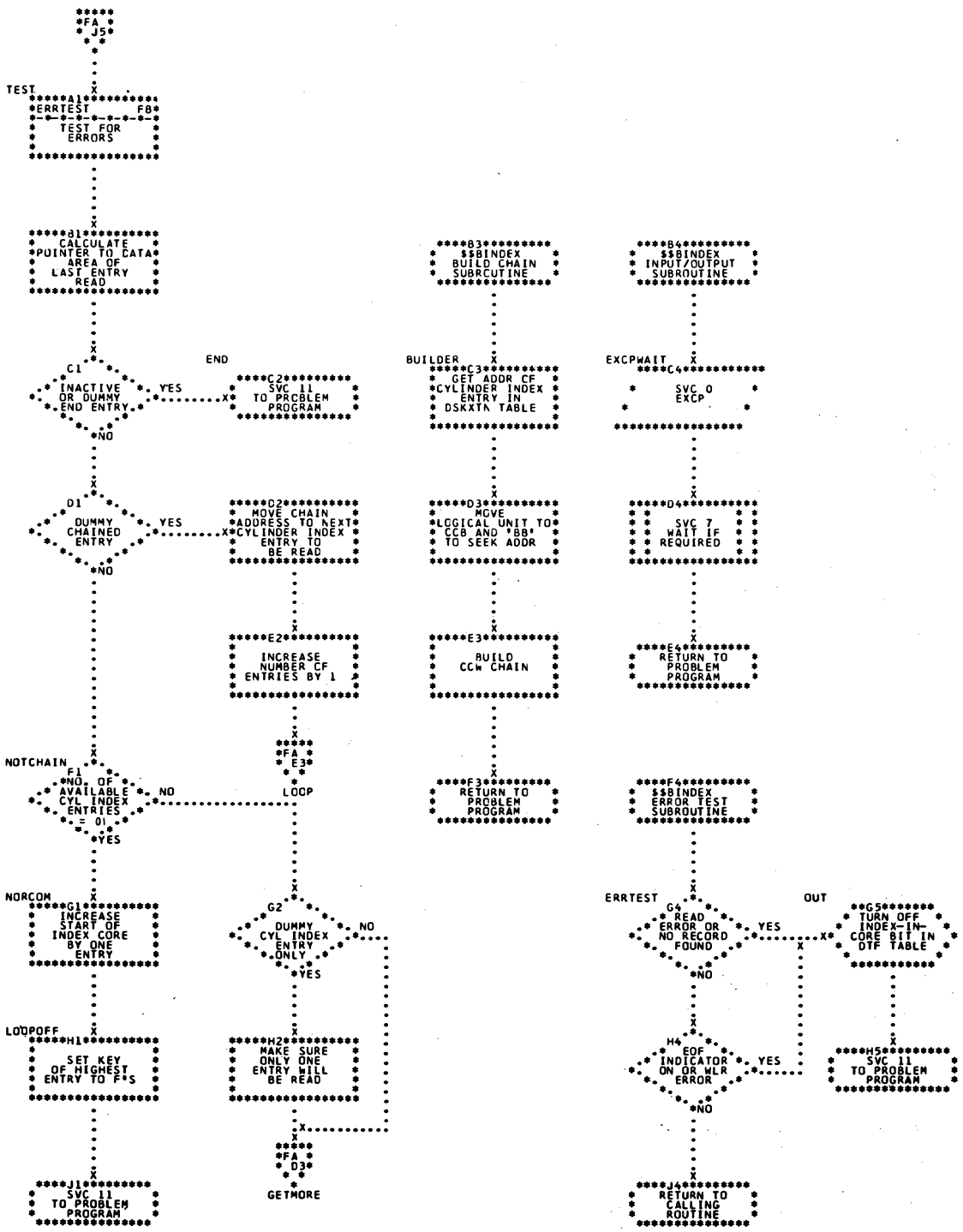


Chart FC. ISAM RETRVE, RANDOM: READ Macro, KEY

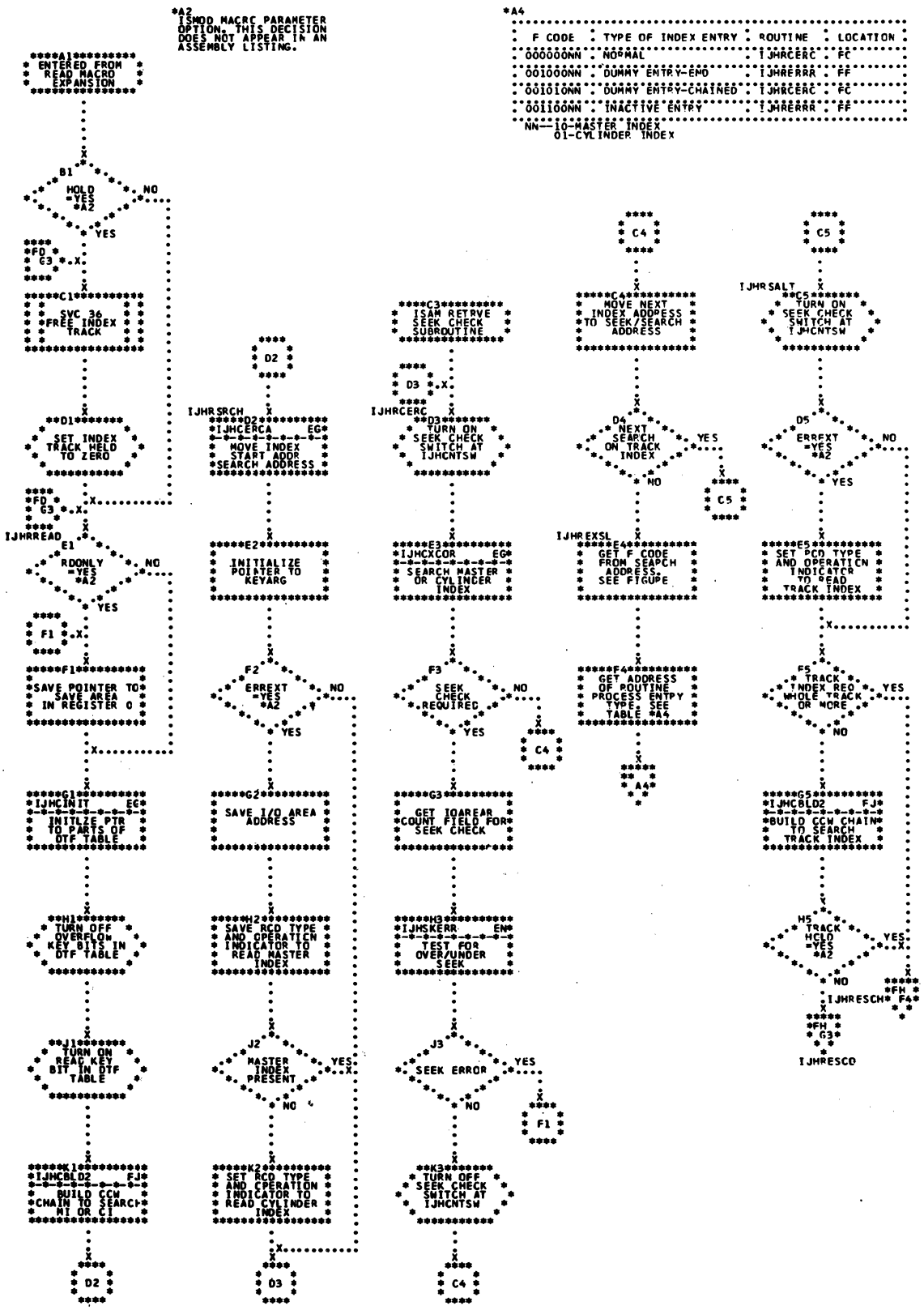


Chart FD. ISAM RETRVE, RANDOM: WAITF Macro (Part 1 of 4)

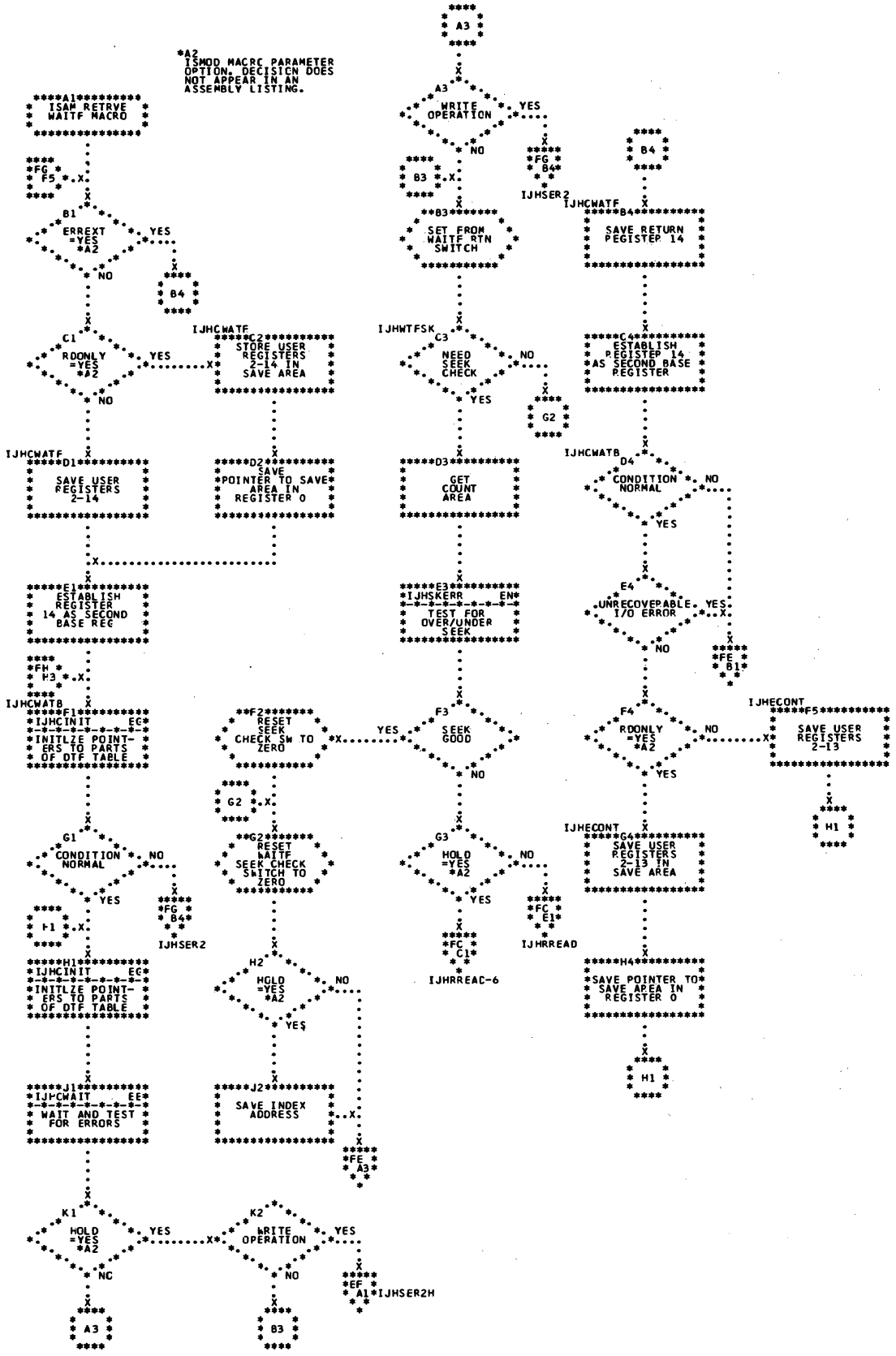


Chart FE. ISAM RETRVE, RANDOM: WAITF Macro (Part 2 of 4)

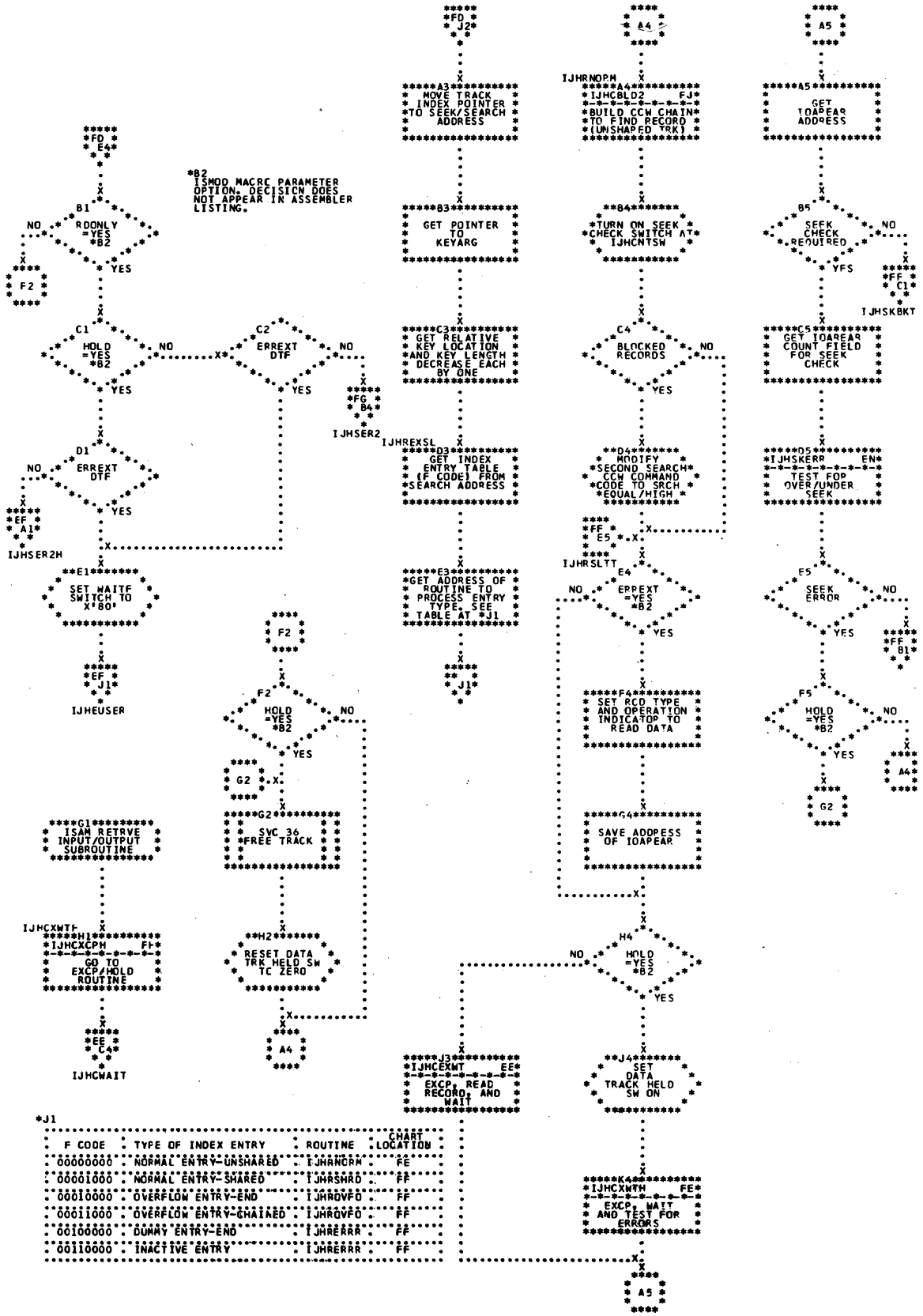


Chart FF. ISAM RETRVE, RANDOM: WAITF Macro (Part 3 of 4)

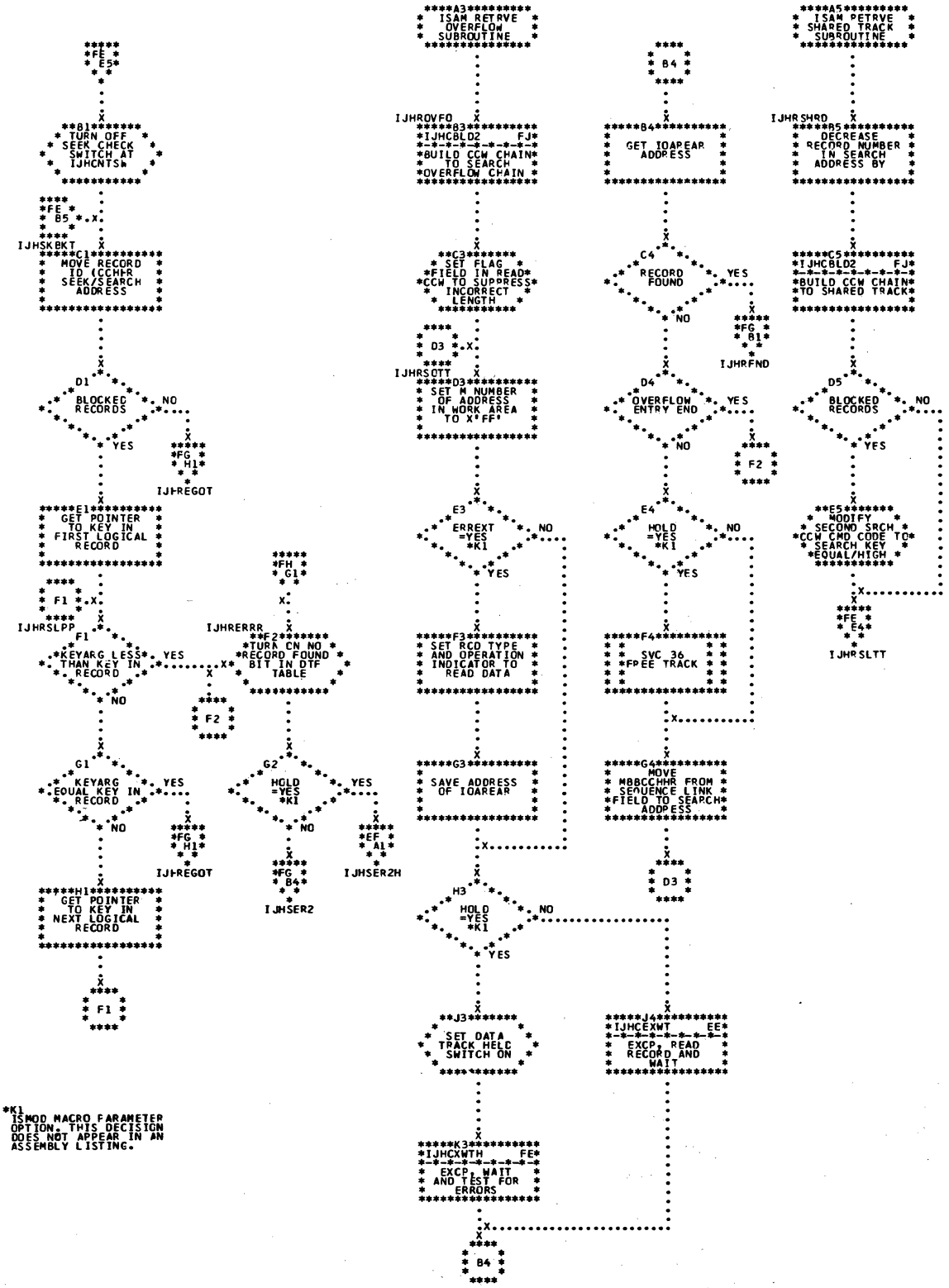


Chart FG. ISAM RETRVE, RANDOM: WAITF Macro (Part 4 of 4)

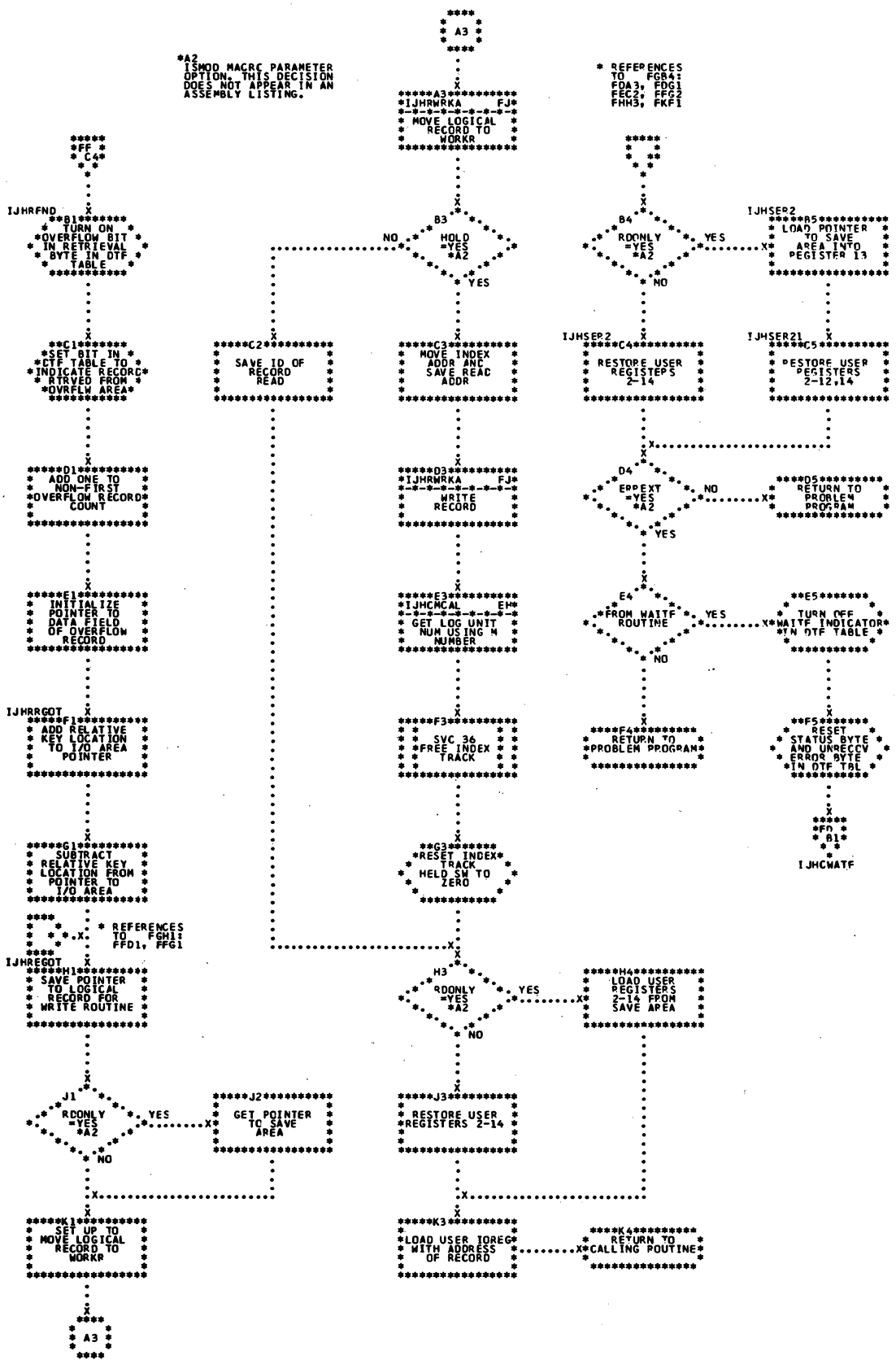


Chart FH. ISAM RETRVE, RANDOM: WRITE MACRO KEY

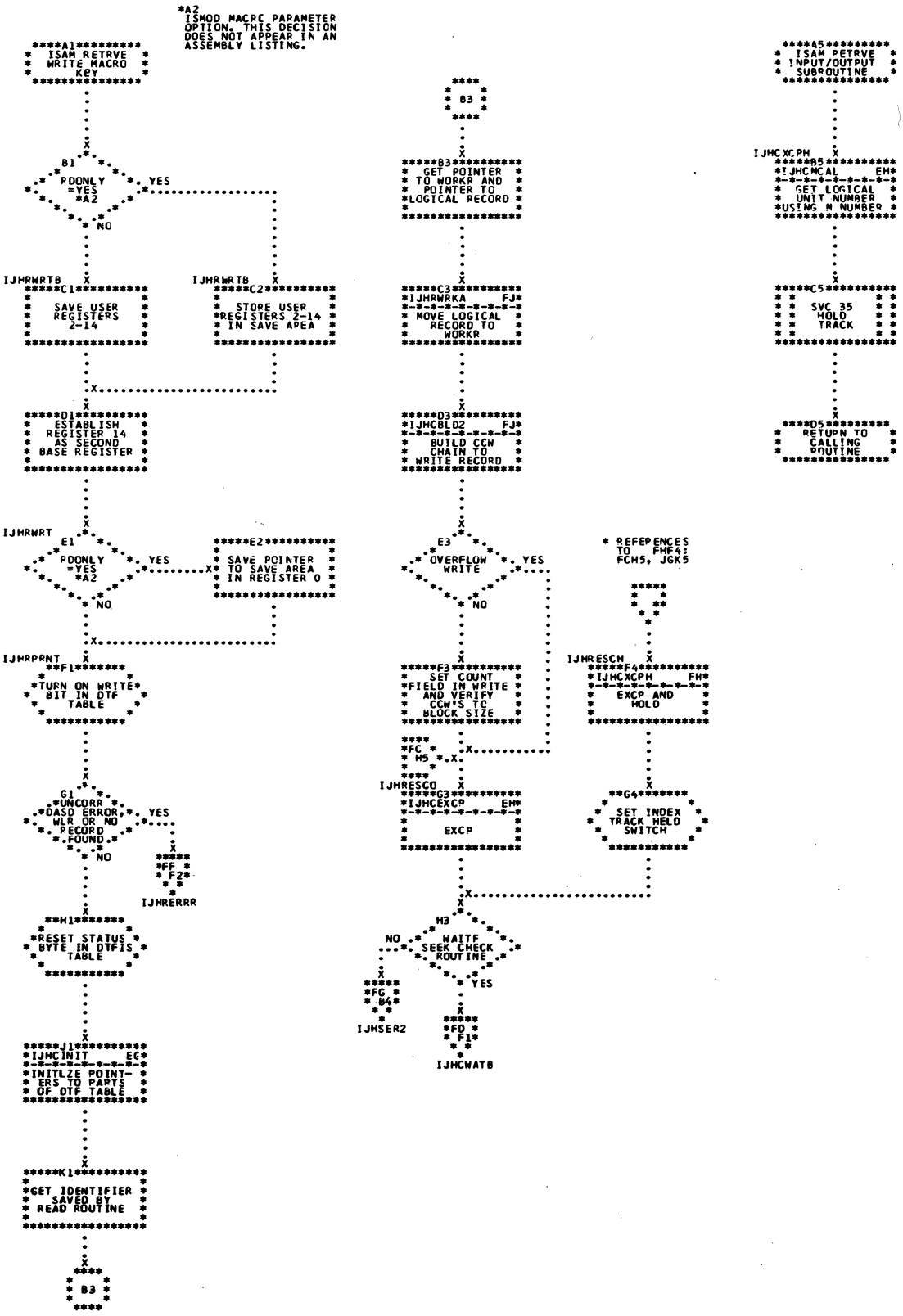


Chart FJ. ISAM RETRVE, RANDOM: Subroutines

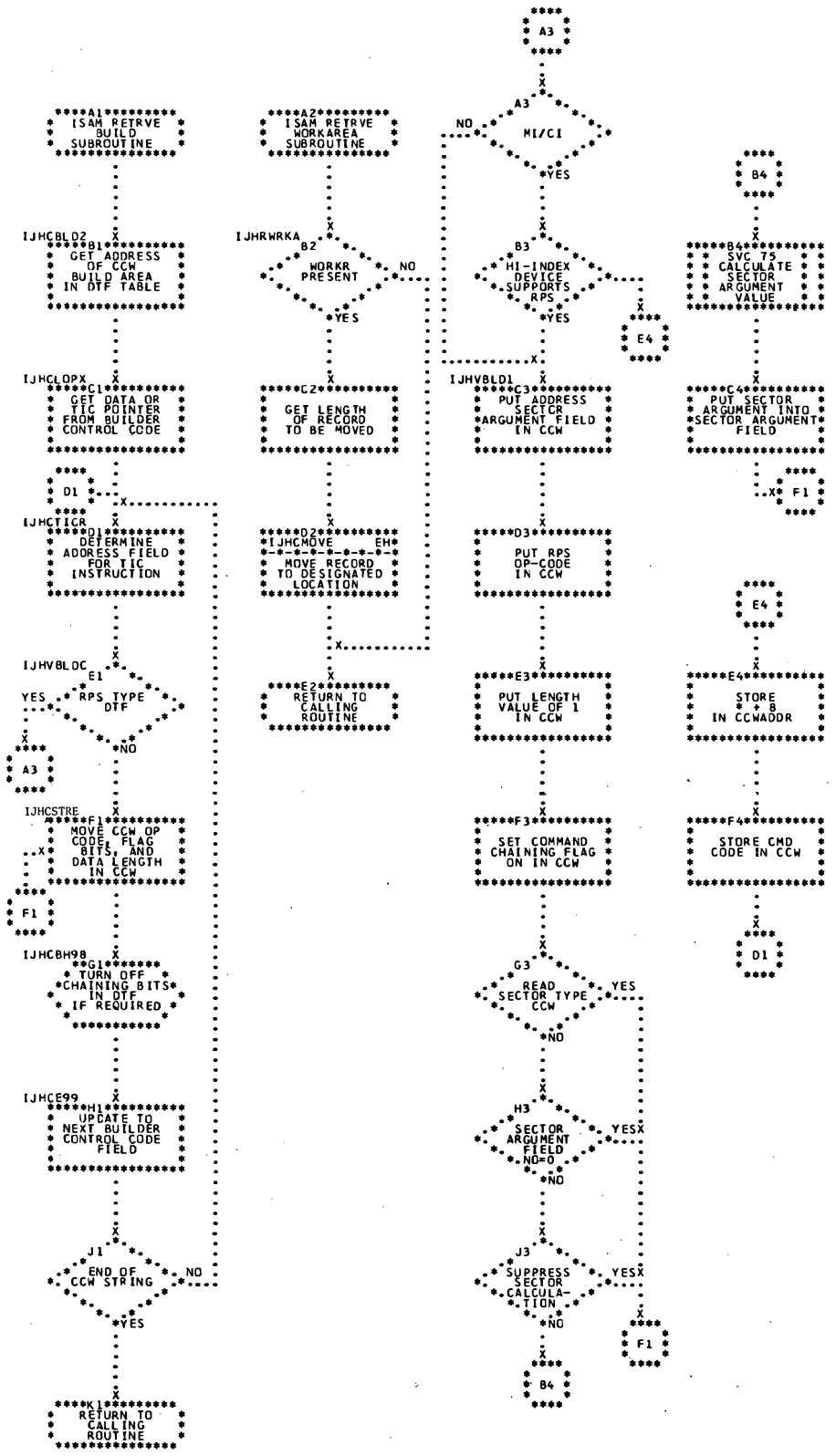


Chart FK. ISAM RETRVE, RANDOM: FREE Macro

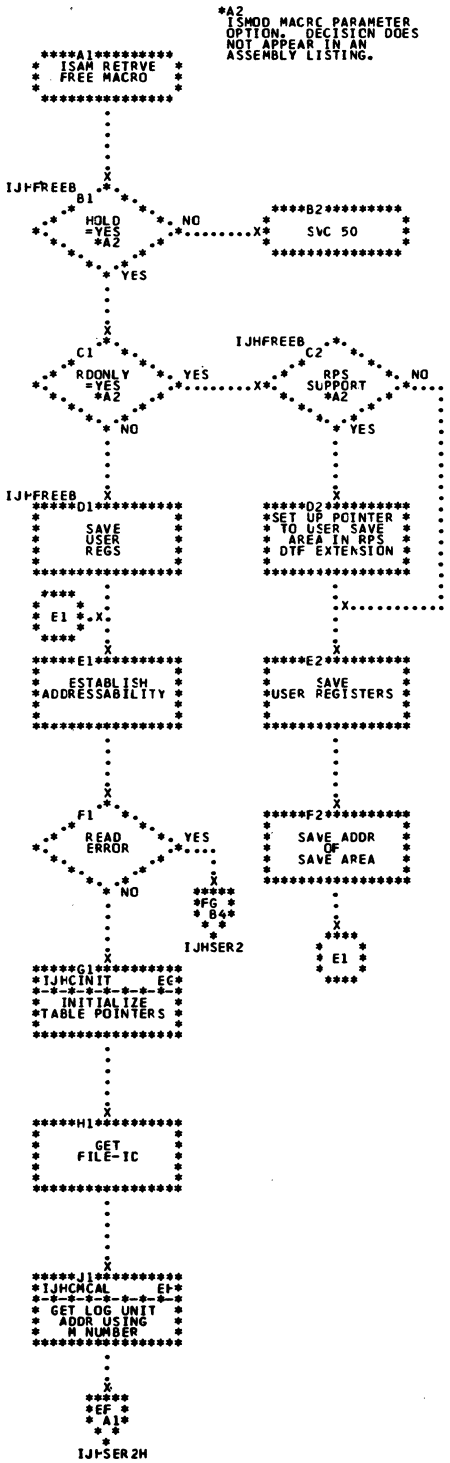


Chart GA. ISAM RETRVE, SEQNTL: ESETL Macro

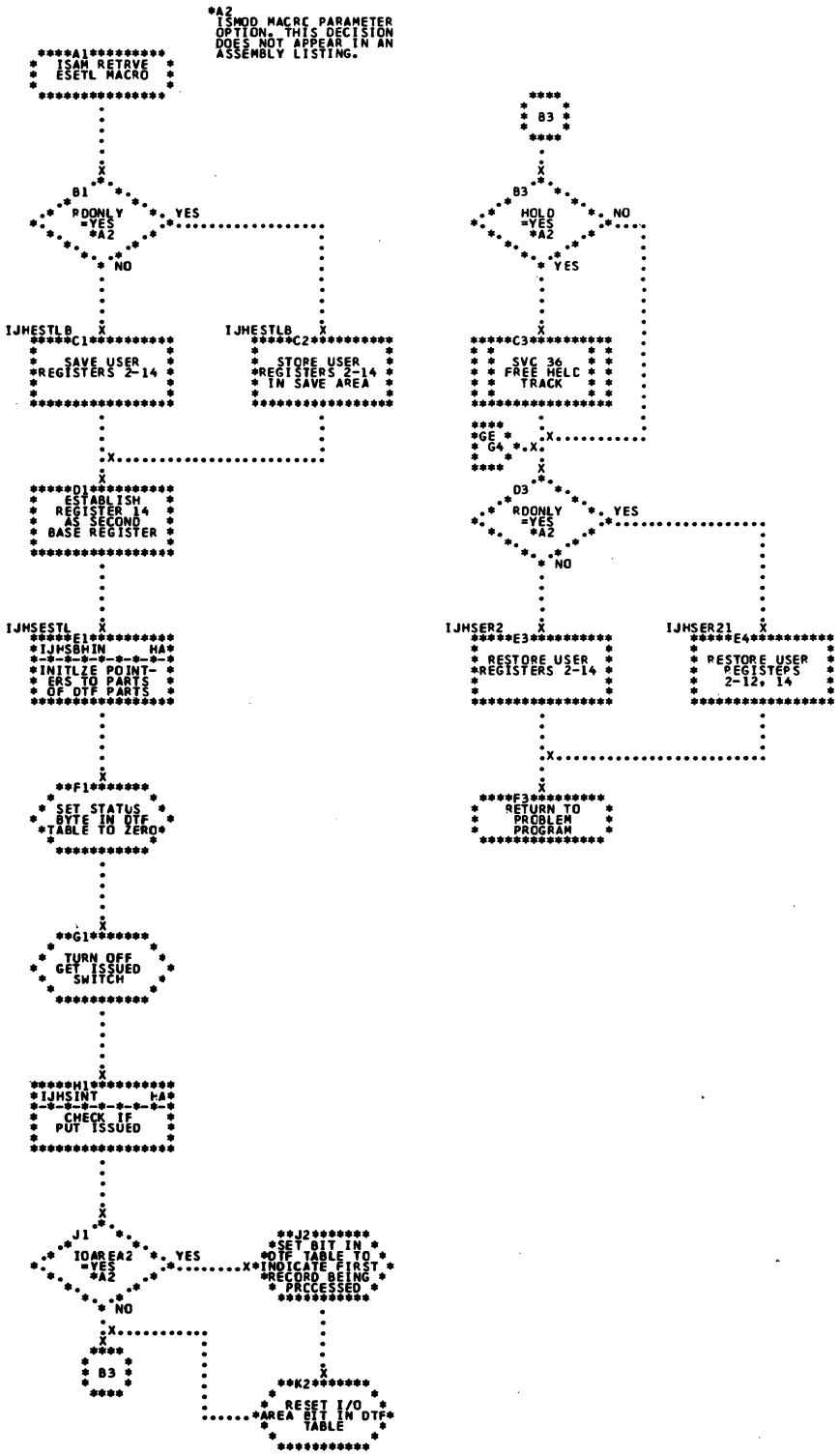


Chart GB. ISAM RETRVE, SEQNTL: GET Macro (Part 1 of 4)

