Contains Licensed Material — Property of IBM
LY24-5210-0
File No. S370/4300-30

Program Product

VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 2 SAM

Program Number 5746-XE9

Component Numbers	**5745-SC-DIO**
	5745-SC-DSK
	5745-SC-IOX
	5745-SC-MCR
	5745-SC-OCR
	5745-SC-PTP
	5745-SC-TAP

Release 2

IBM

SUMMARY OF AMENDMENTS

This manual contains information previously published in DOS/VSE LIOCS Volume 2, SAM Logic, SY33-8560, and VSE Advanced Functions Supplement, LD25-0023. Changes reflect support for:

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

The major result of this support is the removal of descriptions pertaining to Sequential Disk (SD) logic modules, and OPEN and CLOSE B-transients. These functions are supplied by modules executing in the SVA and are described in VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 4, SAM for DASD, LY24-5212.

In addition, magnetic tape message writer B-transients are modified to cause A and D messages to be issued from the SVA so that the LTA is not held in the wait state for the responses. Several new B-transients are supplied to transfer control to and from the LTA and SVA for this purpose.

Other miscellaneous changes are included for 3262 printer support, user-specified label information area on a volume other than SYSRES, and APAR corrections.

First Edition (October 1979)

This edition, LY24-5210-0, applies to Release 2 of VSE/Advanced Functions (Program Number 5746-XE9) and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest System/370 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

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

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

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

PREFACE

This manual is the second in a series of four manuals prcviding detailed information about the VSE/Advanced Functions Logical IOCS prcgrams. The four manuals are:

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

Volume 2: SAM, LY24-5210.

Volume 3: DAM and ISAM, LY24-5211.

Volume 4: SAM for DASD, LY24-5212.

This manual is intended for persons involved in program maintenance and for system programmers who are altering the program design. Program logic information is nct necessary for the operation of the programs described.

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

This volume of the VSE/Advanced Functions Logical IOCS Manuals consists of five parts:

1. LIOCS support for Unit Record files.
2. LICCS support for Magnetic Tape files.
3. LIOCS support for Device Independent files.
4. LIOCS support for Diskette files.
5. Charts.

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

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

1. Label list
2. Messages 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 3, DAM and ISAM, LY24-5211.
- VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 4, SAM for DASD, LY24-5212.
- VSE/Advanced Functions Messages, SC33-6098.
- VSE/Advanced Functions Diagnosis Reference: Supervisor, LY33-9091.
- System/370 and 4300 Processors Bibliography, GC20-0001.

CONTENTS

FIGURES

CHARTS

This volume of the VSE/Advanced Functions Logical IOCS manuals contains detailed information on the logical IOCS support of unit record, magnetic tape, diskette, and device independent files for non-DASD devices. For sequential DASD see _VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 4, SAM for DASD_.

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

Information on all the logical IOCS items (modules, DTF tables, imperative macrcs, open and close routines, etc.), required for the particular file types discussed can be fcund in this manual. The only exceptions are certain common and special-purpose routines which cannot be related to any specific file type or which apply to more than one file type; those routines are covered in Volume 1.

The files discussed in this volume are divided into four categories:

- Unit Record files
- Magnetic Tape files
- Device independent files for non-DASD devices
- Diskette files.

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

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

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

The logic supporting each of the imperative macros has been flowcharted from macro language (source statement) listings. In some instances these flowcharts contain decision blocks to illustrate the logic included in the module for certain xxMOD macro parameter options. You should realize that these decisions do _not_ appear in an assembly listing, but rather that a particular _assembly_ listing is the result of these decision being made at the time the lcgic module is generated.

UNIT RECORD FILES

Logical IOCS supports files on the following IBM devices generally categorized as unit record equipment:

- 125(D) Display Operator Console
- 1017 Paper Tape Reader
- 1018 Paper Tape Punch
- 1255/1259 Magnetic Character Reader
- 1270/1275 Optical Reader/Sorter (these devices are not available in the United States)
- 1287 Optical Readers
- 1403 Printer
- 1419 Magnetic Ink Character Readers
- 1442N1 Card Read Punch
- 1442N2 Card Punch
- 1443 Printers
- 2501 Card Reader
- 2520B1 Card Read Punch
- 2520B2/B3 Card Punch
- 2540R Card Reader
- 2540P Card Punch
- 2560 Multifunction Card Machine
- 2596 Card Read Punch
- 2671 Paper Tape Reader
- 3203 Printer
- 3210 or 3215 Console Printer Keyboard
- 3211 Printer
- 3262 Printer
- 3289-4 Printer
- 3504 Card Reader
- 3505 Card Reader
- 3525P Card Punch
- 3525RP Card Punch with read feature
- 3800 Printer
- 3881 Optical Mark Reader
- 3886 Optical Reader
- 5203 Printer
- 5424/5425 Multifunction Card Unit.

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

The files described in this part include:

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

INITIALIZATION AND TERMINATION

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

With the exception of magnetic ink and optical reader files, which are handled separately, all unit record files are opened by the unit record open logical transient phase, $$BOUR01, fetched by the Open Monitor (refer to VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 1). On the other hand, unit record files (except magnetic ink character reader) require no special termination procedures and are closed by the Close Monitor which simply resets the open indicator in the DTF table for the file. Additional Open and Close processing is required for the 3800 printer.

$$BOUR01: Open Unit Record, Charts AA-AB

Objective: To open a unit record file.

Entry: From the Open Monitor.

Exits:

- To $$BOPEN, if the next file to be opened is not a unit record file.
- To the problem program, if no more files are to be opened.
- To $$BOMSG1, to print out the error message:

  4883I INVALID LOGICAL UNIT

- To $$BOMRCE, if the file to be opened is a 3505 or 3525 card read file using RCE or OMR.
- To $$BOPR3 if the file to be opened is for a 3800 printer.

Method: Entry to $$BOUR01 is from the OPEN Monitor, $$BOPEN1, after the monitor has built the Open table in the top of the transient area and validated the limits of the DTF. This is the proper phase for opening DTFs types 00 through 08 as indicated in byte 20 of the DTF.

The SYSIR macro finds the referenced LUB, and message 4883I is issued if it is unassigned. If the LUB specifies that the logical unit is to be ignored, a bit is set in the DTF but no message is given. If the LUB is assigned, this routine opens unit record files for the following DTF types by performing the functions indicated:

1. Reader. The open switch is turned on.
2. Printer. The open switch is turned on, and an I/O register is loaded if there are two I/O areas. If the printer has the universal character set feature, a set mode instruction is issued to allow or suppress data checks in accordance with user's choice. For the 3800, the set mode instruction is not executed in $$BCUR01, but is executed in $$BOPR3 via a SETPRT macro.
3. Punch. The open switch is turned on, and an I/O register is loaded if there are two I/O areas.
4. Paper Tape. The open switch is turned on, and an I/O register is loaded if an output file with two I/O areas is used.

After the DTF is initialized by turning on the open indicator, this phase determines if there is another unit record file to be opened. If not, control is returned to the problem program, via an SVC 11. If the next file is not for a unit record, the Open Monitor is called.

$$BOMRCE: OMR/RCE Format Open Routine, Charts AC-AD

Objective: Build 80-byte records corresponding to the format descriptor cards read from OMR and RCE and set the device mode.

Entry: From $$BOUR01.

Exits:

- To $$BOPEN, if there is another file to be opened.
- To the problem program, if no more files are to be opened.
- To $$BOMSG1, to print out the error message:

  4899I INVALID FORMAT CARD(S)

Method: Entry to $$BOMRCE is from the $$BOUR01 open unit record routine after $$BOUR01 determines that OMR or RCE is specified for a 3505 or 3525. $$BOMRCE clears the format descriptor record buffer and determines which of the modes, OMR or RCE, is specified.

The format descriptor cards are read and their validity is checked. The message 4899I is issued if they are invalid. An 80-byte format descriptor record is built corresponding to the format descriptor card and the device mode is set.

CARD DEVICE FILES

Card Device (CD) files are those files associated with a card reader and/or a card punch, or a 3881 Optical Mark Reader. The IBM devices supported include:

- 1442N1 Read Punch
- 1442N2 Punch
- 2501 Reader
- 2520B1 Read Punch
- 2520B2 Punch
- 2520B3 Punch
- 2540R Reader
- 2540P Punch
- 2560 MFCM
- 2596 Card Read Punch
- 3504 Card Reader
- 3505 Card Reader
- 3525P Card Punch
- 3525RP Card Punch with read feature
- 3881 Optical Mark Reader
- 5424/5425 Multifunction Card Unit.

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

DTFCD Macro

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

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

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

Specification of DEVICE=2501, DEVADDR=SYSRDR or SYSIPT, and IOAREA2 leads to generation of a special version of double-buffer support, which will provide for increased throughput for SYSIPT and SYSRDR files on an IBM 2501 card reader attached to a Model 115-2, Model 125-1, or Model 125-2.

Bytes*	Bits	Contents	Function
0-15 (00-0F)			CCB.
8 (8)	0		1 = 2501 double-CCW support.
16 (10)	0		1 = OMR[1], 0 = omitted.
	1		1 = ERROPT[2], 0 = omitted.
	2		COBOL open; ignore option.
	3		1 = GET issued[3], 0 = GET not issued[7].
	4		DTF table address constants relocated by OPENR.
			<u>File Association</u>:
	5-7		000 = READ only 010 = READ/PRINT[4] 101 = READ/PUNCH/PRINT[5] 001 = READ/PUNCH[5].
17-19 (11-13)			Address of logic module.
20 (14)		X'02' X'05'	DTF type. DTF type for 2560 or 5424/5425.
21 (15)	0		1 = open; 0 = closed.
	1		First time switch.
	2		1 = 1442 or 2596; 0 = other
	3		1 = 2560, 3525, or 5424/5425; 0 = other.
	4		1 = 3504 or 3505; 0 = other.
	5		1 = 2 I/O areas; 0 = 1 I/O area.
	6		1 = 2520; 0 = other.
	7		1 = 2540; 0 = other.
22 (16)		B'SSF0X010'	Normal command code (not for 2560 or 5424/5425). SS: 00 = pocket1, 01 = pocket2, 10 = pocket3[6]. F : 1 = column binary[3], 0 = EBCDIC. X : 1 = OMR or RCE[3], 0 = neither.
		B'H0B00010'	Read command code (2560). H : 0 = hopper1, 1 = hopper2. B : 0 = EBCDIC, 1 = column binary.
		B'HMMM0011'	SS command code (5424/5425). H : 0 = hopper1,1=hopper2. MMM: 001 = stacker1,010=stacker2. 011 = stacker3,100=stacker4.
23 (17)			Control command code (not for 2560 or 5425).
		B'H0B00010'	Read command code (2560).
		B'HMMM0011'	SS command code (5424/5425). H : 0 = hopper1,1=hopper2. MMM: 001 = stacker1,010=stacker2. 011 = stacker3,100=stacker4.
24-27 (18-1B)			Address of IOAREA2. (If IOAREA2 is not specified, address of IOAREA1.)
28 (1C)	0		1 = 2560; 0 = other.
	1		1 = 5424/5425, 0 = other.
	2-7		Not used.

Figure 1. DTFCD: Input (Reader) (Part 1 of 4)

Bytes*	Bits	Contents	Function
29-31 (1D-1F)			Address of EOF routine.
32-39 (20-27)			Read CCW (2560). Stacker select CCW (5424/5425).
Bytes 40-49 as used for all files except 2560 and 5424/5425 files.			
40-43 (28-2B)		LA &IOREG,0(14) NOP 0	Load user pointer register.
44-49 (2C-31)		MVC 0(&BLKSIZE,13),0(14) NOP 0 DC X'0000'	Move IOAREA to WORKA.
The following bytes (50-105) are used for 2501 double-CCW support.			
50-55 (32-37)			Unused CCB.
56-71 (38-47)			
72 (48)	0		1 = OMR[1], 0 = omitted.
	1		1 = ERROPT[2], 0 = omitted.
	2		COBOL open; ignore option.
	3		1 = GET issued[3], 0 = GET not issued[7].
	4		DTF table address constants relocated by OPENR.
			File Association:
	5-7		000 = READ only 010 = READ/PRINT[4] 101 = READ/PUNCH/PRINT[5] 001 = READ/PUNCH[5].
73-75 (49-4B)			Address of logic module.
76 (4C)		X'02' X'05'	DTF type. DTF type for 2560 or 5424/5425.
77 (4D)	0		1 = open; 0 = closed.
	1		First time switch.
	2		1 = 1442 or 2596; 0 = other.
	3		1 = 2560, 3525, or 5424/5425; 0 = other.
	4		1 = 3504 or 3505; 0 = other.
	5		1 = 2 I/O areas; 0 = 1 I/O area.
	6		1 = 2520; 0 = other.
	7		1 = 2540; 0 = other.
78		B'SSF0X010'	Normal command code (not for 2560 or 5424/5425).
(4E)			SS: 00 = pocket1, 01 = pocket2, 10 = pocket3[6].
			F : 1 = column binary[3], 0 = EBCDIC.
			X : 1 = OMR or RCE[3], 0 = neither.
		B'H0B00010'	Read command code (2560, 5424/5425).
			H : 0 = hopper1, 1 = hopper2.
			B : 0 = EBCDIC, 1 = column binary.

Figure 1. DTFCD: Input (Reader) (Part 2 of 4)

Bytes*	Bits	Contents	Function
79 (4F)			Control command code (not for 2560 or 5424/5425).
		B'H0B00010'	Read command code (2560, 5424/5425).
80-83 (50-53)			Address of IOAREA2. (If IOAREA2 is not specified, address or IOAREA1.)
84 (54)	0		1 = 2560; 0 = other.
	1		1 = 5424/5425; 0 = other.
	2-7		Not used.
85-87 (55-57)			Address of EOF routine.
88-95 (58-5F)			Read CCW.
96-99 (60-63)		LA &IOREG,0(14) NOP 0	Load user pointer register.
100-103 (64-67)		MVC 0(&BIKSIZE,13),0(14) NOP 0	Move IOAREA to WORKA.
104-105 (68-69)		DC X'0000'	
The following bytes (50-57) are used for 3504, 3505, and 3525 associated files.			
50-53 (32-35)		DC A(name) B 16(15) B 20(15) DC F'0'	If ERROPT=name[2]. If ERROPT=SKIP. If ERROPT=IGNORE. If ERROPT=omitted.
54-57 (36-39)		DC A(ASOCFLE)	Address of associated DTF table[7]. (3525 only).
Bytes 40 onward as used for 2560 and 5424/5425 files.			
40-47 (28-2F)			Stacker select CCW (2560). Read CCW (5424/5425).
48-51 (30-33)		LA &IOREG,0(14) NOP 0	
52-57 (34-39)		MVC 0(&BLKSIZE,13),0(14) NOP 0 DC X'0000'	Move IOAREA to WORKA.
58-63 (3A-3F)		CLC 0(L,14),64(1)	Test for end of file. I=4 if MODE=E; L=2 in other cases.
64-67 (40-43)		DC C'/* ' DC X'0C001022'	End-of-file indicator if MODE=E. In other cases.
68-71 (44-47)		DC A(name) B 16(15) B 20(15) DC F'0'	If ERROPT=name[2]. If ERROPT=SKIP. If ERROPT=IGNORE. If ERROPT=omitted.

Figure 1. DTFCD: Input (Reader) (Part 3 of 4)

Bytes*	Bits	Contents	Function
The following bytes are added for 2560 or 5424/5425 associated files.			
72-75 (48-4B)		DC A(ASOCFLE)	Address of associated DTF table[7].
76-81 (4C-51)		MVC 0(&BLKSIZE,14),82(1)	Move card image to IOAREA1.
82 (52)		DC &BLKSIZE.C' '	Buffer for card image.

[1] CMR only for 3504 and 3505.
[2] ERROPT for 2560, 3504, 3505, 3525, or 5424/5425 READ file.
[3] 3504, 3505, and 3525 with or without CONTROL=YES specified.
[4] 2560, 3525, or 5424/5425 with or without CONTROL=YES specified.
[5] 2560, 3525, or 5424/5425 without CONTROL=YES specified.
[6] Defaults to pocket2 for 3504, 3505, and 3525.
[7] Present only when 2560, 3525, or 5424/5425 associated files are specified for the input DTF.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 1. DTFCD: Input (Reader) (Part 4 of 4)

Bytes*	Bits	Contents	Function
0-15 (00-0F)			CCB.
16 (10)	0		Not used.
	1		1 = ERROPT[3]; 0=omitted.
	2		COBOL open; ignore option.
	3		1 = PUT issued[2]; 0 = PUT not issued.
	4		DTF table address constants relocated by OPENR.
			File Association:
	5-7		000 = PUNCH only 011 = PUNCH/PRINT[3] 001 = READ/PUNCH[3] 101 = READ/PUNCH/PRINT[3] 100 = PUNCH/INTERPRET[3]
17-19 (11-13)			Address of logic module.
20 (14)		X'04'	DTF type.
21 (15)	0		1 = open; 0 = closed.
	1		First time switch.
	2		1 = CTLCHR.
	3		1 = fixed unblocked.
	4		1 = variable unblocked.
	5		1 = 2 I/O areas.
	6		1 = work area.
	7		1 = 2 CCWs in table.
	7		0 = 1 CCW in table.
22 (16)		B'SSF00001'	Normal command code. SS: 00 = pocket1; 01 = pocket2; 10 = pocket3[4]. F : 1 = column binary : 0 = EBCDIC.
		B'HSSS0011'	Normal stacker select command code (2560 or 5424/5425). H : 0 = hopper1, 1 = hopper2. SSS: stacker information.
23 (17)			Control command code (not for 2560 or 5424/5425).
		B'HSSS0011'	Actual stacker select command code (2560 or 5424/5425).
24-27 (18-1B)		DC A(IOAREA1+x)	Address of data in IOAREA1.
28-31 (1C-1F)			Bucket.[1]

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

Figure 2. DTFCD: Output (Punch) (Part 1 of 3)

Bytes*	Bits	Contents	Function
32-33 (20-21)		LR 12,(RECSIZE) NOPR 0	Undefined records only.
34-37 (22-25)		LA &IOREG,4(14) NOP 0	Load user pointer register.
38 (26)	0-2		Not used.
	3		1 = 5424/5425.
	4		1 = 2560.
	5		1 = 3525.
	6		1 = 1442 or 2596.
	7		1 = 2520B1.
39 (27)		DC C' '	Blank for eject last card.
For all files except 2560 and 5424/5425 files.			
40-47 (28-2F)			Punch CCW.
48-55 (30-37)			Eject CCW for last card if 2520.
For 2540 files if CRDERR is specified.			
48-55 (30-37)			Retry CCW.
56-135 (38-87)		DC CL80' '	Save area card image.
For 3525 Punch/Interpret files.			
48-55 (30-37)			Load CCW.
56-63 (38-3F)			Print CCW.
64-127 (40-7F)		DC 64C' '	Print buffer.
For 3525 Associated files.			
48-51 (30-33)		DC A(ASOCFLE)	Pointer to associated file.

[1] The bucket bytes handle undefined records.
[2] Valid for 3525 READ/PUNCH, PUNCH/PRINT, and READ/PUNCH/PRINT files.
[3] Valid for 3525 only.
[4] Defaults to pccket2 for 3525.

Figure 2. DTFCD: Output (Punch) (Part 2 of 3)

Bytes*	Bits	Contents	Function
For 2560 and 5424/5425 files.			
40-47 (28-2F)		DC D'0'	Eject CCW. If FUNC=RP or RPW.
48-55 (30-37)			Stacker select CCW.
56-63 (38-3F)			Punch and Feed CCW.
For 2560 Punch/Interpret files.			
64-71 (40-47)			Load print head buffer one CCW.
72-79 (48-4F)			Load print head buffer two CCW.
80-87 (50-57)			Print CCW.
88-151 (58-97)		64C' '	Save area for printing line 2.
For 5424/5425 Punch/Interpret files.			
64-71 (40-47)			Print CCW.
For 2560 and 5424/5425 Associated files.			
64-67 (40-43)		DC A(ASOCFLE)	
68 (44)		DC C' ' DC X'00'	If mode is EBCDIC. If mode is Column Binary.
69- (45-)		DC &BLKSIZE.C' '	Buffer for card image.

[1] The bucket bytes handle undefined length records.
[2] Valid for 2560 or 3525 READ/PUNCH, PUNCH/PRINT, and READ/PUNCH/PRINT files.
[3] Valid for 2560 or 3525 only.
[4] Defaults to pocket2 for 3525.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 2. DTFCD: Output (Punch) (Part 3 of 3)

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

Figure 3. DTFCD: Combined (Reader/Punch) File

CDMCD Macro

The CDMCD macro provides the model statements needed to generate logic modules that contain the GET, PUT, and CNTRL logic for a card reader or punch. It includes the capacity to handle a combined file, which uses the Punch Feed Read (PFR) feature on the IBM 2540. The combined file can also be handled on the IBM 1442-N1.

Over 200 different logical modules can be generated frcm the CDMOD macro. Because it would be impractical to flowchart and describe every possible variation, the CDMOD macro internals are flowcharted to indicate the many parameter variations and combinations that can be handled.

The CDMOD macro can generate logic modules needed to handle all the normal record forms applicable to a card reader/ punch. A particular CD logic module is designed to handle only those items that are indicated by the CDMOD macro parameters. For instance, if TYPEFLE=INPUT and RECFORM=FIXUNB are specified, the generated module does not handle output files, or variable and undefined records. The logic module can include the use of one or two I/O areas, the use of a work area to process the records, and the logic to handle the CNTRI macro if stacker selection is desired.

A separate lcgic module is needed for every reader/punch record format variation.

CDMOD: CNTRI Macro, Chart AE

Objective: Tc stacker select a card into a specified stacker pocket.

Entry: From a CNTRL macro expansion.

Exit: To the prcblem program.

Method: The control code specified by the user passes tc the CNTRL routine in general register 0. The CNTFL routine then modifies the control code as required for the file type and device specified for the file. The modified control code is stored in the command code byte in the DTF table.

If the file is an input file defined for a device other than a 1442 or 2596, the contrcl operation is performed immediately. In all other cases, the control operation is delayed until the next GET or PUT macro is issued.

CDMOD: GET Macro, Charts AF-AH

Objective: To read a card (normal or combined or asscciated file).

Entry: From a GET macro expansion.

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

Method: For regular input files when two I/O areas are used, I/O and processing overlap is possible. (Dual I/O areas are not valid for associated files.) On the first GET operation, one I/O area is filled and another SVC 0 is issued before any processing is allowed. After the second SVC 0 is issued, control returns to the problem program so that processing of data in the first I/O area can begin. If two I/O areas are specified, the first GET macro instruction sets a switch, which changes the logic flow of all the following GET instructions used (in the problem program) with this logic module.

For combined files, IOAREA2 serves as the output area if it is specified; and IOAREA1 is the input area. If IOAREA2 is not specified, IOAREA1 serves for both input and output.

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

For 2560 or 5424/5425 associated files, the feed cycle is always connected with the read cycle (read and feed).

With a 2560 or 5424/5425 read associated file, some overlapping can be achieved. If no card is to be punched, the I/O or work area contains blanks. Reading of the next card is initiated by:

- PUT for the associated punch file (if FUNC=RP).
- PUT for the associated print file (if FUNC=RPW/RW).
- GET for the next card (if FUNC=RPW/RW and a PUT for the print file was not issued).

Although stacker selection is always connected with the punch file if FUNC=RP or RPW, a card that is not to be punched will be properly stacker selected by a special stacker select CCW.

CDMOD: PUT Macro, Charts AJ-AM

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

Entry: From a PUT macro expansion.

Exit: To the problem program.

Method: For regular output card files, the PUT routine punches the next sequential record in the file into a card. If

WORKA=YES is specified in the DTFCD and CDMOD macros, the record is built in the workarea and then moved to the appropriate output area before punching is performed. If IOAREA2=YES is specified in the DTFCD and CDMOD macros, overlap of I/O and processing is possible by alternately switching output areas. (Dual I/O areas are not valid for associated files.)

For combined files (1442 or 2540 with PFR), IOAREA2 if it is specified, is used as the output area. If IOAREA2 is not specified, IOAREA1 serves as both the input and the output area.

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

Punching of a card in a 2560 or 5424/5425 associated file can be initiated by:

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

CONSOLE FILES

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

Console files are neither opened nor closed by logical IOCS.

DTFCN Macro

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

DTFCN: GET Macro, Chart AN

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

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

Exit: To the problem program.

Method: Upon entry to the GET routine, the CCW command code is set to hex '0A' for a read operation. If UNDEF is specified in the RECFORM= parameter in the DTFCN macro, the user specified BLKSIZE is moved into the byte count area of the CCW. (If FIXUNB is specified in the RECFORM= parameter in the DTFCN macro, the byte count area of the generated CCW automatically contains a count of 80.) An SVC 0 is then issued to read the record into IOAREA1. If a work area was not specified in the DTFCN macro, control returns to the problem program immediately after completion of the I/O operation.

If WORKA=YES is specified in the DTFCN macro, the contents of IOAREA1 are moved to the work area before control returns to the problem program.

Bytes*	Contents	Function
0-15 (00-0F)		CCB.
16 (10)	X'20' X'08'	COBOL open; ignore option. DTF table address constants relocated by OPENR.

17-19
(11-13)
		Address of logic module:
		GET and PUT logic if TYPEFLE=INPUT;
		PUT logic if TYPEFLE=OUTPUT;
		GET, PUT, and PUTR logic if TYPEFLE=CMBND.

20 (14)	X'03'	DTF type.
21-23 (15-17)		For input and output: not used. For combined: byte 21 contains X'01' bytes 22-23 contain INPSIZE.
24-31 (18-1F)	X'09',IOAREA1,X'00',BLKSIZE	CCW.
End of table if RECFORM=FIXUNB and WORKA not specified.		
The following bytes are added to the table if WORKA is specified.		
32-35 (20-23)	DC A(ICAREA1)	Address of I/O area.
36-39 (24-27)	DC F'0'	Register save area.
40-43 (28-2B)	DC F'0'	Register save area.
End of table if RECFORM=FIXUNB.		
The following bytes are added to the table if RECFORM=UNDEF.		
	DC F'0'	Register save area.
	DC F'0'	Register save area.
	DC H'BLKSIZE'	I/O area size.
	DC AL2(BLKSIZE-1)	For input files only.
The following bytes are added to the table if TYPEFLE=CMBND		
32-35 (20-23)	DC A(ICAREA1+BLKSIZE)	I/O area address for input.
36-37 (24-25)	DC H'BLKSIZE'	Block size.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 4. DTFCN: Console

DTFCN: PUT Macro, Chart AP

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

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

Exit: To the problem program.

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

(TYPEFLE=INPUT) with a work area specified, this test is bypassed.

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

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

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

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

DTFCN: PUTR Macro, Chart AQ

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

Entry: Form the PUTR macro expansion.

Exit: To the problem program.

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

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

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

MAGNETIC INK CHARACTER RECOGNITION FILES

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

DTFMR Macro

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

Bytes*	Bits	Function
0-5 (00-05)		CCB indicators.
6-7 (06-07)		Logical class and unit numbers (primary if DUAL addressing).
8 (08)		Zero.
9-11 (09-0B)		CCW address.
12-15 (0C-0F)		Zeros.
16 (10)	0-1	Not used.
	2	COBOL open; ignore option.
	3	Not used.
	4	DTF table address constants relocated by OPENR.
	5-7	Not used.
17-19 (11-13)		Address of logic module.
20 (14)		DTF type = X'0B'
21 (15)		Logic module option switches.
	0	User disengage - 0 = off; 1 = on.
	1	Program sort mode - 0 = no; 1 = yes.
	2	First time switch (after engage) - 0 = no; 1 = yes.
	3	Addressing = DUAL - 0 = no; 1 = yes.
	4	Waiting - 0 = no; 1 = yes.
	5	Read logic indicator - 0 = no; 1 = yes.
	6	Not used.
	7	Supervisor initial read (after open) - 0 = no; 1 = yes.
22-29 (16-1D)		Symbolic filename.
30 (1E)		Open/Close option switch.
	0	Open indicator - 0 = closed; 1 = open.
31-33 (1F-21)		Open/Close option switches.
34-35 (22-23)		Logic module option switches.
36-39 (24-27)		Error information status.

Figure 5. DTFMF (Part 1 of 4)

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

Figure 5. DTFMR (Part 2 of 4)

Bytes*	Bits	Function
96-99 (60-63)		Stacker-select routine address.
100-103 (64-67)		Address of stacker select CCW chain.
104-107 (68-6B)		Current buffer address pointer (Supervisor).
108-111 (6C-6F)		Supervisor count.
112-113 (70-71)		Number of buffers minus 7.
114-115 (72-73)		Message indicator.
116-119 (74-77)		ERROPT routine address.
120-121 (78-79)		Logical class and unit numbers (secondary -- for DUAL addressing only).
122-123 (7A-7B)		Reserved for future use.
124-127 (7C-7F)		Address of last buffer given to user.
128-131 (80-83)		Address of first byte of last buffer.
132-139 (84-8B)		Channel status word (CSW).
140-143 (8C-8F)		Address of active GET record.
144-147 (90-93)		GET counter.
148-159 (94-9F)		Reserved for future use.
For SINGLE Addressing		
160-167 (A0-A7)		CCW - Engage.
168-175 (A8-AF)		CCW - Read.
176-183 (B0-B7)		CCW - Sense.
184-191 (B8-BF)		CCW - NOP.

Figure 5. DTFMR (Part 3 of 4)

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

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 5. DTFMF (Part 4 of 4)

MRMOD Macro

The MRMOD macro provides the necessary logic to process Magnetic Ink Character Recognition (MICR) and Optical Reader/ Sorter files. The logic module can handle MICR type files with either dual or single addressing.

The functions of the MR logical IOCS module are:

1. To locate the record (document buffer) to be processed (GET/READ).
2. To test for a "document ready for processing" condition (CHECK).
3. To start and stop the feeding of documents through the device (DISEN; disengage/ engage is an automatic function of GET or READ).
4. To turn on the pocket lights (LITE).
5. To exit from the partition, when no useful processing can be performed (WAITF).

The READ, CHECK, WAITF macro combination is provided for users operating more than one 1255/1259, 1270/1275, or 1419 in a given partition of the system. The READ macro performs all the functions of the GET macro except the inherent wait that occurs when no document is ready for processing.

The READ macro posts an indicator in the current document buffer signifying that the buffer is not ready for processing. The user examines the status of this indicator by issuing the CHECK macro.

The WAITF macro determines if processing can continue. If processing cannot continue, the system is allowed to continue processing in another partition.

MRMOD: CHECK Macro, Charts BA-BB

Objectives: To determine the status of the current document buffer:

1. Buffer ready for processing.
2. Buffer waiting for data.

Entry: From the CHECK macro expansion to the label IJUCHEK.

Exit: To the problem program at one of three locations:

1. The next sequential instruction if the document buffer contains data and is ready for processing.
2. The address specified in the label operand of the macro if the document buffer is waiting for data.
3. The ERROPT address specified in the DTF if the document buffer contains posted error-condition indicators.

Method: The CHECK macro first tests the condition of the wait bit (bit 5 of the first byte) within the current document buffer. If this bit is on, indicating that the device is waiting for a document to be read, a branch is made to the address specified in the second (label) operand of the CHECK macro. This bypasses normal record processing and allows continuation of other processing in the partition. If the second (label) operand is not specified in the CHECK macro, control is passed to the ERROPT address. If neither address is specified, a branch is made to the next sequential instruction following the CHECK macro expansion.

If the wait bit is off, a second test determines if the Buffer Status Indicator (BSI) bit (bit 0 of the first byte within the current document buffer) is on. If the bit is on, indicating that the data is valid and ready for processing, a branch is made to the next sequential instruction following the CHECK macro expansion where normal processing resumes.

If both bits are off (bits 0 and 5), indicating either that the device is not operational or that the document buffer data is in error, a branch is made to the ERROPT location specified in the DTF for further analysis of the error conditions by the user. If the ERROPT location is not specified, a branch is made to the next sequential instruction following the CHECK macro expansion.