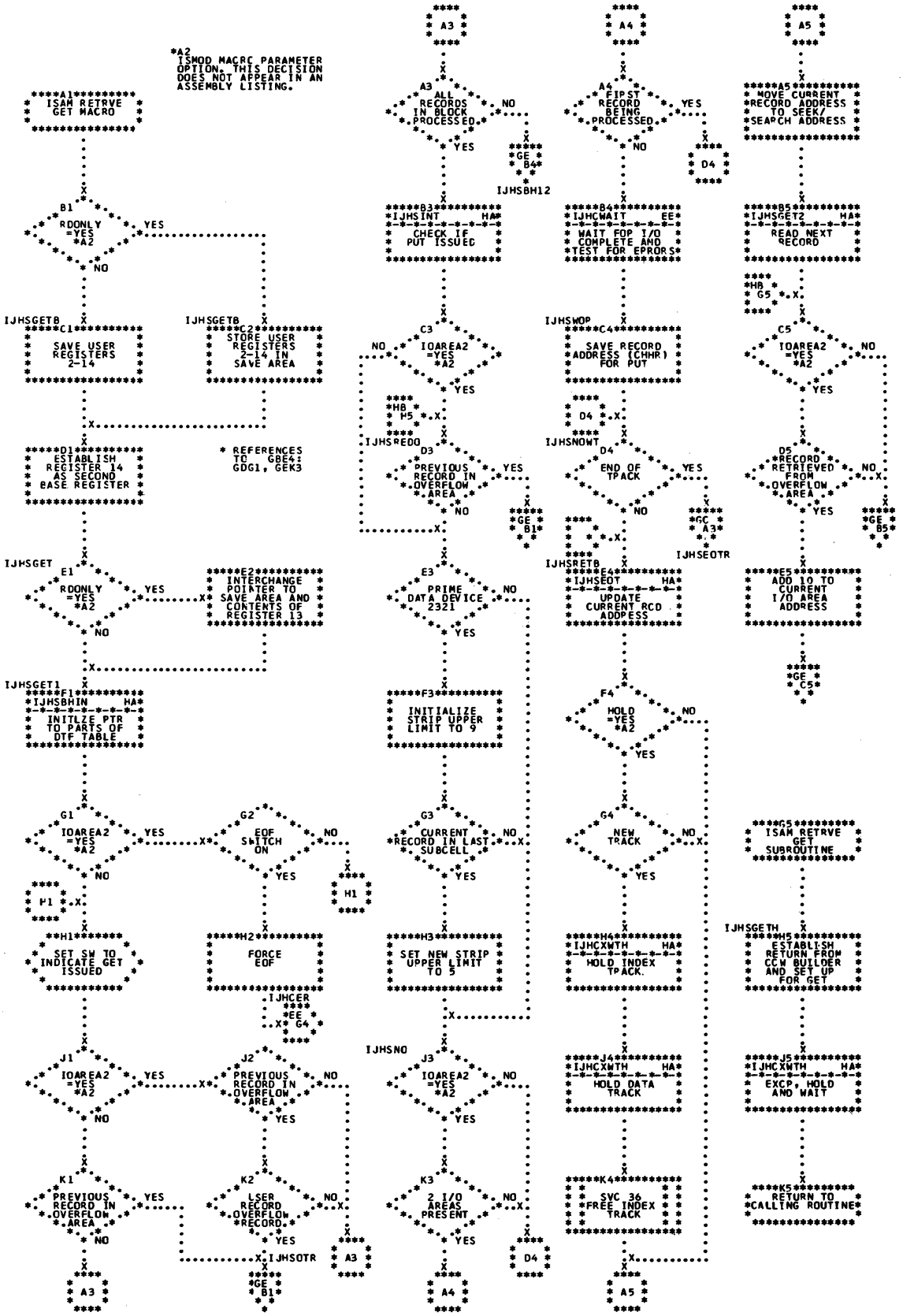


Chart GC: ISAM RETRVE, SEQNTL: GET Macro (Part 2 of 4)

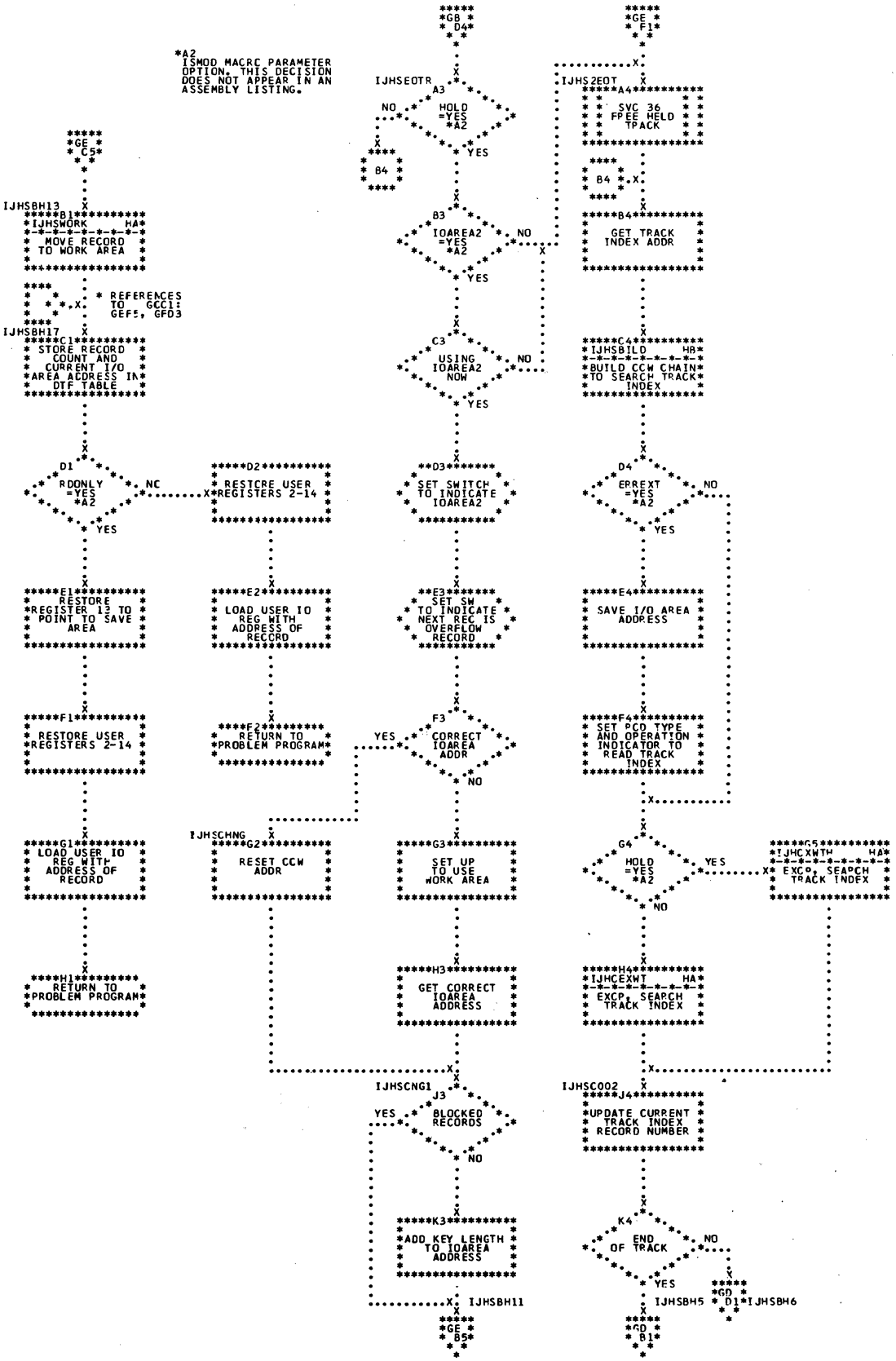


Chart GD. ISAM RETRVE, SEQNTL: GET Macro (Part 3 of 4)

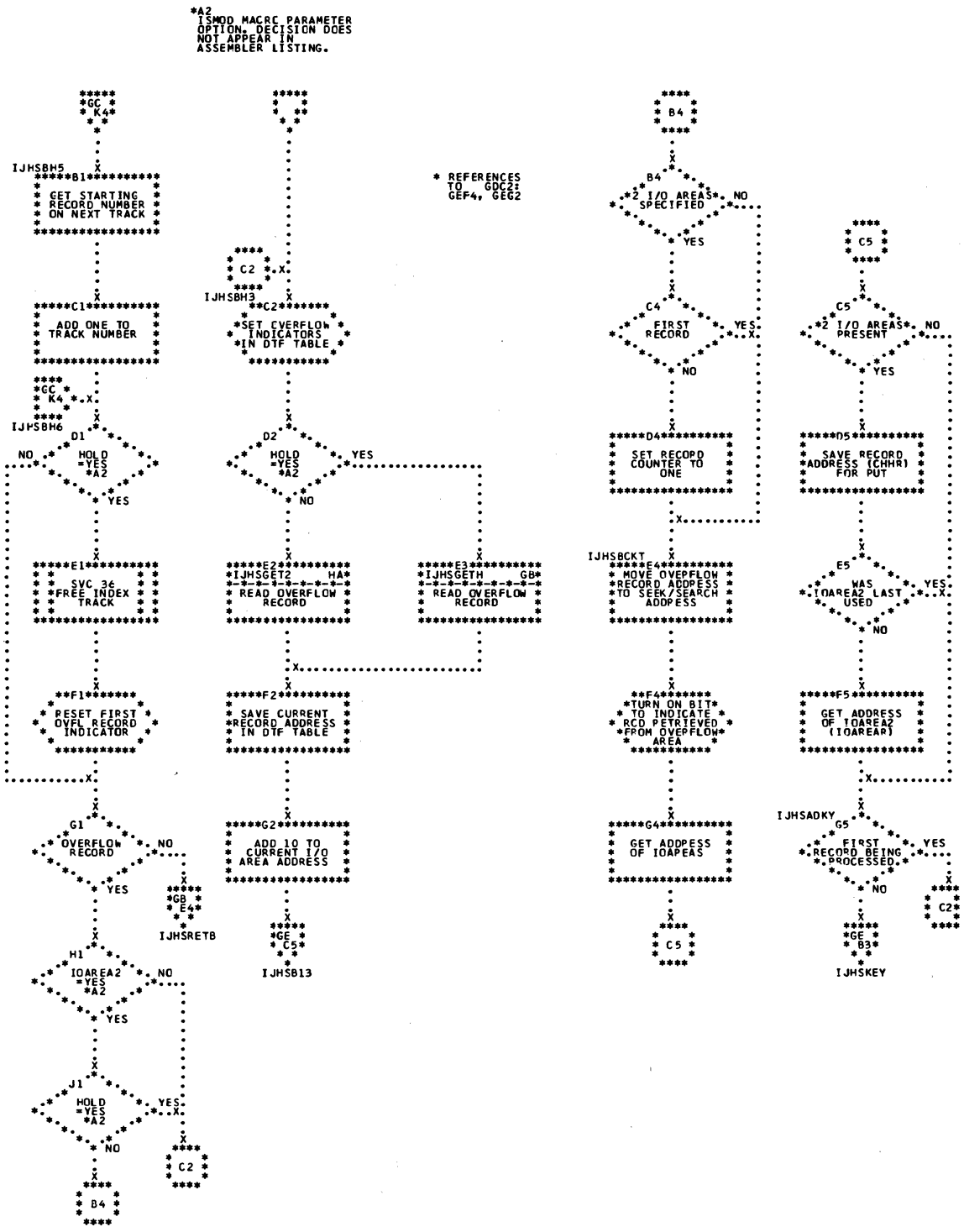


Chart GE. ISAM RETRVE, SEQNTL: GET Macro (Part 4 of 4)

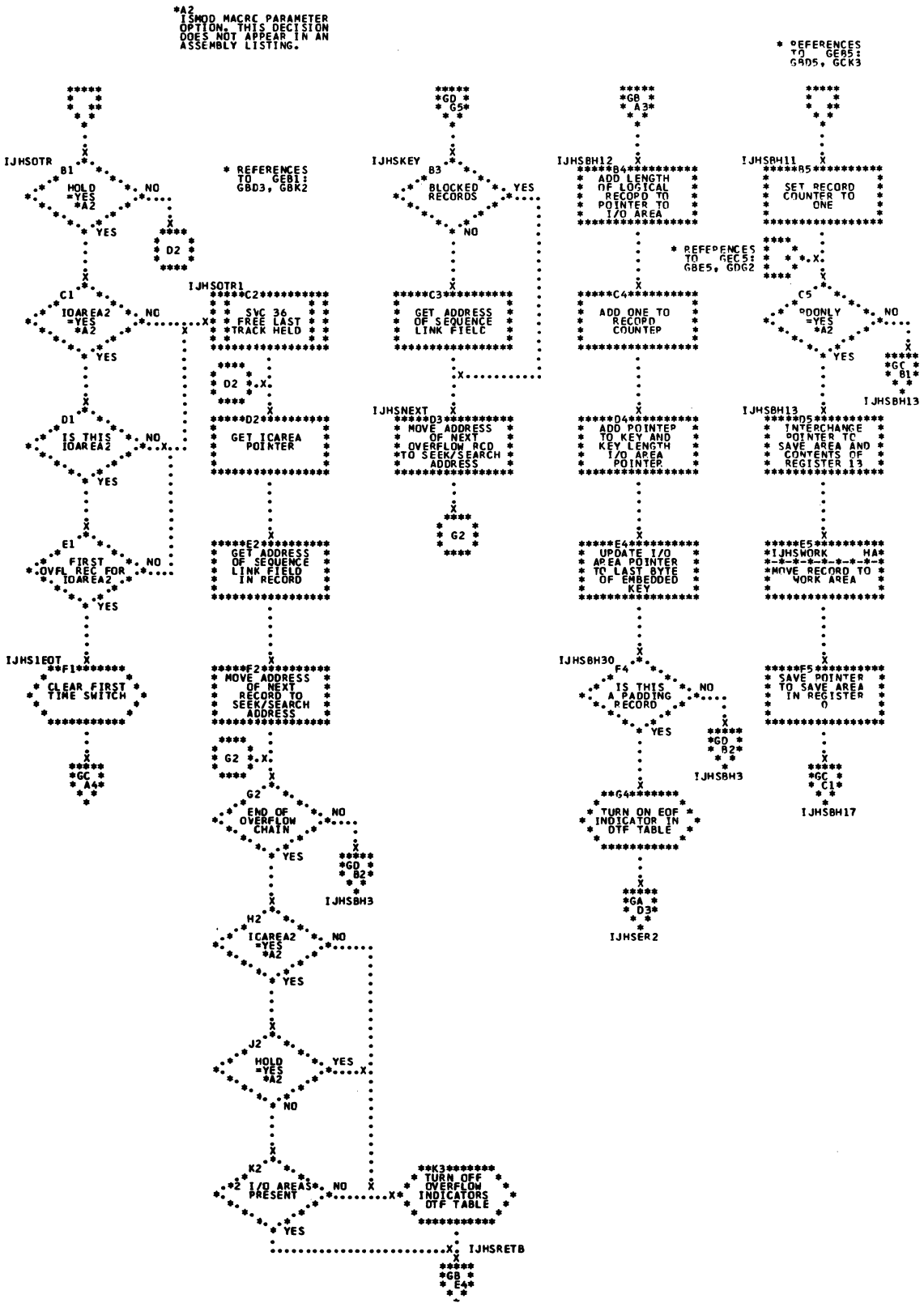


Chart GF. ISAM RETRVE, SEQNTL: PUT Macro

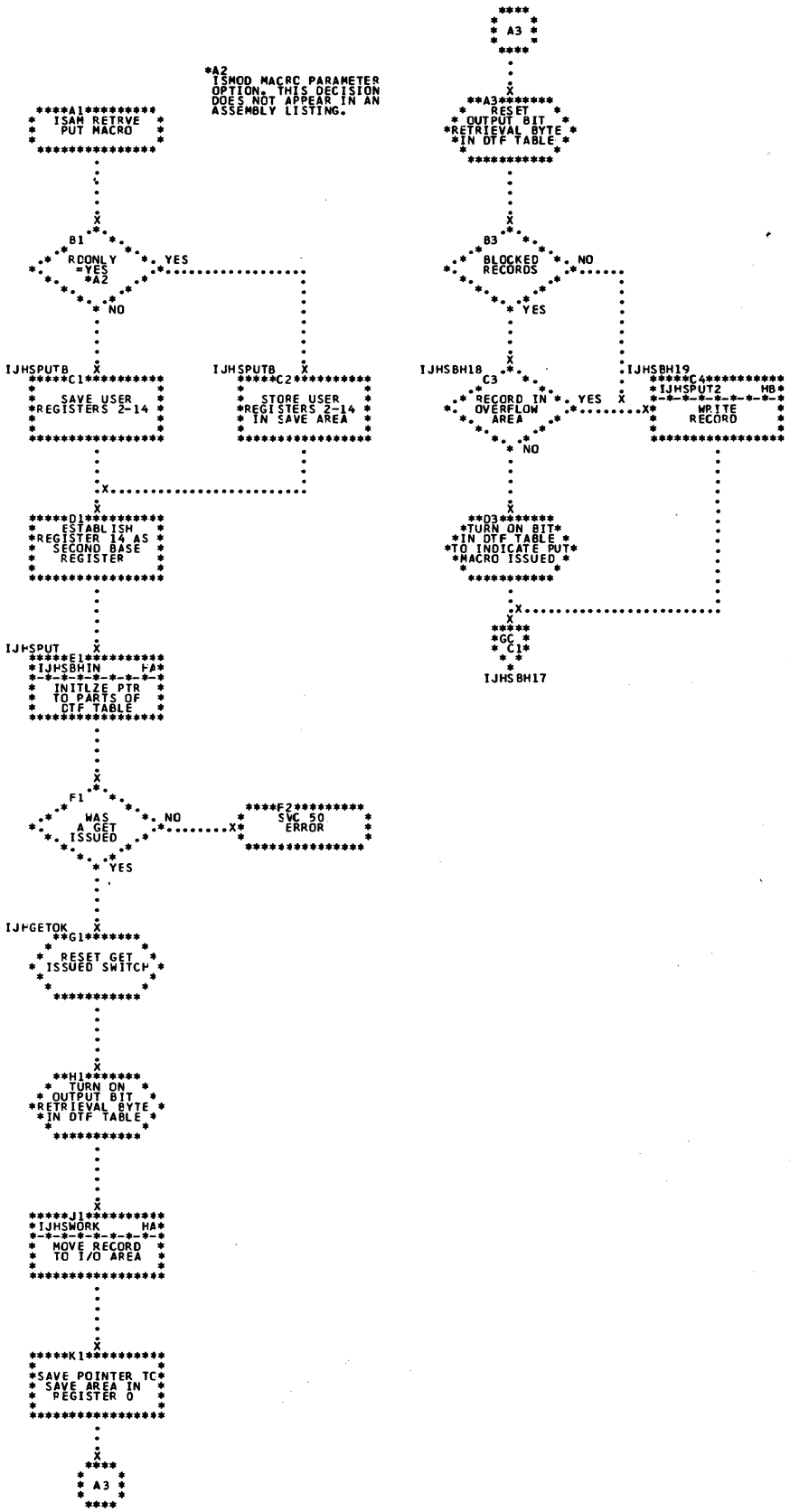


Chart GG. ISAM RETRVE, SEQNTL: SETL Macro, $$BSETL (Part 1 of 5)

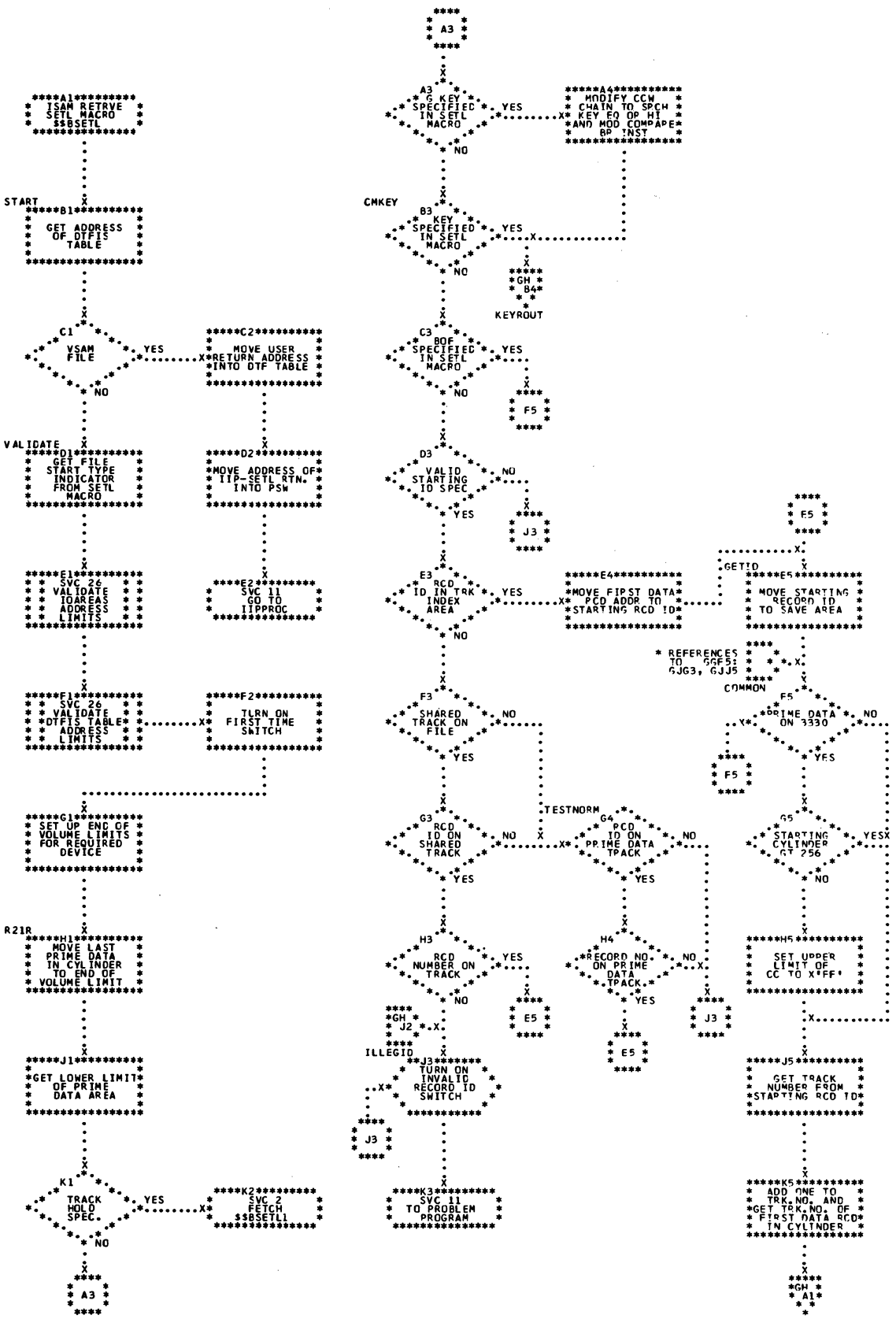


Chart GH. ISAM RETRVE, SEQNTL: SETL Macro, $$BSETL (Part 2 of 5)

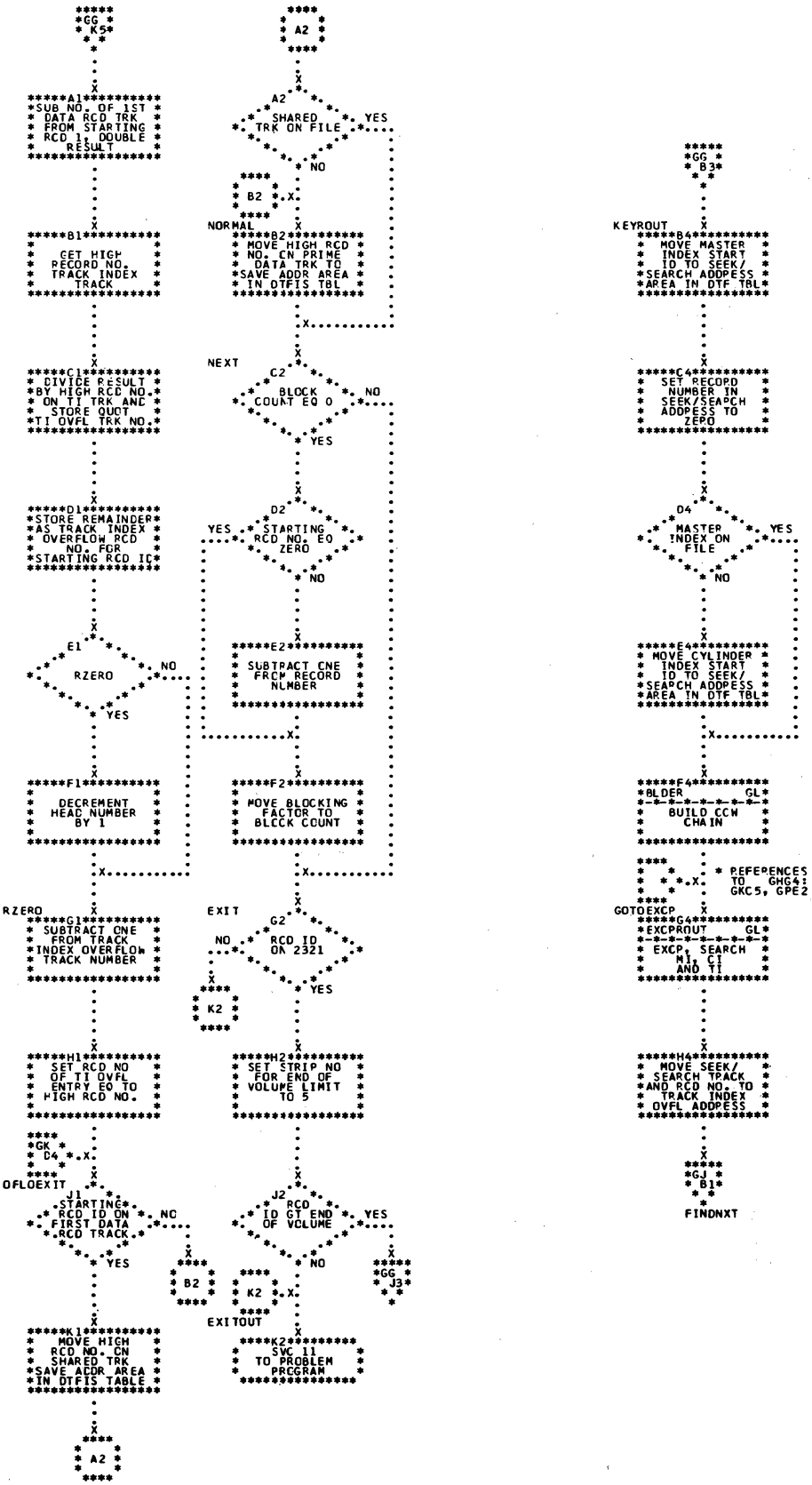


Chart GJ. ISAM RETRVE, SEQNTL: SETL Macro, $$BSETL (Part 3 of 5)

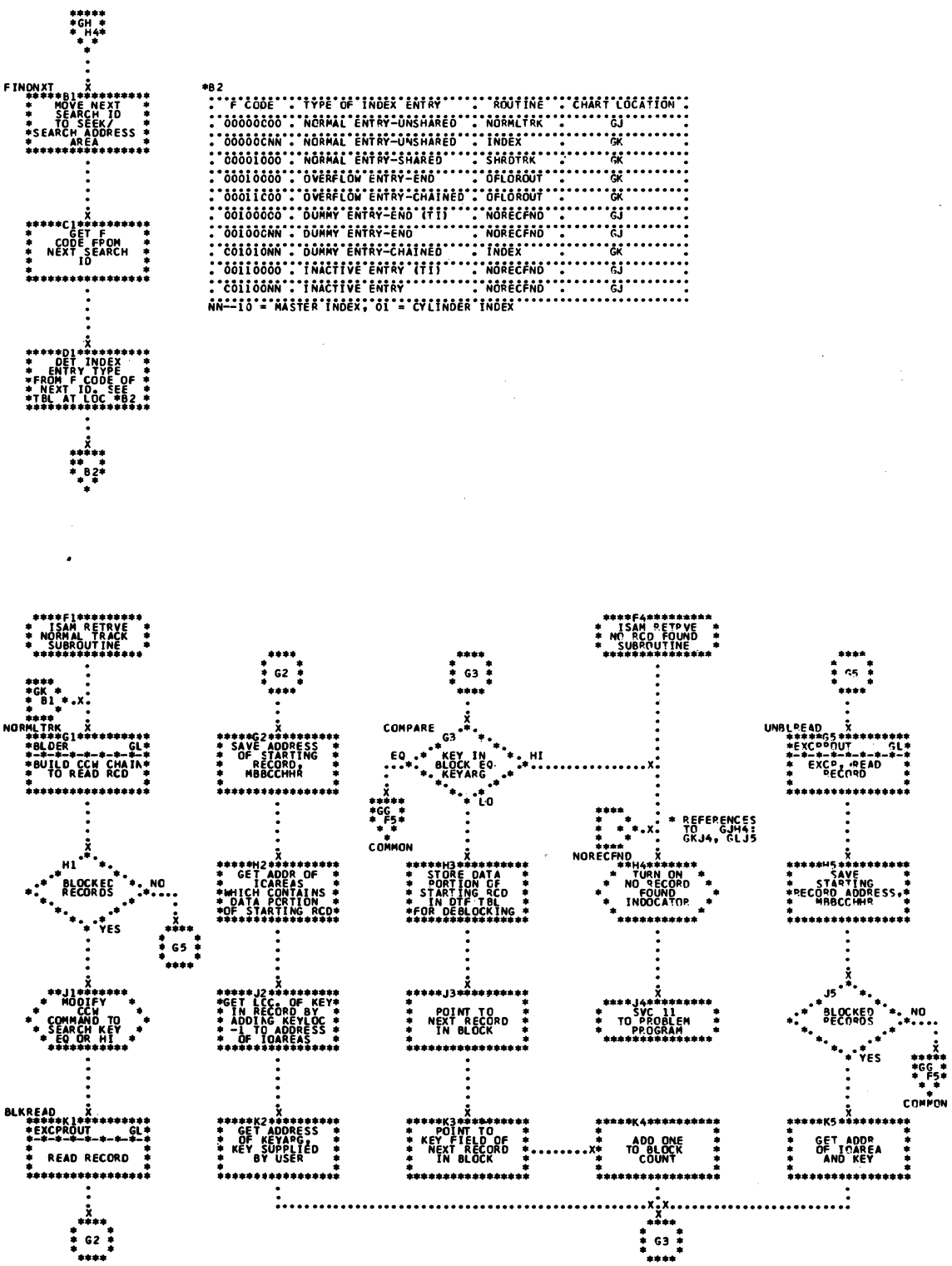


Chart GK. ISAM RETRVE, SEQNTL: SETL Macro, $$BSETL (Part 4 of 5)

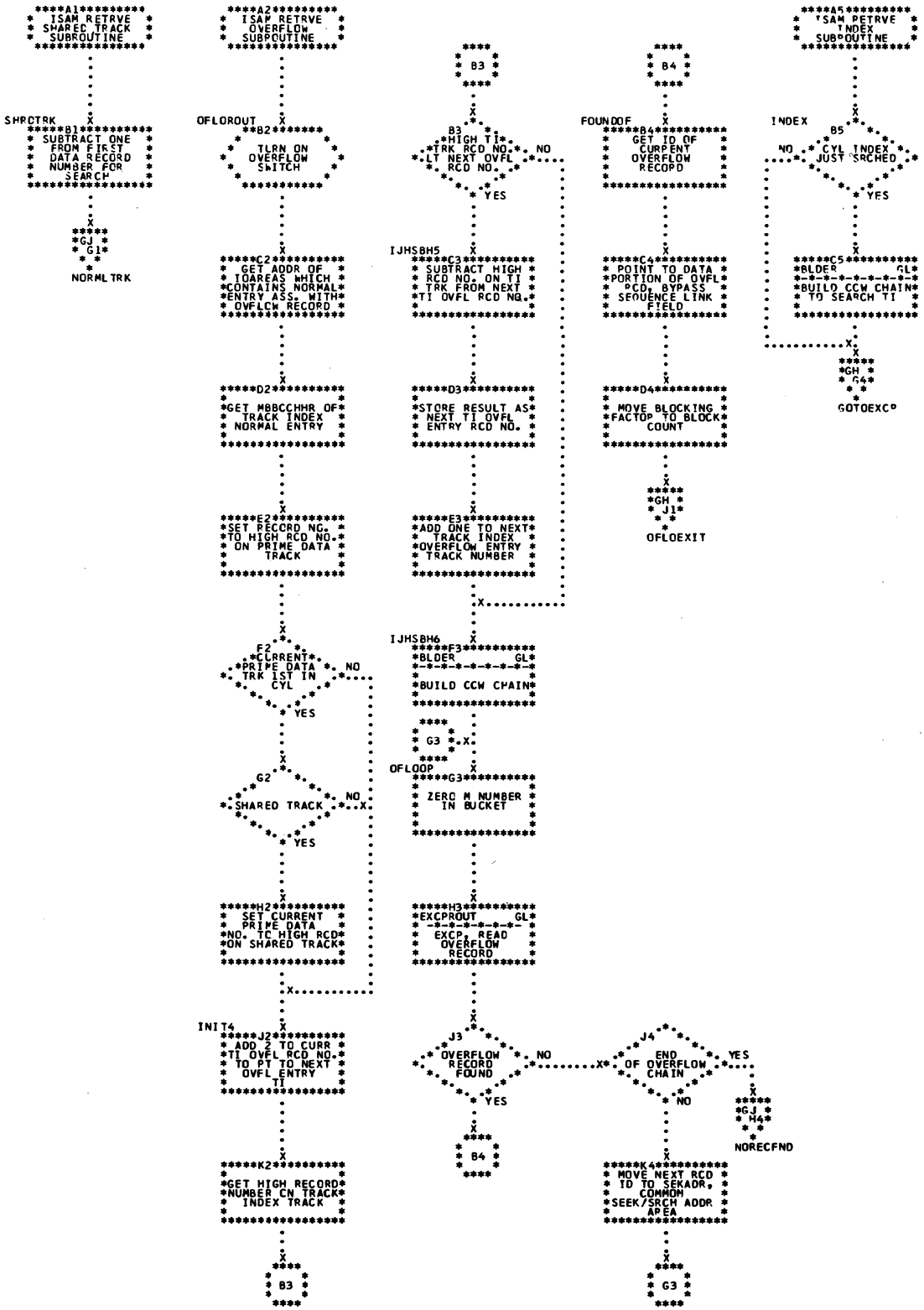


Chart GL. ISAM RETRVE, SEQNTL: SETL Macro, $$BSETL (Part 5 of 5)

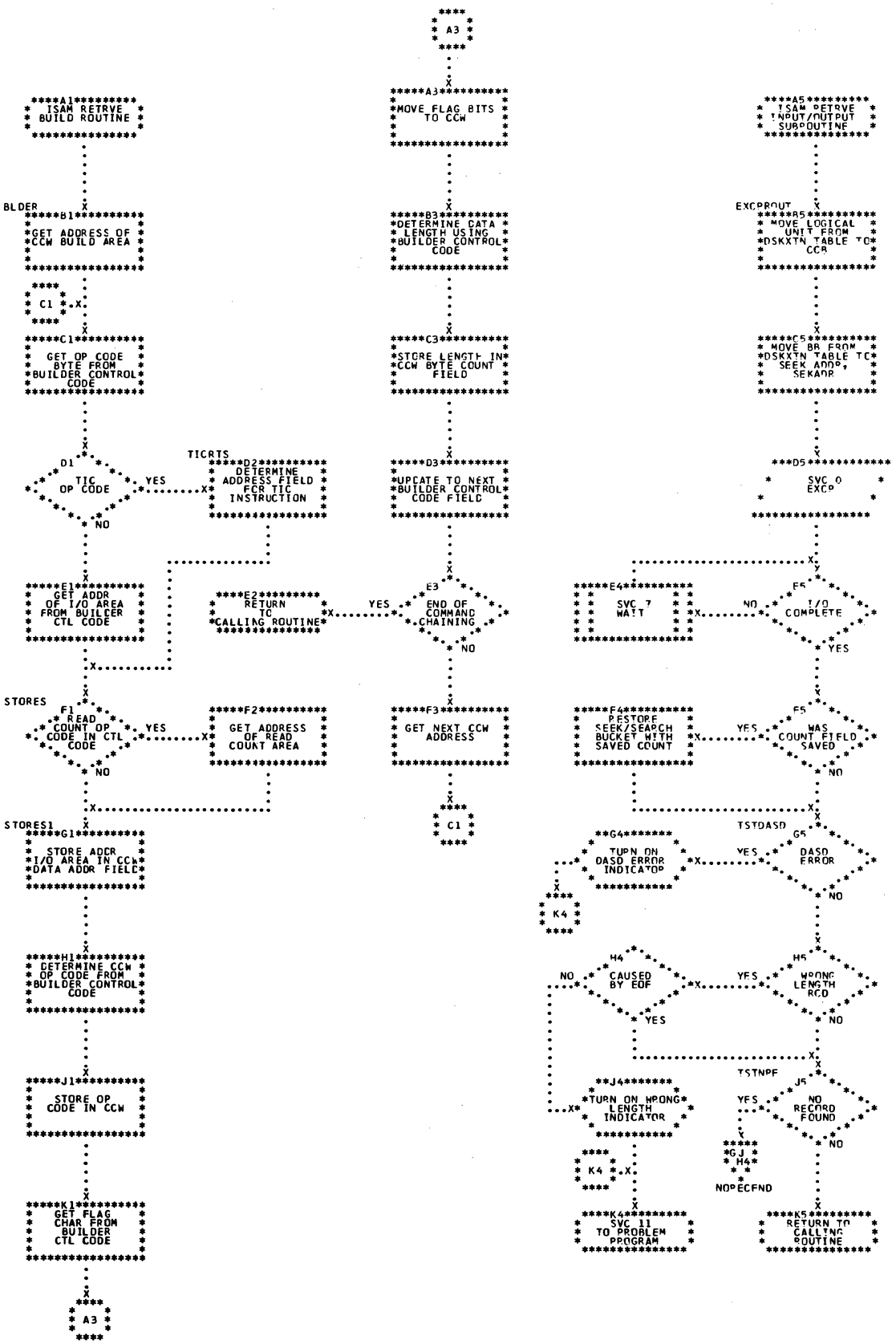


Chart GM. ISAM RETRVE, SEQNTL: SETL Macro $$BSETL1 (Part 1 of 5)

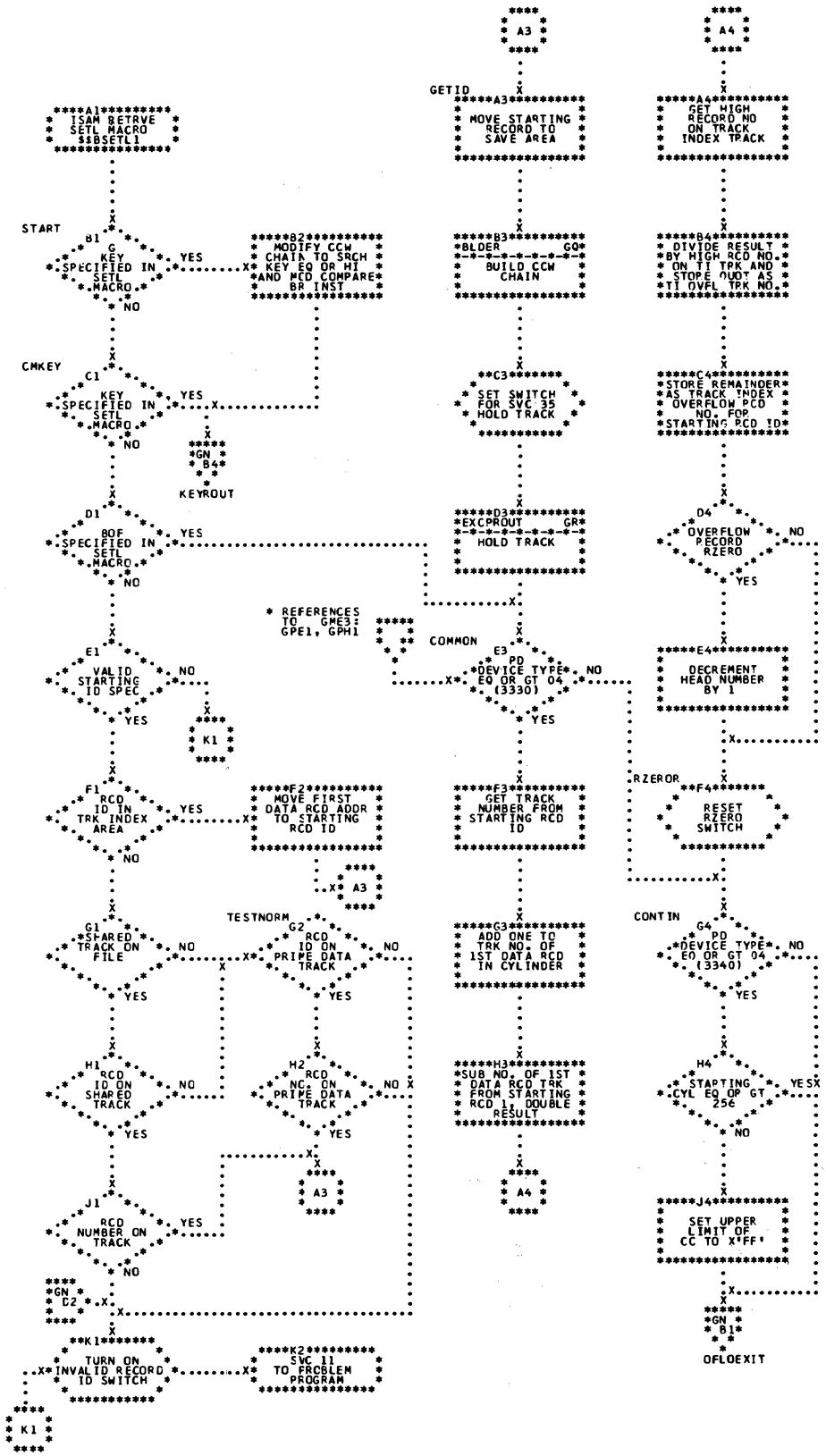


Chart GN. ISAM RETRVE, SEQNTL: SETL Macro, $$BSETL1 (Part 2 of 5)

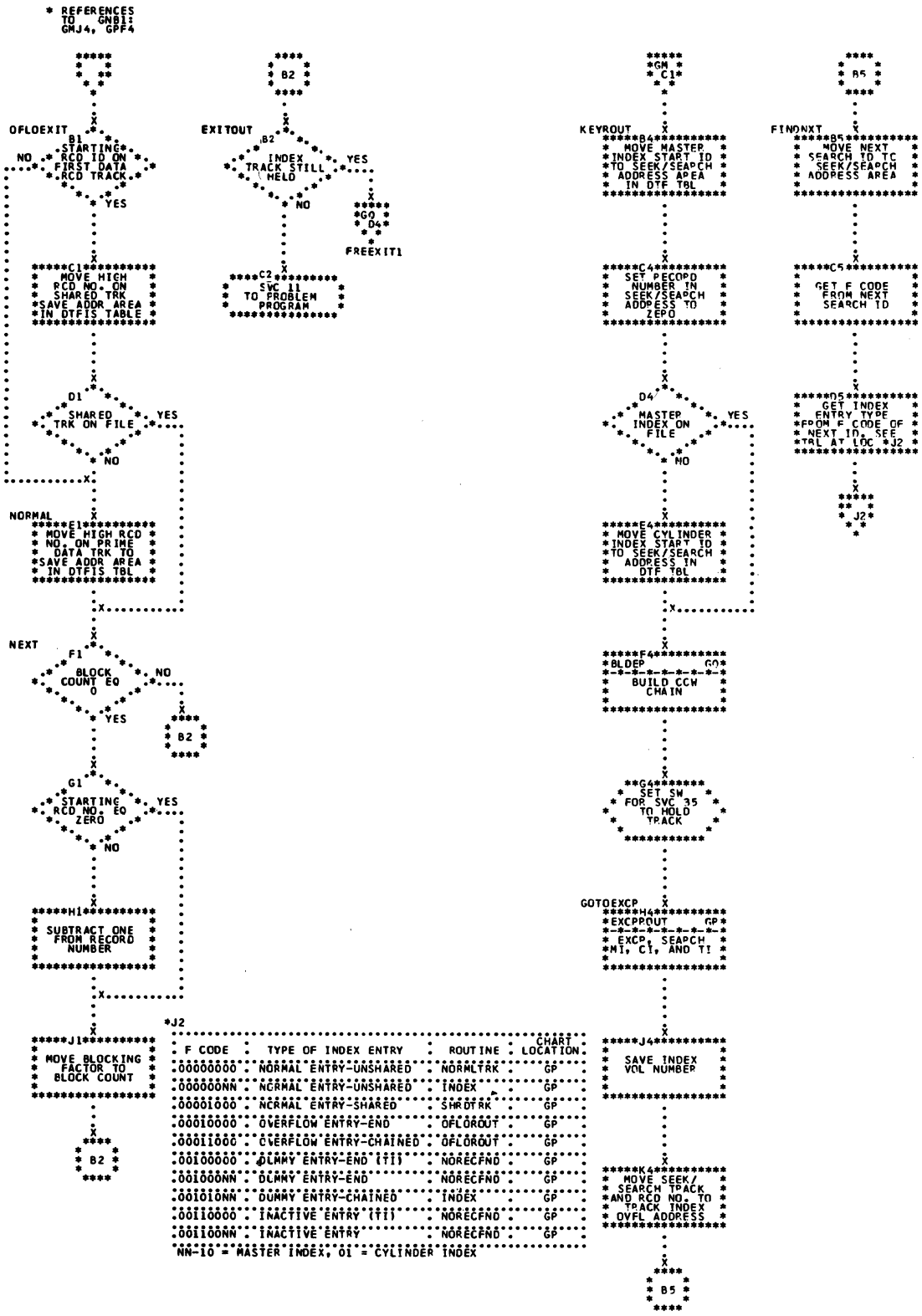


Chart GP. ISAM RETRVE, SEQNTL: SETL Macro, $$BSETL1 (Part 3 of 5)

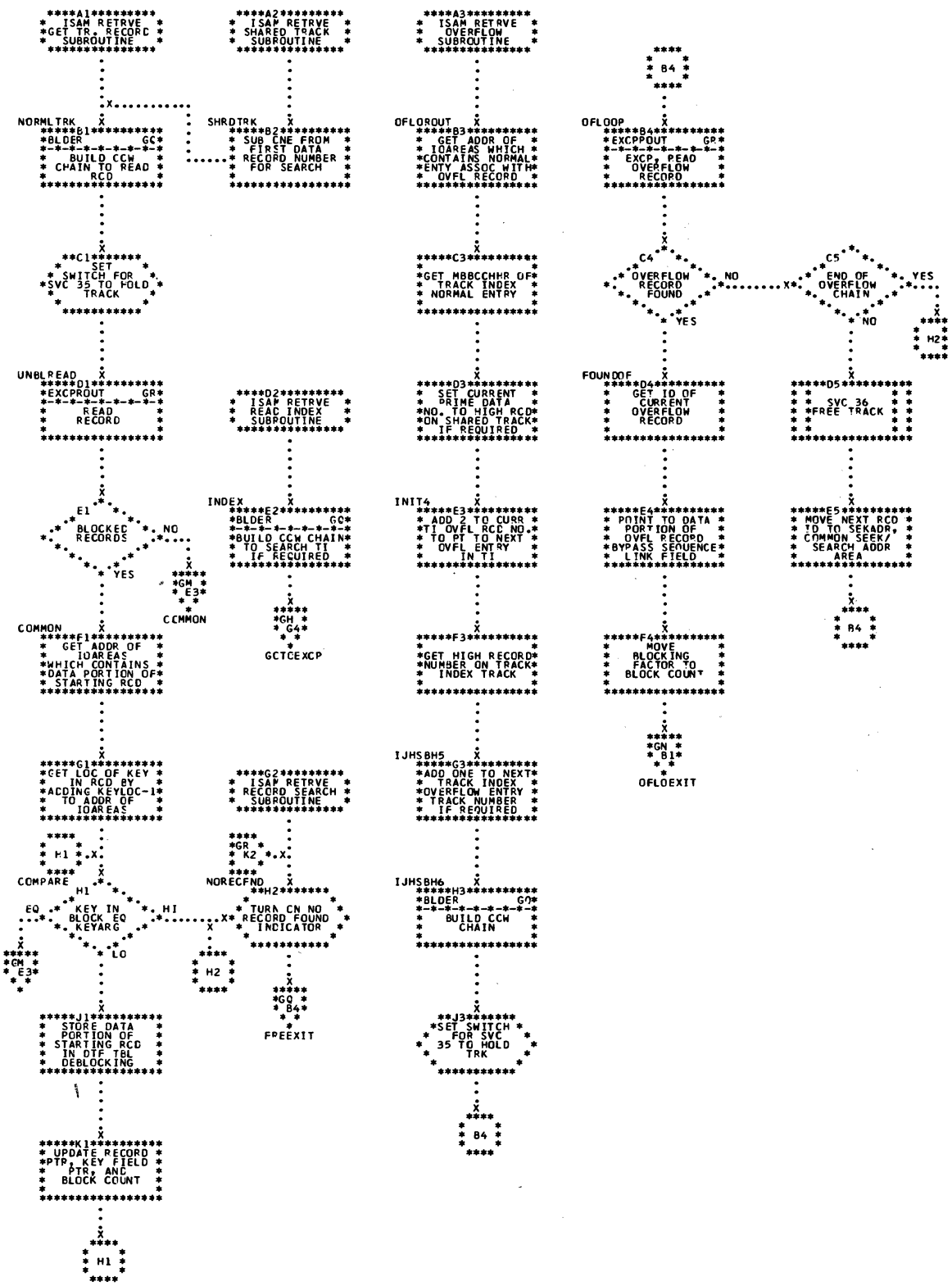


Chart GQ. ISAM RETRVE, SEQNTL: SETL Macro, $$BSETL1 (Part 4 of 5)

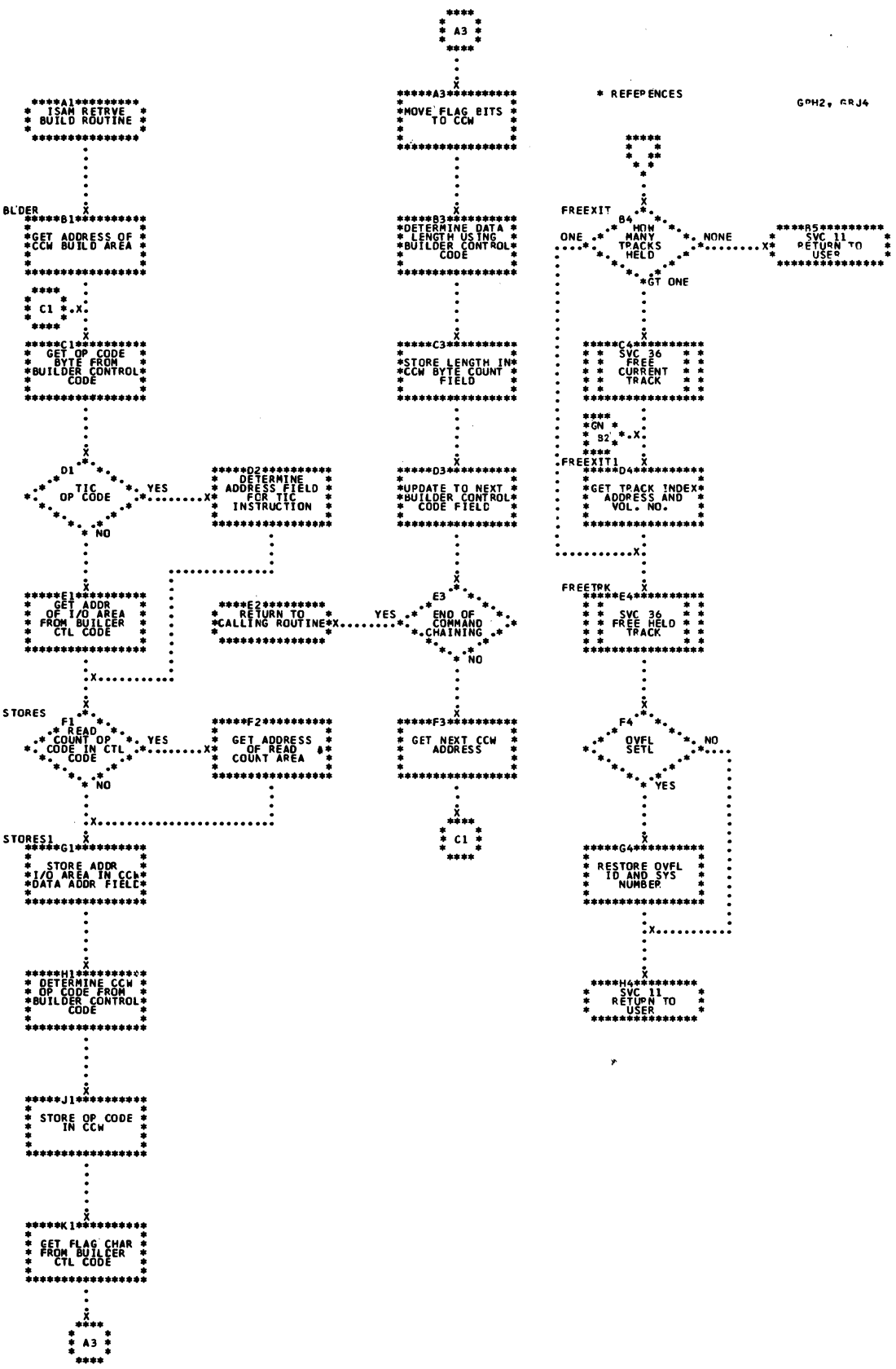


Chart GR. ISAM RETRVE, SEQNTL: SETL Macro, $$BSETL1 (Part 5 of 5)

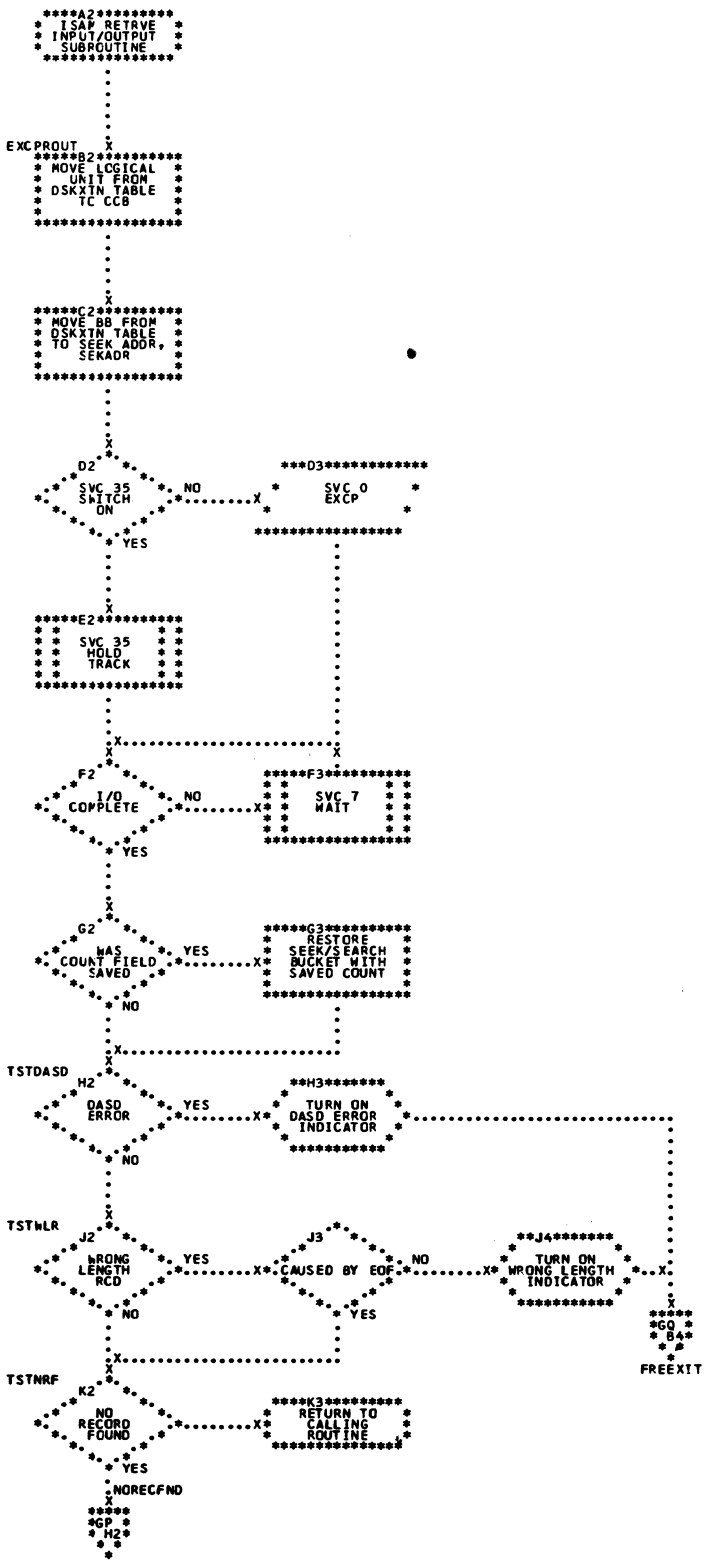


Chart HA. ISAM RETRVE, SEQNTL and ADDRTR: Subroutines (Part 1 of 3)

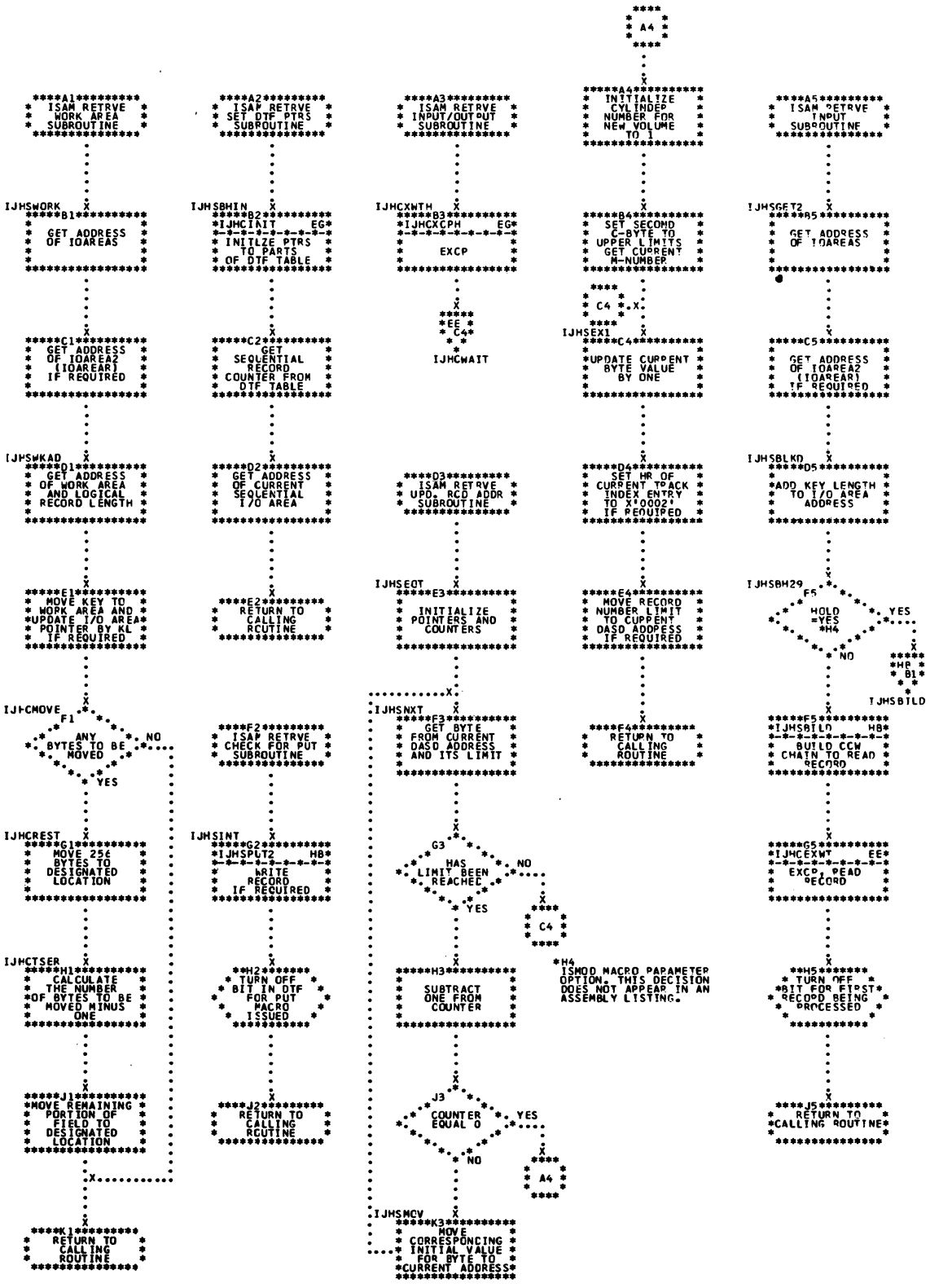


Chart HB. ISAM RETRVE, SEQNTL and ADDRTR: Subroutines (Part 2 of 3)

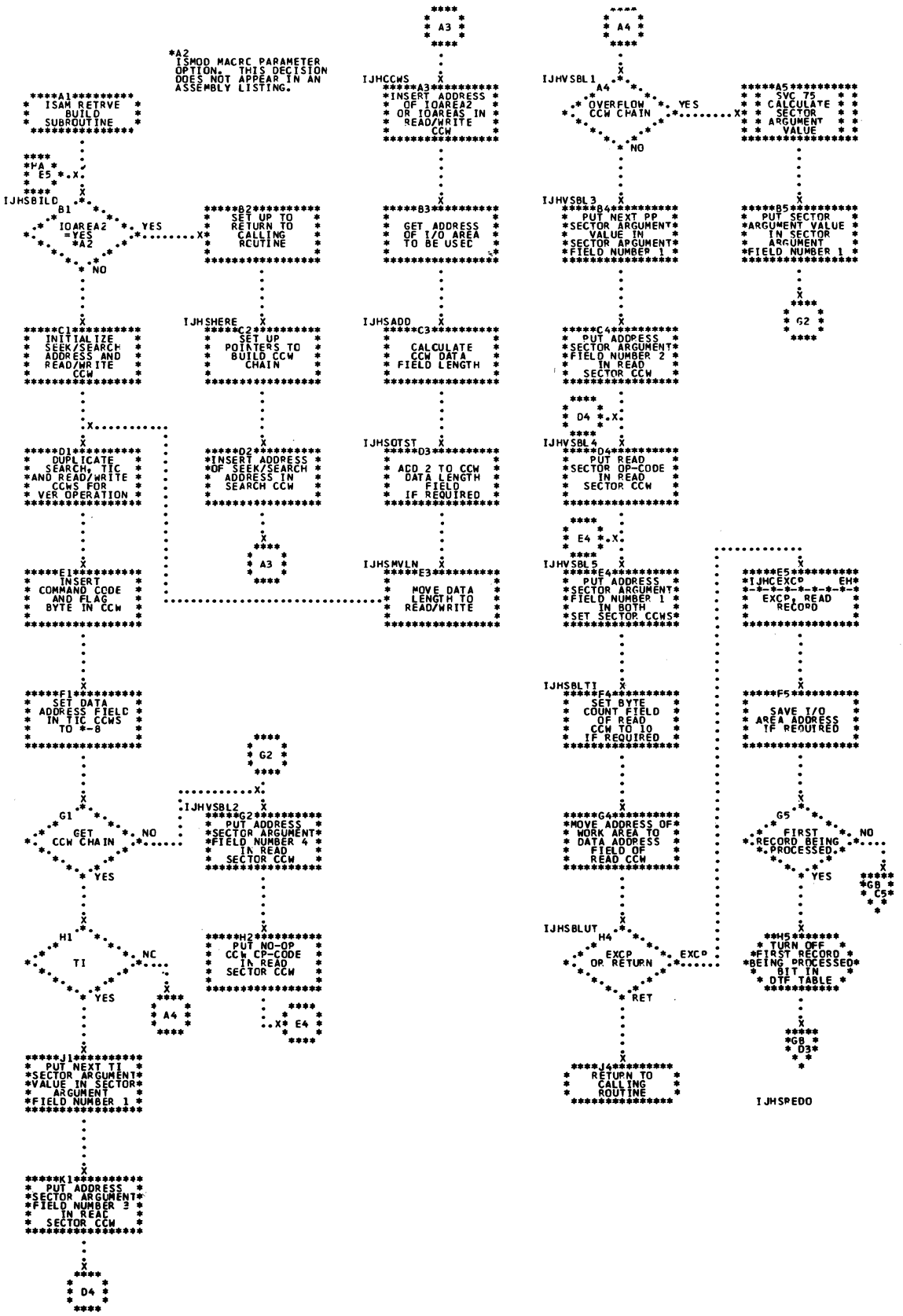


Chart HC. ISAM RETRVE, SEQNTL and ADDRTR: Subroutines (Part 3 of 3).

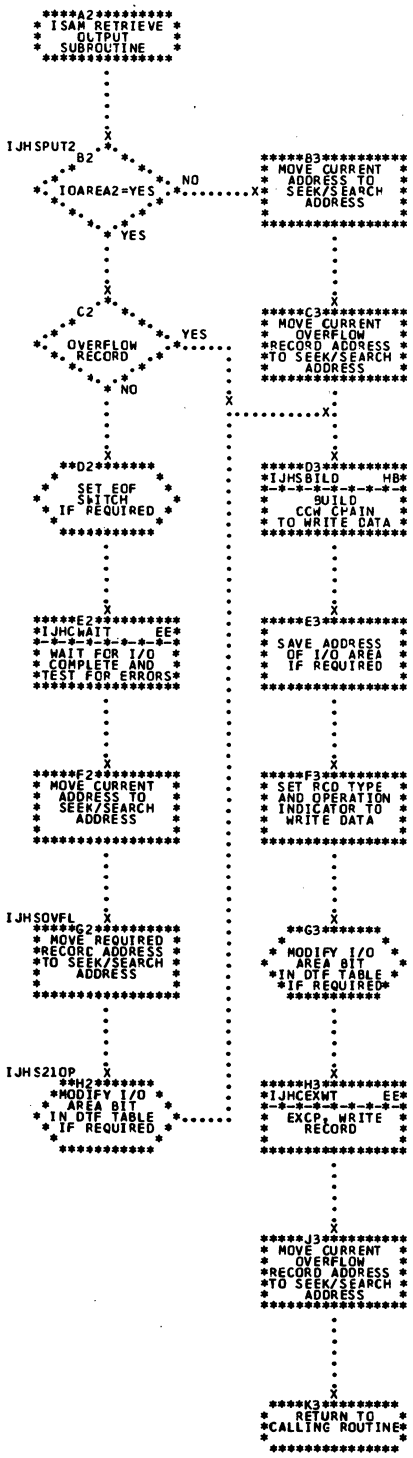


Chart JA. ISAM ADDRTR: ESETL Macro

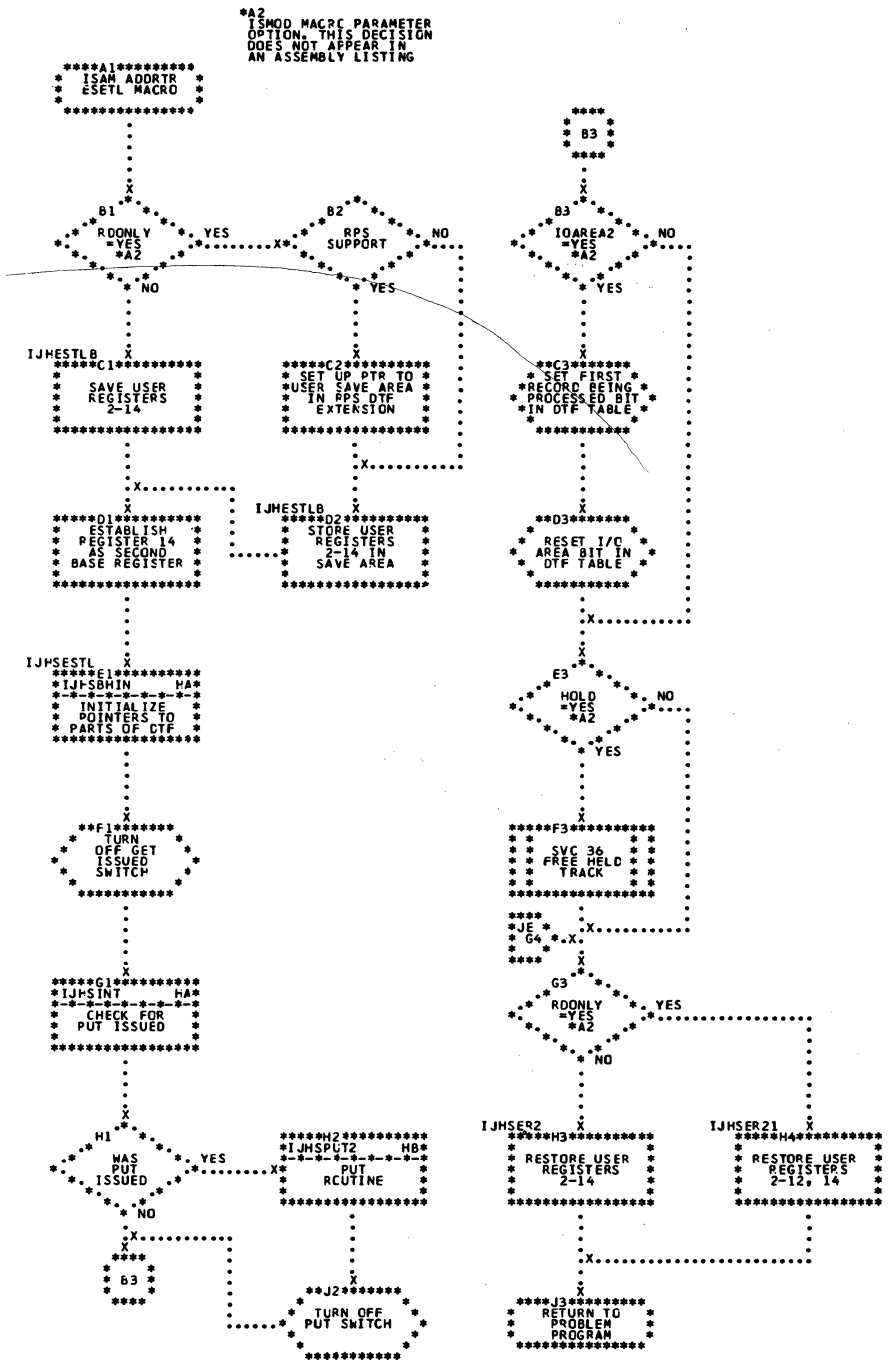


Chart JB. ISAM ADDRTR: GET Macro (Part 1 of 4)

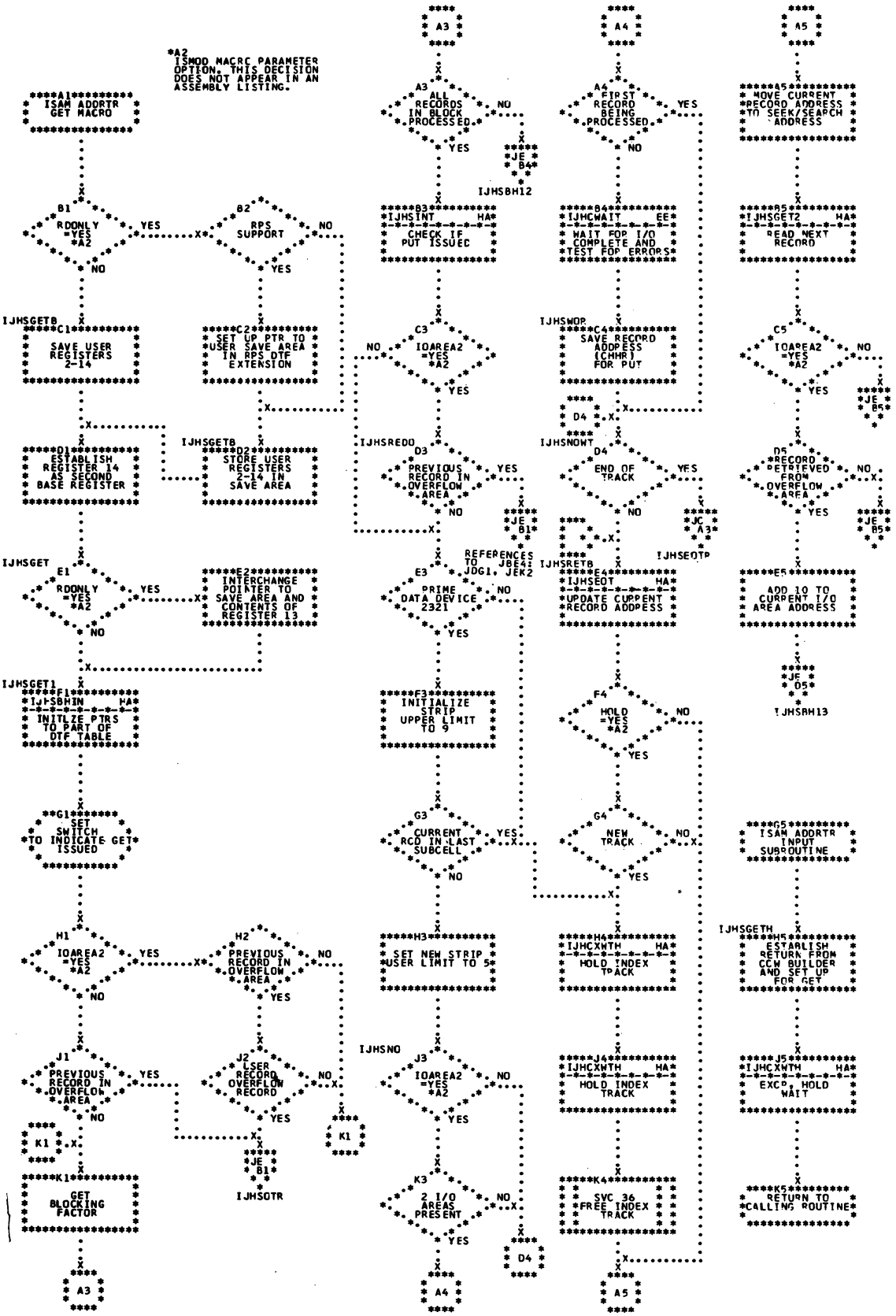


Chart JC. ISAM ADDRTR: GET Macro (Part 2 of 4)