MRMOD: GET Macro, Charts BA-BB

Objectives:

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

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

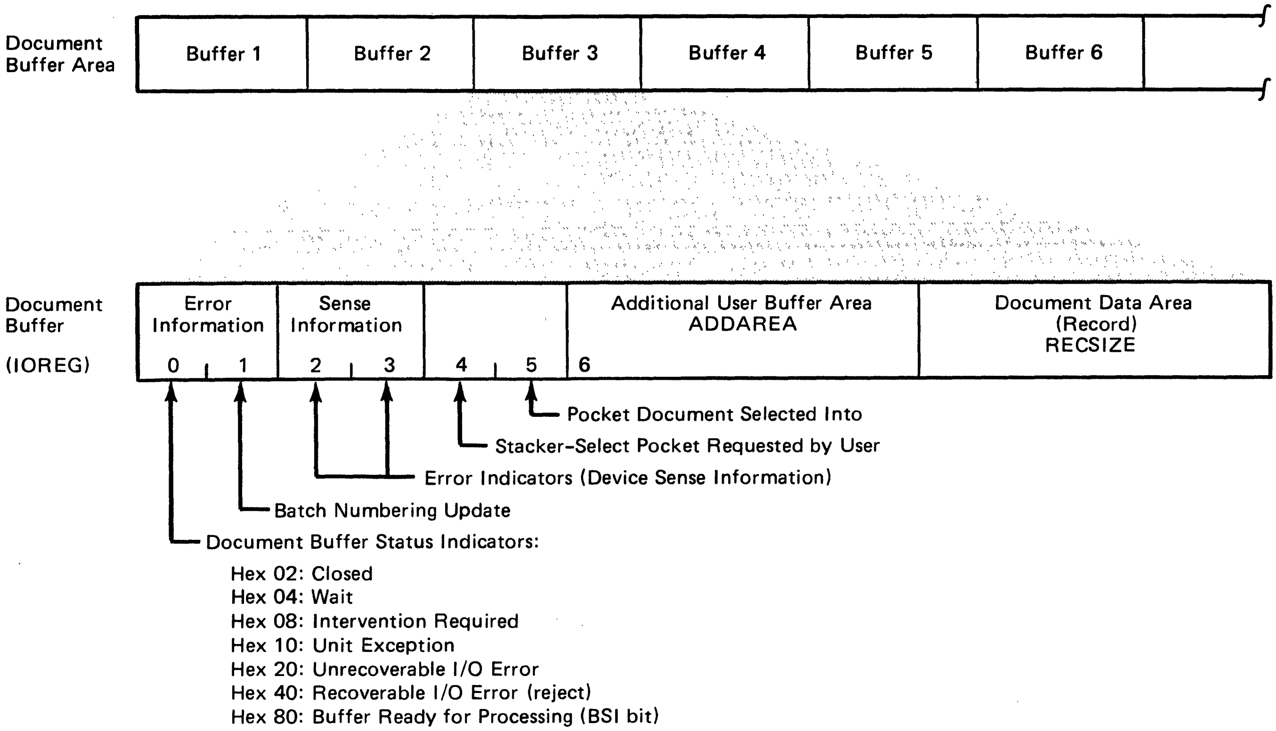

Document Buffer Area
Buffer 1
Buffer 2
Buffer 3
Buffer 4
Buffer 5
Buffer 6
Document Buffer
(IOREG)
Error Information
Sense Information
Additional User Buffer Area ADDAREA
Document Data Area (Record) RECSIZE
0
1
2
3
4
5
6
Pocket Document Selected Into
Stacker-Select Pocket Requested by User
Error Indicators (Device Sense Information)
Batch Numbering Update
Document Buffer Status Indicators:
Hex 02: Closed
Hex 04: Wait
Hex 08: Intervention Required
Hex 10: Unit Exception
Hex 20: Unrecoverable I/O Error
Hex 40: Recoverable I/O Error (reject)
Hex 80: Buffer Ready for Processing (BSI bit)

Figure 6. MICR Buffer Format (1255/1259/1270/1275/1419)

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

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

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

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

MRMOD: READ Macro, Charts BA-BB

Objectives:

1. To provide a pointer, in the user register IOREG, to the next document buffer to be processed in the document buffer area (see Figure 6).

2. To issue engage commands to the MICR type device when necessary.

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

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

Method: Same as the GET macro except for the inherent wait that occurs when the document buffer is not ready for processing. Instead of issuing a wait SVC (SVC 7), the READ macro posts an indicator in the document buffer (bit 5 of first buffer byte) to signal that the buffer is not ready for processing. The user, after examining the indicator by issuing a CHECK macro, can then transfer processing to another location within the partition until the buffer is ready for processing.

<table>
<tr><td></td><td colspan="16">User's Two-Byte Pocket Light Area</td></tr>
<tr><td>Byte Displacement</td><td colspan="8">0</td><td colspan="8">1</td></tr>
<tr><td>Bit Position</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>Pocket Designated if Bit=1</td><td>A</td><td>B</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td colspan="3">Binary Zeros</td><td>Err. Ind.</td></tr>
<tr><td rowspan="2">Pocket indicator bits as rearranged and stored in module for testing</td><td colspan="16">MRMOD Test Area "IJULBKT"</td></tr>
<tr><td colspan="4">Binary Zeros</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>A</td><td>B</td></tr>
</table>

Pocket light test starts at Pocket B and continues through Pocket 0.

Figure 7. 1275/1419 Pocket Light Indicator User Area and Work Area

MRMOD: LITE Macro, Chart BC

Objective: To turn on the pocket lights specified in the pocket light indicator by the user.

Entry: From a LITE macro expansion to the label IJULITE.

Exit: To the prcblem program.

Prerequisites:

1. A DISEN macro must be issued before the LITE macro is issued.

2. The unit exception bit must be turned on in the CCB in the DTF table before the LITE macro is issued.

Method: The LITE macro checks the pocket light indicator (see Figure 7) bit by bit, to determine and turn on the lights specified by the user. After each specified pocket light is turned on, a check is made for the occurrence of an unrecoverable I/O errcr during the turning on of the light. If an error occurred, bit 15 of the pocket light indicator is set.

The pocket light indicators are tested in the sequence: E, A, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0. Tc turn on a particular pocket light, a pocket code is combined with a stacker select command code.

Example: Assume that the light in pocket B is to be turned on, that is, bit 1 of the pocket indicator is on.

Upon entry to the pocket light routine of the MRMOD, a count of 12 is initially loaded into a register. When the actual light program lccp is entered (at location IJUNXTLT) and it is determined that the pocket B light is to be turned on, the count is reduced by one to 11. The 11 is then multiplied by 16. The result is 176, or a hexadecimal B0.

For the single address adapter, the hex B0 is ORed with a hex 0F and results in a command code of hex BF. For the dual address adapter, the hex B0 is ORed with a hex 07 and results in a command code of hex B7. After the pocket light is successfully turned on, the initial count (12) is reduced by one and the routine returns to location IJUNXTLT to test for the next pocket light.

MRMOD: DISEN Macro, Chart BD

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

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

Exit: To the problem program.

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

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

MRMOD: WAITF Macro, Chart BD

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

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

Exit: To the problem program.

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

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

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

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

Initialization and Termination of MICR Type Files

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

$$BOMR01: Open MICR Type Files, Chart BE

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

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

Exit: To the TES processor, $$BOPEN.

Method: The $$BOMR01 routine clears the entire document buffer area and checks for device assignment. It then calculates the Physical Unit Block (PUB) entry address for the device, and determines from the PUB information which entry in the Supervisor table of DTF addresses (PDTABB) is to be used. The address of the DTF table is then inserted into the proper entry in the PDTABB table. Refer to VSE/Advanced Functions Diagnosis Reference: Supervisor, LY33-9091, for the format and use of the PDTABB table.

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

$$BCMR01: Close MICR Type Files, Chart BE

Objective: To close 1255/1259/1270/1275/1419 MICR type files and to reinitialize the DTFMR table.

Entry: From the Close Monitor, $$BCLOSE, to the label FRSTINST.

Exit: To the Close Monitor, $$BCLOSE.

Method: The $$BCMR01 routine resets the open indicator in the DTF and the traffic (wait) bit in the CCB, and turns off the external line status indicator for the file.

MICR Error Messages

Error conditions occurring on 1255/1259/1270/1275/1419 MICR type devices are reported to the operator via a special MICR error message writer logical transient, $$BMMR20.

$$BMMR20: MICR Message Writer, Chart BF

Objective: To print one of two error messages for MICR type devices:

1. 4MR1I - EXTERNAL INTERRUPT I/O ERROR

2. 4MR2I - SCU NOT OPERATIONAL

Entry: From the MR logical IOCS module when an error occurs during a GET or READ.

Exit: To the logical IOCS module after the proper message is printed.

Method: The $$BMMR20 message writer routine examines the message indicator in the DTF table (bytes 114-115) to determine the appropriate message. It then extracts information from the DTF and CSW, converts the informaticn to printable form, and inserts it into the message.

The proper message is then printed on SYSLOG and control is returned to the logical IOCS module.

OPTICAL READER FILES

Optical reader files are input files associated with a 1287 optical reader (OR) or with a 3886 optical reader (DR). Files for a 1287 optical reader are defined by a DTFOR macro; an ORMOD macro provides the data handling logic module for this type of file. Files for a 3886 optical reader are defined by a DTFDR macro; a DRMOD macro provides the data handling logic module for this type of file. The DFR and DLINT macros build format records that are used to process 3886 optical reader files.

DTFOR Macro

The DTF table generated by the DTFOR macro services the 1287 optical reader; either journal tapes or documents can be processed (see Figure 8).

Bytes*	Bits	Function
0-15 (00-0F)		Dummy CCB.
16 (10)	0-1	Not used.
	2	COBOL open; ignore option.
	3	Not used.
	4	DTF table address constants relocated by OPENR.
	5-7	Not used.
17-19 (11-13)		Address of logic module.
20 (14)		DTF type, (X'09'). DTF type, (X'0A' if HEADER=YES).
21 (15)		PIOCS switches.
	0	1 = Open; 0 = Closed.
	1	1 = Input.
	2	1 = Control.
	3	1 = Device is 1287.
	4	1 = Header.
	5	Reserved for future use.
	6	1 = RDLNE.
	7	Not used.
22 (16)		Not used.
23 (17)	0-6	Not used.
	7	1 = LIOCS posts a hopper empty condition to DTF.
24-39 (18-27)		CCB.
40-47 (28-2F)		Sense CCW.
48-51 (30-33)		Lost lines (equipment check).
52-55 (34-37)		After nine retries for journal tape, or after two retries for documents.
56-59 (38-3B)		Wrong-length records.
60-63 (3C-3F)		After four retries for journal tape, or after two retries for documents.
64-67 (40-43)		Keyboard corrections.
68-71 (44-47)		Count of data check errors.
72-75 (48-4B)		Lines marked.
76-79 (4C-4F)		Total lines read (CCW chains executed).

Figure 8. DTFCR (Part 1 of 2)

Bytes*	Bits	Function
80 (50)		Error indicators.
	0	1 = EOP.
	1	1 = Lost reference mark indicator.
	2	1 = Late stacker selection.
	3	1 = Non-recovery error.
	4	1 = Equipment check.
	5	1 = Wrong-length record.
	6	1 = Hopper empty.
	7	1 = Data check.
81 (51)		LIOCS switches.
	0	1 = First time.
	1	1 = Two I/O areas.
	2	1 = WORKA=YES
	3	1 = RECFORM=FIXUNB
	4	1 = RECFORM=UNDEF
	5-7	Not used.
82 (52)		Normal command code.
83 (53)		Control command code.
84-87 (54-57)		IOAREA2 address.
88-95 (58-5F)		Read CCW.
96-103 (60-67)		Go to next line CCW.
104-111 (68-6F)		Control CCW.
112-115 (70-73)		EOF address.
116-119 (74-77)		Correction exit address.
120-123 (78-7B)		IOAREA1 address.
124-127 (7C-7F)		DC A(&BLKS-1)
128-129 (80-81)		SR 13,&RECS
130-131 (82-83)		LR &RECS,13
132-133 (84-85)		LR &IOR,13
134-135 (86-87)		Sense.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 8. DTFOR (Part 2 of 2)

ORMOD Macro

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

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

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

ORMOD: CNTRL Macro, Charts BG-BH

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

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

Entry: From a CNTRL macro expansion.

Exits:

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

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

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

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

ORMOD: GET Macro, Charts BJ-BN

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

Entry: From the GET macro expansion.

Exit: To the problem program.

Method: The GET routine reads a record from the 1287 optical reader and places it in main storage starting at the high end of the I/O area and going toward the low end until the last character has been read.

If two I/O areas or one I/O area and a work area are used, I/O and processing overlap is possible. On the first GET operation, one area is filled and another EXCP is issued before any processing is allowed. After the second EXCP is issued, control is returned to the problem program so that processing can begin on the data in the first I/O area. If two I/O areas are specified, the first GET macro instruction sets a switch that changes the logic flow of all the following GET instructions used in the problem program with this logic module. If a work area is specified, the contents of the I/O area are moved to the work area. The next EXCP is issued, and an exit to the problem program is made. If RECFORM=UNDEF, the contents of the I/O area are left-justified in the work area.

ORMOD: RDLNE Macro, Chart BP

Objective: To cause a line of data from a journal tape to be read in the online mode while processing is in the offline correction mode.

Entry: From the RDLNE macro expansion.

Exit: To the problem program.

Method: The routine indicates a keyboard correction, modifies the CCW with a read backward online command, and issues an EXCP. When the I/O operation is complete, the routine checks for equipment, wrong-length record, and data checks. If present, these conditions are posted to the DTF, and an exit is made to the problem program.

ORMOD: DSPLY Macro, Chart BQ

Objective: To cause a complete field of data from a document to be displayed on the display scope of the 1287.

Entry: From the DSPLY macro expansion.

Exit: To the problem program.

Method: The routine modifies the read backward CCW by changing the command code, input address, and chaining flags. It then blanks out the portion of I/O area into which it is reading. An EXCP is then issued for this CCB. When the I/O operation is complete, the routine checks for a unit exception indicating an error was made in keying in corrections. The command is reexecuted if necessary. If a unit exception is not present, an exit is made to the problem program.

ORMOD: READ Macro, Chart BQ

Objective: To access a record or records from the 1287 optical character reader when processing documents.

Entry: From the READ macro expansion.

Exit: To the problem program.

Method: The READ routine reads a record or records from a document by executing the users CCWs. It is the user's responsibility to properly set up the CCWs before the READ macro is issued.

ORMOD: RESCN Macro, Chart BR

Objectives: To allow up to nine retries of an unreadable line of data on a document. Also to allow a read in the online mode while processing in the offline correction mode.

Entry: From the RESCN macro expansion.

Exit: To the problem program.

Method: The routine modifies the CCW to allow data checks, and an EXCP is issued. After the I/O operation is complete, the routine checks for and posts equipment checks, data checks, and wrong-length records. The line of data is retried the number of times specified.

When the retries are completed, control returns to the problem program unless forced online correction was specified. In the case of forced online correction, the CCW is modified to force online correction and another EXCP is issued.

ORMOD: WAITF Macro, Charts BS-BU

Objective: To provide storage resident error recovery in addition to its normal function of waiting for I/O completion.

Entry: From the WAITF macro expansion.

Exit: To the user-specified COREXIT routine if a read error occurred or to the next sequential instruction following the WAITF macro expansion in the problem program if there was no read error.

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

DFR and DLINT Macros

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

The DFR macro builds the first part of the format record, called the Document Information Record (see Figure 9); it also generates two fields preceding the format record to provide information about that record. The first field is eight bytes long; when the field is opened or when the SETDEV macro is issued it contains the name of the format record. The second field is a 2-byte binary field that contains the total length of the format record plus the two preceding fields. These two fields are not part of the format record and therefore not included when the format record is loaded into the 3886 control unit.

The DLINT macro builds the following for each line described:

- A line information record to describe an individual line (see Figure 10).
- A field information record that describes an individual field on the line that is to be scanned (see Figure 11).
- A field information record that describes an individual field on the line that is not to be scanned (see Figure 12).

- A sync byte of X'FF' to indicate the end of the DIINT expansion.

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

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

Bytes	Bits	Contents	Functions
0		X'3F' or user-specified character code	Hex code sent by the 3886 for an unrecognizable character.
1			Bit on indicates which font is on document.
	0		Numeric A font.
	1		Alphanumeric A font.
	2		Numeric B font.
	3		Alphanumeric B font.
	4		NHP/Gothic font.
	5		National NHP/Gothic font.
	6-7		Not used.
2	0	1	Allow group and character erase characters.
	1-3		Serial and batch number control.
		B'000'	No numbering.
		B'001'	Batch and serial number stacker A documents.
		B'010'	Batch and serial number stacker B documents.
		B'011'	Batch and serial number all documents.
		B'101'	Batch number stacker A documents.
		B'110'	Batch number stacker B documents.
		B'111'	Batch number all documents.
	4-7		National symbol set options.
		B'0000'	Allows recognition of different font characters for specific hex codes.
		.	
		.	
		B'0101'	
3-8		Edit character or X'00'	Edit characters: characters that can be removed from a field read.

Figure 9. DFR Macro Instruction Expansion - Document Information Record

Bytes	Bits	Contents	Functions
0		X'00' - X'3F'	Line format record number; a unique number for this group of line format data (0-63).
1			Left margin; the address in BCD of the leftmost side of the leftmost field to be scanned on this line.
2	0	B'0'	Edit format selected for data record. Also allows edit function.
		B'1'	Image format selected for data record. Inhibits add/delete functions.
	1-7	B'0000000'	Not used.