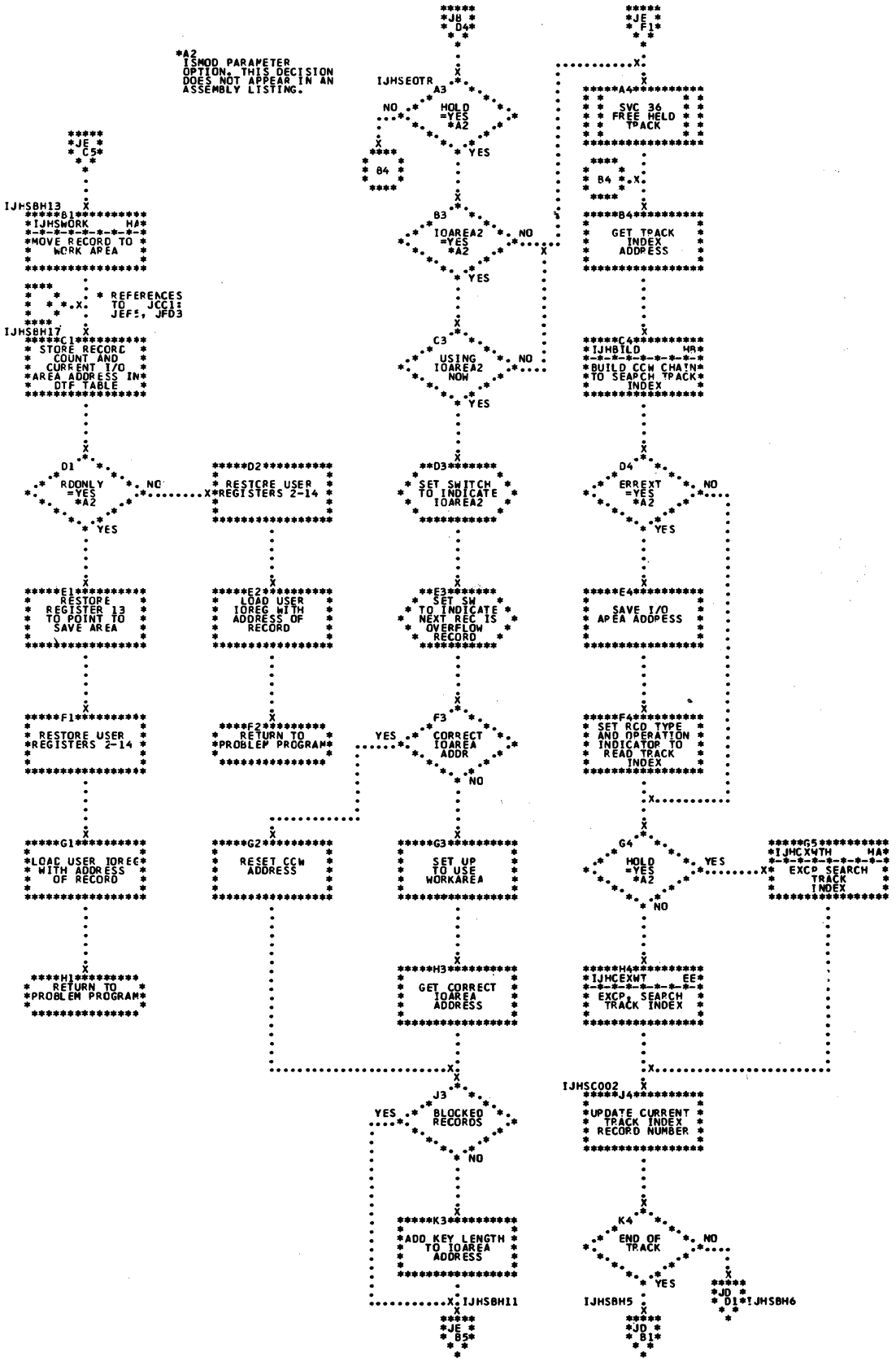


Chart JD. ISAM ADDRTR: GET Macro (Part 3 of 4)

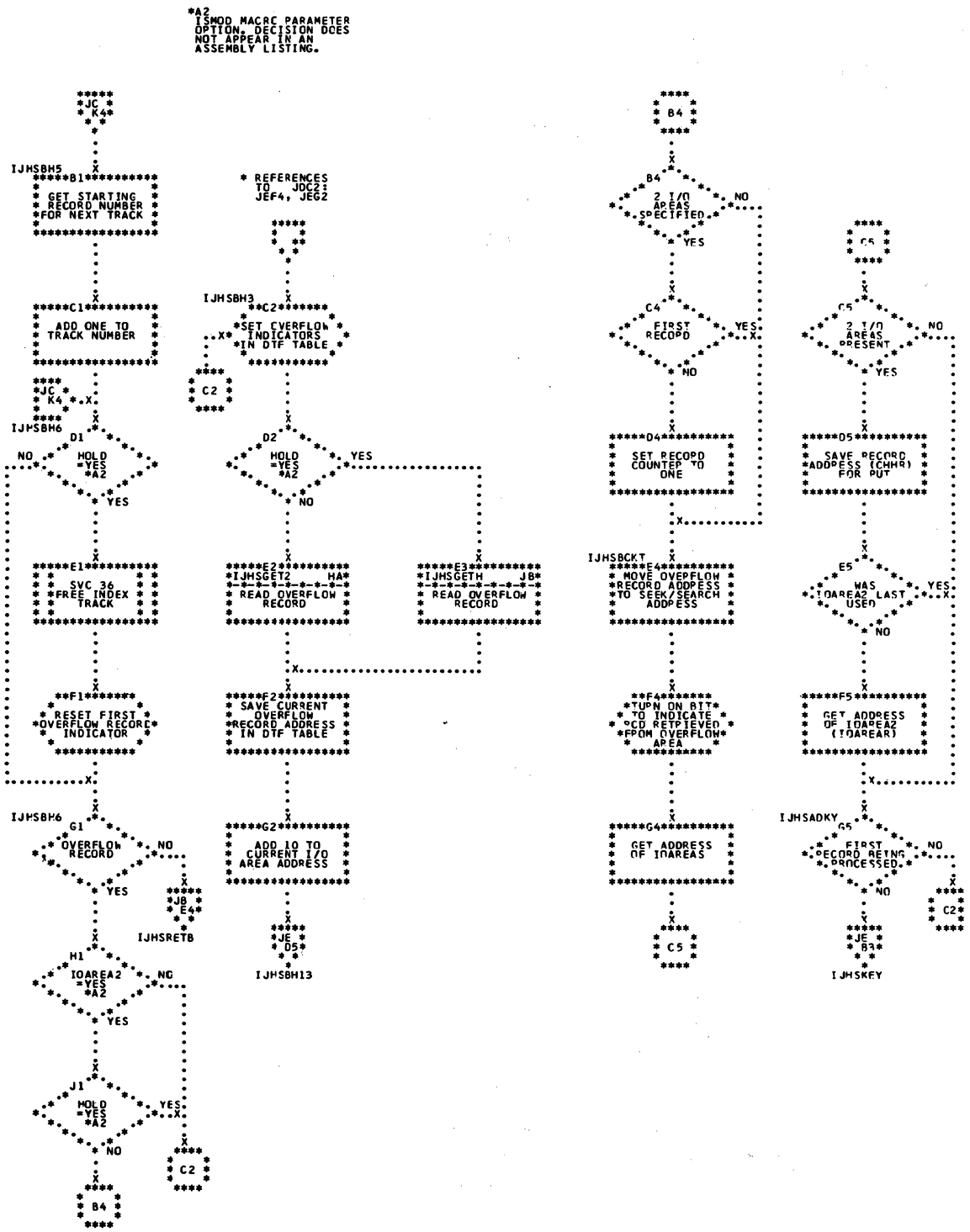


Chart JE. ISAM ADDRTR: GET Macro (Part 4 of 4)

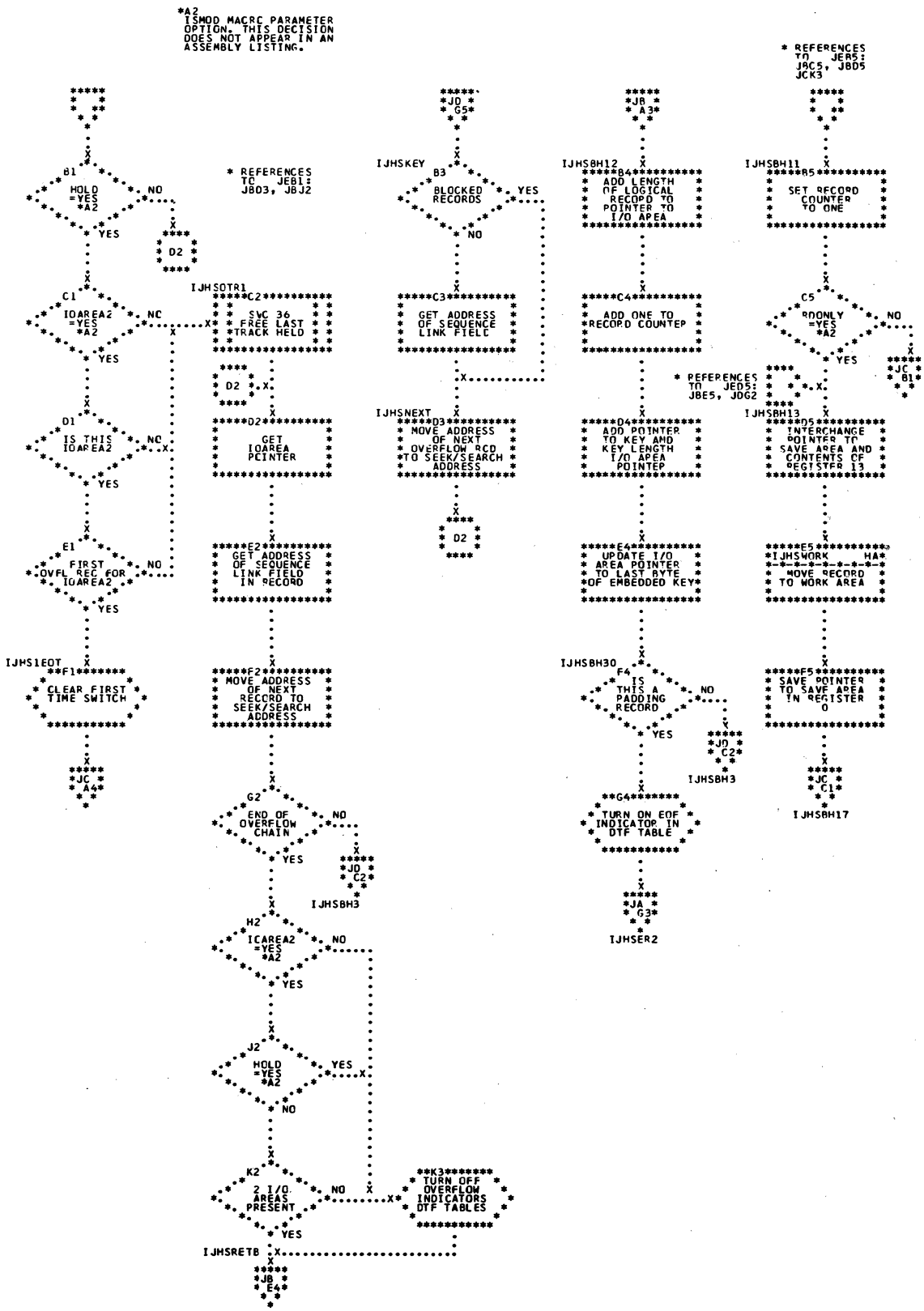


Chart JF. ISAM ADDRTR: PUT Macro

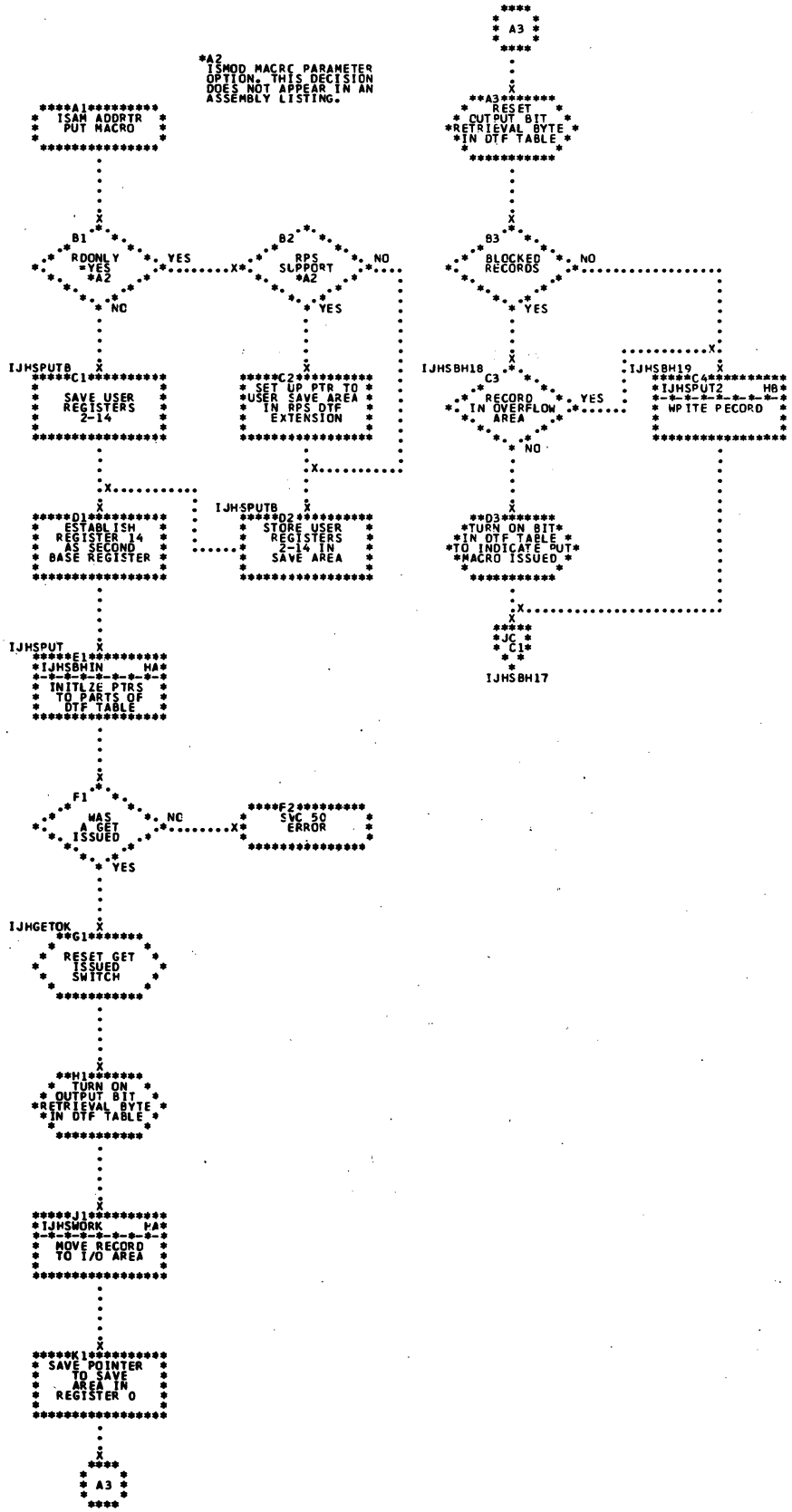


Chart JG. ISAM ADDRTR: READ Macro, KEY

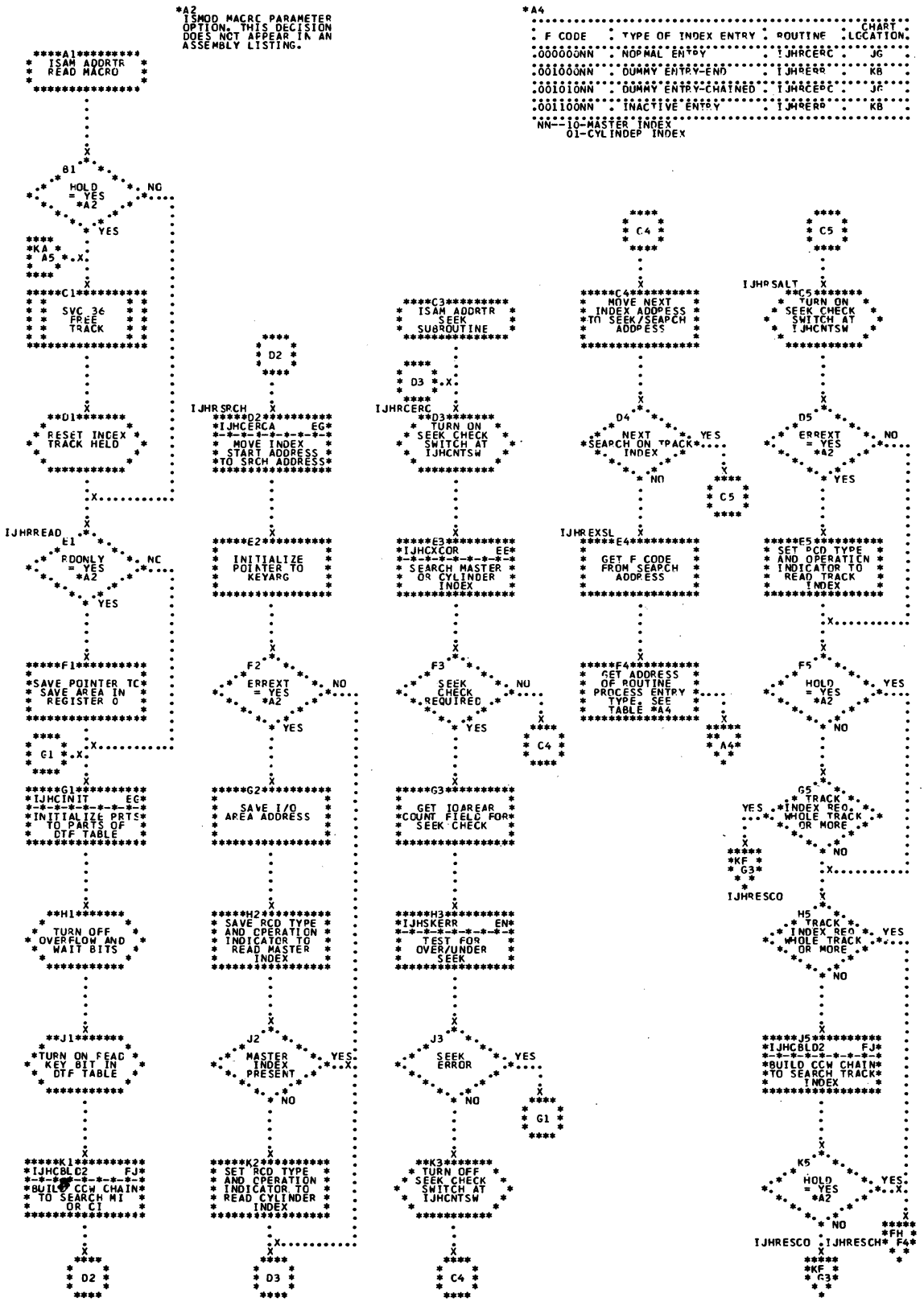


Chart JH. ISAM ADDRTR: SETL Macro, $$BSETL (Part 1 of 3)

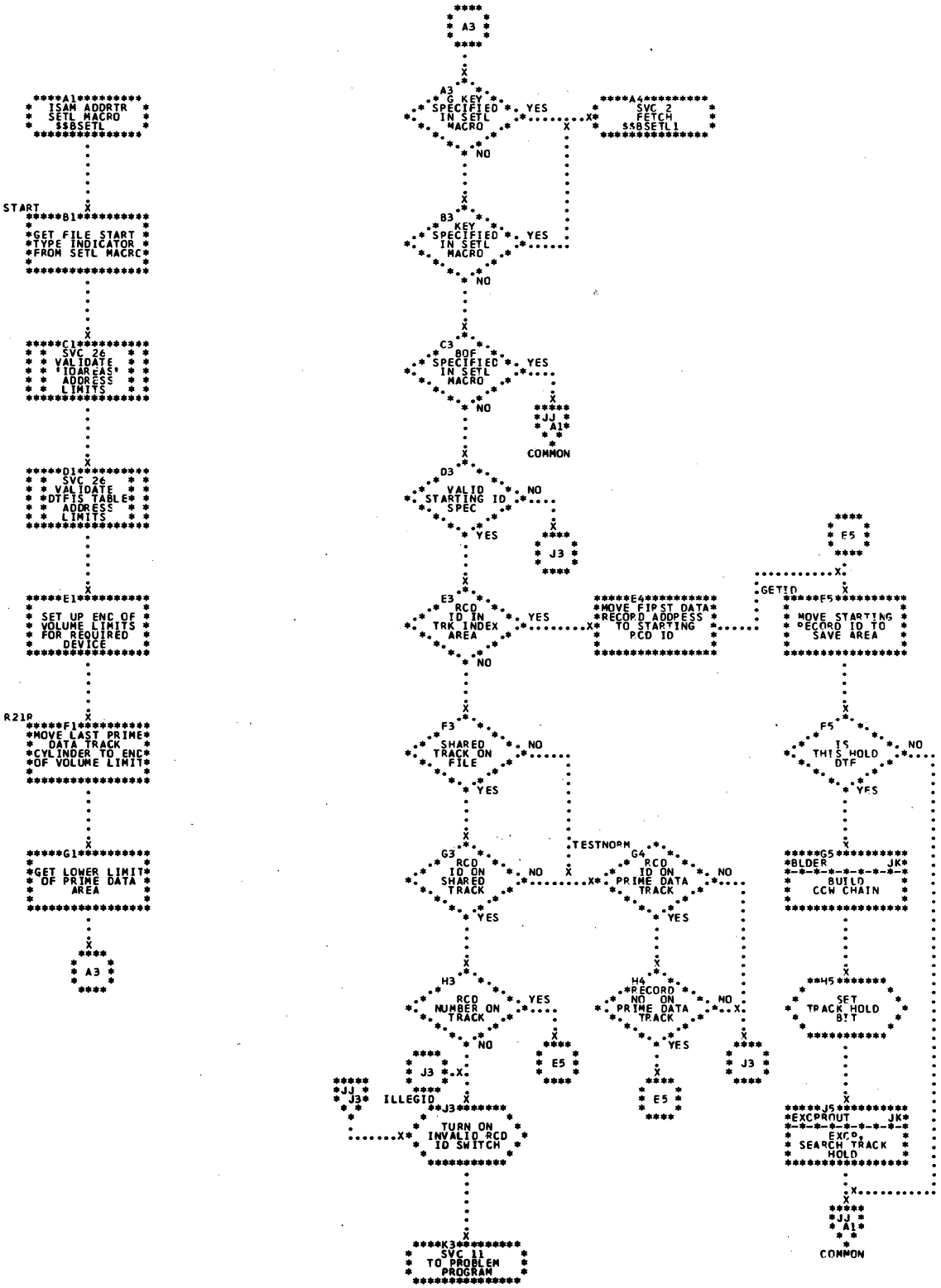


Chart JJ. ISAM ADDRTR: SETL Macro, $$BSETL (Part 2 of 3)

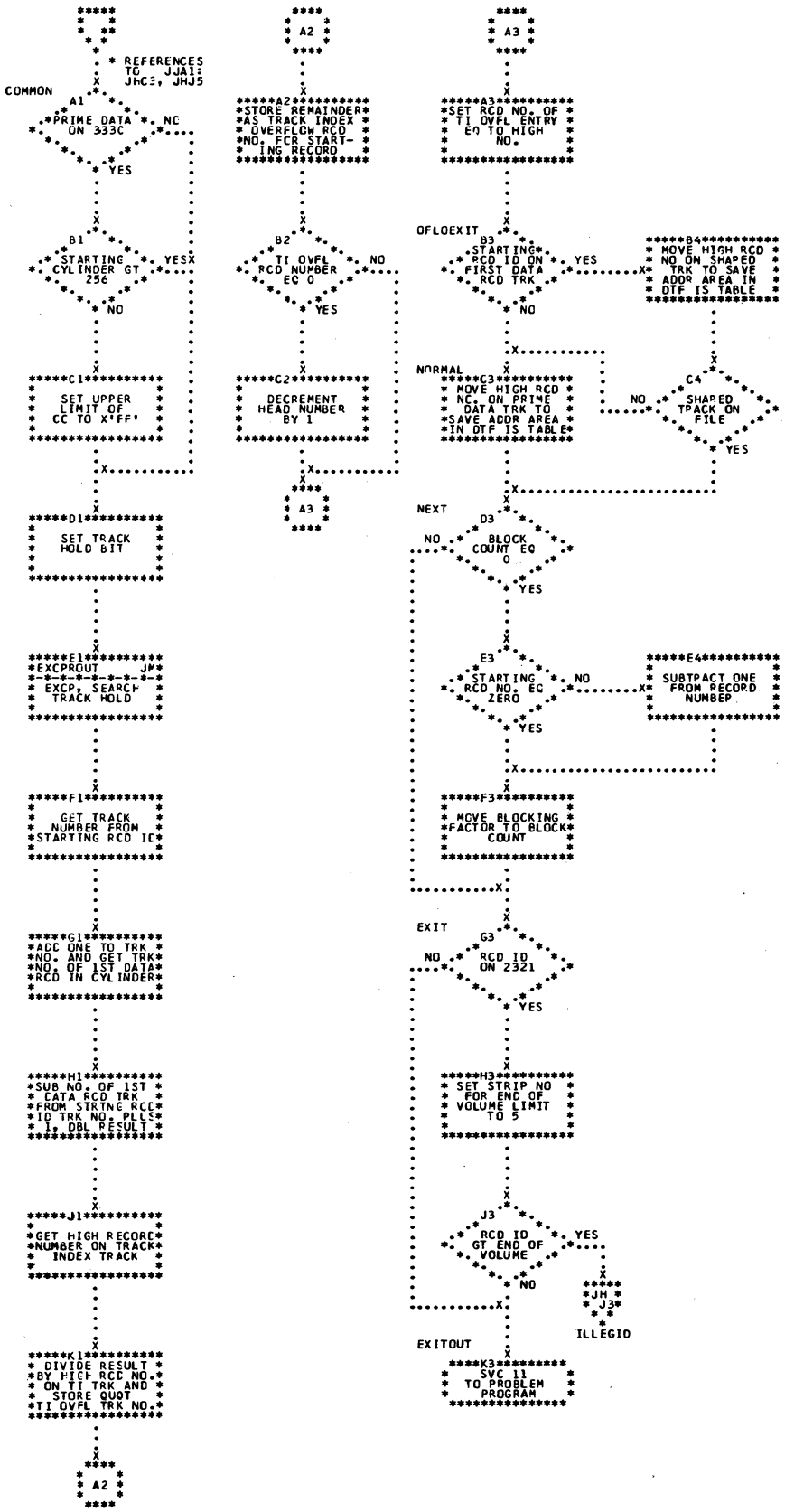


Chart JK. ISAM ADDRTR: SETL Macro, $$BSETL (Part 3 of 3)

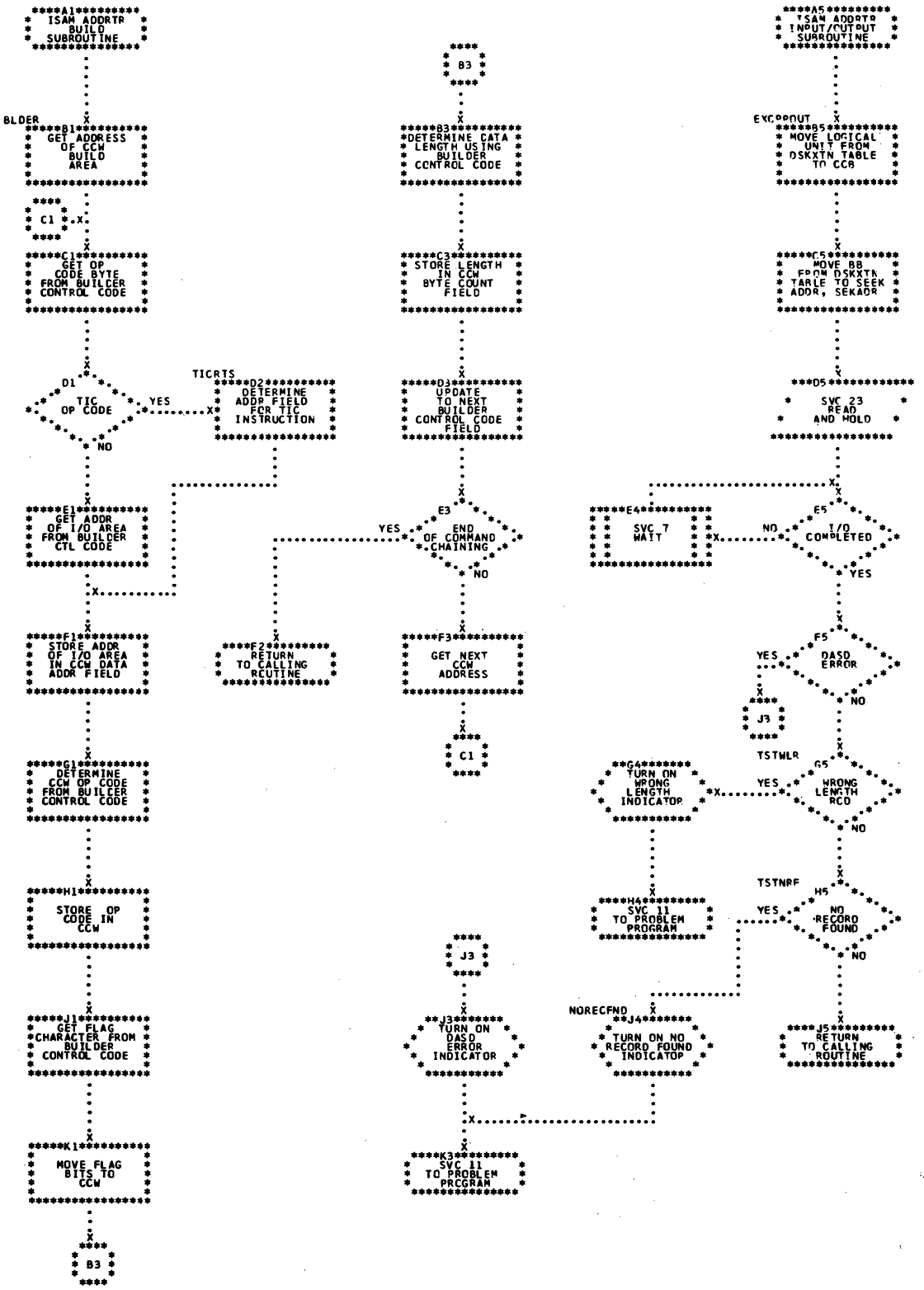
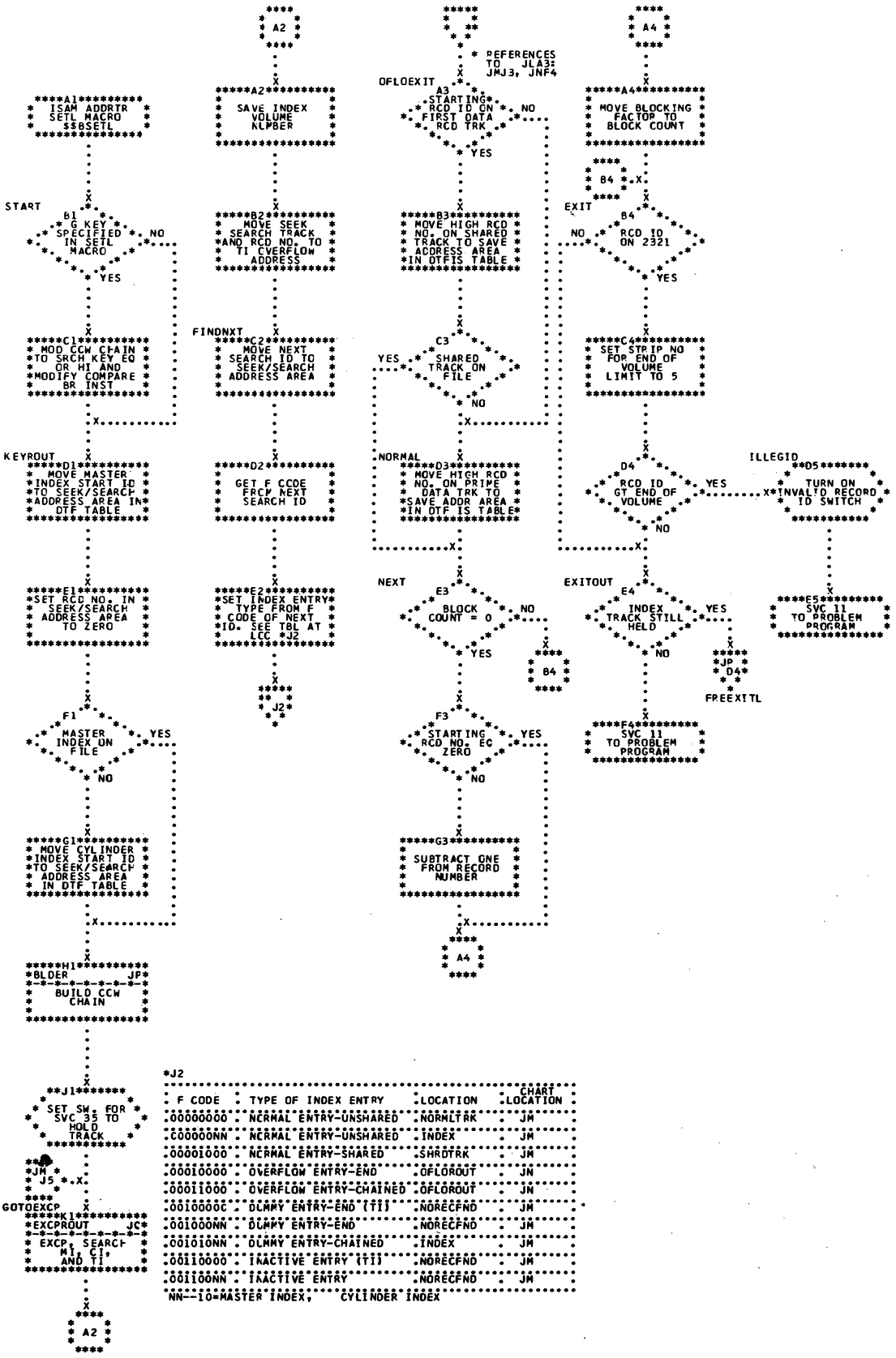


Chart JL. ISAM ADDRTR: SETL Macro, $$BSETL1 (Part 1 of 5)



*J2

F CODE	TYPE OF INDEX ENTRY	LOCATION	CHART LOCATION
00000000	NORMAL ENTRY-UNSHARED	NORMLTRK	JM
CCCCCONN	NORMAL ENTRY-UNSHARED	INDEX	JM
00001000	NORMAL ENTRY-SHARED	SHRDTRK	JM
00010000	OVERFLOW ENTRY-END	OFLOROUT	JN
00011000	OVERFLOW ENTRY-CHAINED	OFLOROUT	JN
00100000	DUMMY ENTRY-END (III)	NORECFND	JM
00100ONN	DUMMY ENTRY-END	NORECFND	JM
00101ONN	DUMMY ENTRY-CHAINED	INDEX	JM
00110000	INACTIVE ENTRY (III)	NORECFND	JM
00110ONN	INACTIVE ENTRY	NORECFND	JM

NN--10=MASTER INDEX, CYLINDER INDEX

Chart JM. ISAM ADDRTR: SETL Macro, $$BSETL1 (Part 2 of 5)