Figure 10. DLINT Macro Instruction Expansion - Line Information Record

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

Figure 11. DLINT Macro Instruction Expansion - Field Information Record for a Scannable Field

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

Figure 12. DLINT Macro Instruction Expansion - Field Information Record for a Field that is not to be Scanned

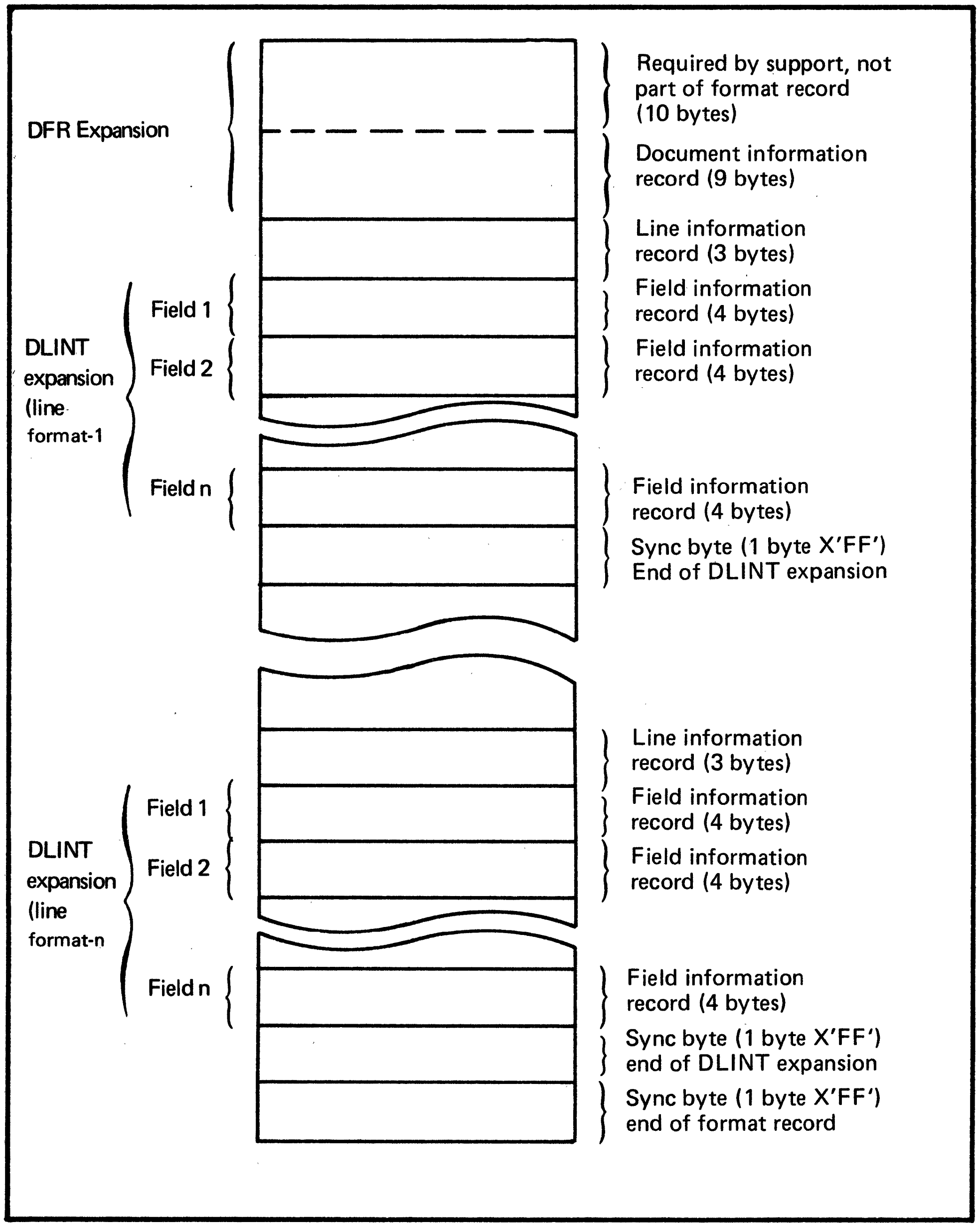

DFR Expansion
Required by support, not part of format record (10 bytes)
Document information record (9 bytes)
Line information record (3 bytes)
DLINT expansion (line format-1
Field 1
Field information record (4 bytes)
Field 2
Field information record (4 bytes)
Field n
Field information record (4 bytes)
Sync byte (1 byte X'FF') End of DLINT expansion
Line information record (3 bytes)
DLINT expansion (line format-n
Field 1
Field information record (4 bytes)
Field 2
Field information record (4 bytes)
Field n
Field information record (4 bytes)
Sync byte (1 byte X'FF') end of DLINT expansion
Sync byte (1 byte X'FF') end of format record

Figure 13. Format Record Relationship to DFR and DLINT Expansions

DTFDR Macro

The DTF table generated by the DTFDR macro services the 3886 optical reader (see Figure 14). It requires a format record built by the DFR and DLINT macros.

Bytes*	Bits	Contents	Function
0-15 (00-0F)			CCB.
16 (10)	0-1		Not used.
	2		COBOL open; ignore option.
	3		Not used.
	4		OPENR relocates DTF table addresses.
	5-7		Not used.
17-19 (11-13)			Address of logic module.
20 (14)		X'0C'	DTF type.
21 (15)			PIOCS switches:
	0		1 = open; 0 = closed.
	1	B'1'	Input.
	2-5	B'0000'	Not used.
	6	B'1'	Device is 3886.
	7	B'0'	Not used.
22 (16)			Not used.
23 (17)			LIOCS switches:
	0-4	B'00000'	Not used.
	5		1 = SETDEV.
	6		1 = Control passed to COREXIT.
	7		1 = FR loaded from disk.
24-31 (18-1F)			FR phasename at open time.
32-39 (20-27)			Phasename of currently used FR.
40-43 (28-2B)		X'00000000'	Not used.
44-47 (2C-2F)			Start address of FR area in DTF.
48-51 (30-33)			Address of 4-byte pointer at the end of the FR area in the DTF.
52-55 (34-37)			EOF routine address.
56-63 (38-3F)			Scan CCW.
64-71 (40-47)			Read CCW.
72-79 (48-4F)			Control CCW.
80-87 (50-57)			Load format record CCW.

Figure 14. DTFDR (Part 1 of 2)

Bytes*	Bits	Contents	Function
88-91 (58-5B)			COREXIT routine address.
92-95 (5C-5F)			IOAREA1 area address.
96-99 (60-63)			Header area address.
100-103 (64-67)			Exit indicator address.
104 (68)			Start of DR area.
105-107 (69-6B)			Header area address.
108-111 (6C-6F)			Exit indicator address.
112 (70)			Start of FR area.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 14. DTFDR (Part 2 of 2)

DRMOD Macro

The module generated by the DRMOD macro provides the logic to perform the READ, WAITF, CNTRL, and SETDEV functions for the 3886 optical reader. The DRMOD macro is capable of generating a number of logic modules, each tailored to the specific DRMOD parameters. Because it is not practical to describe each variation individually, the internals of the DR module are described and flowcharted to indicate all variations.

DRMOD: CNTRL Macro, Chart CA

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

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

Entry: From the CNTRL macro expansion.

Exits:

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

Method: The CNTRL routine sets up the field address and the length count in the CCW. If the CCW command is eject or stacker select, the routine builds a 1-byte field to be passed to the 3886. The control CCW is then executed. When the I/O operation is completed, the routine checks for unit exception. If unit exception occurs, control is passed to the user's EOF routine. The routine then checks to see if any errors occurred. If an error occurred, it is posted to the user and control is passed to the user's COREXIT routine. On return from the COREXIT routine, control is passed to the problem program.

DRMOD: READ Macro, Chart CA

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

Entry: From the READ macro expansion.

Exit: To the problem program.

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

DRMOD: SETDEV Macro, Chart CB

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

Entry: From the SETDEV macro expansion.

Exits:

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

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

DRMOD: WAITF Macro, Chart CB

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

Entry: From the WAITF macro expansion.

Exits:

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

Method: After checking for completion of the previous I/O operation, the routine moves the header record (20 bytes) to the address specified in the header parameter of the DTF. The recognition data is then moved to the user's I/O area specified by the IOAREA1 parameter in the DTF. The routine then checks for unit exception. If unit exception has occurred, control is passed to the user's EOF routine. The routine then checks if any error occurred during the previous I/O operation. If so, the error is posted to the user and control is passed to the user's COREXIT routine. On return from the COREXIT routine, control is returned to the problem program.

Note: If more than one file is specified in the operand field of the WAITF macro, this routine cancels the problem program by issuing an illegal supervisor call.

Initialization and Termination of Optical Reader Files

Optical reader files are opened by the logical transient $$BOOR01 that is fetched by the Open Monitor (refer to VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 1). These files are closed by the Close Monitor, $$BCLOSE, which simply resets the open indicator in byte 21 of the DTF table.

$$BOOR01: Open Optical Reader, Chart CC

Objective: To open an optical reader file.

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

Exit: To $$BOPEN or cancel.

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

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

Method - 3886: This phase opens the file and loads the format record from disk into the DTF. If the format record is within the correct limits, it is loaded into the 3886 control unit. If the format record is not of the correct length the open routine is canceled by issuing message 4186I. If a device error occurs when loading the format record in the 3886, the command is retried. If the error persists, the open routine is canceled by issuing message 4186I.

PRINTER FILES

Printer (PR) files are defined by a DTFPR macro for output files on 1403, 1443, 3203, 3211, and 5203 printers, the 3525 card punch with print feature, the 2560 Multifunction Card Machine, and the

5424/5425 Multifunction Card Unit. In addition, the Selective Tape List (STL) feature for printing journal tapes on a 1403 is also supported.

Data handling logic for printer files is supplied by the associated PRMOD macro or the 3800 printer extended buffering module, IJDPR3. Data handling for the 3800 printer files can utilize extended buffering. Printer files are cpened and closed by the Open and Close Mcnitors, respectively, (refer to VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 1).

DTFPR Macro

The DTFPR macro (see Figure 15) generates a DTF table containing all the necessary information tc describe an output printer file on any of the following devices:

- IBM 1403 Printer (with or without the STL feature)
- IBM 1443 Printer
- IBM 2560 MFCM
- IBM 3203 Printer
- IBM 3211 Printer
- IBM 3262 Printer
- IBM 3289-4 Printer
- IBM 3525 Card Punch with print feature
- IBM 3800 Printer
- IBM 5203 Printer
- IBM 5424/5425 Multifunction Card Unit.

Bytes*	Bits	Contents	Function
0-15 (00-0F)			CCB. If 3800 extended buffering is selected, the CCW address is changed by OPEN to point to a DTF extension work area in the user virtual area. CLOSE restores it.
16 (10)	0		1 = 2-line printer[3],[4]; 0 = other.
	1		1 = ERROPT[3],[4]; 0 = omitted.
	2		COBOL open; ignore option.
	3		1 = 3525; 0 = other.
	4		OPENR relocates DTF address constants.
	5-7		3525 Modes: 000 = PRINT only 011 = PUNCH/PRINT[3] 010 = READ/PRINT[3] 101 = READ/PUNCH/PRINT[3] 3800 Modes: (Bit 7 not used) 11 = TRC=YES specified on DTF[7]. 01 = TRC=Y specified via SETPRT (set by OPEN)[7].
17-19 (11-13)			Address of logic module. If 3800 extended buffering is selected, OPEN changes this address to point to extended buffering logic module IJDPR3 in system virtual area. CLOSE restores it.
20 (14)		X'08'	DTF type.
		X'07'	DTF type for 2560 and 5424/5425.
21 (15)	0		1 = open; 0 = closed.
	1		First time switch.
	2		1 = Control character.
	3		1 = Fixed unblocked records.
	4		1 = Variable unblocked records.
	5		1 = Two I/O areas.
	6		1 = Work area.
	7		1 = Print overflow channel 9.
For printer and card punch devices.			
22 (16)		X'09'	Normal command code[5].
23 (17)		X'09'	Control command code[5].
24-27 (18-1B)		DC A(IOAREA1+x)	Address of data in IOAREA1.
28-31 (1C-1F)			Bucket.[1]
32-33 (20-21)		LR 12,(RECSIZE) NOPR 0	For undefined records only.
34-37 (22-25)		LA &IOREG,4(14) NOP 0	Only if IOREG=(r).
38-39 (26-27)			Bucket.[2]
40-47 (28-2F)		11,*,X'60',1 9,IOAREA,X'20',121	CCW -- Set up Selective Tape List Control[6]. STLIST not specified.

Figure 15. DTFPR (Part 1 of 3)

Bytes*	Eits	Contents	Function
For printer and card punch devices (Cont.)			
48-55 (30-37)		9,IOAREA,X'20',121	CCW -- STLIST specified[6].
		A(Name)	Address of user error routine (for all the 3211-compatible printers identified by device type code PRT1).
		DC A(ASOCFLE)	If ASOCFLE=filename[3].
For the 2560 and 5424/5425 Multifunction Card Machine.			
22 (16)		X'00'	Not used.
23 (17)		B'HHHHHH00'	Print head selection byte. H = 1 specifies the corresponding head.
24-27 (18-1B)			Address of IOAREA1.
28-31 (1C-1F)			Bucket.
32-33 (20-21)		LR 12,(RECSIZE) NCPR 0	For undefined records only.
34-37 (22-25)		LA &IOREG,4(14) NOP 0	Only if IOREG=(r)
38-39 (26-27)			Number of bytes to be printed by the last specified print head.
40-43 (28-2B)		DC A(ASOCFLE) DC F'0'	If FUNC=RW, PW, or RPW. In all other cases.
44 (2C)	0		1 = 2560.
	1		Not used.
	3		1 = print control switch for 2560 associated files.
	4-7		Not used.
45-47 (2D-2F)		DC 3X'00'	Reserved for future use.
For 2560 sirple files.			
48-55 (30-37)			Eject CCW.
56-63 (38-3F)			Load print head buffer CCW.
64-71 (40-47)			Print CCW.
For 2560 associated files.			
48-55 (30-37)			Load print head buffer CCW.
56-63 (38-3F)			Print CCW.

Figure 15. DTFPR (Part 2 of 3)

Bytes	Bits	Contents	Function
For 5424/5425 files.			
48-55 (30-37)			Print CCW.

[1] The bucket bytes handle undefined records. Bit 0 of byte 28 at open time determines the mode set of a printer with UCS. If bit 0 is 1, the mode is set so that data checks occur if an invalid character is printed. Otherwise, mode is set to suppress data checks. The use of the UCS parameter determines the setting of this bit. If STLIST=YES, byte 31 saves the STLIST control byte provided by the PUT macro.

[2] The 2 byte bucket saves print overflow conditions if CTLCHR=ASA. If STLIST=YES, byte 38 contains the current STLIST control byte. Byte 39 is set by the PUT macro to indicate spacing or skipping. (X'00' no spacing, no skipping; X'01' spacing; X'02' skipping.)

[3] Valid for 3525 READ/PRINT, PUNCH/PRINT, and READ/PUNCH/PRINT files.

[4] Valid for 3525 PRINT only files.

[5] X'05' for 3525, X'09' for other devices.

[6] Valid for 1403 only.

[7] Valid for 3800 only.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 15. DTFPR (Part 3 of 3)

PRMOD Macro

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

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

PRMOD: CNTRL Macro, Chart CD

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

Entry: From a CNTRL macro expansion.

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

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

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

PRMOD: PRTOV Macro, Chart CE

Objective: To cause and control an overflow skip.

Entry: From a PRTOV macro expansion.

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

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

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

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

PRMOD: PUT Macro, Charts CF-GG

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

Entry: From a PUT macro expansion.

Exit: To the problem program or cancel.

Method: This routine causes a record to be printed on the cutput device. The logic determines if two I/O areas are used, if a work area is specified, if a table reference character (TRC) exists in the data, and if CTICHR controls the carriage. A test is made to determine if CNTRL is specified. If sc, the CTICHR cannot be used; CTLCHR and CNTRL are mutually exclusive (if one is used, the other cannot be specified).

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

If the TRC=YES cption is used, a TRC byte can be processed.

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

For 2560 and 5424/5425 associated files, the print module initiates the read and/or punch command of the associated read and/or punch file processed by a CDMOD.

If the Selective Iape List feature is used (1403 only), selected tapes are controlled through the use of a 1-byte control field. This field is accessed by the optional operand, either STLSP=label, or STLSK=label, of the PUT macro. Figure 16 shows the format of the field specified by label.

Bit	0	1	2	3	4	5	6	7
Tape Positions	8	7	6	5	4	3	2	1
Note: Tape 1 is the leftmost tape on the Selective Tape Lister.								

Figure 16. STL Control Field

The control field is used as the data address in the control CCW in the DTF table. Double width tapes are controlled by using the two bits corresponding to the tape positions occupied.

Note: The label CONCMD used in this logic is a one byte save area in the DTF table used to store the control character.

EYTENDED BUFFER PROCESSING

SAM logic causes one, two, and sometimes three EXCPs per print line on existing printers. This method cannot be used to drive the 3800 at rated speed because of the CPU instruction overhead and channel selection overhead. For the 3800, extended buffering writes multiple lines per EXCP using a command-chained channel program. Extended buffering logic provides seven functions for a 3800 printer file: CNTRL, PRTOV, PUT, buffer truncation, TRC/FCB update, OPEN, and CLOSE as depicted in Figure 17. A 3800 printer file defined by a DTFPR, DTFDI, or DTFCP is supported by extended buffering logic.

IJDPR3: CNTRI Macro, Chart XA

Objective: To control the form space and skip operations for a 3800 DTFPR printer file opened for extended tuffering mode.

Entry: IJDPR3 + 0 from a CNTRL macro expansion.

Exits: To the caller or cancel.

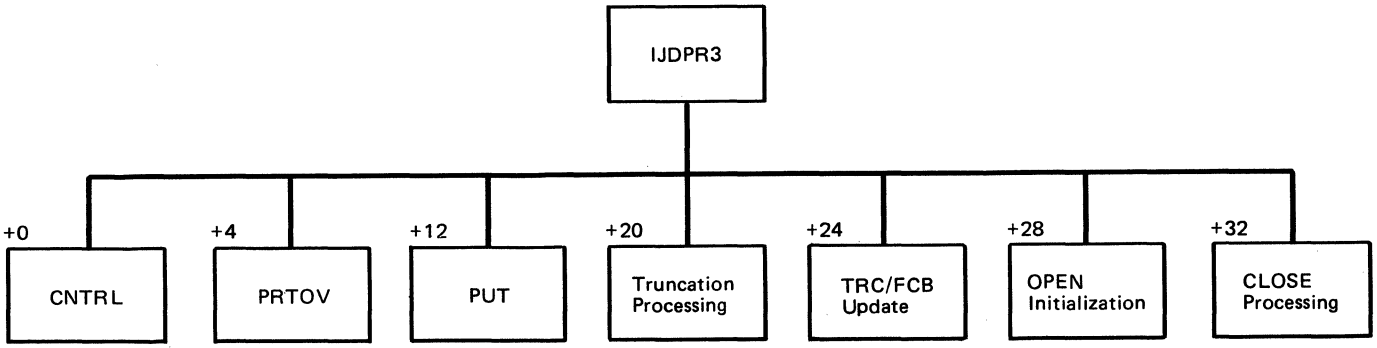


Figure 17. 380C Extended Buffering

Method: This rcutine provides independent forms position control of the 3800 printer. It is used only for a DTFPR 3800 file. The FOLD and UNFCLD operations are not supported for the 3800 printer.

The extended buffering CNTRL routine builds a control CCW in the 3800 extended buffer as part of a command-chained channel program. The control operation will not occur until either: (1) the extended buffer is subsequently fcund filled and the channel program therein is executed, or (2) the buffer undergoes truncation processing.

IJDPR3: PRTCV Macro, Chart XB

Objective: To cause and control an overflow skip for a 3800 printer file (DTFPR only) opened for extended buffering mode.

Entry: IJDPR3 + 4 from a PRTOV macro expansion.

Exits:

1. To the caller's overflow processing rcutine if the address of the routine is provided.

2. To the caller or cancel.

Method: This routine checks for a printer overflow condition that may have been detected during the immediately preceding PUT (or CNTRL) processing. The overflow condition is indicated in the DTFPR for which the PRTOV macrc was issued.

Because extended buffer processing builds a command-chained channel program during PUT cr CNTRL processing, an overflow condition must be initially detected at that time by predetermining whether a channel 9 or 12 would be encountered by a particular PUT or CNTRL. (Refer to the FCB tracking descripticn given in the method section under IJDPR3: PUT macro.)

When the PRTOV routine finds an overflow condition, a skip-to-channel 1 CCW is added to the extended buffer channel program. However, if the address of a user routine has been prcvided, a branch is made to that routine where any desired function except another PRTCV can be performed.

IJDPR3: PUT Macro, Chart XC

Objective: To provide output processing for 3800 printer files that are opened for extended buffering mcde. To provide table reference character (TRC) mode for DTFPR, DTFDI, and DTFCP 3800 printer files.

Entry: IJDPR3 + 12 from a PUT macro expansion.

Exits: To the caller or cancel.

Method: This routine is entered when a PUT macro is issued on a DTFPR, DTFDI, or DTFCP (type 31 or 32), that has been opened in extended buffering mode for a file assigned to a 3800 printer. Note that only a DTFPR can have a record format other than fixed unblocked.

Other existing support found in the PUT routine of the access method (SAM) logic modules exists in the PUT routine of IJDPR3; for example, control character support, multiple I/O area support and work area support.

The DTFB (extended buffer) gotten by OPEN is utilized to build a chain of CCWs to minimize the number of EXCPs to be issued. The DTFB contains two channel programs and data areas, so that a channel program built in one area can be executing while another channel program is being built in the other area by subsequent PUTs, CNTRL, or PRTOV macros. The channel program in each area consists of (1) CCWs built from the front-end and (2) related data blocks built from the back-end of the area. When the space between the CCWs and the data blocks has diminished so that another CCW-data block combination cannot fit into the area, an EXCP is issued on the channel program just built. The other channel program previously EXCPed is completed before being utilized to build another channel program. A separate CCB exists at the beginning of each of the two channel program areas.

When extended buffer processing is to occur with TRC mode, the appropriate select-translate-table CCWs will be built. When more than one DTF is concurrently opened to the same 3800 printer, extended buffering for that 3800 will still occur, even when PUTs on the different DTFs are intermixed. The lines of output will appear in the same order as the PUTs that were issued on the different DTFs. To accomplish this, the extended buffering channel control program in the common buffer, DTFB, will be built to include CCWs and data from PUTs issued on any of the DTFs concurrently opened to the same 3800 printer.

Tracking logic exists to follow the movement of line position down a form in support of printer overflow processing for the DTFPR file type.

Because extended buffer processing for a 3800 printer causes a series of command-chained CCWs to be built in the DTFB, it is necessary to predetermine whether a channel 9 or 12 printer overflow condition would be caused by each movement-causing operation code encountered when building each CCW. This is accomplished by matching line movement down a form with corresponding movement through the current FCB image, a copy of which exists in the DTFB.

Tracking occurs for all DTF types supported for extended buffer processing, even though printer overflow support exists only for the DTFPR file type. This must be done to handle multiple DTFs concurrently open and processing output to the same 3800. Even if only one of those DTFs is a DTFPR (for which printer overflow support is requested), any potential form movement caused for any DTF type must be tracked in order to maintain printer overflow support for the DTFPR.

FCB tracking is not necessary if: (1) a channel 9 or 12 does not exist in the current FCB image, or (2) the current CCW operation code does not cause form movement.

IJDPR3: Buffer Truncation Processing, Chart XD

Objective: To cause data currently in the 3800 extended buffer to be printed.

Entry: IJDPR3 + 20 from SETPRT, SEGMENT, and also the CLOSE function within IJDPR3 itself.

Exits: To the caller.

Method: The previously scheduled extended buffer's channel program is waited upon for completion. The current extended buffer's channel program (if any) is scheduled for I/O processing by an EXCP. Processing is completed before control is returned to the user.

IJDPR3: TRC/FCB Update Processing, Chart XE

Objective: To update TRC and FCB data maintained in the extended buffer.

Entry: IJDPR3 + 24 from SETPRT and also the OPEN function within IJDPR3 itself.

Exit: To the caller.

Method: The current TRC value maintained in the extended buffer is reset to zero (a non-TRC value). A new FCB image address may be passed to the routine as input, requiring initialization of that image in the extended buffer. The extended buffer is reformatted to accommodate the new FCB image.

IJDPR3: OPEN Processing, Chart XF

Objective: To perform OPEN initialization for 3800 extended buffer processing (runs in key zero).

Entry: IJDPR3 + 28 from OPEN B-transient phase $$BOPR3.

Exit: To $$BOPR3 at the next sequential instruction following the call to IJDPR3.

Method: Each 3800 that has extended buffering logic in effect requires a 4K GETVIS area in the partition for CCWs and data. Each DTF that is opened requires a 128-byte GETVIS area in the partition for a work area. An OPEN of a 3800 DTF when no other DTF is opened to the same 3800 device will obtain the 4K virtual storage area and the 128-byte virtual storage area from the user GETVIS area. Any subsequent OPEN of a 3800 DTF when at least one other DTF is opened to the same 3800 device will obtain only the 128-byte virtual storage area from the user GETVIS area.

The 4K area, called the 3800 DTF Extension Buffer (DTFB), is used by the new extended buffering logic module as a buffer for print lines for the user, thus minimizing the number of EXCPs issued to the 3800 printer. OPEN initializes the DTFB for the subsequent building of channel programs. A small portion of the DTFB is initialized with a copy of the current FCB image if necessary.

The 128-byte area, called the 3800 DTF Extension Work Area (DTFXWA), is used to save addresses, registers, DTF fields modified for extended buffer processing, and other values unique to each DTF. OPEN initializes the first word of the 3800 DTFXWA to point to the DTFB. Additionally, OPEN changes the DTF CCW address at bytes 9-11 (decimal) to point to the DTFXWA, and changes the DTF logic module address constant at bytes 17-19 to point to IJDPR3, the extended buffering logic module in the system virtual area (SVA). The original CCW and logic module addresses are saved in the DTFXWA.

OPEN also chains together the DTFXWAs related to each DTF opened within the partition to a 3800 pointer. The chain base is in the partition-related GETVIS area and is called the Anchor Table Extension (ATX).The ATX exists for each partition only as is necessary. Each subsequent chain pointer is placed in the related DTFXWA. While all DTFXWAs related to DTFs opened within the partition to a 3800 pointer will be on this single chain, only the DTFXWAs of those DTFs opened to the same 3800 printer will point to the same DTFB. This is done so that if multiple DTFs are opened to the same 3800, the lines will be intermixed in the order of issuance of PUT.

IJDPR3: CLOSE Processing, Chart XG

Objective: To perform CLOSE processing for 3800 printer files opened for extended buffering mode (runs in key zero).

Entry: IJDPR3 + 32 from CLOSE B-transient phase $$BCLCS2.

Exit: To $$BCLCS2 at the next sequential instruction following the call to IJDPR3.

Method: CLOSE will (1) print the partial performance buffer when necessary, by calling the buffer truncation function in IJDPR3; (2) reestablish the original DTF CCW and logic module addresses that were saved in the DTFXWA; (3) remove the DTFXWA from the ATX chain and issue FREEVIS for the 3800 DTFXWA obtained during OPEN; and (4) issue FREEVIS for the 3800 DTFB, if necessary. Item (1) is necessary only if a partially built channel program exists in the DTFB. Item (4) is necessary only if the DTFXWA related to the DTF being closed is the only one on the DTFXWA chain that points to the subject DTFB.

PAPER TAPE FILES

Paper tape (PT) files are input or output files associated with 2671/1017 paper tape readers or 1018 paper tape punches and defined by a DTFPT macro. A corresponding PTMOD macro supplies the data handling logic module which supports only fixed unblocked and undefined records.

Paper tape files are opened and closed by the Open and Close Monitors, respectively (refer to VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 1).

DTFPT Macro

The DTFPT macro is capable of generating several different DTF tables depending on the parameters specified in the macro operand. The ten basic table types are:

- DTFPT: No Translation, Shifts or Deletes, Device=2671 (see Figure 18)
- DTFPT: Translation, No Shifts or Deletes, Device=2671 (see Figure 19)
- DTFPT: Translation, Shifts, and Deletes with Fixed Unblocked Records, Device=2671 (see Figure 20)
- DTFPT: Translation, Shifts and Deletes with Undefined Records, Device=2671 (see Figure 21)
- DTFPT: No Translation, Shifts or Deletes, Device=1017 (see Figure 22)
- DTFPT: Translation, No Shifts or Deletes, Device=1017 (see Figure 23)
- DTFPT: Translation, Shifts and Deletes, Fixed Unblocked Records, Device=1017 (see Figure 24)
- DTFPT: Translation, Shifts and Deletes, Undefined Records, Device=1017 (see Figure 25)
- DTFPT: No Shifts, Device=1018 (see Figure 26)
- DTFPT: Shifts, Device=1018 (see Figure 27)

Translation refers to code translation of paper tapes punched in other than EBCDIC code. Shifts refers to letter or figure shift required for 5-channel paper tape. Deletes refers to acceptance of delete characters punched in the tape.

Bytes*	Contents	Function
0-15 (00-0F)		CCB (X'88' in byte 2).
16 (10)	X'20' X'08'	COBOL open; ignore option. OPENR relocates DTF address constants.
17-19 (11-13)		Address of logic module.
20 (14)	X'01'	DTF type.
21 (15)		Bit 0 OPEN indicator. 1 Two I/O areas. 2 EOF indicator. 3 Read error. 4-6 Not used. 7 Undefined record.
22-25 (16-19)	MVI 26(1),X'07' NOP 0(0)	NOP first SVC 0 if two areas. If one area.
26-27 (1A-1B)	SVC 0	Read a record.
28-29 (1C-1D)	SVC 0 NOPR 0	Read another record if two areas. If one area.
30-31 (1E-1F)	LR &RECSIZE,14 NOPR 0	Put record length in user's register. No RECSIZE entry.

32-35
(20-23)
	A(&IOAREA2)
	A(&IOAREA1)
		If two areas.
		If one area.

36-39 (24-27)	A(&EOFADDR)	End-of-file address.

40-43
(28-2B)
	A(&ERROPT)
	SR 0,0
	SVC 6
	B 12(15)
	B 138(15)
		Addr. of user's error routine if ERROPT=name.
		ERROPT omitted.
		
		ERROPT=SKIP.
		ERROPT=IGNORE.

44-47 (2C-2F)	A(&WLRERR) B 12(15) B 152(15) B 152(15)	Address of user's WLR routine if WLRERR=name. WLRERR omitted and ERROPT=SKIP. WLRERR and ERROPT both omitted, or WLRERR omitted and ERROPT=IGNORE. RECFORM=FIXUNB or omitted.
48-55 (30-37)	X'02',&IOAREA1,X'00',&BLKSIZE X'06',&IOAREA1,X'00',&BLKSIZE	CCW: if RECFORM=FIXUNB or omitted. CCW: if RECFORM=UNDEF.
56-63 (38-3F)		Duplicate CCW.
64-67 (40-43)	F'0'	Save area for register 14.
68-71 (44-47)	L &IOREG,48(1) NOP 0(0)	Put input area address into user's register. No IOREG entry.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 18. DTFPT: No Translations, no Shifts or Deletes; Device=2671

Bytes*	Contents	Function
0-15 (00-0F)		CCB (X'88' in byte 2).
16 (10)	X'20' X'08'	COBOL open; ignore option. OPENR relocates DTF address constants.
17-19 (11-13)		Address of logic module.
20 (14)	X'01'	DTF type.
21 (15)		Bit 0 OPEN indicator. 1 Two I/O areas. 2 EOF indicator. 3 Read error. 4-6 Not used. 7 Undefined record.
22-25 (16-19)	MVI 26(1),X'07' NOP 0(0)	NOP first SVC 0 if two areas. If one area.
26-27 (1A-1B)	SVC 0	Read a record.
28-29 (1C-1D)	SVC 0 NOPR 0	Read another record if two areas. If one area.
30-31 (1E-1F)	LR &RECSIZE,14 NOPR 0	Put record length in user's register. No RECSIZE entry.

32-35
(20-23)
	A(&IOAREA2)
	A(&IOAREA1)
		If two areas.
		If one area.

36-39 (24-27)	A(&EOFADDR)	End-of-file address.

40-43
(28-2B)
	A(&ERROPT)
	SR 0,0
	SVC 6
	B 12(15)
	B 138(15)
		Addr. of user's error routine if ERROPT=name.
		ERROPT omitted.
		
		ERROPT=SKIP.
		ERROPT=IGNORE.

44-47 (2C-2F)	A(&WLRERR) B 12(15) B 182(15)	Addr. of user's WLR routine if WLRERR=name. WLRERR omitted and ERROPT=SKIP or IGNORE; or WLRERR and ERROPT both omitted. RECFORM=FIXUNB or omitted.
48-55 (30-37)	X'02',&IOAREA1,X'00',&BLKSIZE X'06',&IOAREA1,X'00',&BLKSIZE	CCW: if RECFORM=FIXUNB or omitted. if RECFORM=UNDEF.
56-63 (38-3F)		Duplicate CCW.
64-67 (40-43)	A(&TRANS)	Address of user's translate table.
68-71 (44-47)	L &IOREG,32(1) NOP 0(0)	Put input area address into user's register. No RECSIZE entry.
72-75 (48-4B)	LH &RECSIZE,30(1) NOP 0(0)	Put record length into user's register. No IOREG entry.