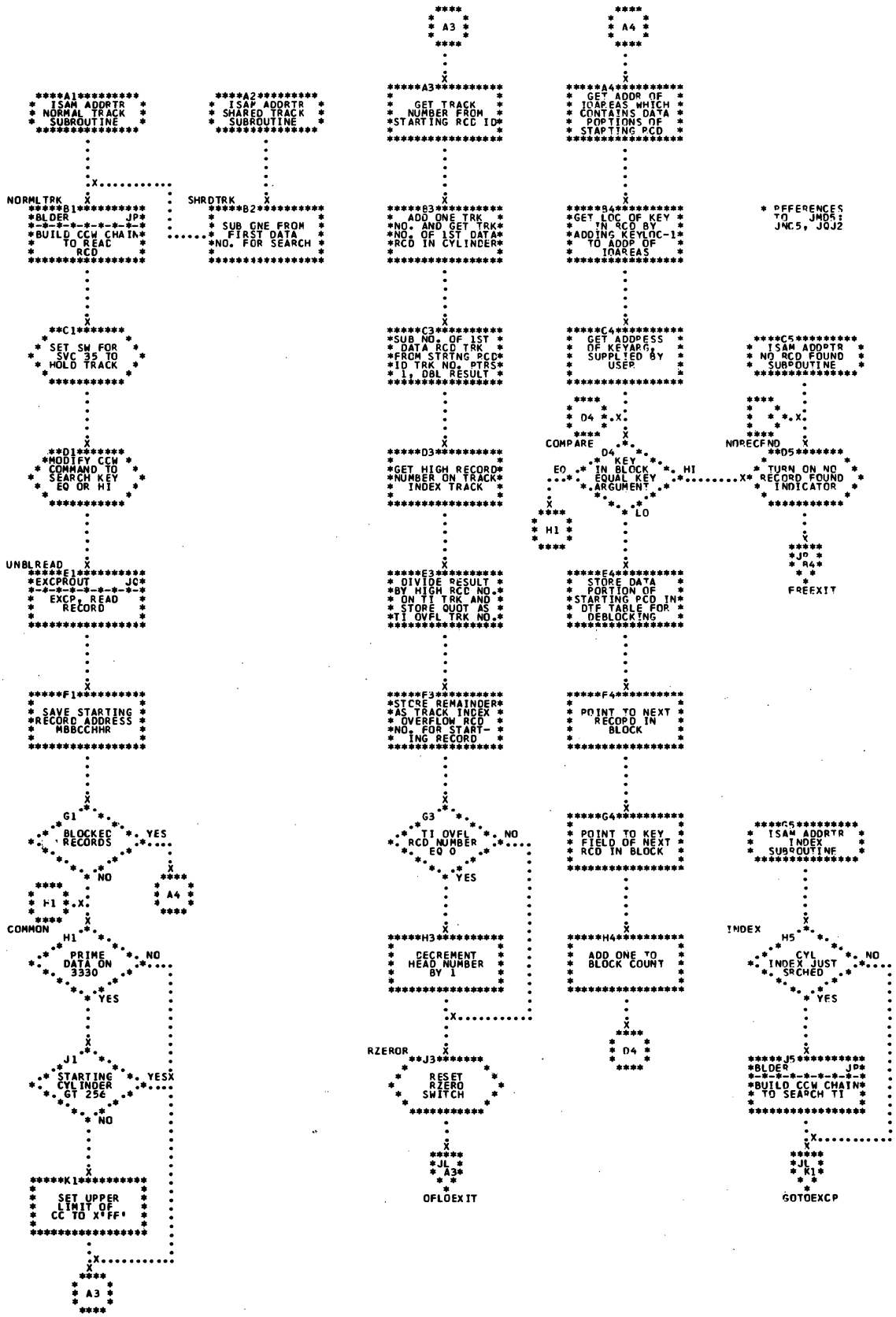


Chart JN. ISAM ADDRTR: SETL Macro, $$BSETL1 (Part 3 of 5)

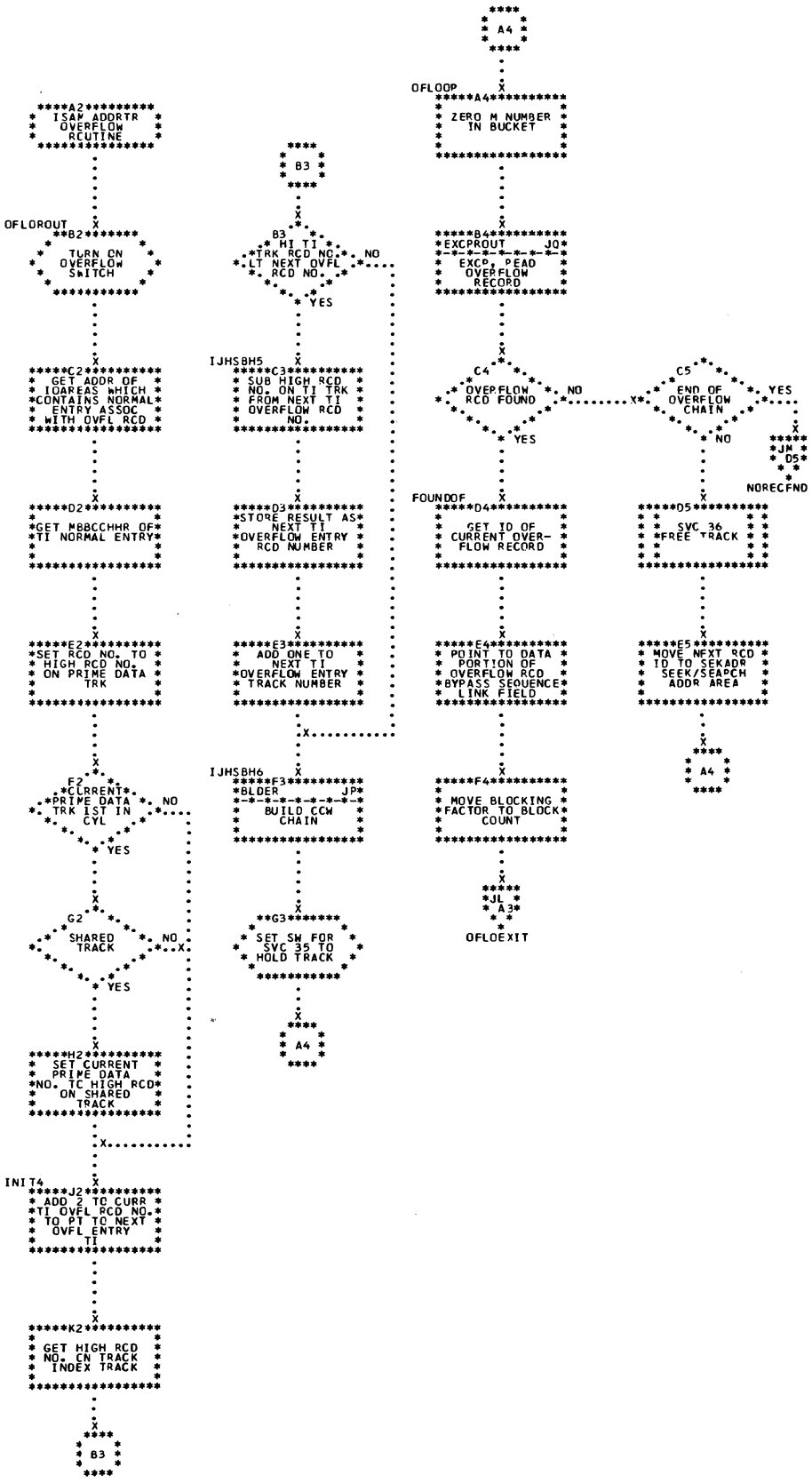


Chart JP. ISAM ADDRTR: SETL Macro, $$BSETL1 (Part 4 of 5)

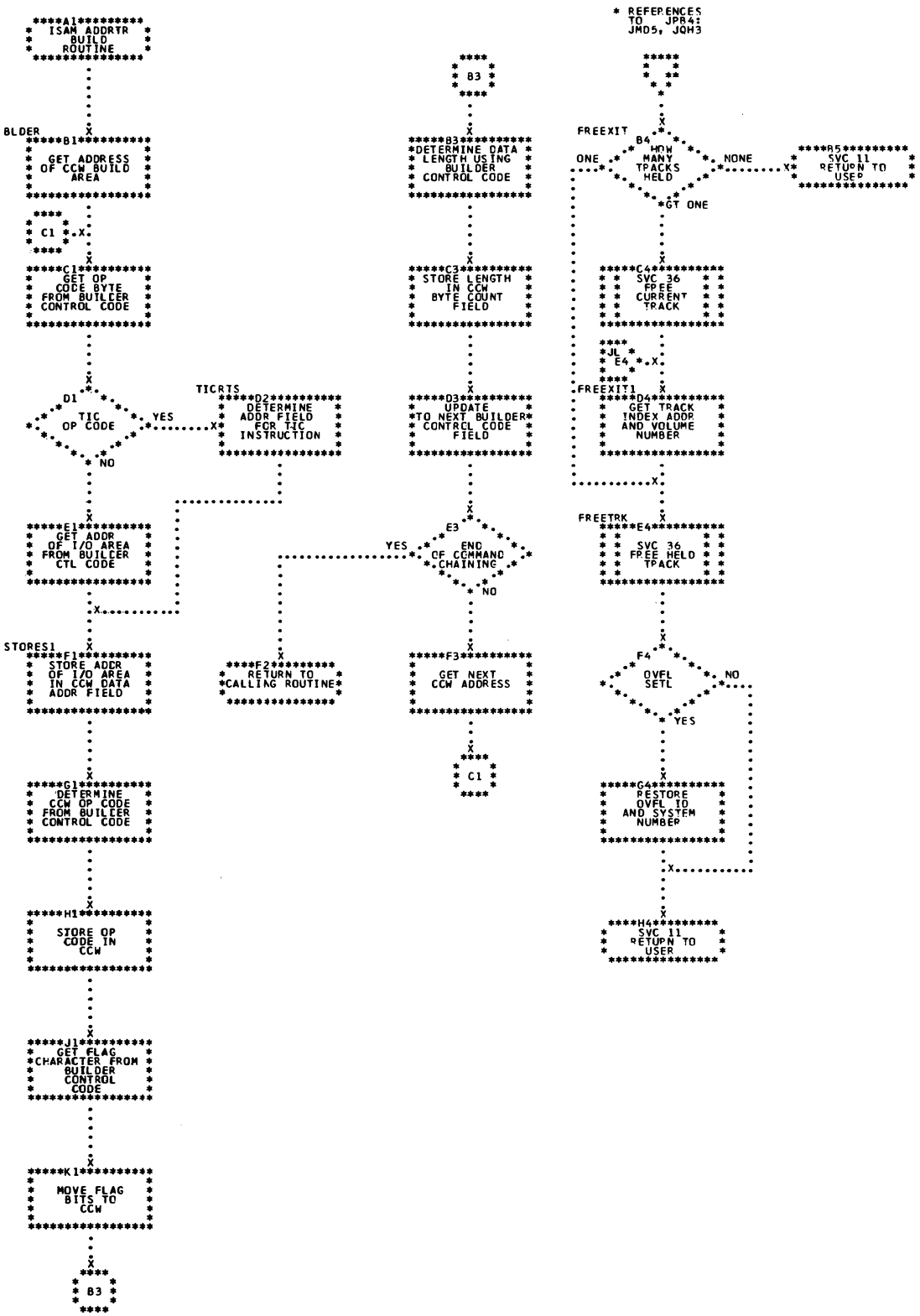


Chart JQ. ISAM ADDRTR: SETL Macro, $$BSETL1 (Part 5 of 5)

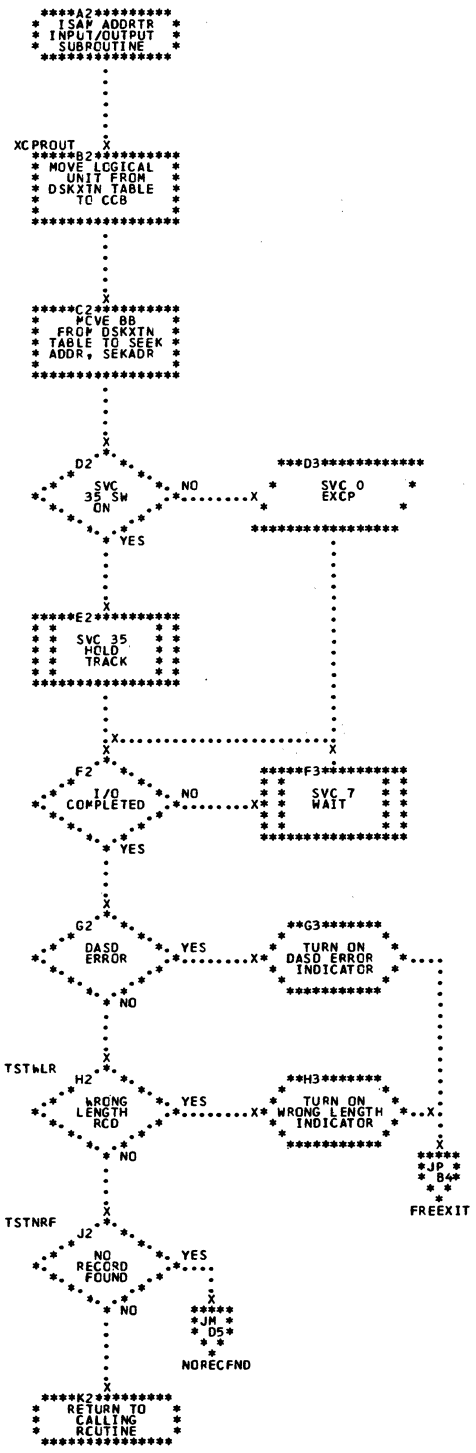


Chart KA. ISAM ADDRTR: WAITF Macro (Part 1 of 5)

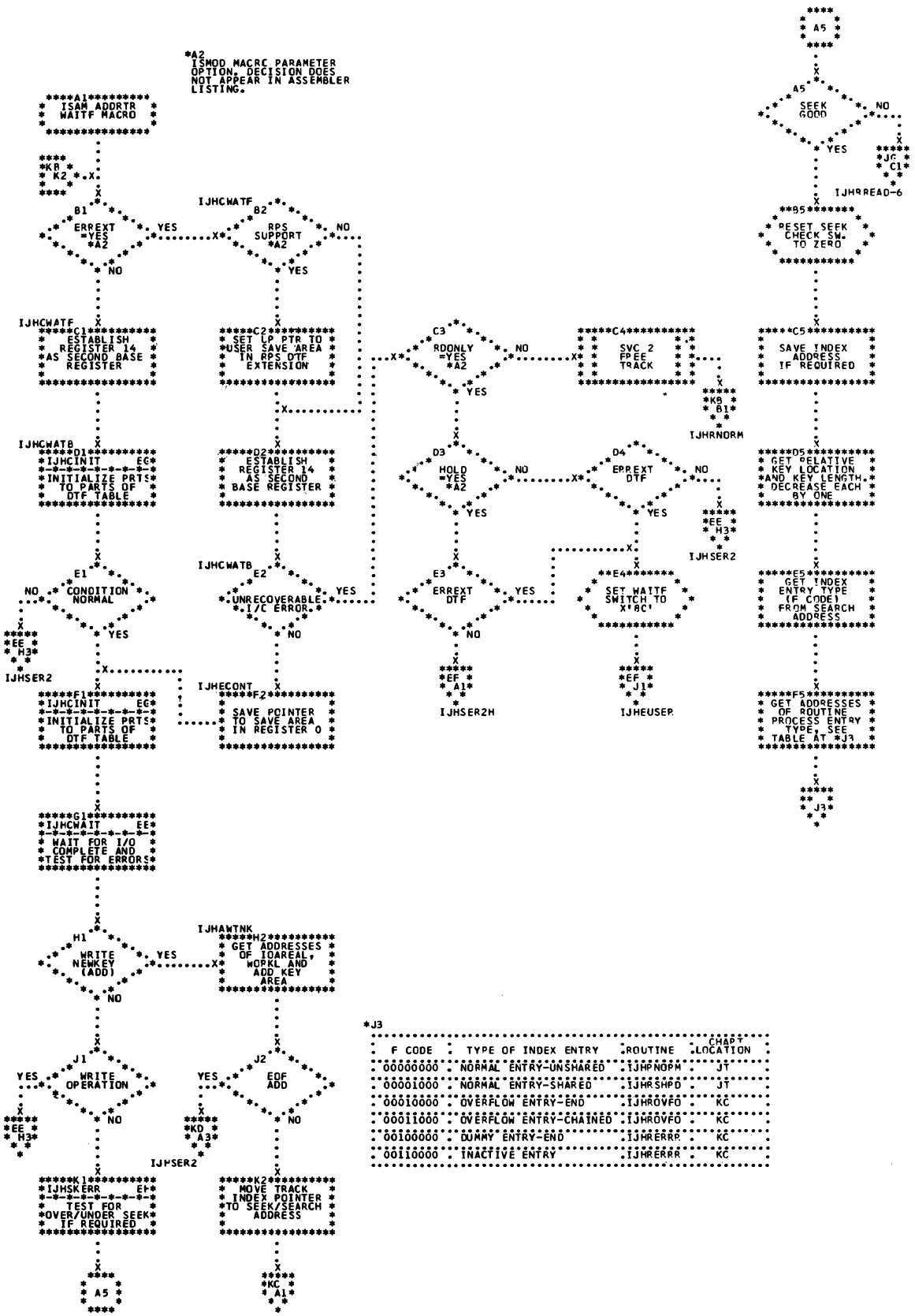


Chart KB. ISAM ADDRTR: WAITF Macro (Part 2 of 5)

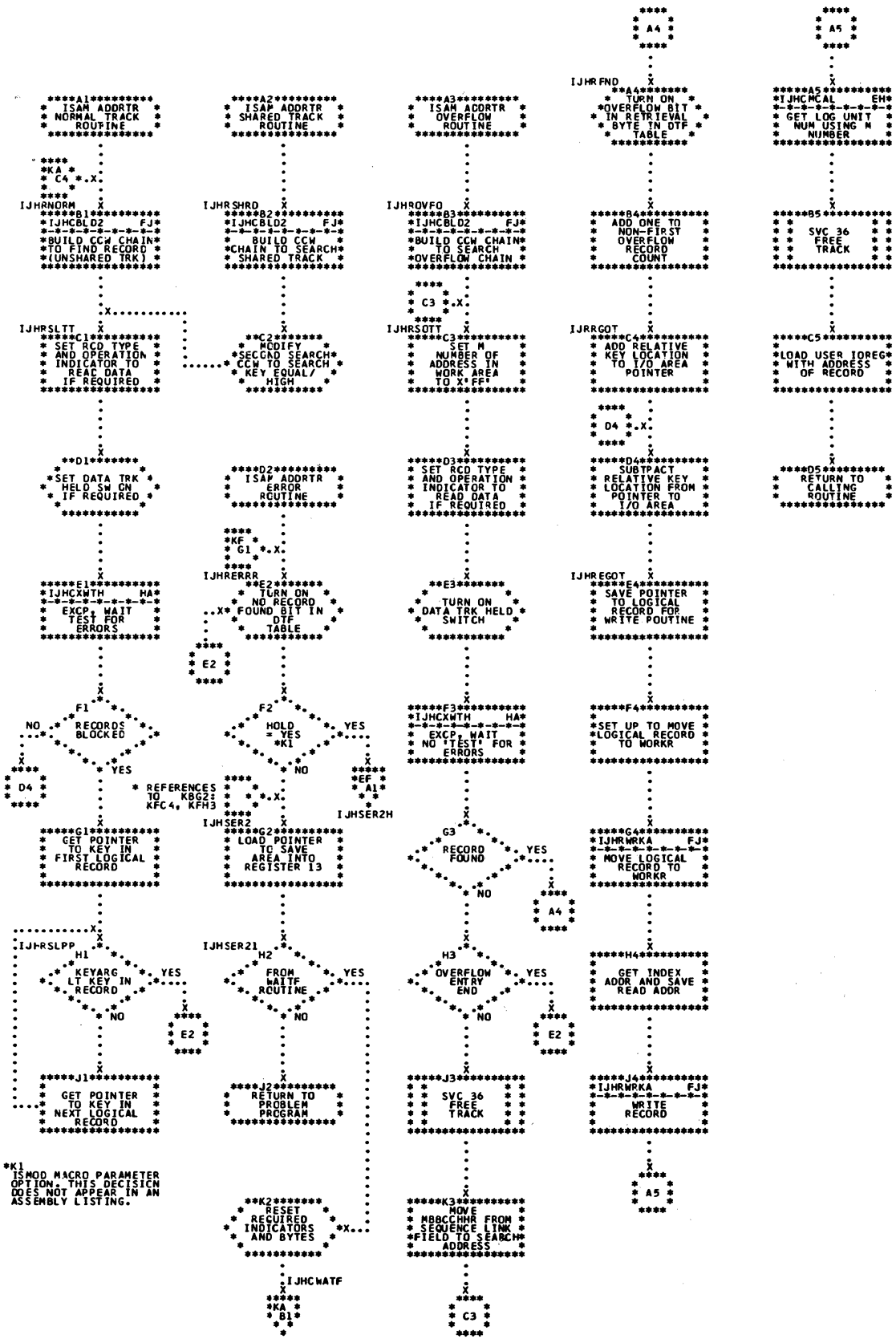


Chart KC. ISAM ADDRTR: WAITF Macro (Part 3 of 5)

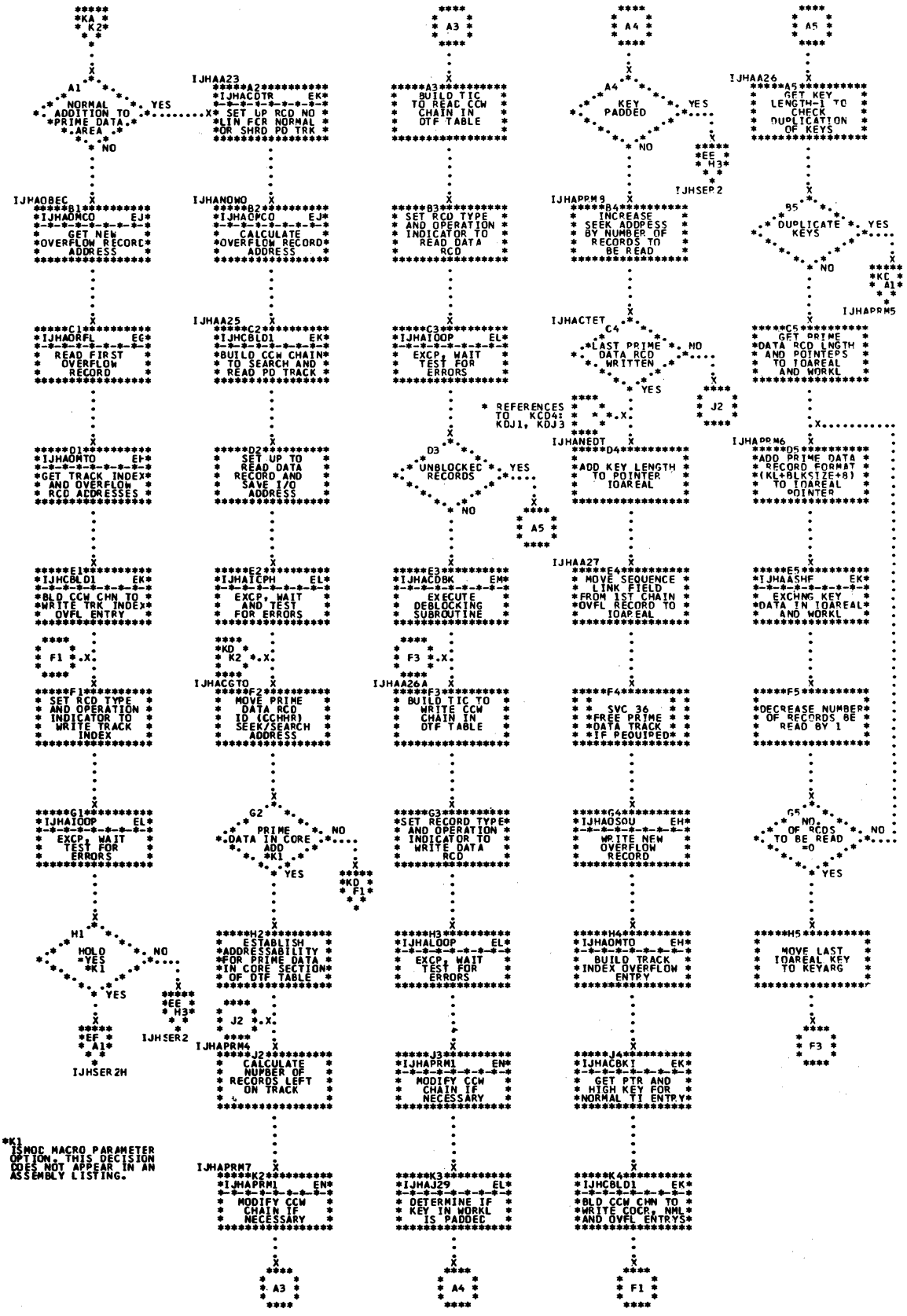


Chart KD. ISAM ADDRTR: WAITF Macro (Part 4 of 5)

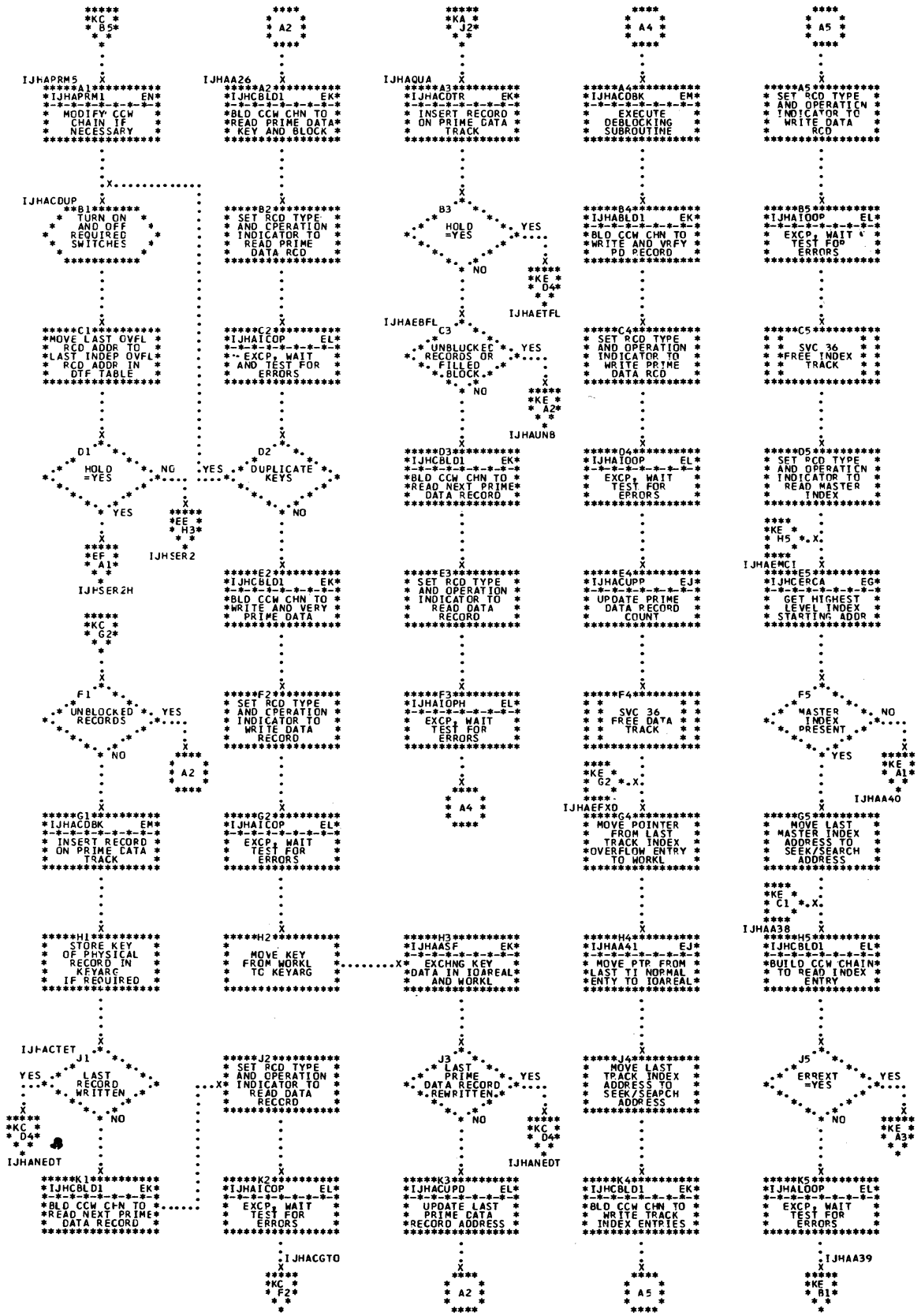


Chart KE. ISAM ADDRTR: WAITF Macro (Part 5 of 5)

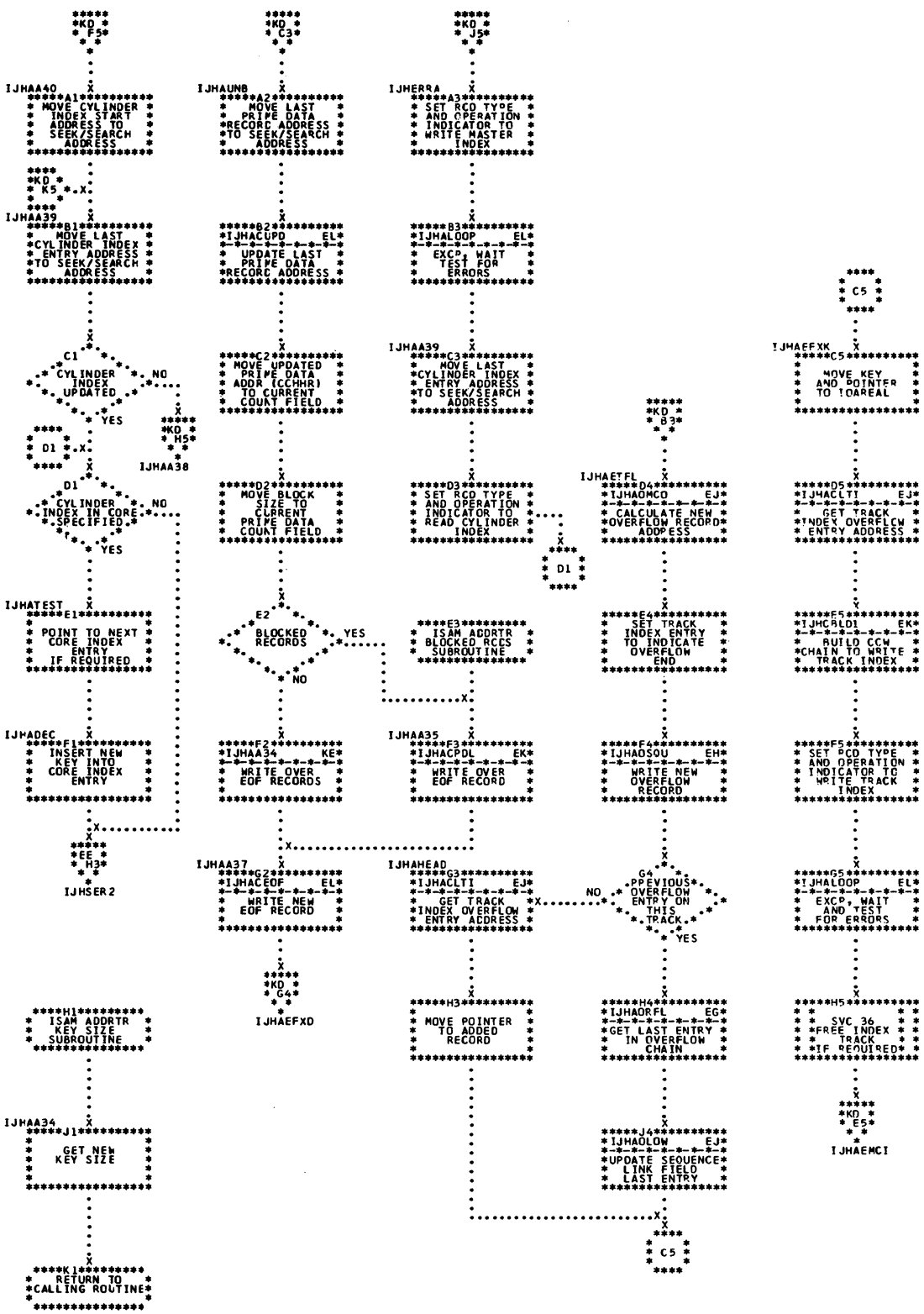


Chart KF. ISAM ADDRTR: WRITE Macro, KEY

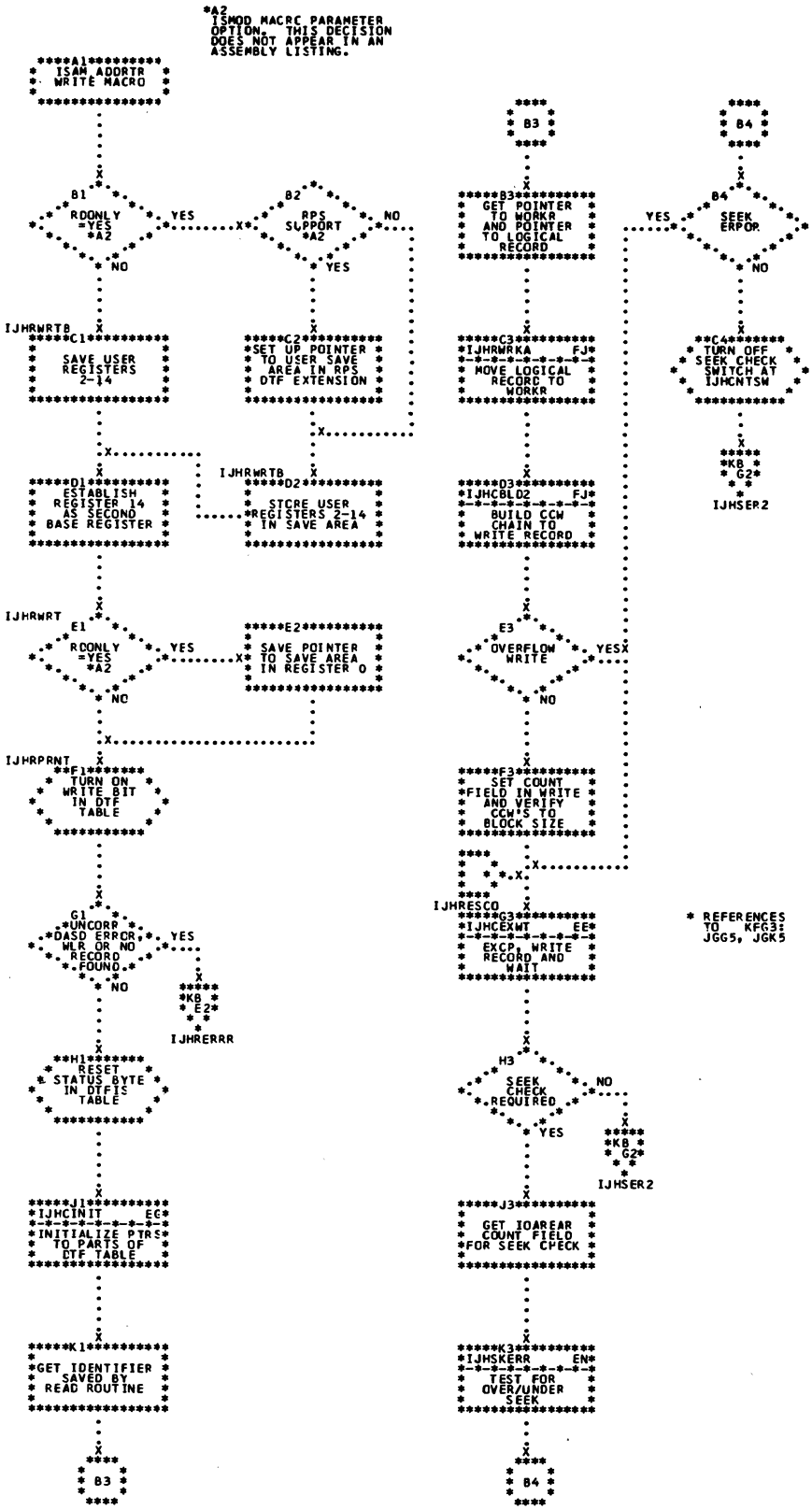


Chart IA. $$BOIS01: ISAM Open, Phase 1 (Part 1 of 2)