Figure 19. DTFPT: Translation, no Shifts or Deletes; Device=2671

Bytes*	Contents	Function
0-15 (00-0F)		CCB (X'80' in byte 2).
16 (10)	X'20' X'08'	COBOL open; ignore option DTF table address constants relocated by OPENR.
17-19 (11-13)		Address of logic module.
20 (14)	X'01'	DTF type.
21 (15)		Bit 0 OPEN indicator. 1 Two I/O areas. 2 EOF indicator. 3-5 Not used. 6 Scanning. 7 Not used.
22-25 (16-19)	MVI 26(1),X'07' NOP 0(0)	NOP first SVC 0 if two areas. If one area.
26-27 (1A-1B)	SVC 0	Read a record.
28-29 (1C-1D)	SVC 0 NOPR 0	Read another record if two areas. If one area.
30-31 (1E-1F)	H'0'	Record length field.
32 (20)	C'02'	

33-35
(21-23)
	AL3(&ICAREA2)
	AL3(&ICAREA1)
		If two areas.
		If one area.

36-39 (24-27)	A(&EOFADDR)	End-of-file routine address.
40-43 (28-2B)	A(0)	
44-47 (2C-2F)	NOP 0(0)	
48-55 (30-37)	X'02',&IOAREA1,X'00',&OVBLKSZ X'02',&IOAREA1,X'00',&BLKSIZE	CCW: if OVBLKSZ specified. if OVBLKSZ not specified.

Figure 20. DTFPT: Translation, Shifts and Deletes, with Fixed Unblocked Records; Device=2671 (Part 1 of 2)

Bytes*	Contents	Function
56-59 (38-3B)	L &IOREG,96(1) NOP 0(0)	Put input area address into user's register. No IOREG entry.
60-63 (3C-3F)	LH &RECSIZE,30(1) NOP 0(0)	Put record length into user's register. No RECSIZE entry.
64 (40)	X'02'	
65-67 (41-43)	AL3(&IOAREA1)	Address of current input area.
68-71 (44-47)	A(0)	Address of remainder in input area.
72-75 (48-4B)	F'0'	Length of remainder.
76-79 (4C-4F)	A(&FTRANS)	Address of figure shift table.
80-83 (50-53)	A(&LTRANS)	Address of letter shift table.

84-87
(54-57)
	A(&TRANS)
	A(&FTRANS)
	A(0)
		If LTRANS and FTRANS not specified.
		Address of current translate table.
		If no translation is specified.

88-91 (58-5B)	A(&SCAN)	Address of scan table.
92-95 (5C-5F)	F'&BLKSIZE'	Required record length.
96-103 (60-67)		Duplicate CCW.
104-109 (68-6D)	TR 0(0,4),0(5) TM 0,0 NOPR 0	If TRANS, LTRANS or FTRANS specified. If no translation specified.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 20. DTFPT: Translation, Shifts and Deletes, with Fixed Unblocked Records; Device=2671 (Part 2 of 2)

Bytes*	Contents	Function
0-15 (00-0F)		CCB (X'88' in byte 2).
16 (10)	X'20' X'08'	COBOL open; ignore option. DTF table address constants relocated by OPENR.
17-19 (11-13)		Address of logic module.
20 (14)	X'01'	DTF type.
21 (15)		Bit 0 OPEN indicator. 1 Two I/O areas. 2 EOF indicator. 3-5 Not used. 6 Scanning. 7 Read error.
22-25 (16-19)	MVI 26(1),X'07' NOP 0(0)	NOP first SVC 0 if two areas. If one area.
26-27 (1A-1B)	SVC 0	Read a record.
28-29 (1C-1D)	SVC 0 NOPR 0	Read another record if two areas. If one area.
30-31 (1E-1F)	H'0'	Record length field.

32-35
(20-23)
	A(&IOAREA2)
	A(&IOAREA1)
		If two areas.
		If one area.

36-39 (24-27)	A(&EOFADDR)	End-of-file routine address.

40-43
(28-2B)
	A(&ERROPT)
	SR 0,0
	SVC 6
	B 16(15)
	B 246(15)
		Address of user's routine if ERROPT=name.
		ERROPT omitted.
		
		ERROPT=SKIP.
		ERROPT=IGNORE.

44-47 (2C-2F)	A(&WLRERR) B 16(15) B 260(15)	Address of user's WLR routine if WLRERR=name. WLRERR omitted and ERROPT=SKIP. WLRERR omitted and ERROPT=IGNORE, or both omitted.

Figure 21. DTFPT: Translation, Shifts and Deletes, with Undefined Records; Device=2671 (Part 1 of 2)

Bytes*	Contents	Function
48-51 (30-33)	L &IOREG,32(1) NOP 0(0)	Put input area address into user's register. No IOREG entry.
52-55 (34-37)	LH &RECSIZE,30(1) NOP 0(0)	Put record length into user's register. No IOREG entry.
56-59 (38-3B)	A(&FTRANS)	Address of figure shift table.
60-63 (3C-3F)	A(LTRANS)	Address of letter shift table.

64-67
(40-43)
	A(&FTRANS)
	A(&TRANS)
	A(0)
		Address of current translate table.
		If LTRANS and FTRANS not specified.
		If no translation specified.

68-71 (44-47)	A(&SCAN)	Address of scan table.
72-79 (48-4F)		Duplicate CCW.
80-87 (50-57)	X'06',&IOAREA1,X'00',&BLKSIZE	CCW.
88-93 (58-5D)	TR 0(0,4),0(5) TM 0,0 NOPR 0	If TRANS, LTRANS or FTRANS specified. If no translation specified.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 21. DTFPT: Translation, Shifts and Deletes, with Undefined Records; Device=2671 (Part 2 of 2)

Bytes*	Contents	Function
0-15 (00-0F)		CCB (X'88' in byte 2).
16 (10)	X'08'	Indicates DTF table relocated by OPENR.
17-19 (11-13)		Address of logic module.
20 (14)	X'01'	DTF type.
21 (15)		Bit 0 CPEN indicator. 1 Two I/O areas. 2 EOF indicator. 3 Read error. 4-6 Not used. 7 Undefined record.
22-25 (16-19)	MVI 26(1),X'07' NCP 0(0)	NOP first SVC 0 if two areas. If one area.
26-27 (1A-1B)	SVC 0	Read a record.
28-29 (1C-1D)	SVC 0 NOPR 0	Read another record if two areas. If one area.
30-31 (1E-1F)	LR &RECSIZE,14 NOPR 0	Put record length into user's register. No RECSIZE entry.

32-35
(20-23)
	A(&IOAREA2)
	A(&IOAREA1)
		If two areas.
		If one area.

36-39 (24-27)	A(&EOFADDR) SVC 50 H'0'	End-of-file address. No EOFADDR operand.

40-43
(28-2B)
	A(&ERRCPT)
	SR 0,0
	SVC 6
	B 0(14)
	B 4(14)
		Addr. of user's error routine if ERROPT=name.
		ERROPT omitted.
		
		ERROPT=SKIP
		ERROPT=IGNORE

44-47 (2C-2F)	A(&WLRERR) A(&ERROPT) B 0(14) B 8(14) B 8(14)	Address of user's WLR routine if WLRERR=name. WLRERR absent and ERROPT=name. WLRERR absent and ERROPT=SKIP. WLRERR and ERROPT both absent or WLRERR absent and ERROPT=IGNORE. RECFORM=FIXUNB or absent.
48-55 (30-37)	X'02',&IOAREA1,X'00',&BLKSIZE X'06',&IOAREA1,X'00',&BLKSIZE	CCW: if RECFORM=FIXUNB or absent. CCW: if RECFORM=UNDEF.
56-63 (38-3F)		Duplicate CCW.
64-67 (40-43)	F'0'	Save area for register 14.
68-71 (44-47)	L &IOREG,48(1) NOP 0(0)	Put input area address into user's register. No IOREG entry.

Figure 22. DTFPT: No Translation, no Shifts or Deletes; Device=1017

Bytes*	Contents	Function
0-15 (00-0F)		CCB (X'88' in byte 2).
16 (10)	X'08'	Indicates DTF table relocated by OPENR.
17-19 (11-13)		Address of logic module.
20 (14)	X'01'	DTF type.
21 (15)		Bit 0 OPEN indicator. 1 Two I/O areas. 2 EOF indicator. 3 Read error. 4-6 Not used. 7 Undefined record.
22-25 (16-19)	MVI 26(1),X'07' NOP 0(0)	NOP first SVC 0 if two areas. If one area.
26-27 (1A-1B)	SVC 0	Read a record.
28-29 (1C-1D)	SVC 0 NOPR 0	Read another record if two areas. If one area.
30-31 (1E-1F)	H'0'	Record length field.
32 (20)	X'00'	
33-35 (21-23)	AL3(&IOAREA1)	One input area.
36-39 (24-27)	A(&EOFADDR) SVC 50 H'0'	End-of-file address. No EOFADDR operand.

40-43
(28-2B)
	A(&ERROPT)
	SR 0,0
	SVC 6
	B 0(14)
	B 4(14)
		Addr. of user's error routine if ERROPT=name.
		ERROPT omitted.
		
		ERROPT=SKIP.
		ERROPT=IGNORE.

44-47 (2C-2F)	A(&WLRERR) A(&ERROPT) B 0(14) B 8(14) B 8(14)	Address of user's WLR routine if WLRERR=name. WLRERR absent and ERROPT=name. WLRERR absent and ERROPT=SKIP. WLRERR and ERROPT both absent or WLRERR absent and ERROPT=IGNORE. RECFORM=FIXUNB or absent.
48-55 (30-37)	X'02',&IOAREA1,X'00',&BLKSIZE X'06',&IOAREA1,X'00',&BLKSIZE	CCW: if RECFORM=FIXUNB or absent. CCW: if RECFORM=UNDEF.
56-63 (38-3F)		Duplicate CCW.
64-67 (40-43)	A(&TRANS)	Address of user's translate table.
68-71 (44-47)	L &IOREG,32(1) NOP 0(0)	Put input area address into user's register. No IOREG operand.
72-75 (48-4B)	LH &RECSIZE,30(1) NOP 0(0)	Put record length into user's register. No RECSIZE operand.

Figure 23. DTFPT: Translation, no Shifts or Deletes; Device=1017

Bytes*	Contents	Function
0-15 (00-0F)		CCB (X'80' in byte 2).
16 (10)	X'08'	Indicates DTF table relocated by OPENR.
17-19 (11-13)		Address of logic module.
20 (14)	X'01'	DTF type.
21 (15)		Bit 0 OPEN indicator. 1 Two I/O areas. 2 EOF indicator. 3-5 Not used. 6 Scanning. 7 Not used.
22-25 (16-19)	MVI 26(1),X'07' NCP 0(0)	NOP first SVC 0 if two areas. If one area.
26-27 (1A-1E)	SVC 0	Read a record.
28-29 (1C-1D)	SVC 0 NOPR 0	Read another record if two areas. If one area.
30-31 (1E-1F)	H'0'	Record length field.
32 (20)	C'02'	

33-35
(21-23)
	AL3(&IOAREA2)
	AI3(&ICAREA1)
		If two areas.
		If one area.

36-39 (24-27)	A(&ECFADCR) SVC 50 H'0'	End-of-file address. No EOFADDR operand.
40-43 (28-2B)	A(0)	
44-47 (2C-2F)	NOP 0(0)	
48-55 (30-37)	X'02',&IOAREA1,X'00',&OVBLKSZ X'02',&IOAREA1,X'00',&BLKSIZE	CCW: if CVBLKSZ specified. if OVBLKSZ not specified.
56-59 (38-3E)	L &ICFEG,96(1) NCP 0(0)	Put input area address into user's register. No IOREG cperand.

Figure 24. DTFPT: Translation, Shifts and Deletes, Fixed Unblocked Records; Device= 1017 (Part 1 of 2)

Bytes*	Contents	Function
60-63 (3C-3F)	LH &RECSIZE,30(1) NOP 0(0)	Put record length into user's register. No RECSIZE operand.
64 (40)	X'02'	
65-67 (41-43)	AL3(&IOAREA1)	Address of current input area.
68-71 (44-47)	A(0)	Address of remainder in input area.
72-75 (48-4B)	F'0'	Length of remainder.

76-79
(4C-4F)
	A(&FTRANS)
	A(0)
		Address of figure shift table.
		No FTRANS operand.

80-83
(50-53)
	A(&LTRANS)
	A(0)
		Address of letter shift table.
		No LTRANS operand.

84-87
(54-57)
	A(&FTRANS)
	A(&TRANS)
	A(0)
		Address of current translate table.
		If FTRANS not specified.
		If no translation is specified.

88-91 (58-5B)	A(&SCAN)	Address of scan table.
92-95 (5C-5F)	F'&BLKSIZE'	Required record length.
96-103 (60-67)		Duplicate CCW.
104-109 (68-6D)	TR 0(0,4),0(5) TM 0,0 NOPR 0	If TRANS, LTRANS or FTRANS specified. If no translation specified.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 24. DTFPT: Translation, Shifts and Deletes, Fixed Unblocked Records; Device= 1017 (Part 2 of 2)

Bytes*	Contents	Function
0-15 (00-0F)		CCB (X'88' in byte 2).
16 (10)	X'08'	Indicates DTF table relocated by OPENR.
17-19 (11-13)		Address of logic module.
20 (14)	X'01'	DTF type.
21 (15)		Bit 0 OPEN indicator. 1 Two I/O areas. 2 EOF indicator. 3-5 Not used. 6 Scanning. 7 Read error.
22-25 (16-19)	MVI 26(1),X'07' NOP 0(0)	NOP first SVC 0 if two areas. If one area.
26-27 (1A-1B)	SVC 0	Read a record.
28-29 (1C-1D)	SVC 0 NOPR 0	Read another record if two areas. If one area.
30-31 (1E-1F)	H'0'	Record length field.

32-35
(20-23)
	A(&IOAREA2)
	A(&IOAREA1)
		If two areas.
		If one area.

36-39 (24-27)	A(&EOFADDR) SVC 50 H'0'	End-of-file address. No EOFADDR operand.
40-43 (28-2B)	A(&ERROPT)	Address of user's error routine if ERROPT=name.

	SR 0,0
	SVC 6
	B 0(14)
	B 4(14)
		ERROPT omitted.
		
		ERROPT=SKIP
		ERROPT=IGNORE

44-47
(2C-2F)
	A(&WLRERR)
	A(&ERROPT)
	B 0(14)
	B 8(14)
		Address of user's WLR routine if WLRERR=name
		WLRERR absent and ERROPT=name.
		WLRERR absent and ERROPT=SKIP.
		WLRERR and ERROPT both absent or WLRERR absent and ERROPT=IGNORE.

Figure 25. DTFPT: Translation, Shifts and Deletes, Undefined Records; Device=1017 (Part 1 of 2)

Bytes*	Contents	Function
48-51 (30-33)	L &IOREG,32(1) NOP 0(0)	Put input area address into user's register. No IOREG operand.
52-55 (34-37)	LH &RECSIZE,30(1) NOP 0(0)	Put record length into user's register. No IOREG operand.

56-59
(38-3B)
	A(&FTRANS)
	A(0)
		Address of figure shift table.
		No FTRANS operand.

60-63
(3C-3F)
	A(&LTRANS)
	A(0)
		Address of letter shift table.
		No LTRANS operand.

64-67
(40-43)
	A(&FTRANS)
	A(&TRANS)
	A(0)
		Address of current translate table.
		If FTRANS not specified.
		If no translation specified.

68-71 (44-47)	A(&SCAN)	Address of scan table.
72-79 (48-4F)		Duplicate CCW.
80-87 (50-57)	X'06',&IOAREA1,X'00',&BLKSIZE	CCW.
88-93 (58-5D)	TR 0(0,4),0(5) TM 0,0 NOPR 0	If TRANS, LTRANS or FTRANS specified. If no translation specified.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 25. DTFPT: Translation, Shifts and Deletes, Undefined Records; Device=1017 (Part 2 of 2)

Bytes*	Contents	Function
0-15 (00-0F)		CCB (X'8A' in byte 2).
16 (10)	X'08'	Indicates DTF table relocated by OPENR.
17-19 (11-13)		Address of logic module.
20 (14)	X'01'	DTF type.
21 (15)		Bit 0 OPEN indicator. 1 Two I/O areas. 2 Write error. 3 CLOSE indicator. 4-5 Retry counter. 6 Not used. 7 Undefined record.
22-25 (16-19)	L &IOREG,48(1) NOP 0(0)	Put output area address into user's register. No IOREG operand.
26 (1A)	X'&DELCHAR' X'00'	Delete character. No delete character specified.
27 (1B)		Not used.

28-31
(1C-1F)
	A(&IOAREA2)
	A(&IOAREA1)
		If two areas.
		If one area.

32-47 (20-2F)	X'01',&IOAREA1,X'00',&BLKSIZE D'0' X'01',&IOAREA1,X'00',&BLKSIZE X'01',68,X'00',01	CCW if RECFORM=FIXUNB. CCW if RECFORM=UNDEF.
48-55 (30-37)		Duplicates CCW.

56-59
(38-3B)
	A(&TRANS)
	A(0)
		Address of user's translate table.
		If no translation specified.

60-63
(3C-3F)
	A(&ERROPT)
	
	SR 0,0
	SVC 6
	B 0(14)
		Address of user's error routine if
		ERROPT=none.
		ERROPT absent.
		
		ERROPT=IGNORE.

64-67 (40-43)	STH &RECSIZE,54(1) NOP 0(0)	Save user's record length. No RECSIZE operand.
68 (44)	X'&EORCHAR' X'00'	End of record character. No EORCHAR operand or RECFORM=FIXUNB.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 26. DTFPT: No Shifts, Device=1018

Bytes*	Contents	Function
0-15 (00-0F)		CCB (X'82' in byte 2).
16 (10)	X'08'	Indicates DTF table relocated by OPENR.
17-19 (11-13)		Address of logic module.
20 (14)	X'01'	DTF type.
21 (15)		Bit 0 OPEN indicator. 1 Two I/O areas. 2 Not used. 3 CLOSE indicator. 4-5 Retry counter. 6 Shifted. 7 Undefined record.
22-25 (16-19)	L &IOREG,56(1) NOP 0(0)	Put output area address into user's register. No IOREG operand.
26 (1A)	X'&DELCHAR' X'00'	Delete character. No delete character specified.
27 (1B)		Reserved for first shift character.

28-31
(1C-1F)
	A(&IOAREA2)
	A(&IOAREA1)
		If two areas.
		If one area.

32-55 (20-37)	X'01',27,X'80',01 X'01',&IOAREA1,X'00',&BLKSIZE D'0'	CCWs if RECFORM=FIXUNB.
	X'01',27,X'80',01 X'01',&IOAREA1,X'80',&BLKSIZE X'01',82,X'00',01	CCWs if RECFORM=UNDEF.
56-63 (38-3F)		Duplicate CCW.

64-67
(40-43)
	A(&TRANS)
	A(0)
		Address of user's translate table.
		If no translation specified.

68-71 (44-47)	A(&FSCAN)	Address of user's figure shift status table.
72-75 (48-4B)	A(&LSCAN)	Address of user's letter shift status table.
76-79 (4C-4F)	STH &RECSIZE,82(1) NOP 0(0)	Save user's record length. No RECSIZE operand.
80-81 (50-51)	H'&OVBLKSZ' H'&BLKSIZE'	If OVBLKSZ specified. If OVBLKSZ not specified, or RECFORM=UNDEF.
82 (52)	X'&EORCHAR' X'00'	End of record character. No EORCHAR operand.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 27. DTFPT: Shifts, Device=1018

PTMOD Macro

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

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

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

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

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

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

Entry: From a GFT macro expansion.

Exit: Tc the prcblem program, or to the user's EOF routine.

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

Statement	One Input Area	Two Input Areas
IJE1MCVE	NOP 0(0)	MVI IJE1SVC1,X'07' (NOP first SVC 0 instruction)
IJE1SVC1	SVC 0	SVC 0 (first)
IJE1SVC2	NOPR 0	SVC 0 (second)

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

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

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

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

PTMOD: GET Macro, Translation, Shifted Code, Fixed-Unblocked Records, Device=2671, Chart CK

The basic module is the same as the GET with Translation of Shifted Code except for the method of obtaining the correct number of bytes to complete the fixed length record. Because the number of control characters throughout the tape record is unpredictable, a routine is included to ensure that the correct number of valid characters is included. The user specifies the number of characters to be read in (OVBLKSZ) to produce a specified number of bytes (BLKSIZE) after the control characters have been deleted.

For example, a group of 70 punched tape characters (OVBLKSZ) are to be read to produce a fixed length record of 50 characters (BLKSIZE). However, because excessive delete characters are encountered, the record contains only 45 characters. To reduce the number of physical I/O operations, a second group of 25 punched tape characters is read to ensure that the 5 characters needed to fill the original record are obtained. On the next GET, the remaining characters (if any) from the second group of punched tape characters are moved into the leftmost positions of the input area. The byte count and the data address in the CCW are modified accordingly before the next SVC 0 is issued.

PTMOD: GET Macro, Translation, Shifted Code, Undefined Records, Device=2671, Chart CL

This GET routine for undefined records functions in much the same manner as the GET routine for fixed length records. The routine differs in the use of two translate tables; one table is required for letter shift, and the other for figure shift.

Shift characters and delete characters contained in the punched paper tape are removed from the record as translation takes place. The record in main storage is thereby compressed and left-justified.

GET LOGIC FOR THE 1017 PAPER TAPE READER -- BASIC PRINCIPLES

The DTF table contains three key statements for the four basic logic modules generated by the macro instruction PTMOD for the GET logic. The three statements are:

Statement	One Input Area	Two Input Areas
IJEMOVE	NOP 0(0)	MVI IJESVC1,X'07'
IJESVC1	SVC 0	SVC 0
IJESVC2	NOPR 0	SVC 0

First Macro Instruction GET: The first GET macro instruction requires that the two input areas be filled. At first, the statement IJESVC1 requires SVC 0. Then the statement IJEMOVE changes IJESVC1 to NOP 0 if two input areas are specified.

Logic for One Input Area: The following steps are necessary when creating the GET logic for one input area:

1. Execute IJESVC1 (read a record).

2. Wait.

3. Process record (all translating and editing).

4. Change input areas. (If only one input area has been specified, input areas are not changed because the same name is generated in both areas of the DTF table.)

5. Execute IJEMOVE (NOP instruction).

6. Execute IJESVC2 (NOP instruction).

7. Return to user's program.

Logic for Two Input Areas: The following steps are necessary when creating the GET logic for two input areas:

1. Execute IJESVC1 (NOP instruction, except the first time a record is read).

2. Wait for read operation of the preceding GET macro instruction.

3. Restore IJESVC1 (in case of end of file for possible reopening).

4. Process record.

5. Change input areas.

6. Execute IJEMOVE. This statement changes IJESVC1 into a NOP instruction.

7. Execute IJESVC2 (read a record).

8. Return to user's program.

Error Checking: The same standard error checking routines are provided for one or two input areas.

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

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

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

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

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

Entry: From the GET macro expansion.

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

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

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

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

Entry: From the GET macro expansion.

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

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

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

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

Entry: From the GET macro expansion.

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

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

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

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

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

Entry: From the GET macro expansion.

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

Method: Only the translating and editing functions are considered. The logic is described under "Basic Principles". Editing is done as follows:

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

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

3. The preceding segment is translated by means of the current translate tables LTRANS or FTRANS, depending on the current shift status. Then, the translated segment is moved by one character for left-justification to delete the shift or delete character found.

4. If the remaining record length is not zero, the procedure is resumed at step 1. If the remaining record length is zero, the new record length is computed and indicated to the user's program.

PUT LOGIC FOR THE 1018 PAPER TAPE PUNCH -- BASIC PRINCIPLES

For the PUT logic, the PTMOD macro instruction generates only the two following logic modules:

- PUT without the shift code procedure.

- PUT with the shift code procedure.

The same logic module performs the different functions by means of the switch byte in the DTFPT table, which contains all the information required by the PTMOD logic module.

The bit positions of the switch byte and the corresponding functions are:

Bit	Function
0	OPEN indicator
1	0 = 1 IOAREA, 1 = 2 IOAREAS
2	Reserved (write error indicator)
3	CLOSE indicator for logical ERP transient
4	Used as retry counter
5	Used as retry counter
6	0 = no shifted codes, 1 = shifted codes
7	Record format: 0 = FIXUNB, 1 = UNDEF

PTMOD: PUT Macro, No Shifted Code, Device=1018, Chart CQ

Objective: To perform the translate function where required and to punch the record on the paper tape punch.

Entry: From the PUT macro expansion.

Exit: To the user's program one instruction after the PUT macro instruction (normal return), or to the user's error routine.

Method: The logic for one output area requires the following steps:

1. Translate the record, if the translate table address in the DTFPT table is not zero.

2. Punch the output area (write operation).

3. Wait.

4. Check the operation. On data check, the logical transient $$BERPTP is called. If it detects a permanent write error in the CCB, the logic:

    - punches the remainder of the record, because the punch has stopped on the erroneous character.

    - gives control to the user's error routine, if any, or takes the error option.

5. Return to the user's program.

The logic for two output areas requires the following steps:

1. Translate the record if the translate table address in the DTFPT table is not zero.

2. Wait until the end of the preceding macro instruction.

3. Check this operation.

4. Punch the output area.

5. Return to the user's program.

PTMOD: PUT Macro, Shifted Code, Device=1018, Chart CR

Objective: To perform the translating and editing functions and to punch the record on the paper tape punch.

Entry: From the PUT macro expansion.

Exit: To the user's program one instruction after the PUT macro expansion.

Method: Only the translating or editing function is considered here, because the remaining logic is described under "PUT Macro, No Shifted Code."

An unrecoverable I/O error cannot be ignored (X'80' in the communication byte of the CCB). If it is ignored, an unrecoverable I/O error on a shift character may occur. The method is mainly based on a channel program having three CCWs.

CCW0 Write at beginning of record the shift character depending on the first character of the record.

CCW1 Write the whole record.

CCW2 Write the EOR character in the UNDEF record format.

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

The following steps describe the logic of the editing function:

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

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

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

4. Test to make sure that the end of the record does not overstep the bounds of

the output area (BLKSIZE or OVBLKSZ). If space is left, the remainder of the record is moved by one character for right-justification tc include the new shift character. Then, the procedure is resumed at step 2. If no space is left, the segment considered is punched. The remainder of the record is moved to the beginning of the output area, for left-justification. Then, the procedure is resumed at step 2.

Note that OVBLKSZ is ignored for undefined records. For further information on OVBLKSZ, refer to "Paper Tape File (DTFPT)" in VSE/Advanced Functions Macro Reference.

MAGNETIC TAPE FILES

Magnetic tape (MT) files are defined for logical IOCS by a DTFMT macro. Files so defined can be either input or output data files, or work files. The function of a particular file is determined, by the user, in the TYPEFIE= parameter of the DTFMT macrc.

Magnetic tape files can also be defined for physical ICCS if the user intends to use physical ICCS macros such as EXCP, WAII, etc. These files are defined by a DTFPH macro.

In addition, magnetic tape system files can be defined by the device independent macros, DTFDI and DTFCP. These files are described under "Device Independent Files" later in this manual.

The data handling logic modules for files defined fcr logical IOCS by the DTFMT macro are provided by the associated MTMOD macrc.

Files defined fcr logical IOCS by the DTFDI macro can be EBCDIC files only. Files defined fcr logical IOCS by DTFMT, DTFPH, or DTFCP can be either EBCDIC or ASCII (The American National Standard Code for Information Interchange) files.

Magnetic tape files are opened and closed by logical transient routines fetched by the Open and Close monitors respectively (refer to VSE/Advanced Functions Diagncsis Reference: LIOCS Volume 1). The open rcutine provide procedures for checking each file before any records are processed. The close routines provide procedures tc terminate the use of each file after all records are processed.

DTFMT MACRO

DATA FILES

The DTFMT macro used to define magnetic tape data files can generate four basic DTF tables, one for each of the following record types:

- Fixed length
- Variable length
- Undefined
- Spanned.

The four basic DTF tables have been combined into a single DTFMT table in Figure 28 of this publication. The values included in the table for a specified record type are noted.

WORK FILES

Figure 29 illustrates the DTF table that is generated when TYPEFLE=WORK, RECFORM=SPNBLK, or RECFORM=SPNUNB is specified by the user in the operand of a DTFMT macro. ASCII cannot be specified for work files.

DTFPH MACRO (MAGNETIC TAPE)

Figure 30 is generated by the DTFPH macro when the parameter DEVICE=TAPE is specified in the macro operand. The table contains the information necessary to define a magnetic tape file for physical IOCS.

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

Figure 28. DTFMT: Data Files (Part 1 of 8)

Bytes*	Bits	Contents[1]	Function	Record[2] Format
31 (1F)	0-4		Bits 0-4 are used as displacements by OPEN to determine the location of variable fields of the DTF.	
		B'01110'	Input.	F
		B'01100'	Output.	F
		B'10001'	Input.	V
		B'01111'	Output.	V
		B'01101'	Input.	U
		B'01011'	Output.	U
	5		1 = Tape label information included in DTF (see bytes 88-95). 0 = Tape label information not included in DTF.	
	6		Used by COBOL.	
	7		1 = Header label and EOV information wanted. 0 = No header label and EOV information wanted.	
32 (20)	0		Standard labels: 1 = yes, 0 = no.	
	1		Labels: 1 = nonstandard, 0 = unlabeled.	
	2		Rewind unload: 1 = yes, 0 = no.	
	3		Rewind option: 1 = no rewind, 0 = rewind.	
	4		Drive direction: 1 = backwards, 0 = forward.	
	5		User label address: 1 = yes, 0 = no.	
	6		Tapemark option: 1 = no, 0 = yes.	
	7		EOF-EOV switch (used by IBM SORT): 1 = yes, 0 = no.	
33-35 (21-23)			User label routine address.	
36 (24)	0		DTFPH: 1 = yes, 0 = no.	
	1		COBOL indicator: 1 = yes, 0 = no.	
	2		File type: 1 = input, 0 = output.	
	3		FEOV switch: 1 = yes, 0 = no.	
	4		EOF-EOV switch (output): 1 = EOF, 0 = EOV.	
	5		Open indicator: 1 = open, 0 = closed.	
	6		1 = variable or spanned records.	V,S
	7		1 = undefined records.	U

Figure 28. DTFMT: Data Files (Part 2 of 8)

Bytes*	Bits	Contents[1]	Function	Record[2] Format
37-39 (25-27)			EOF address.	
40-43 (28-2B)			Block count.	
44-47[1] (2C-2F)		BXH 11,12,24(15)	Forward.	F
		BXLE 11,12,24(15)	Backward.	F
		L &VARBLD,DEBLOCKER	If VARBLD parameter is used.	V
		NOP 0(0)		S
		DC F'0'	DEBLOCKER1.	U
48-51 (30-33)		LA 14,1(14)		F,V,S
		BCTR 14,0+NOPRO	Backward.	F,V,S
		L &RECSIZE,DEBLOCKER1	If RECSIZE given.	U
		NOP 0(0)	For input if not NOP.	U
52-55 (34-37)		L &IOREG,DEBLOCKER1	If IOREG specified.	F
		L &IOREG,DEBLOCKER5	If IOREG specified.	V
		L &IOREG,DEBLOCKER2	If IOREG specified.	U
		NOP 0(0)	If no IOREG.	
		L &RECSIZE,IJFVSREC	If spanned input.	S
		ST &RECSIZE,IJFVSREC	If spanned output.	S
56-63 (38-3F)			CCW.	
64-67 (40-43)		DC A(IOAREA1)	One I/O area.	
		DC A(IOAREA1+BLKSIZE-1)	One I/O area, read backward.	
		DC A(IOAREA2)	Two I/O areas.	
		DC A(IOAREA2+BLKSIZE-1)	Two I/O areas, read backward.	
68-71 (44-47)		DC F'0'	Input.	F
		DC A(IOAREA1+BLKSIZE -RECSIZE)	Input backward: DEBLOCKER1.	F
		DC A(IOAREA1)	One I/O area, output: DEBLOCKER1.	F
		DC A(IOAREA2)	Two I/O areas, output: DEBLOCKER1.	F
		DC A(BLKSIZE)	DEBLOCKER1: EBCDIC	V,S
		DC A(IOAREA1)	One I/O area: DEBLOCKER2.	U
		DC A(IOAREA2)	Two I/O areas: DEBLOCKER2.	U

Figure 28. DTFMT: Data Files (Part 3 of 8)