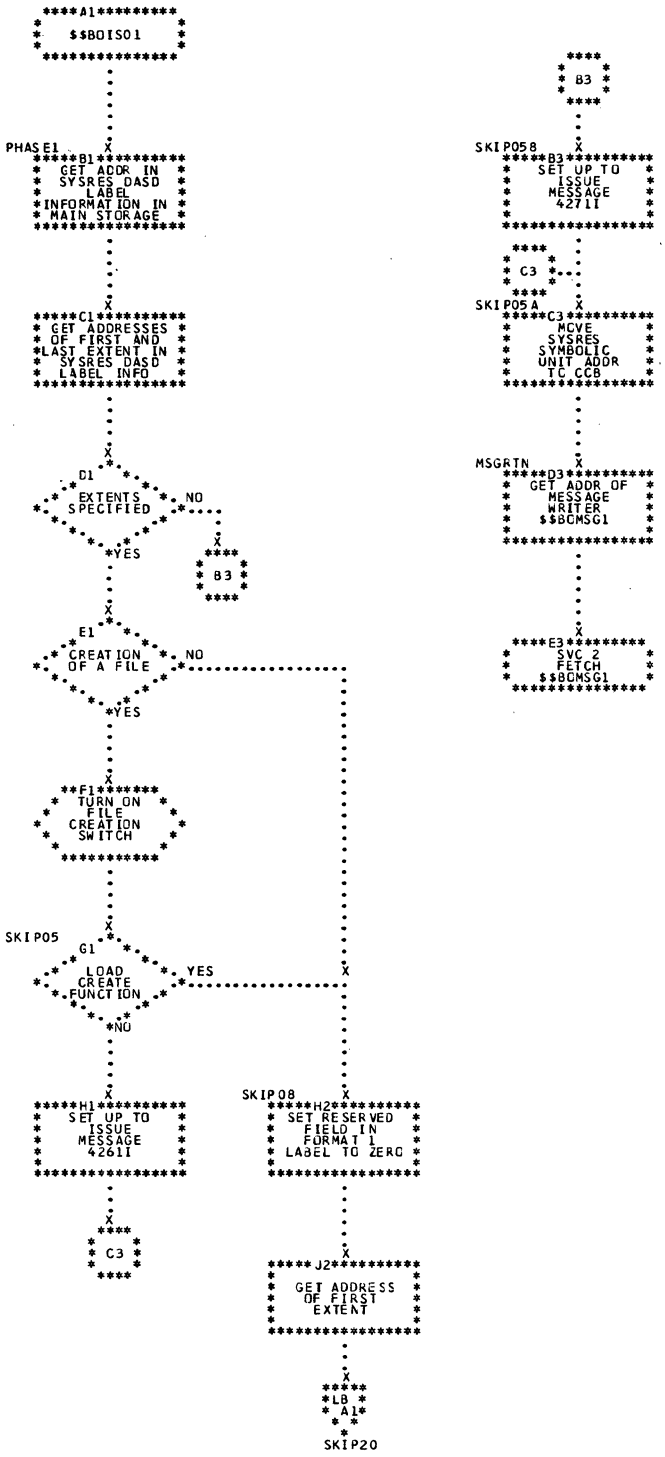


Chart LB. $$BOIS01: ISAM Open, Phase 1 (Part 2 of 2)

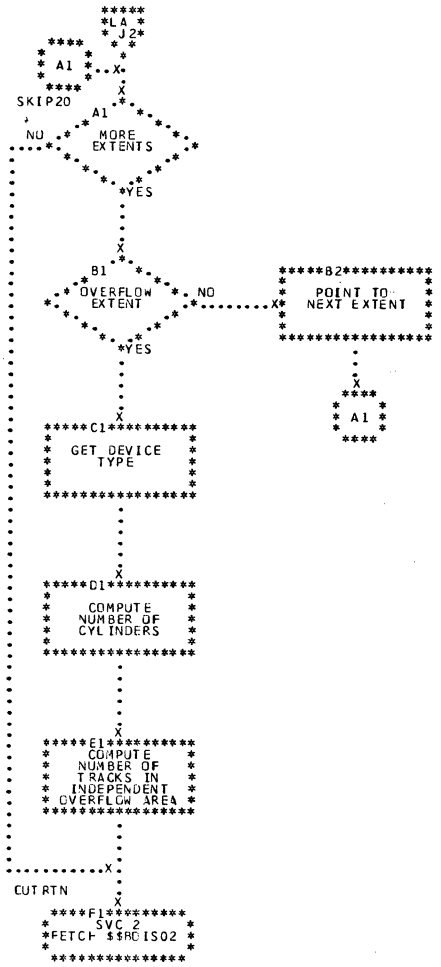


Chart LC. $$BOIS02: ISAM Open, Phase 2

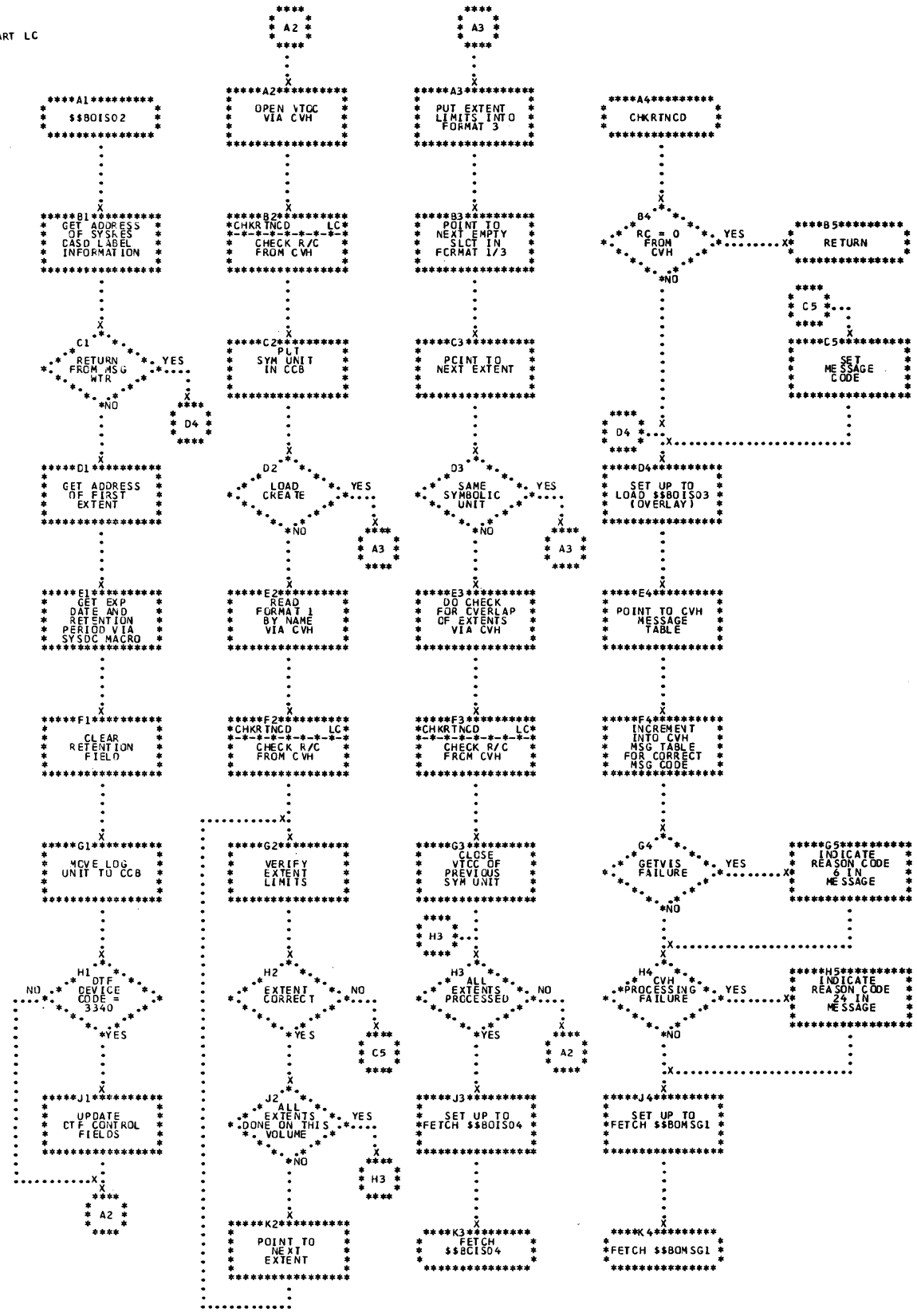


Chart ID. $$BOIS04: ISAM Open, Phase 4

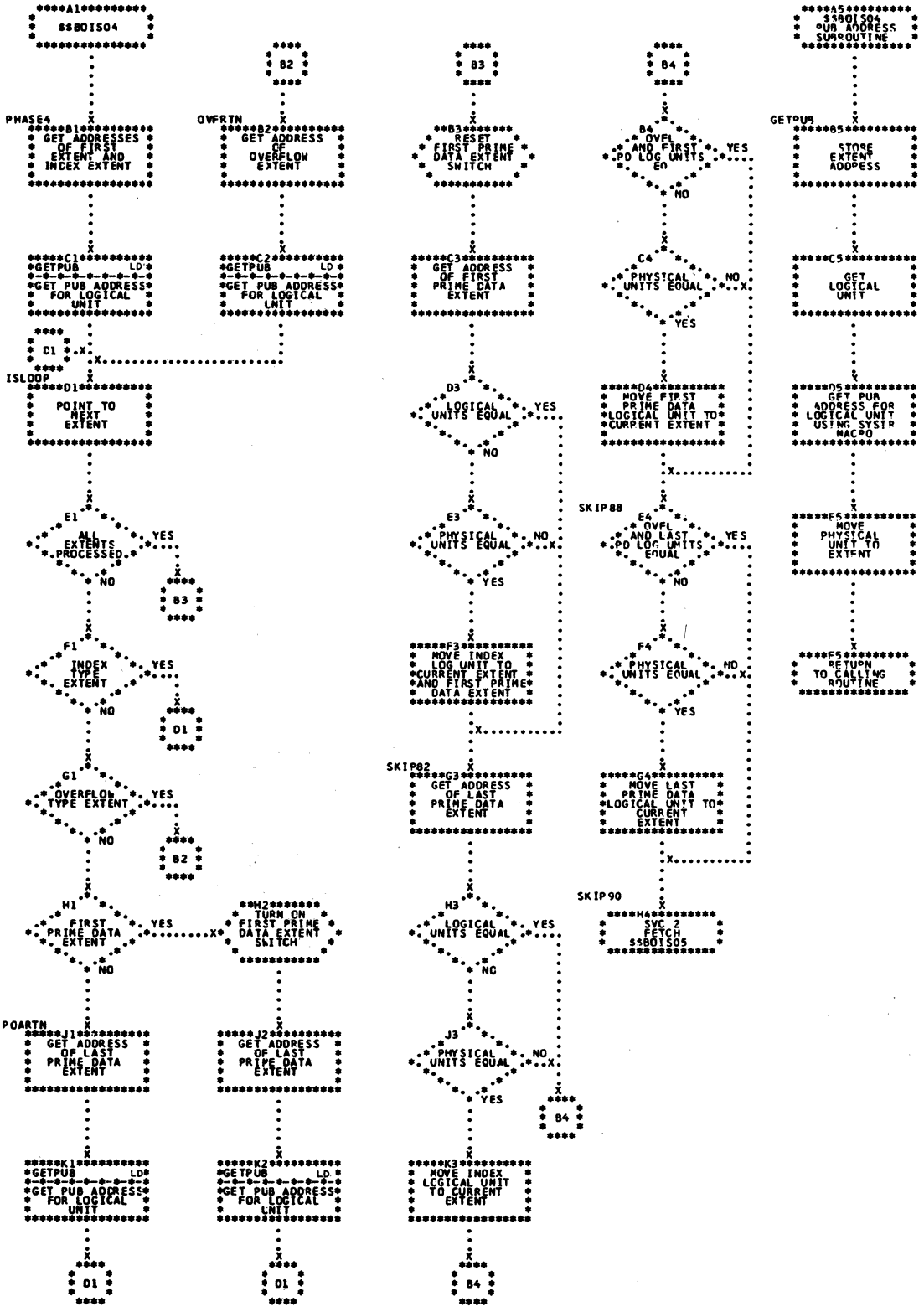


Chart LE. $$BOIS05: ISAM Open, Phase 5 (Part 1 of 3)

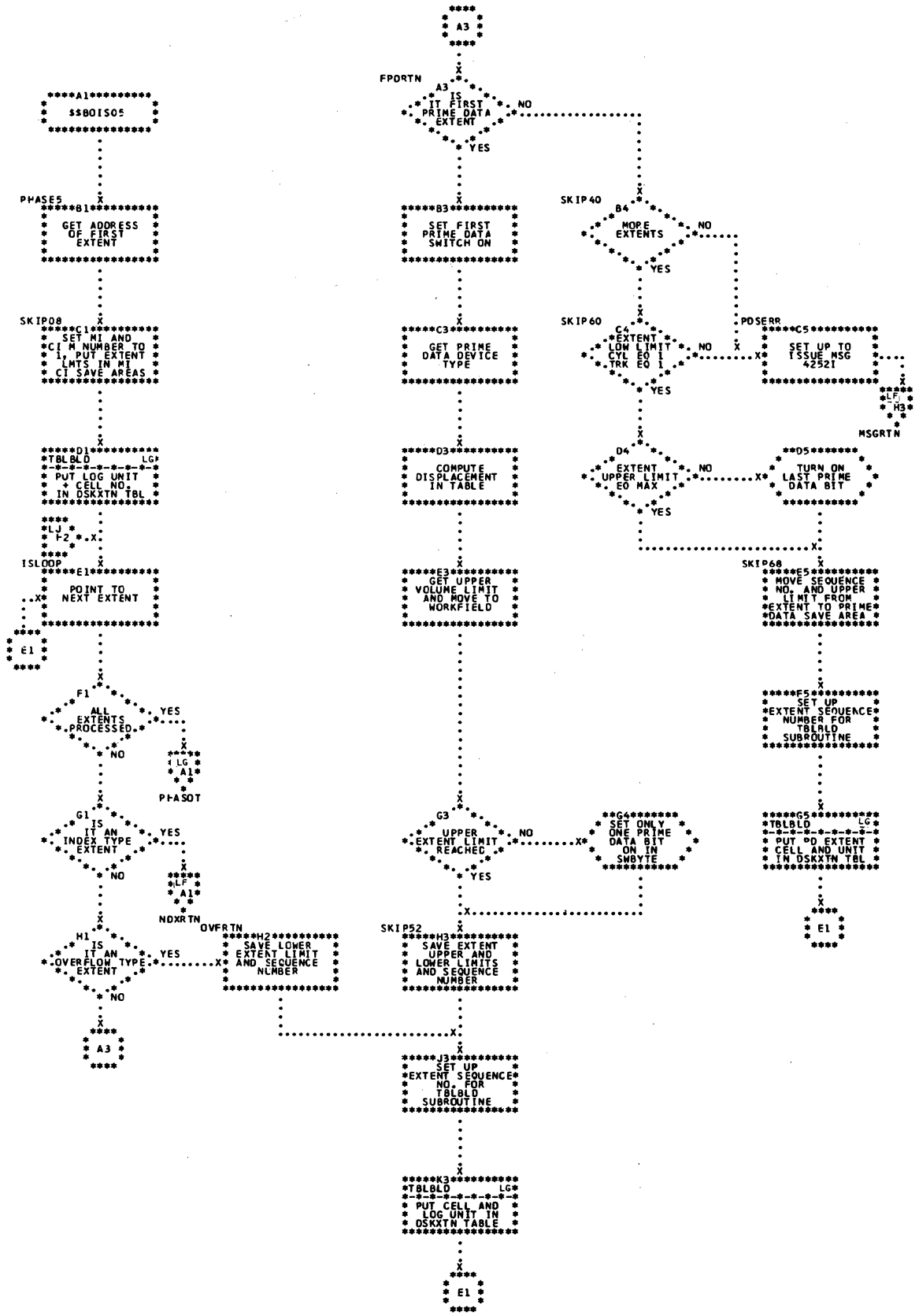


Chart IF. $$BOIS05: ISAM Open, Phase 5 (Part 2 of 3)

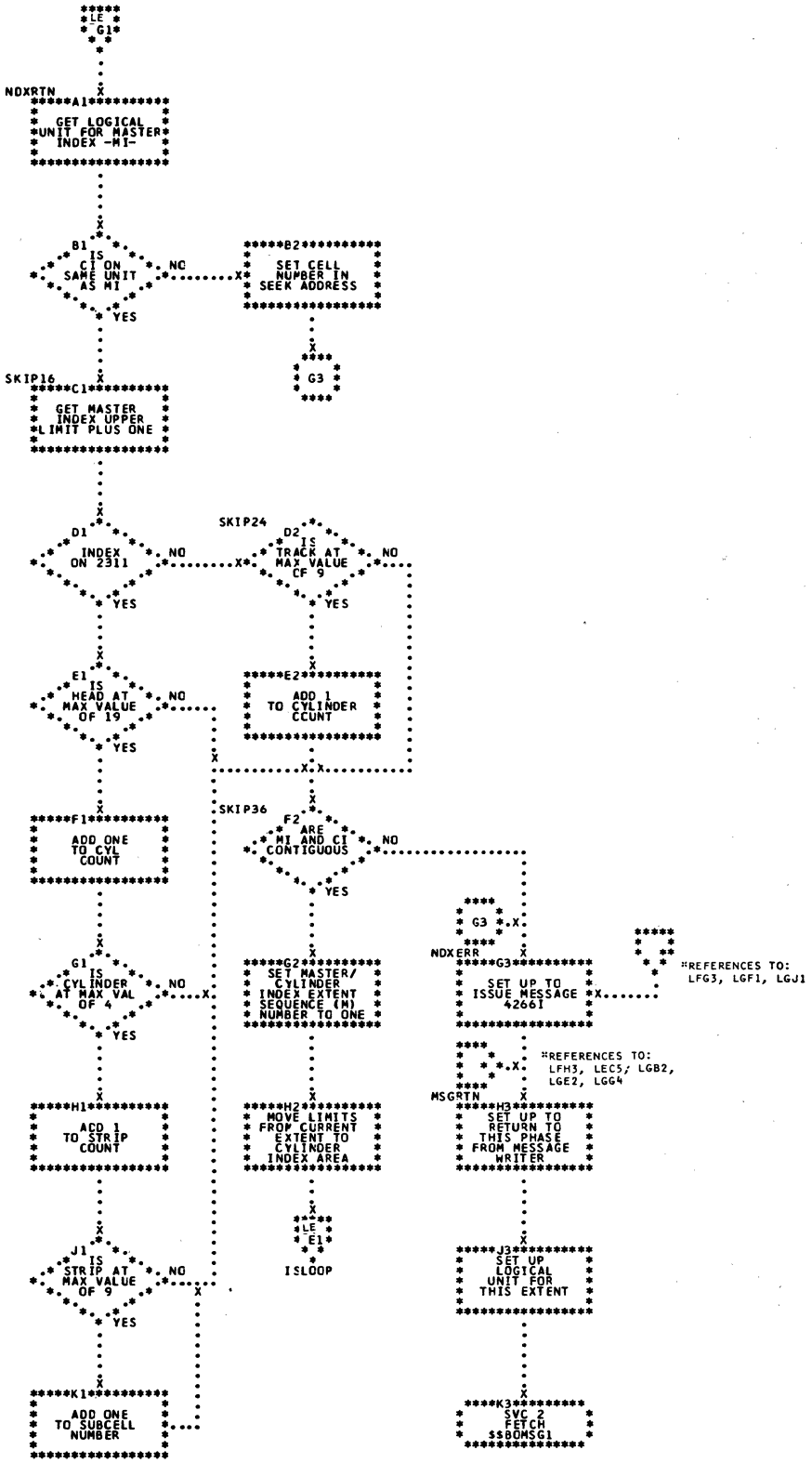


Chart LG. $$BOIS05: ISAM Open, Phase 5 (Part 3 of 3)

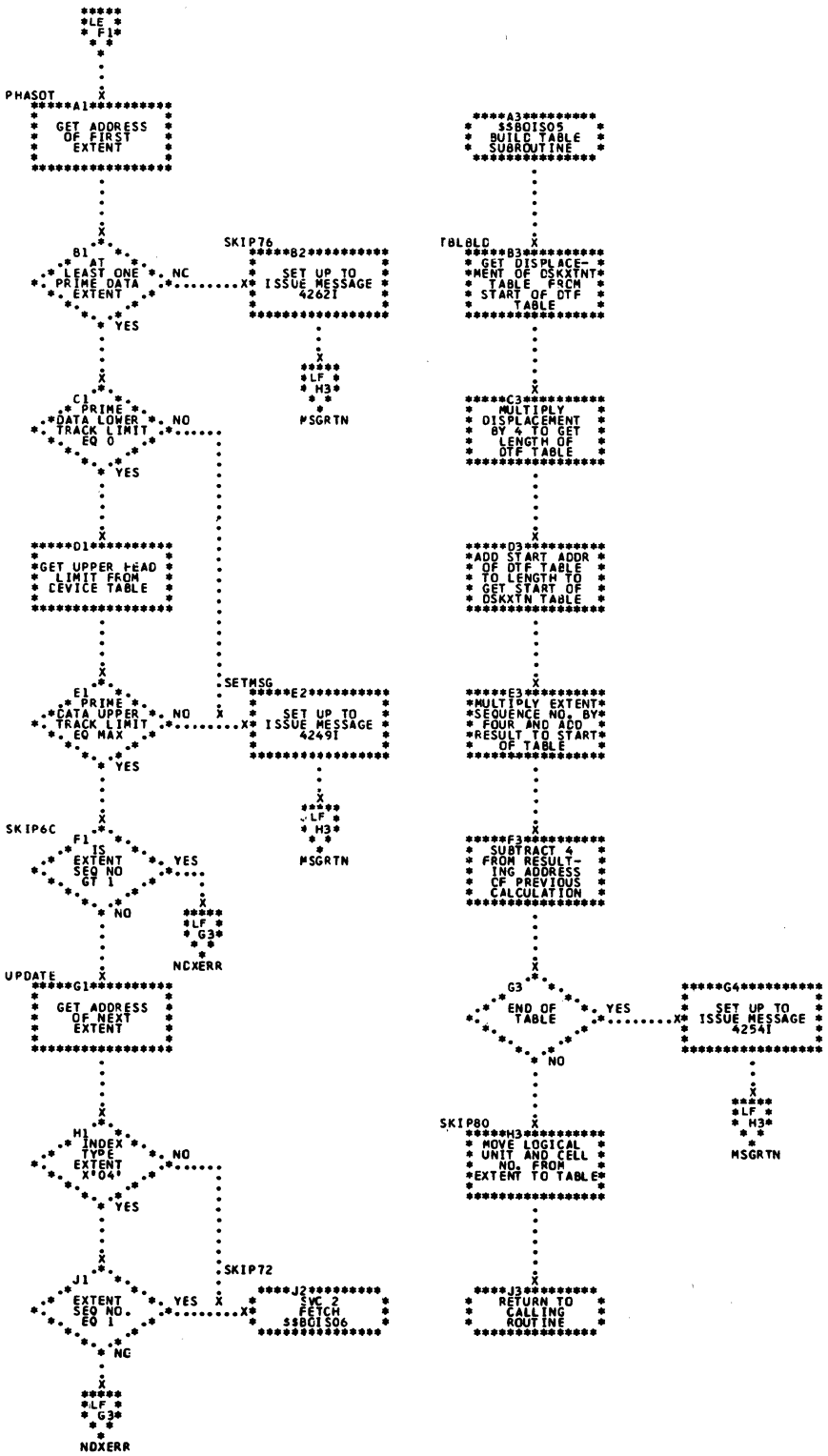


Chart IH. $$BOIS06: ISAM Open, Phase 6 (Part 1 of 2)

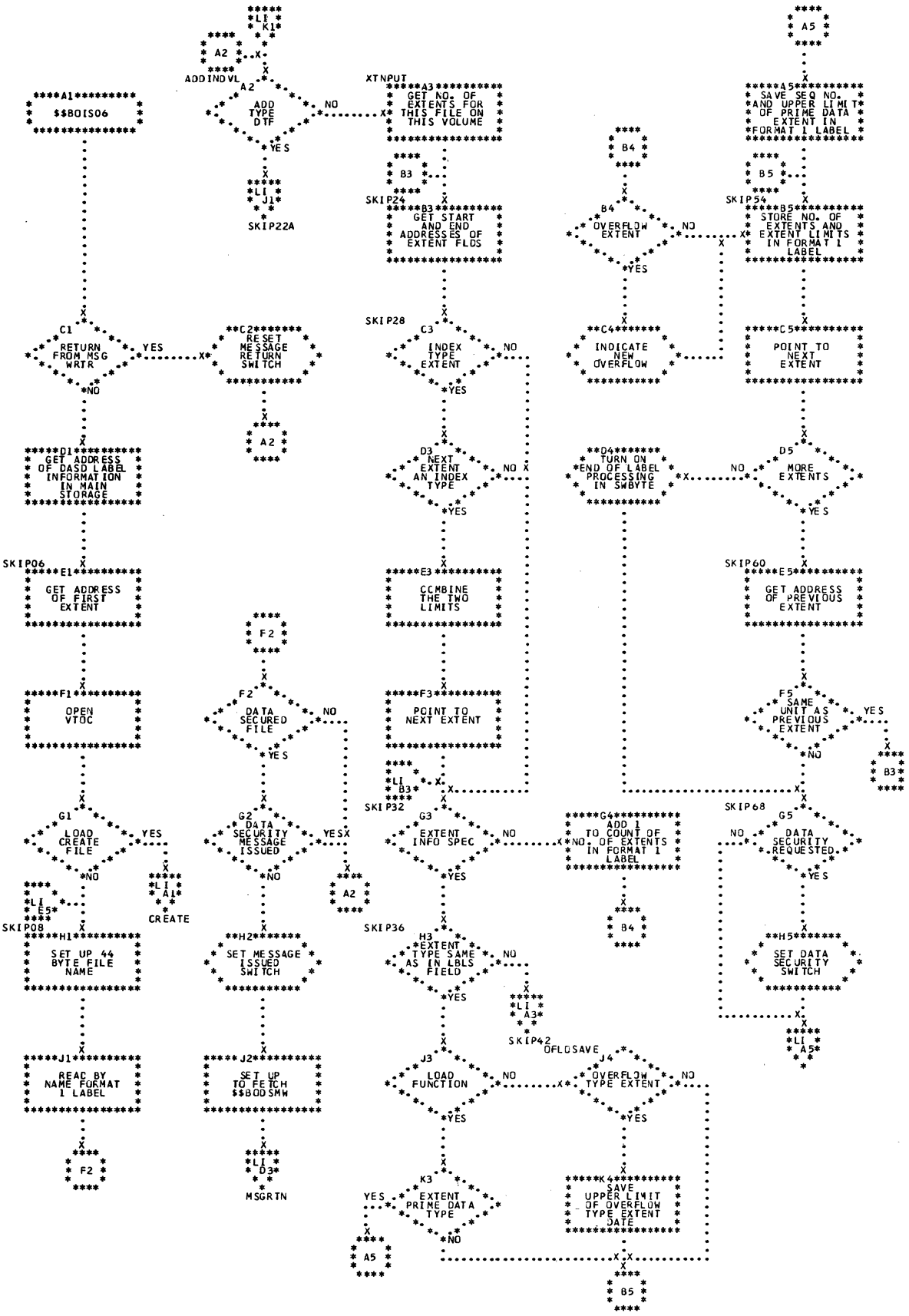


Chart LI. $$BOIS06: ISAM Open, Phase 6 (Part 2 of 2)

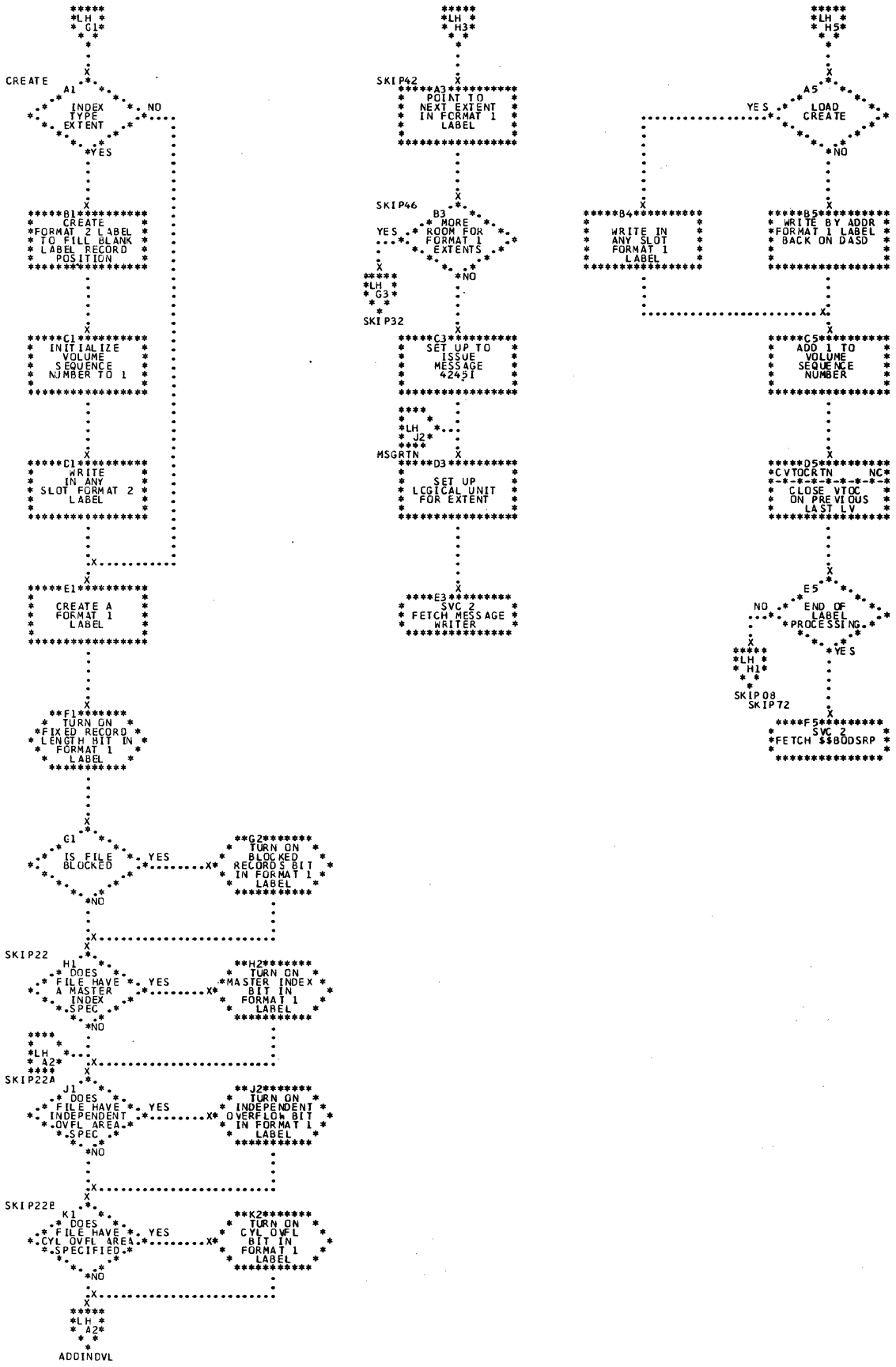


Chart LJ. $$BOISRP: ISAM Open, Phase RPS

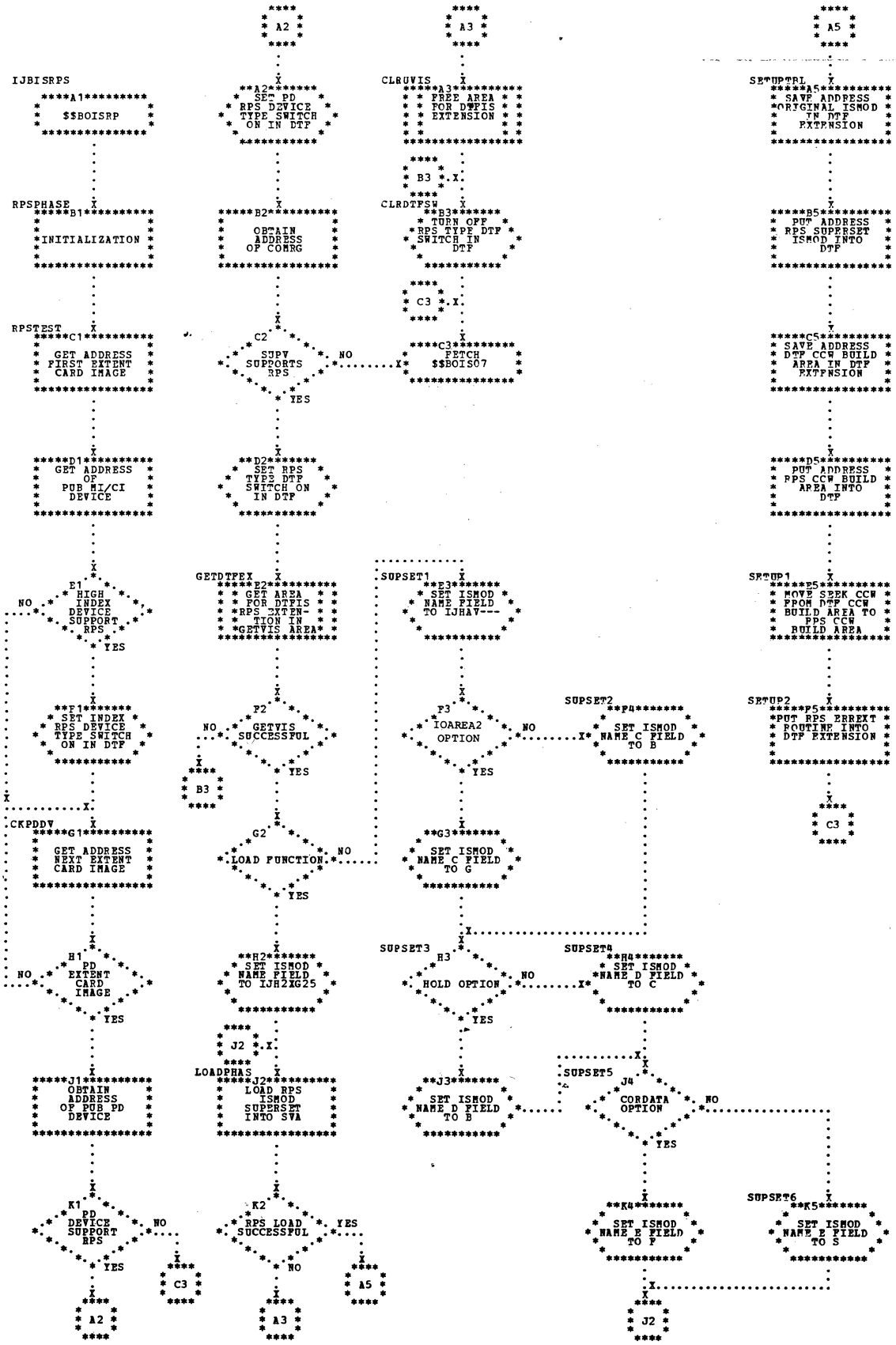


Chart MA. $$BOIS07: ISAM Open, Phase 7 (Part 1 of 4)

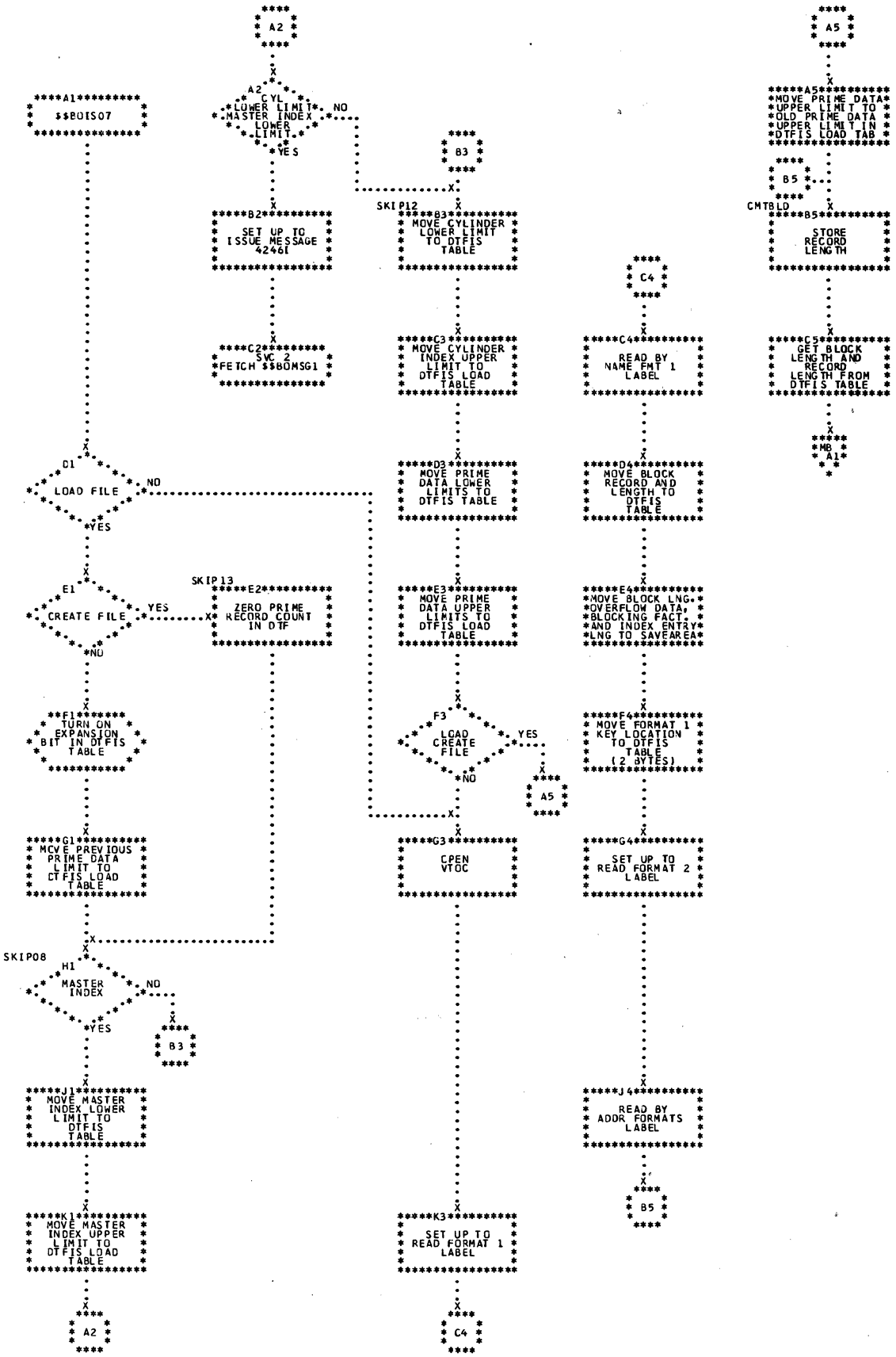


Chart MB. $$BOIS07: ISAM Open, Phase 7 (Part 2 of 4)

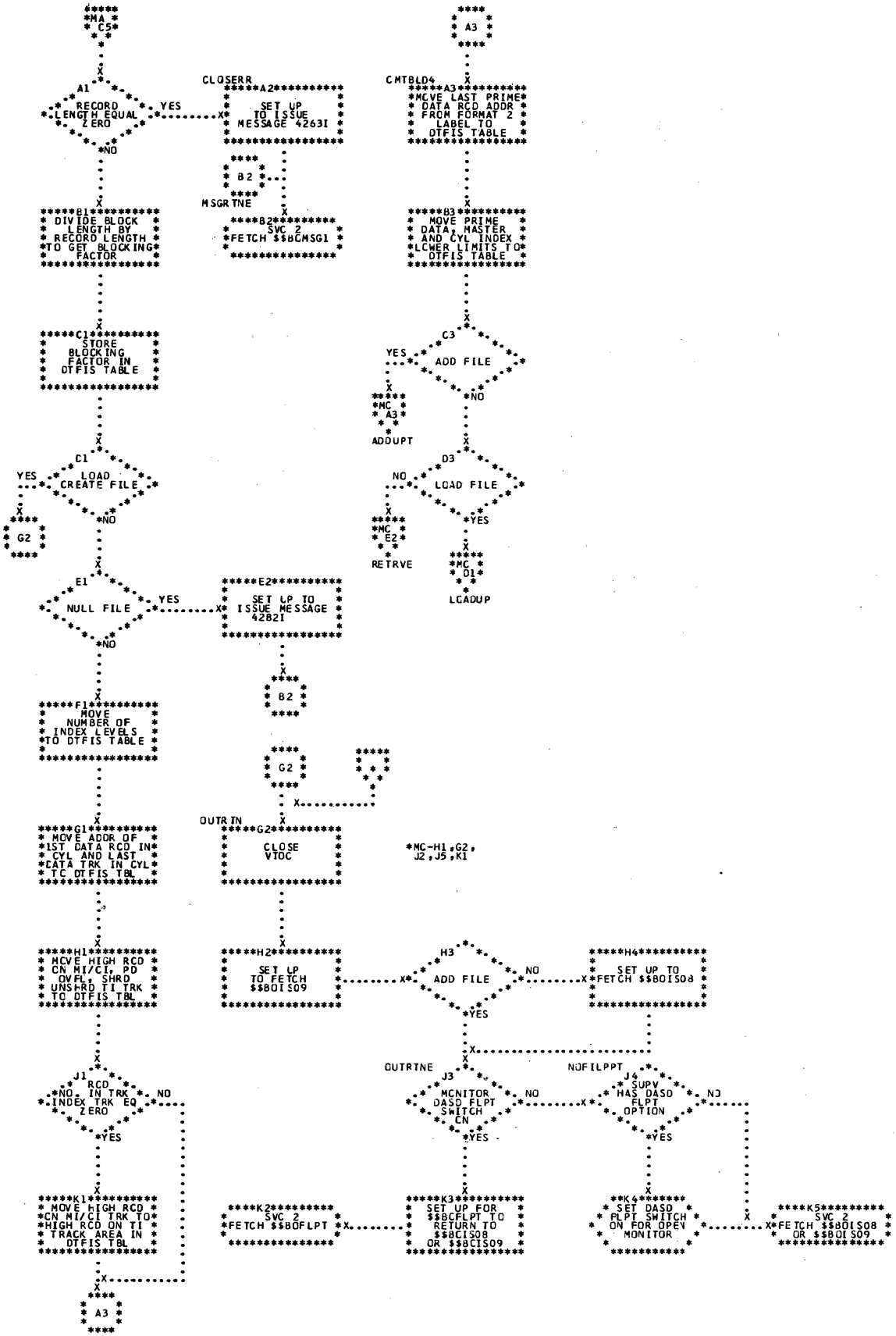


Chart MC. $$BOIS07: ISAM Open, Phase 7 (Part 3 of 4)

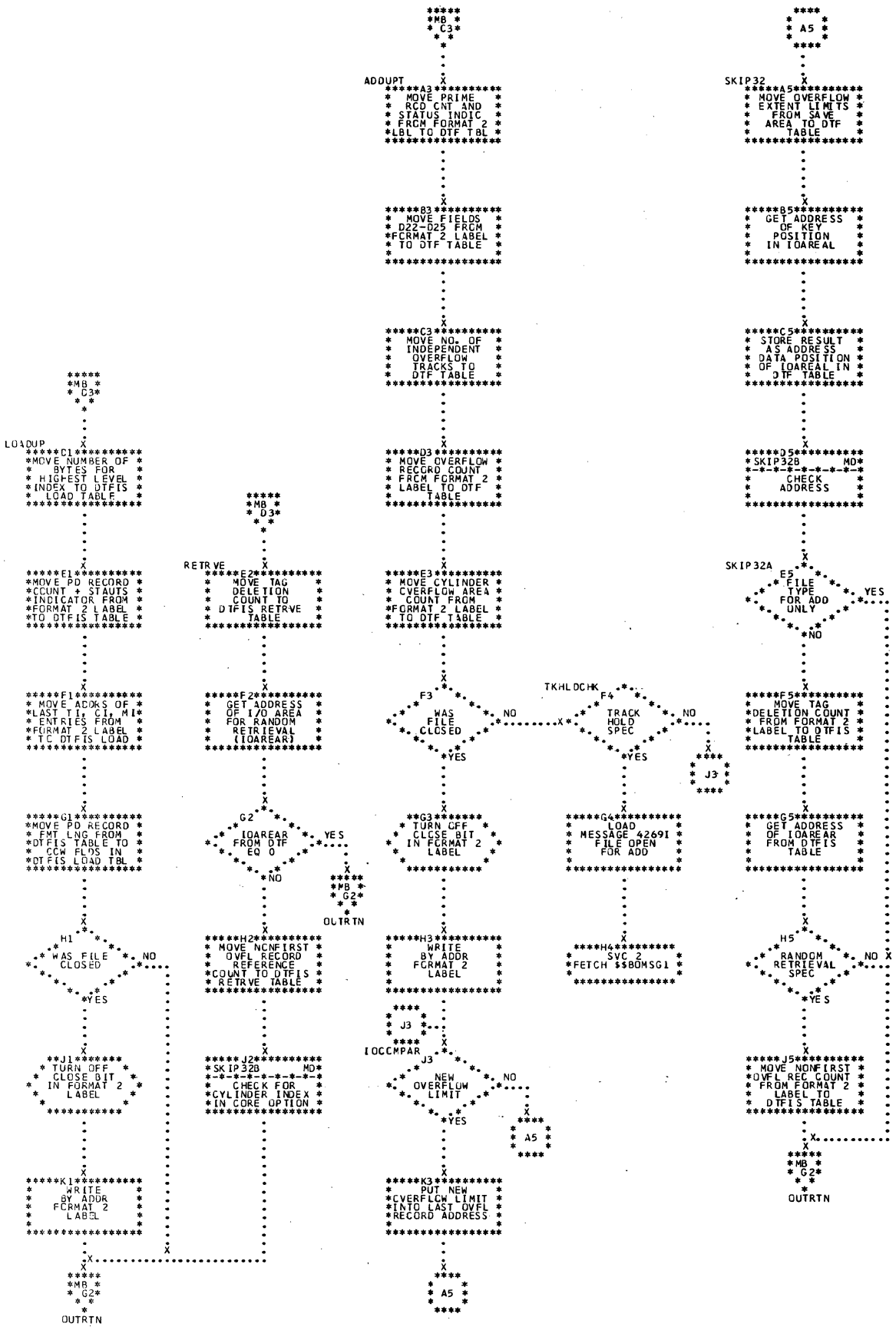


Chart MD. $$BOIS07: ISAM Open, Phase 7 (Part 4 of 4)

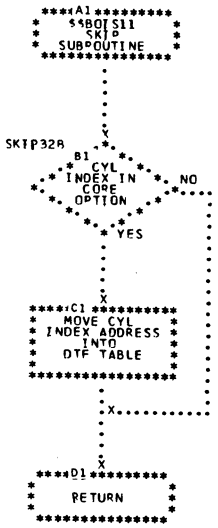


Chart ME. $$BOIS08: ISAM Open, Phase 8 (Part 1 of 2)

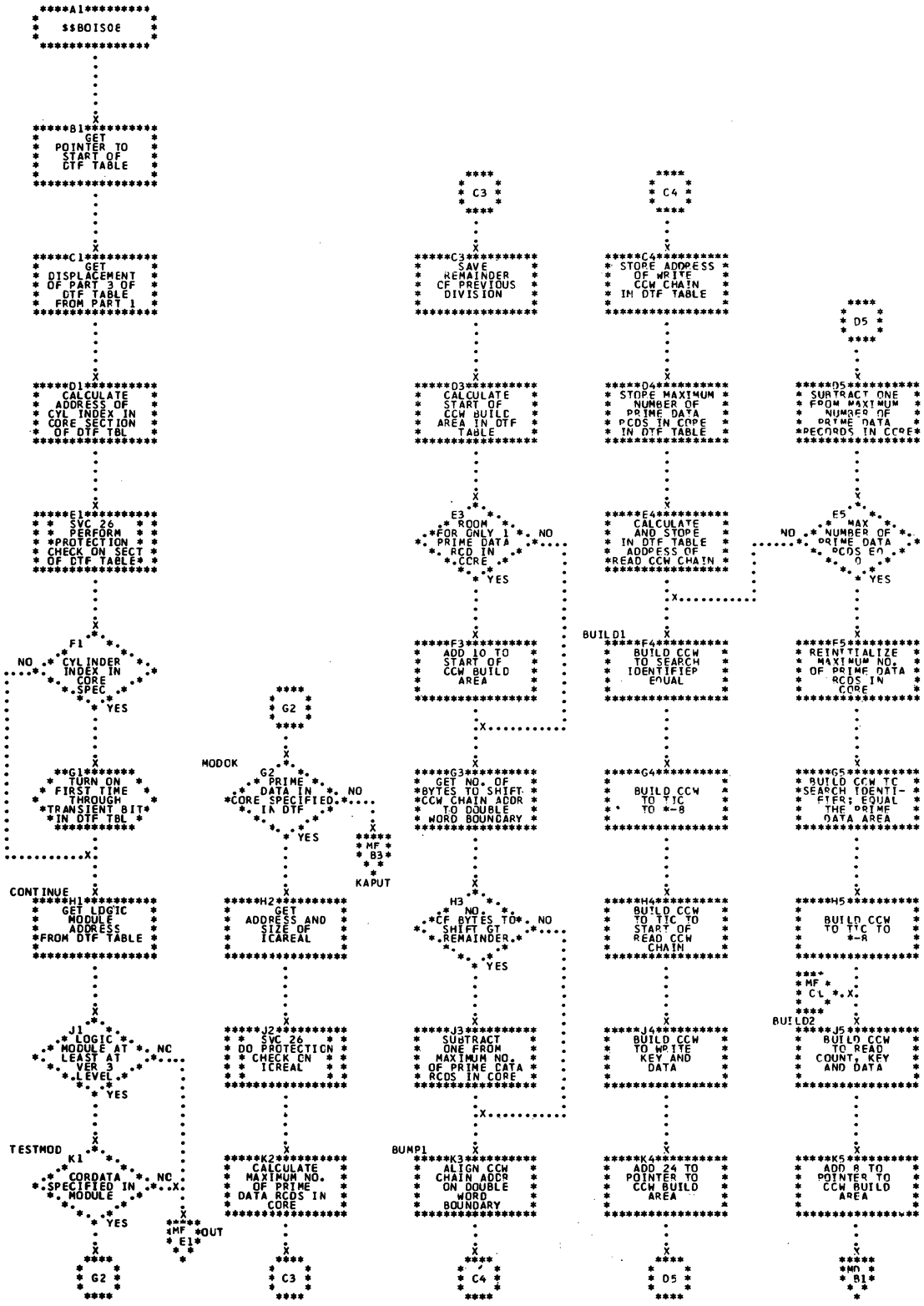


Chart MF. $$BOIS08: ISAM Open, Phase 8 (Part 2 of 2)

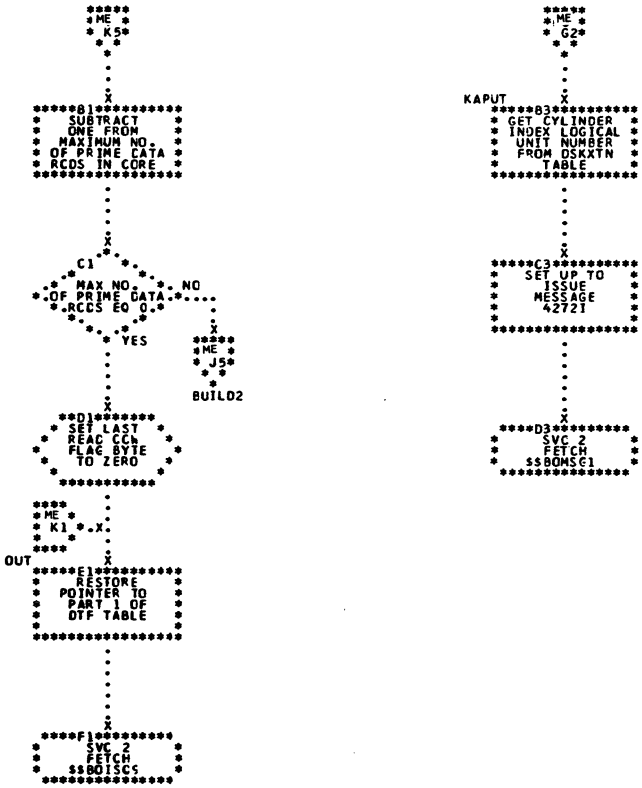


Chart MG. $$BOIS09: ISAM Open, Integrity Phase 1 (Part 1 of 3)

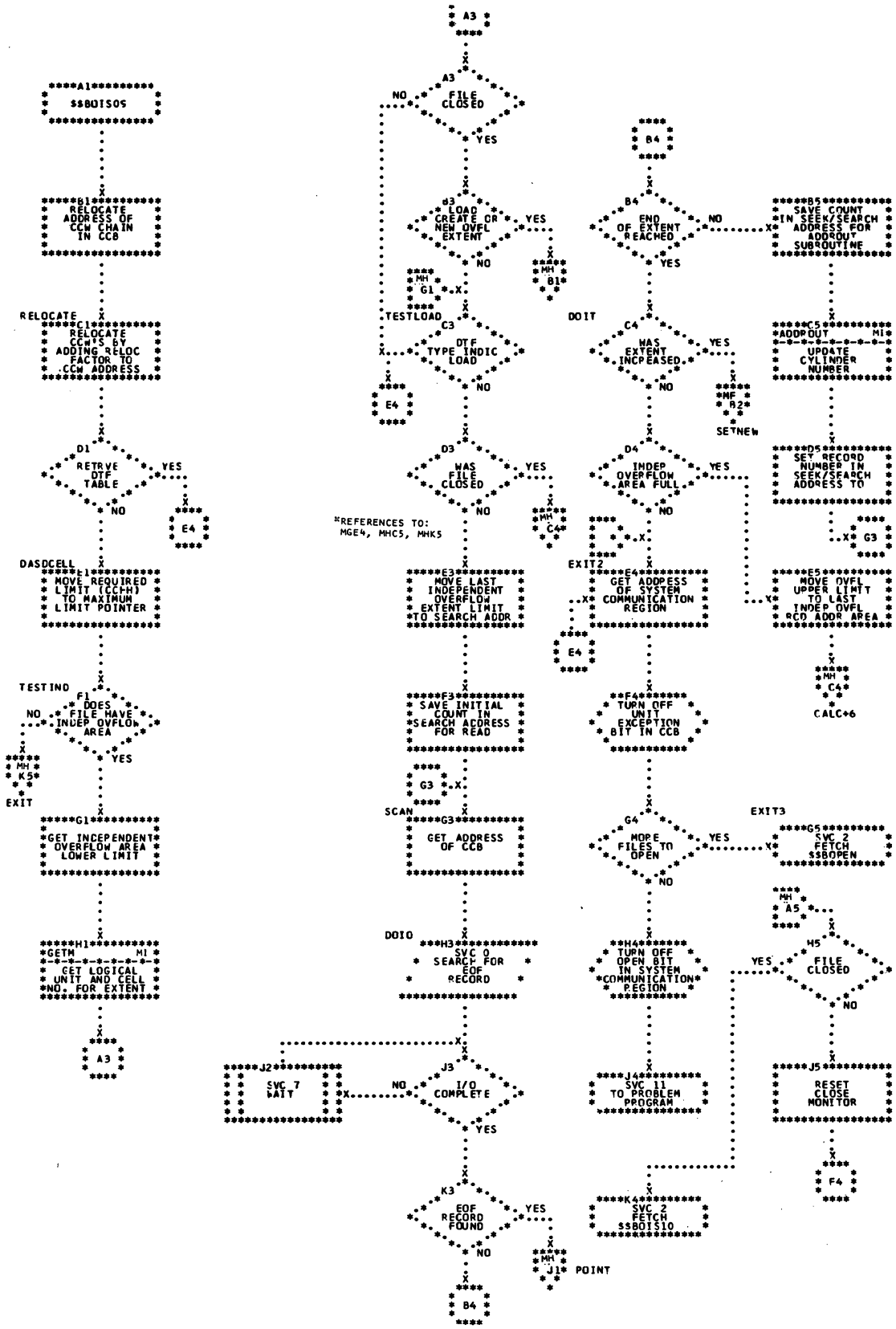


Chart MH. $$BOIS09: ISAM Open, Integrity Phase 1 (Part 2 of 3)

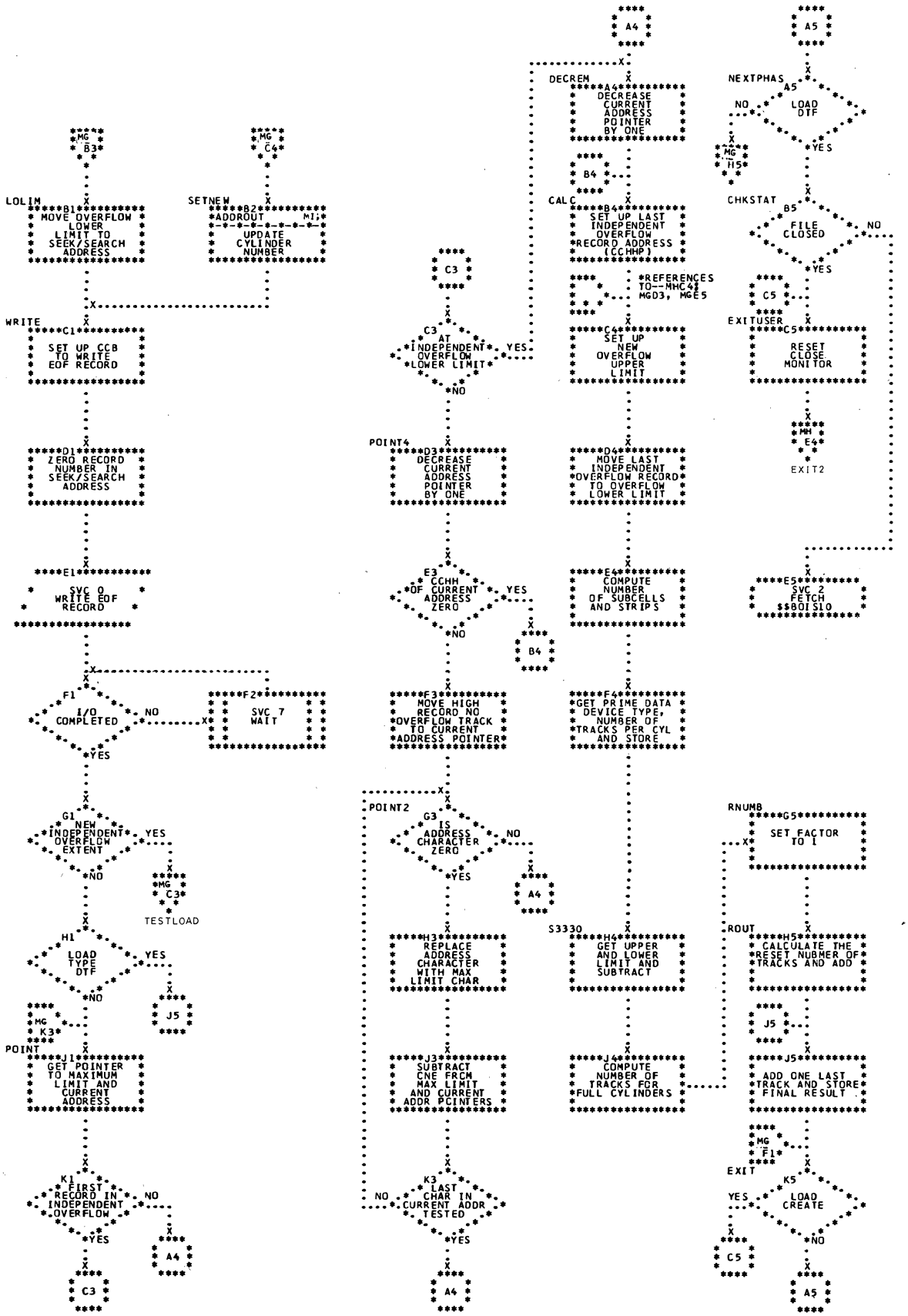


Chart MI. $$BOIS09: ISAM Open, Integrity Phase 1 (3 of 3)

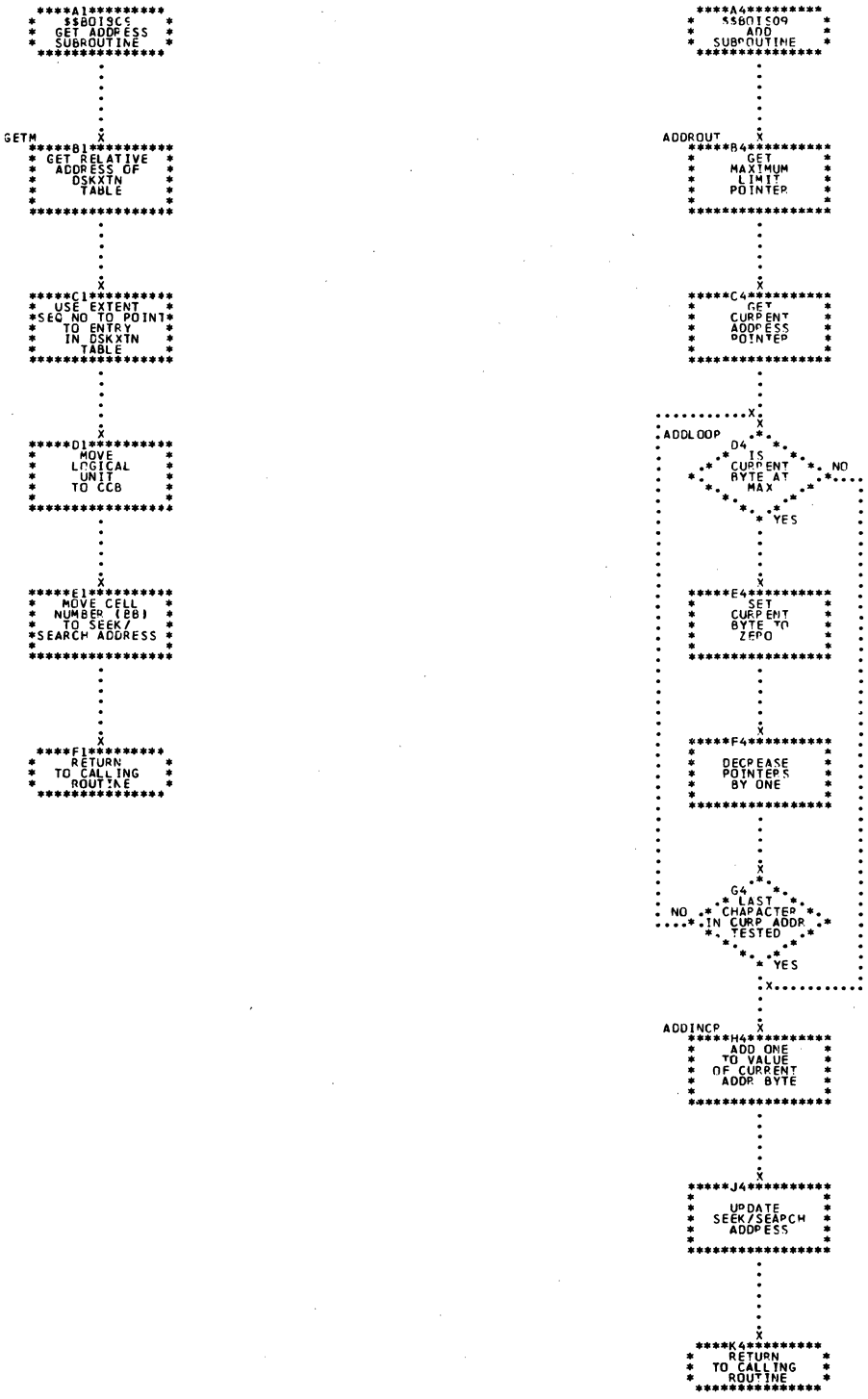


Chart MJ. $$BOIS10: ISAM Open, Integrity Phase 2 (Part 1 of 2)

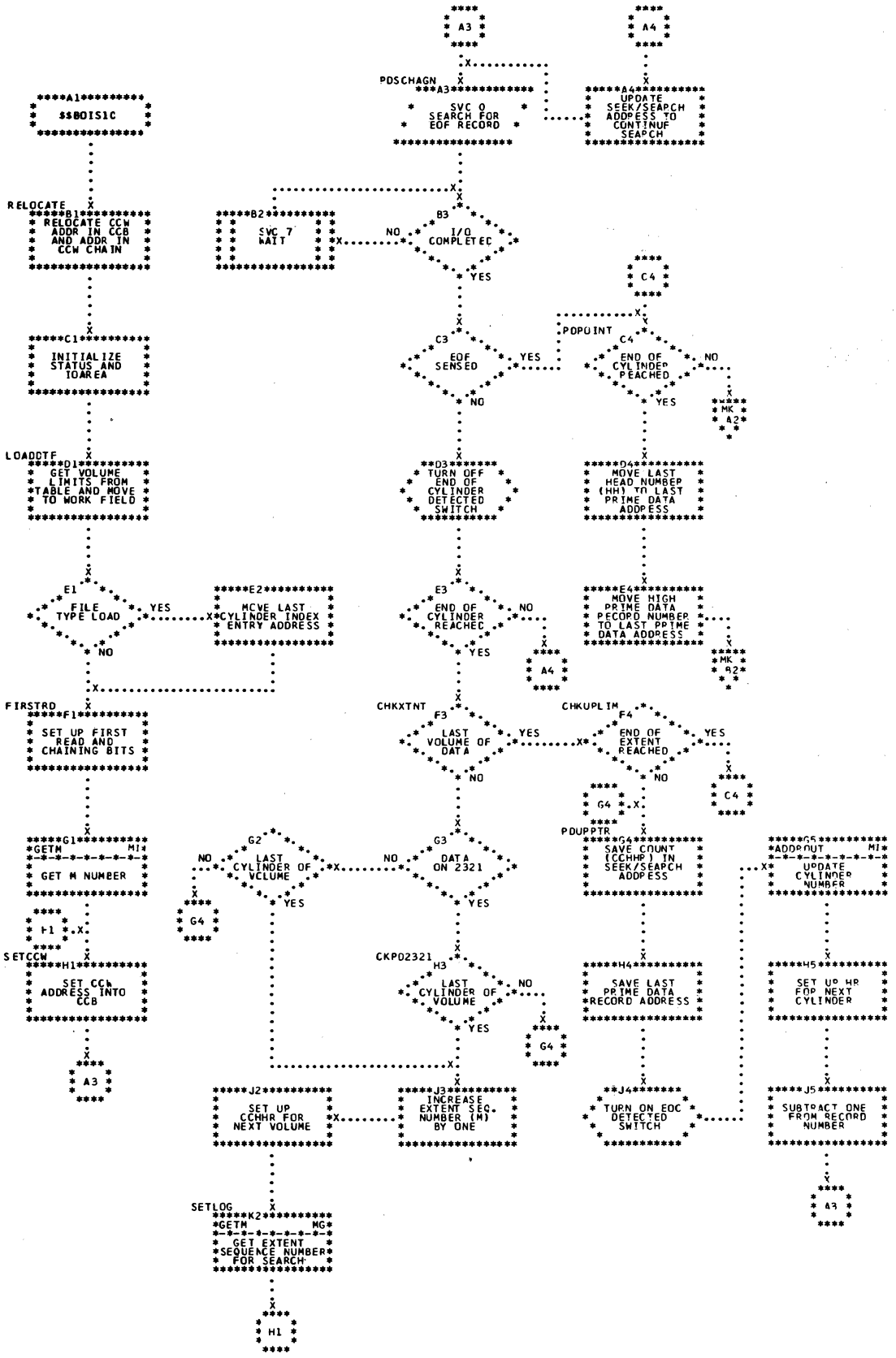


Chart MK. $$BOIS10: ISAM Open, Integrity Phase 2 (Part 2 of 2)

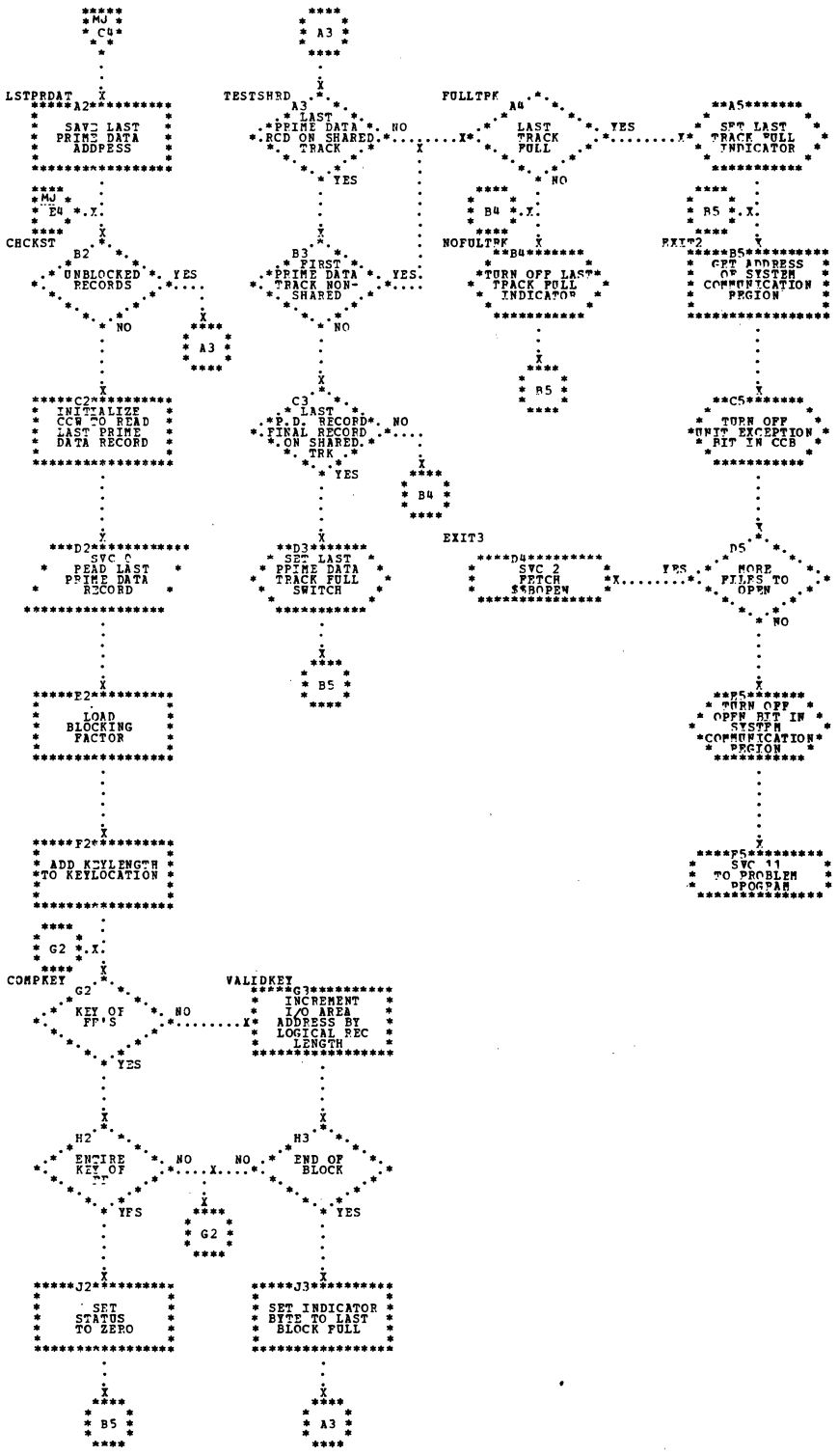


Chart NA. $$BCISOA: ISAM Close (Part 1 of 3)

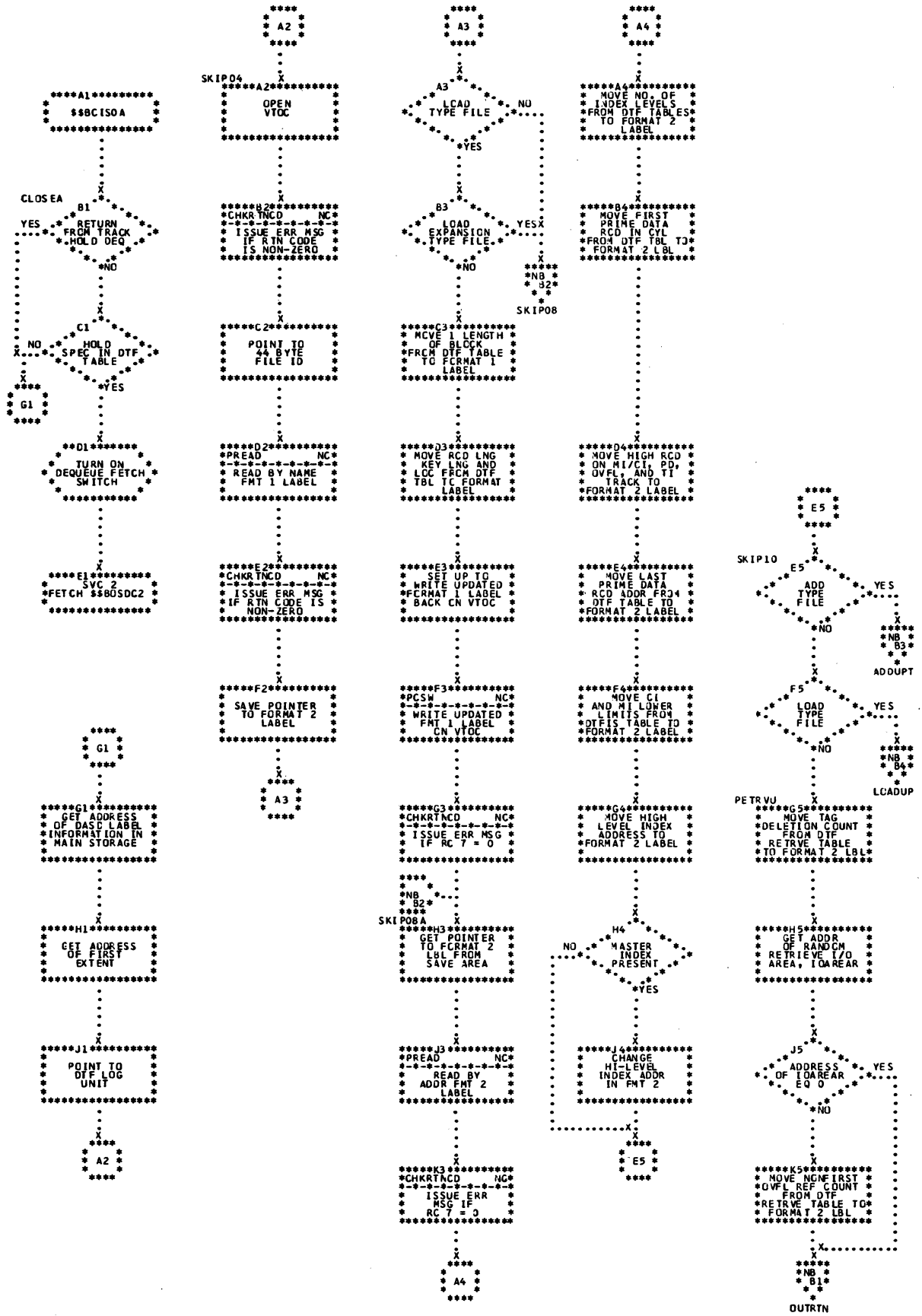


Chart NB. $$BCISOA: ISAM Close (Part 2 of 3)

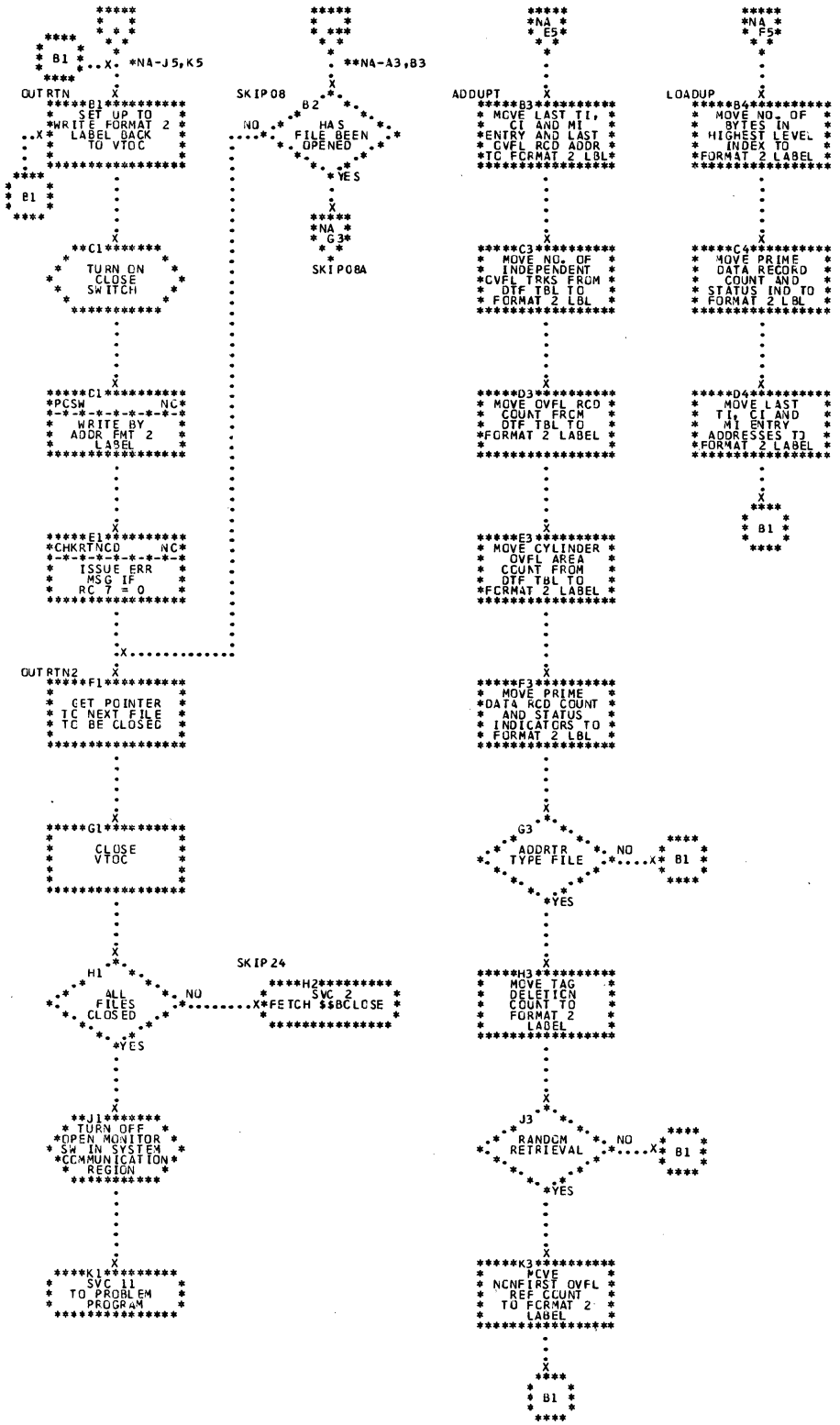


Chart NC. $$BCISOA: ISAM Close (Part 3 of 3)

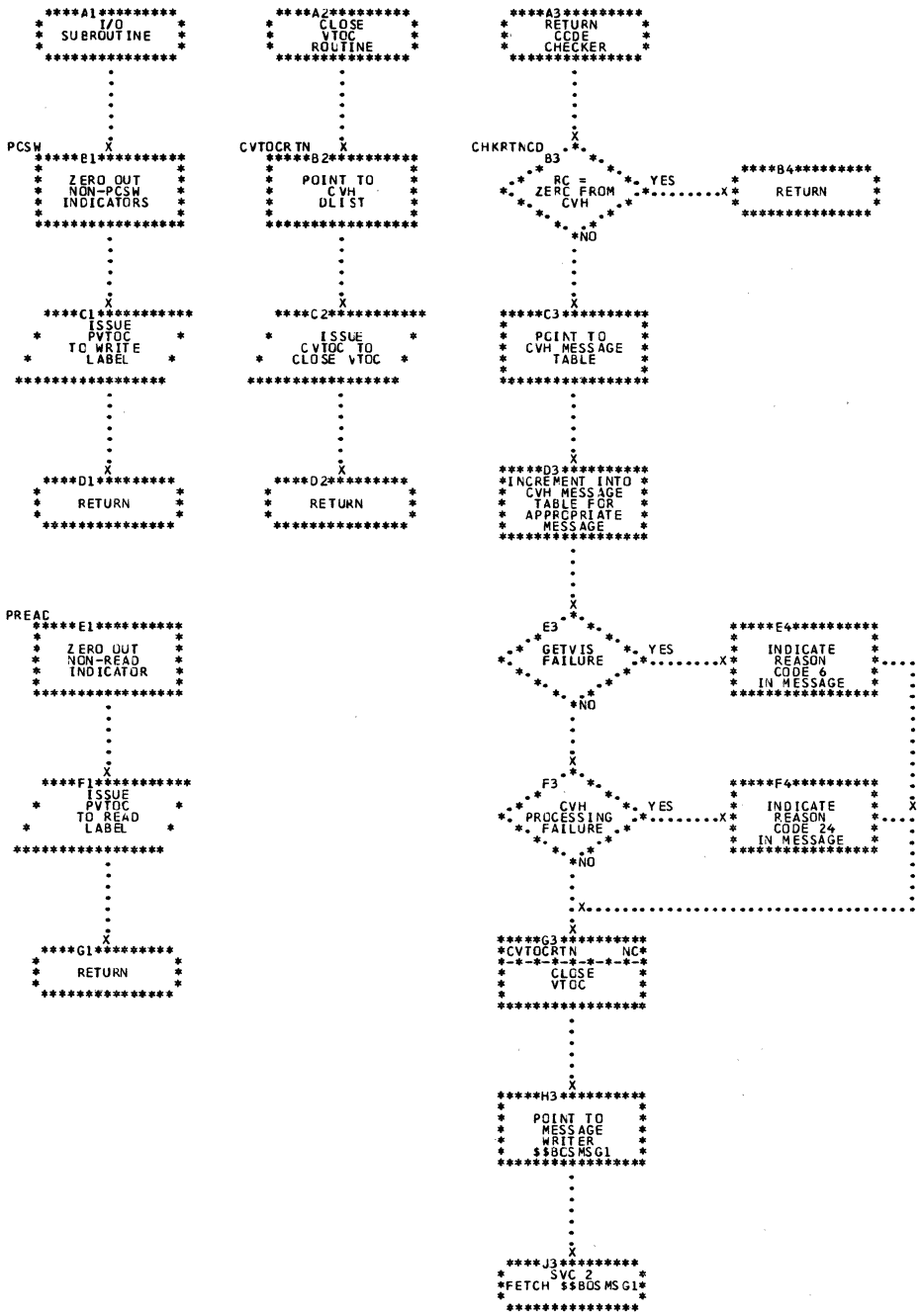


Chart ND. $$BORTV1: ISAM RETRVE Open, Phase 1 (Part 1 of 3)

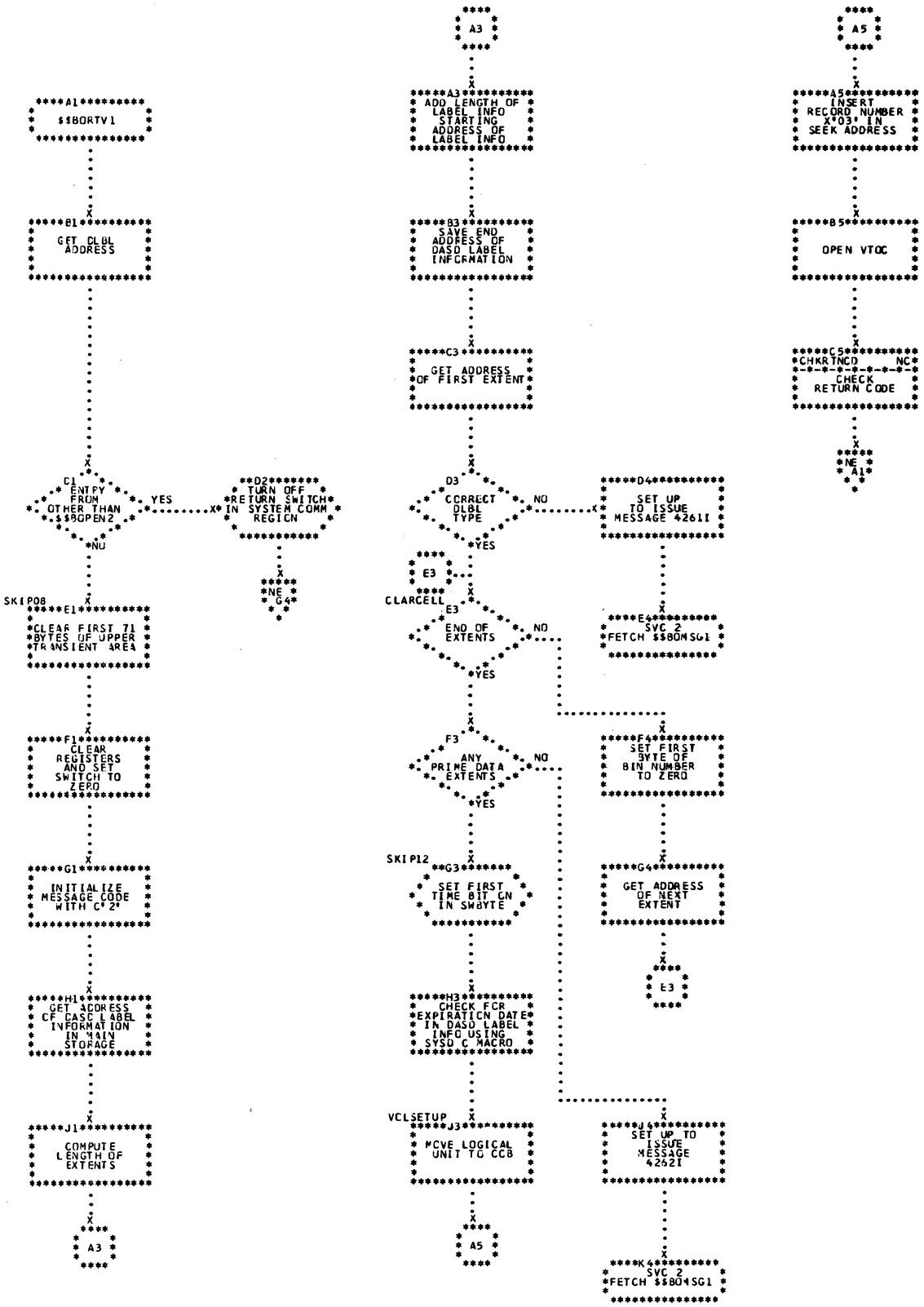


Chart NE. $$BORTV1: ISAM RETRVE Open, Phase 1 (Part 2 of 3)

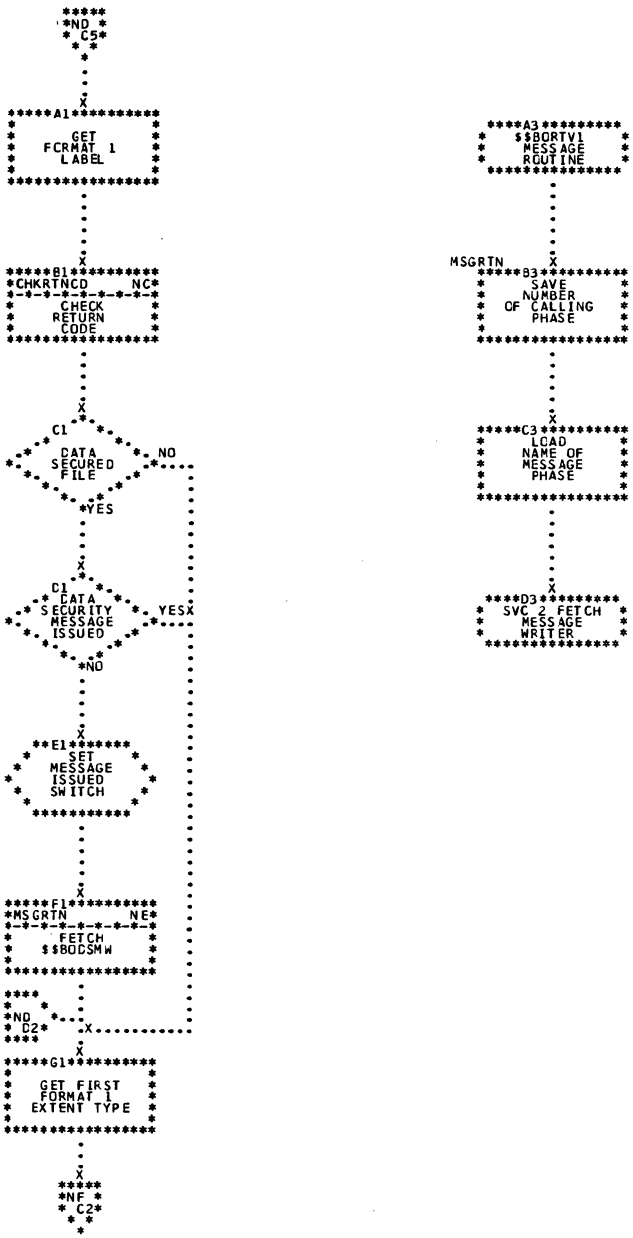


Chart NF. $$BORTV1: ISAM RETRVE Open, Phase 1 (Part 3 of 3)

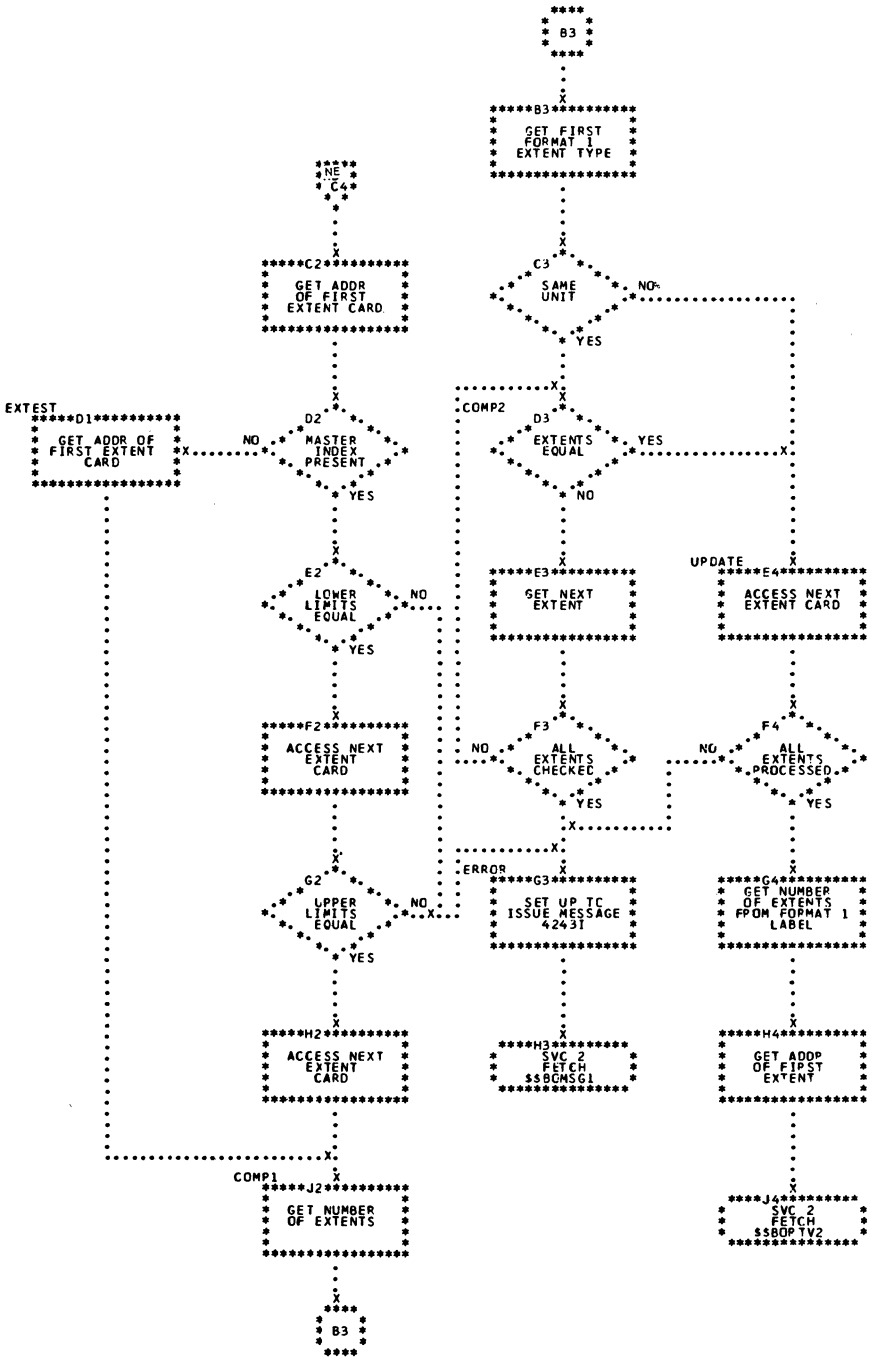


Chart NG. $$BORTV2: ISAM RETRVE Open, Phase 2 (Part 1 of 2)

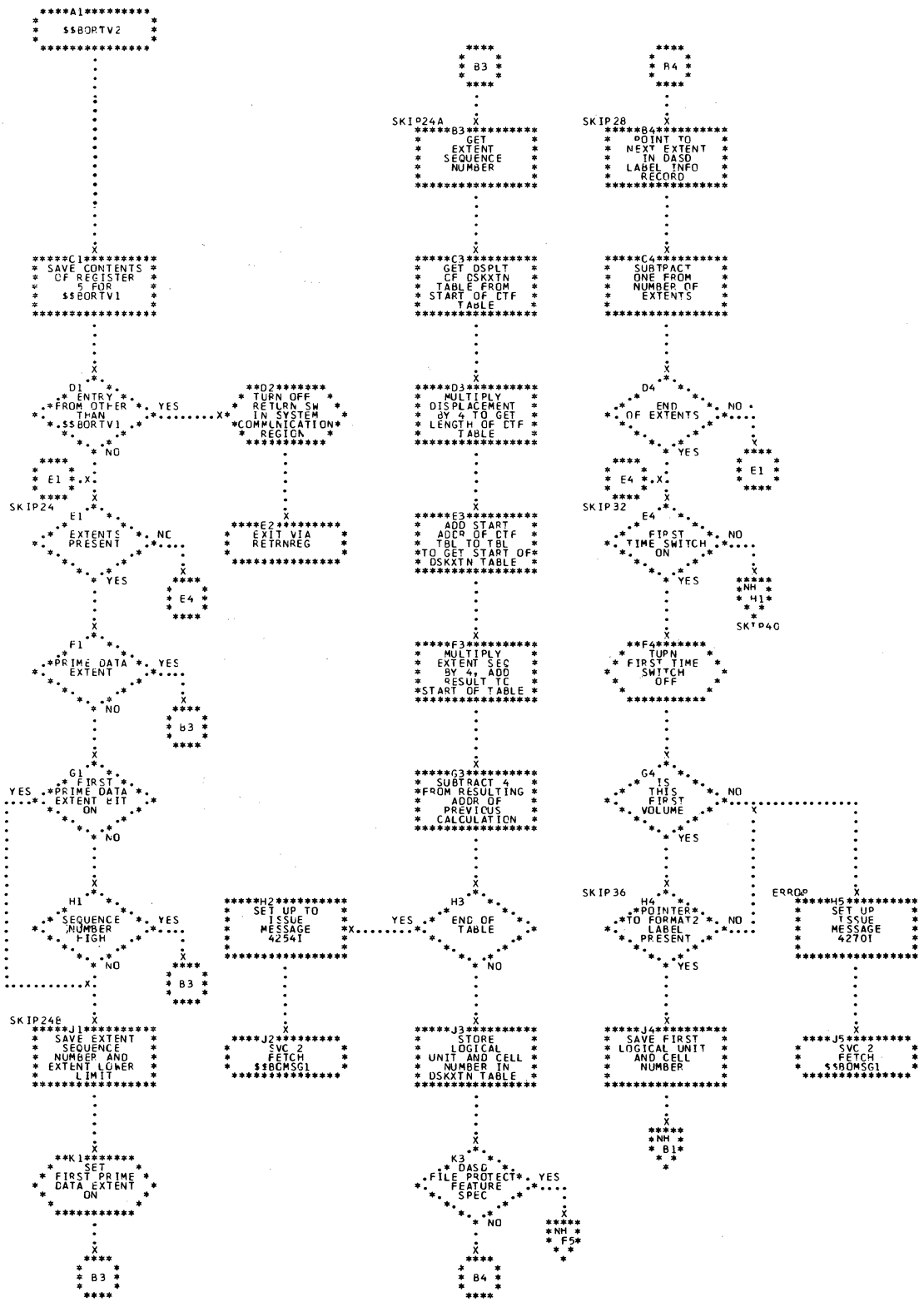
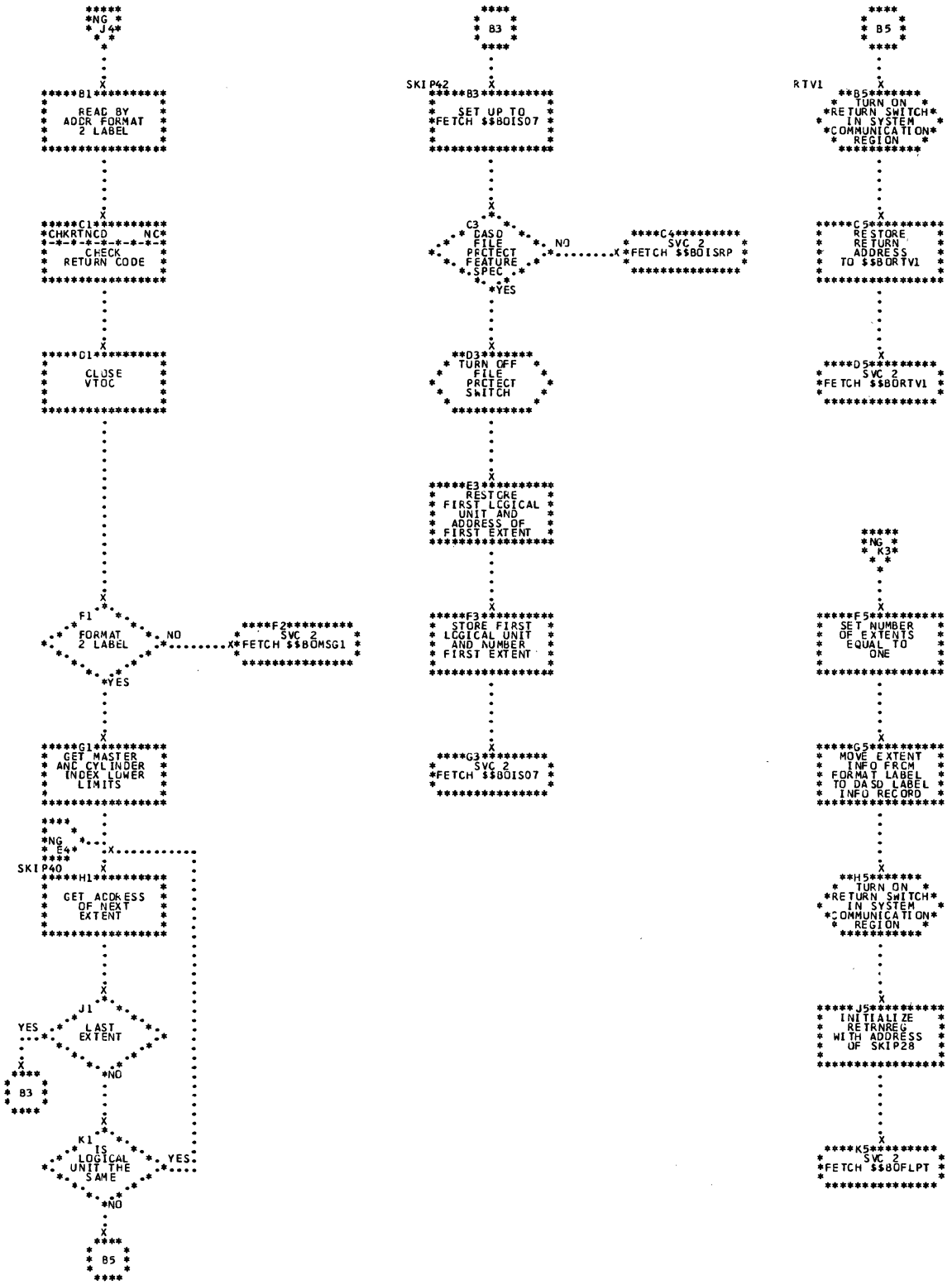


Chart NH. $$BORTV2: ISAM FETRVE Open, Phase 2 (Part 2 of 2)



APPENDIX A: LABEL CROSS-REFERENCE LIST

IJBISRPS	LJ
IJHAASHF	EL
IJHAA12	EE
IJHAA13	EM
IJHAA14	EM
IJHAA16	EM
IJHAA17	EM
IJHAA20	EJ
IJHAA23	EA, KC
IJHAA25	EA, KC
IJHAA26	EE, KC, KD
IJHAA26A	EA, KC
IJHAA27	EE, KC
IJHAA28	EN
IJHAA29	EN
IJHAA33	EI
IJHAA34	EL, KE
IJHAA35	EC, KE
IJHAA36	EC
IJHAA37	EC, KE
IJHAA38	EC, KD
IJHAA39	EC, KE
IJHAA40	EC, KE
IJHAA41	EJ
IJHAA42	EE
IJHAA43	EL
IJHAA44	EI
IJHAA45	EL
IJHABEOF	EH
IJHAB29	EN
IJHACACH	EM
IJHACADX	EJ
IJHACBKI	EL
IJHACBMP	EN
IJHACDBK	EN
IJHACDOF	EM
IJHACDTR	EL
IJHACDUP	EE, KD
IJHACDVP	EJ
IJHACEOF	EM
IJHACFLL	EE
IJHACGTC	EA, KC
IJHACINX	EE
IJHACITF	EM
IJHACITI	EM
IJHACITX	EM
IJHACLOP	EN
IJHACLTI	EJ
IJHACPAD	EN
IJHACPD1	EI
IJHACTET	EA, EB, KC, KD
IJHACUOV	EM
IJHACUPD	EM
IJHACUPP	EJ
IJHAC29	EN
IJHADEC	EC, KC
IJHAD29	EN
IJHAEBEG	EE
IJHAEBLF	EB, KD
IJHAEFXD	EE, KD
IJHAEFXK	ED, KE
IJHAEMCI	KD
IJHAEMC1	EC
IJHAENT	EL
IJHAETFL	ED, KE
IJHAE29	EN
IJHAF29	EN
IJHAG29	EN
IJHAHEAD	ED, KE
IJHAH29	EN
IJHAIOOP	EM
IJHAICPH	EM
IJHAJ29	EM
IJHANDWO	EA
IJHANEDT	EA, KC
IJHANEOF	EH
IJHANOWO	KC
IJHAOBEC	KC
IJHAOBEG	EA
IJHAODN1	EJ
IJHAOLOW	EJ
IJHAOMCO	EJ
IJHAOMTO	EH, EJ
IJHAONGO	EG
IJHAORFL	EG
IJHAOSQU	EH
IJHAPRET	EN
IJHAPRM1	EP
IJHAPRM2	EP
IJHAPRM3	EP
IJHAPRM4	EA, KC
IJHAPRM5	EB, KD
IJHAPRM6	EB, KC
IJHAPRM7	EA, KC
IJHAPRM8	EA, KC
IJHAQUA	EB, KD
IJHATEST	EC, KE
IJHAUNB	EC, KE
IJHAWRKY	EE
IJHAWRNK	EE
IJHAWTNK	EA, KA
IJHCAAB	EE
IJHCBH98	EK, FJ
IJHCBLD1	EK
IJHCBLD2	EJ
IJHCBUMP	EG
IJHCCWS	HB
IJHCER	EE
IJHCERCA	EG
IJHCERRT	EE
IJHCEXCP	EH
IJHCE99	EK, FJ
IJHCHKIN	EE
IJHCHRFP	EG
IJHCINDX	EG
IJHCINIT	EG
IJHCLGUT	EH
IJHCLOPX	EK, FJ
IJHCLOP1	EE
IJHCMCAL	EH
IJHCMOVE	EH, HA
IJHCPLST	EE
IJHCRES	EF
IJHCREST	EH, HA
IJHCRES2	EF
IJHCSTRE	EK, FJ
IJHCTICR	EK, FJ
IJHCTRDX	EG
IJHCTSER	EH, HA
IJHCWAIT	EE
IJHCWATB	EA, FD, KA
IJHCWATF	EA, FD, KA
IJHCXCOP	EE
IJHCXCOR	EG
IJHCXCPH	EG, fh
IJHCXWTH	FE, HA
IJHECCNT	EA, FD, KA
IJHELDR	EF
IJHENDFL	DM
IJHENOTE	EA
IJHERRA	EC, KE
IJHESTLB	GA, JA
IJHEUSER	EF
IJHFREEB	FK

IJIAFX	AN
IJIANT	AM
IJIANT1	AN
IJIANV	AN
IJIANVL	AN
IJIARD	AM
IJIAWR	AN
IJIBLD	AJ
IJICMC	AH
IJICTL	AF
IJICTL1	AF
IJIDIC	AE
IJIDUMY	AB
IJIDV2	AH
IJIDV3	AH
IJIEOF	AA
IJIEOF1	AC
IJIEOV	AE
IJIFLG	AJ
IJIFREE	AF
IJIFRM	AE
IJIFXL	AA
IJIGCH	BD
IJIGET	BD
IJIGFT1	BD
IJIGNXT	BD
IJIGSKB	BD
IJIGSKB1	BD
IJIGSKE	BD
IJIGSKM	BD
IJIGTTT	BD
IJIJCK	AD
IJILADR	AC
IJILBK	AH
IJILOH	AA
IJIMXK	AA
IJINCD	AH
IJINCT	AE
IJINEF	AE
IJINFIT	AM
IJINID	AD
IJINRF	AM
IJINRM	AA
IJINRTO	AB
IJIOVCH	BF
IJIOVP	AG
IJIOV2	AG
IJIPRT	AA
IJIRDY	AJ
IJIREAD	AC, BB
IJIREST	AF
IJIRNO	AA
IJIRON	AA
IJIRTER	AB
IJIRZO	AA
IJISAFIT	AM
IJISAFT	AM
IJISANUL	AK
IJISASRO	AM
IJISBK	AA
IJISCMC	BE
IJISCTL	AF
IJISCTL1	AF
IJISDV3	BF
IJISEOV	BA
IJISERF	AM
IJISFREE	AF
IJISFRM	BA
IJISGDL	AK
IJISGS	AL
IJISICL	BA
IJISING	AL
IJISJCK	BC
IJISKIF	AL
IJISKIP	AL
IJISKIR	AL
IJISKI1	AL
IJISKS	AL
IJISKS1	AL
IJISK1	AL
IJISLBK	BE
IJISMID	BB
IJISNEF	BB
IJISNF	AL
IJISNID1	BB
IJISNL	AK
IJISNRM	AM
IJISNRMU	BD
IJISNRTO	BA
IJISOVP	BE
IJISOV1	BE
IJISOV2	BF
IJISOV2&6	BF
IJISPW	AK
IJISREST	AF
IJISRFD	AK
IJISRON	AK
IJISRTER	BC
IJISSEOF	AK
IJISSF	AK
IJISSTO	AK
IJISSTRT	AK
IJISSWL	AK
IJISTDL1	AL
IJISTDL2	AL
IJISTI	AK
IJISTK	AK
IJISTKW	AL
IJISTNR	BC
IJISTO	AA
IJISTRT	AA
IJISTT	AK
IJISUBPS	AA
IJISUID	BB
IJISVR	AL
IJISVT	AK
IJISVT1	AK
IJISVW	AL
IJISWC	AL
IJISWEOF	AM
IJISWLL	AM
IJISWLR	AM
IJISWND	AF, AM, AN
IJISWRT	AK
IJISWTM	BA
IJISXTFD	BA
IJISYID	BB
IJISZER	AM
IJITIC	AJ
IJITICR	AJ
IJITNR	AD
IJITOM	AD
IJITRHLD	AA
IJITWRU	AA
IJIWAFT	AN
IJIWFTR	BB
IJIWND	AA, AF
IJIWRI	AJ
IJIWTM	AB
IJIWT3	AB
IJIXNF	AH
IJIXTFD	AB
IJIYID	AB
IJIZER	AA

For explanations and actions to be taken for the various messages, refer to VSE/Advanced Functions Messages.

Note: The second digit of the message number indicates the type of DASD file issuing the message. The file types are:

2 = ISAM file

6 = Direct access - Input

7 = Direct access - Output

Message Number	Issuing Phase	Chart	Message
4201I	$$BOIS02	LC	NO FORMAT 1 LABEL FOUND
	$$BOIS0A	NA	or
			NO RECORD FOUND
4701I	$$BODAO2	CD	
	$$BODAU1	CK	
4202I	$$BCIS0A	NA	NO RECORD FOUND
4204I	$$BOIS02	LC	NO FORMAT 4 LABEL FOUND
			or
			NO RECORD FOUND
4206I	$$BOIS02	LC	NO STANDARD VOL1 LABEL FOUND
	$$BCIS0A	NA	or
			NO RECORD FOUND
4608D	$$BODACL	CQ	NO RECORD FOUND
4639D	$$BODACL	CQ	USER TRL LBL IS NOT STD.
4240I	$$BOIS02	LC	EXTENT OVERLAP ON ANOTHER
4241I	$$BOIS02	LC	EXTENT OVERLAP ON VTOC
4243I	$$BOIS02	LC	INV EXTENT HI/LO LIMITS
	$$BORTV1	NF	
4244A	$$BOIS02	LC	OVERLAP ON UNEXPRD FILE
4245I	$$BCIS06	LI	TOO MANY EXTENTS
4246I	$$BOIS07	MA	DISCONT INDEX EXTENTS
4249I	$$BCIS05	LG	DATA TRACK LIMIT INVALID
4252I	$$BOIS05	LE	DISCONT TYPE 1 EXTENTS

Figure 114. Message Cross-Reference List (Part 1 of 2)

Message Number	Issuing Phase	Chart	Message
4254I	$$BOIS05	LG	DSK XTN ENTRY TABLE FULL
	$$BORTV2	NG	
4261I	$$BOIS01	LA	INVALID DLBL FUNCTION
	$$BORTV1	ND	
4262I	$$BOIS05	LG	NO PRIME DATA EXTENT
	$$BORTV1	ND	
4263I	$$BCIS07	MB	LOAD FILE NOT CLOSED
4266I	$$BOIS05	LF	1 TRACK USER LABEL EXTENT
4269I	$$BOIS07	MC	FILE IS OPEN FOR ADD
4270I	$$BORTV2	NG	1ST XTNT CD NOT INDX VOL
4271I	$$BOIS01	LA	EXTENT INFO NEEDED
4272I	$$BOIS08	MF	MOD AND DTF INCOMPATIBLE
4282I	$$BCIS07	MA	ISAM NULL FILE
4297I	$$BOIS02	LC	OVLAP EXPIRED SECRD FILE
4298I	$$BOIS02	LC	OVLAP UNEXPRD SECRD FILE

Figure 114. Message Cross-Reference List (Part 2 of 2)

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

LY24-5211-0

VSE/Advanced Functions: LIOCS Volume 3 (File No. S370/4300-30) Printed in U.S.A. LY24-5211-0

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

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

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

VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 3
DAM and ISAM
LY24-5211-0

READER'S COMMENT FORM

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

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

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

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

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

_____

_____

_____

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

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

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

LY24-5211-0

Reader's Comment Form

Cut or Fold Along Line

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

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE:

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

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

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

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

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

VSE/Advanced Functions: LIOCS Volume 3 (File No. S370/4300-30) Printed in U.S.A. LY24-5211-0

VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 3
DAM and ISAM
LY24-5211-0

READER'S COMMENT FORM

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

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

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

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

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

_____

_____

_____

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

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

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

Reader's Comment Form

Cut or Fold Along Line

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

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE:

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

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

VSE/Advanced Functions: LIOCS Volume 3 (File No. S370/4300-30) Printed in U.S.A. LY24-5211-0

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

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

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