Bytes*	Bits	Contents[1]	Function	Record[2] Format
72-75 (48-4B)		DC F'RECSIZE'	Forward: DEBLOCKER2.	F
		DC F'-RECSIZE'	Backward: DEBLOCKER2.	F
		DC A(IOAREA1)	One I/O area: DEBLOCKER2.	V,S
		DC A(IOAREA2)	Two I/O areas: DEBLOCKER2.	V,S
		LA 14,1(14)	Forward.	U
		BCTR 14,0+NOPRO	Backward.	U
76-79 (4C-4F)		DC F'0'	Input forward: DEBLOCKER3.	F
		DC A(IOAREA1+BLKSIZE -RECSIZE)	Input backwards: DEBLOCKER3.	F
		DC A(IOAREA1+BLKSIZE-1)	Output, one I/O area: DEBLOCKER3.	F
		DC A(IOAREA2+BLKSIZE-1)	Output, two I/O areas: DEBLOCKER3.	F
		DC F'0'	DEBLOCKER3.	V,S
		DC Y(BLKSIZE)	(Bytes 76-77 only)	U
		DC Y(BLKSIZE-1)	(Bytes 78-79 only)	U
80-83 (50-53)		DC Y(BLKSIZE)+Y(BLKSIZE-1)	Forward.	F
		DC &(BLKSIZE)+Y(BLKSIZE+1)	Backward.	F
		DC F'0'	DEBLOCKER4.	V,S
		LR 12,RECSIZE	(Bytes 80-81 only)	U
		DC H'0'	(Bytes 82-83)	U
84-87 (54-57)		DC Y(RECSIZE-1)	(Bytes 84-85)	F
		DC 2X'00'	(Bytes 86-87) Output, standard labels.	F
		DC A(IOAREA1+4)	One I/O area: DEBLOCKER5, EBCDIC.	V,S
		DC A(IOAREA2+4)	Two I/O areas: DEBLOCKER5, EBCDIC.	V,S
		DC A(IOAREA1+BUFOFF)	One I/O area: DEBLOCKER5, ASCII.	V
		DC A(IOAREA2+BUFOFF)	Two I/O areas: DEBLOCKER5, ASCII.	V
		DC 2X'00'	(Bytes 84-85 output only) standard labels. Reserved for OPEN.	F, U
		B 28(15)	Input only, ERROPT=omitted.	U
		B 24(15)	Input only, ERROPT=SKIP.	U
		B 28(15)	Input only, ERROPT=IGNORE.	U
		DC A(ERROPT)	Input only, ERROPT=ADDRESS	U

Figure 28. DTFMT: Data Files (Part 4 of 8)

Bytes*	Bits	Contents[1]	Function	Record[2] Format
88-91 (58-5B)		DC A(WLRERR)	Input only WLRERR=ADDRESS.	For Fixed-Length Records only
		B 24(15)	Input only, WLRERR omitted and ERROPT=SKIP.	
		B 28(15)	Input only, WLRERR omitted and ERROPT=IGNORE or omitted.	
		DC 2X'00'	Output only, standard labels (bytes 88-89), reserved for OPEN.	
		DC A(ERROPT)	Input only, WLRERR omitted and ERROPT=ADDRESS.	
90-95 (5A-5F)		DC 6X'00'	File Serial Number, standard labels, output only.	
92-95 (5C-5F)		DC A(ERROPT)	Input only, ERROPT=ADDRESS. Output, nonstandard labels only. ERROPT=ADDRESS	
		B 28(15)	Input only, ERROPT=omitted.	
		B 24(15)	Input only, ERROPT=SKIP.	
		B 28(15)	Input only, ERROPT=IGNORE.	
96-99 (60-63)		DC 4X'00'	Volume Sequence Number, standard labels, output only.	
96-97 (60-61)		DC 2X'00'	Standard labels, input only. Reserved for OPEN.	
98-103 (62-67)		DC 6X'00'	File Serial Number, standard labels, input only.	
100-103 (64-67)		DC 4X'00'	File Sequence Number, standard labels, output only.	
104-107 (68-6B)		DC 4X'00'	Volume Sequence Number, standard labels, input only.	
		DC A(ERROPT)	Output only. Standard labels only. ERROPT=ADDRESS	
108-111 (6C-6F)		DC 4X'00'	File Sequence Number, standard labels, input only.	
88-91 (58-5B)		DC F'0'	DEBLOCKER6.	For Variable Length and Spanned Records
92-95 (5C-5F)		DC A(ERROPT)	Output only. Nonstandard labels only. ERROPT=ADDRESS	
92-93 (5C-5D)		DC Y(BLKSIZE)	Input only.	
		DC Y(BLKSIZE-4)	Output only: EBCDIC	V,S
		DC Y(BLKSIZE-BUFOFF)	Output only: ASCII	V
94-95 (5E-5F)		DC Y(BLKSIZE-1)		
96-97 (60-61)		DC Y(RECSIZE-1)		

Figure 28. DTFMT: Data Files (Part 5 of 8)

Bytes*	Bits	Contents[1]	Function	Record[2] Format
98-99 (62-63)		DC H'0'	Input only: residual count	For Variable Length and Spanned Records
100-103 (64-67)		DC A(WLRERR)	Input only, WLRERR=ADDRESS.	
		B 24(15)	Input only, WLRERR=omitted and ERROPT=SKIP.	
		B 32(15)	Input only, WLRERR=omitted and ERROPT=IGNORE or omitted.	
100-101 (64-65)		DC 2X'00'	Output only, standard labels. Reserved for OPEN.	
104-107 (68-6B)		DC A(ERROPT)	Input only, ERROPT=ADDRESS.	
		B 28(15)	Input only, ERROPT=omitted.	
		B 24(15)	Input only, ERROPT=SKIP.	
		B 28(15)	Input only, ERROPT=IGNORE.	
		DC A(ERROPT)	Output, nonstandard labels only (Version 3 onward). ERROPT=ADDRESS	
108-111 (6C-6F)		DC 4X'00'	Volume sequence number. Standard labels, output only.	
108-109 (6C-6D)		DC 2X'00'	Standard labels, input only. Reserved for OPEN.	
110-115 (6E-73)		DC 6X'00'	File serial number. Standard labels, input only.	
112-115 (70-73)		DC 4X'00'	File sequence number. Standard labels, output only.	
116-119 (74-77)		DC A(ERROPT)	Output only, ERROPT=ADDRESS, standard labels only.	
116-119 (74-77)		DC 4X'00'	Volume sequence number. Standard labels, input only.	
121-123 (78-7B)		DC 4X'00'	File sequence number. Standard labels, input only.	
86-91 (56-5B)		DC 6X'00'	File Serial Number. Standard labels, output only.	For Undefined Records only
88-91 (58-5B)		DC A(WLRERR)	Input only, WLRERR=ADDRESS.	
		B 24(15)	Input only, WLRERR=omitted and ERROPT=SKIP.	
		B 28(15)	Input only, WLRERR=omitted and ERROPT=IGNORE or omitted.	
		DC A(ERROPT)	Input only, WLRERR=omitted and ERROPT=ADDRESS. Output only, nonstandard labels, ERROPT=ADDRESS.	
92-95 (5C-5F)		DC 4X'00'	Volume Sequence Number. Standard labels, output only.	

Figure 28. DTFMT: Data Files (Part 6 of 8)

Bytes*	Bits	Contents[1]	Function	Record[2] Format
92-93 (5C-5D)		DC 2X'00'	Standard labels, input only. Reserved for OPEN.	For Undefined Records Only
94-99 (5E-63)		DC 6X'00'	File Serial Number. Standard labels, input only.	
96-99 (60-63)		DC 4X'00'	File Sequence Number. Standard labels, output only.	
100-103 (64-67)		DC 4X'00'	Volume Sequence Number. Standard labels, input only.	
100-103 (64-67)		DC A(ERROPT)	Output only, standard labels only. ERROPT=ADDRESS	
100-103 (64-67)		DC 4x'00'	Volume Sequence Number. Standard labels, output only.	
104-107 (68-6B)		DC 4X'00'	File Sequence Number. Standard label, input only.	
100-103 (64-67)		DC A(WLRERR)	Input only, WLRERR=ADDRESS.	For Spanned Records Only
		B 24(15)	Input only, WLRERR=omitted and ERROPT=SKIP.	
		B 32(15)	Input only, WLRERR=omitted and ERROPT=IGNORE or omitted.	
100-101 (64-65)		DC 2X'00'	Output only, standard labels. Reserved for OPEN	
102-107 (66-6B)		File Serial Number	Standard labels, output only.	
100-103 (64-67)		DC 4X'00'	Output only, ERROPT=ADDRESS, nonstandard labels only.	
100-123 (64-7B)		DC 24X'00'	Output only, ERROPT=omitted, nonstandard labels.	
104-107 (68-6B)		DC A(ERROPT) B 24(15) B 24(15) B 28(15)	Input only, ERROPT=ADDRESS. Input only, ERROPT=omitted. Input only, ERROPT=SKIP. Input only, ERROPT=IGNORE.	
104-107 (68-6B)		DC A(ERROPT)	Output only, ERROPT=ADDRESS, nonstandard labels.	
108-123 (6C-7B)		DC 16X'00'	Output only, ERROPT=ADDRESS, nonstandard labels.	
108-111 (6C-6F)		Volume Sequence Number	Standard labels, output only.	
112-115 (70-73)		File Sequence Number	Standard labels, output only.	
116-119 (74-77)		DC A(ERROPT)	Output only, ERROPT=ADDRESS, standard labels.	

Figure 28. DTFMT: Data Files (Part 7 of 8)

Bytes*	Bits	Contents[1]	Functions	Record[2] Format
120-123 (78-7B)		DC 4X'00'	Output cnly, ERROPT=ADDRESS, standard labels.	For Spanned Records Only
108-123 (6C-7B)		DC 16X'00'	Input only, nonstandard labels.	
1C8-109 (6C-6D)		DC 2X'00'	Standard labels, input only. Reserved for OPEN.	
110-115 (6E-73)		File Serial Number	Standard labels, input only.	
116-119 (74-77)		Volume Sequence Number	Standard labels, input only.	
120-123 (78-7B)		File Sequence Number	Standard labels, input only.	
124-127 (7C-7F)		DC F'0'	Full word for loading and storing USER RECSIZE: IJFVSREC	
128 (80)		DC X'00'	IJFVSFLG	
	0		Sign bit, not used.	
	1		Skip to first segment.	
	2		First segment.	
	3		Segment out of sequence, input only.	
	4		Read back for EOV, output only.	
	5		File reversed fcr logical spacing, input CNTRL only. TRUNC issued, output only.	
	6		User TRUNC issued, output only.	
	7		Multisegment, output only. Skip Get Segment, input CNTRL only.	
129-131 (81-83)		DC 3X'00'	Pointer within WORKA.	

[1] The format of the tape data file DTF table is different starting at tyte 44. The lccation irdicated by the numbers in the left hand column can contain only one of the factors listed under "Contents". The factor used for any given DTF table is determined by whether the file record format is fixed, variable, or undefined, and by other DTF parameters as indicated. A blank in the record column indicates that the contents apply to all record types.

[2] Record Format Explanation.

F = Fixed Fecord
V = Variable
U = Undefined
S = Spanned (variable format superset)

The deblcckers are scratch areas used by the modules to save data from one GET/PUT macrc instruction to another. In the text and listings, they are referred to by the names DEBLCCKER1 to 6. These are not labels, they are comments used to make it easier to follcw the listings.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 28. DTFMT: Data Files (Part 8 of 8)

Bytes*	Bits	Function
0-15 (00-0F)		CCB.
16	0-1	Not used.
(10)	2	COBOL open; ignore option.
	3	1 = VOL1 label is at user specified density.
	4	1 = DTF table address constants relocated by OPENR.
	5-7	Not used.
17-19 (11-13)		Address of logic module.
20 (14)		DTF type = X'10'.
21	0	1 = no rewind.
(15)	1	1 = rewind unload.
	2	1 = work file.
	3	1 = read backward.
	4	1 = write.
	5	1 = POINTW.
	6	Not used.
	7	1 = forward-space file before next operation.
22-23 (16-17)		Not used.
24-25 (18-19)		Record length.
26-27 (1A-1B)		Maximum BLKSIZE.
28 (1C)		Read command code (X'02' for read forward, X'0C' for read backward).
29-31 (1D-1F)		EOF address.
32-39 (20-27)		CCW.
40-43 (28-2B)		Block count, initialized 00000000 for read forward, 00400000 for read backward.
44	0	1 = error routine.
(2C)	1	1 = ignore.
	2	Not used.
	3	1 = record fixed unblocked.
	4-7	Not used.
45-47 (2D-2F)		DC A(ERROPT) Address of error routine.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 29. DTFMT: Work Files

Bytes*	Bits	Contents	Function
0-15 (00-0F)			CCB.
16 (10)	0-1		Not used.
	2		COBOL open; ignore option.
	3		Not used.
	4		DTF table address constants relocated by CPENR.
	5		Not used.
	6		1 = ASCII, 0 = EBCDIC.
	7		Not used.
17-19 (11-13)		3X'00'	
20 (14)		X'12'	Standard labeled, output
		X'14'	Standard labeled, input, forward.
21 (15)	0-3		Not used.
	4		1 = input, 0 = output.
	5-7		Not used.
22-29 (16-1D)			Symbolic filename.
30 (1E)			Not used.
31 (1F)	0-4	B'01100'	Used as displacement by OPEN.
	5		
	6-7		Reserved.
32 (20)	0		1 = standard labels.
	1-2		Not used.
	3		1 = no rewind.
	4		Not used.
	5		User label address: 1 = yes, 0 = no.
	6-7		Not used.
33-35 (21-23)			User label routine address.
36 (24)	0		1 = DTFPH table.
	1		Not used.
	2		File switch: 1 = input, 0 = output.
	3		Not used.
	4		1 = EOF switch.
	5-7		Not used.
37-39 (25-27)			User label exit.

Figure 30. DTFPH: Magnetic Tape (Part 1 of 2)

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

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 30. DTFPH: Magnetic Tape (Part 2 of 2)

MTMOD MACRO

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

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

DATA FILES

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

1. Fixed, blocked and unblocked.

2. Variable and spanned, blocked and unblocked.

3. Undefined.

These modules provide the logic to perform the functions of the LIOCS imperative macros CNTRL, GET, PUT, RELSE, and TRUNC. IF RECFORM=SPNBLK or SPNUNB, a work area is required.

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

WORK FILES

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

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

In addition to the READ and WRITE imperative macros, the work file module supports the NOTE, POINTS, POINTR, POINTW, and CHECK imperative macros. CHECK must always be issued after a READ or WRITE if the user wishes to ensure completion of the operation before issuing another instruction. NOTE is used with the POINTR and POINTW macros to position the tape to a predetermined record. The POINTS macro instruction repositions the tape at the beginning of a file and sets an indicator which causes the tape to be spaced over any tapemark or header labels when the next READ or WRITE macro is issued.

Error Option Extensions

When MTMOD specifies ERREXT=YES, checking is done for two types of errors:

1. READ Errors: If a data check occurs, the unrecoverable I/O error bit (byte 2, bit 2 of the CCB) is turned off. If no data check occurs, this bit will be left on. Pointers and linkage addresses are saved before returning to the user. A pointer to a parameter list is placed in register 1. The parameter list contains:

    a. DTF address (4 bytes).

    b. Storage address of the block in error (4 bytes).

2. WRITE Errors: The problem program ERRCPT routine is entered if a write error occurs. The device is considered to be inoperable, and a SVC 50 is issued to cancel the job if the problem program tries to return to the module. Register 1 points to the parameter list described for READ errors.

MTMOD: CHECK Macro, Work File, Chart DA

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

Entry: From a CHECK macro expansion to the label IJFWCHEK.

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

Method: This macro instruction waits for the completion of the input/output operation started by a READ or WRITE macro instruction. If the I/O operation is completed without detecting an error or an exceptional condition, CHECK returns control to the next sequential instruction following the CHECK macro expansion in the problem program. If the operation results in a read error, CHECK processes the user's option specified in the ERROPT entry of the DFT. If CHECK finds an end-of-file condition, control passes to the user's end-of-file routines.

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

MTMOD: CNTRL Macro, Chart DB

Objective: To execute a nondata physical operation on the file.

Entry: From a CNTRL macro expansion to the label:

IJFFCNT for files containing fixed length records in EBCDIC.

IJFXCNT for files containing fixed length records in ASCII.

IJFUCNT for files containing undefined records in EBCDIC.

IJFNCNT for files containing undefined length records in ASCII.

IJFVCNT for files containing variable length records in EBCDIC.

IJFSCNT for files containing spanned records in EBCDIC.

IJFRCNT for files containing variable length records in ASCII.

Exit: To the problem program.

Method: The CNTRL macro is available to the user for nondata operations on the file. For a magnetic tape file, these operations are: backspace, write a tapemark, forward space file, etc.

The control routine first checks to ensure that any previous I/O operation has been completed. Then the routine sets the control operation code in the CCW, and causes the control function to be performed. The orginal operation code is restored in the CCW, and, when I/O complete is indicated, the routine returns to the problem program.

MTMOD: CNTRL Macro Work File, Chart DB

Objective: To perform a nondata physical operation.

Entry: From a CNTRL macro expansion to the label IJFWCTL.

Exit: To the problem program.

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

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

MTMOD: FEOV Macro, Chart DC

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

Entry: From an FEOV macro expansion to the label:

IJFFFEO for files containing fixed length records in EBCDIC.

IJFXFEO for files containing fixed length records in ASCII.

IJFUFEO for files containing undefined records in EBCDIC.

IJFNFEO for files containing undefined records in ASCII.

IJFVFEO for files containing variable length records in EBCDIC.

IJFSFEO for files containing spanned records in EBCDIC.

IJFRFEO for files containing variable length records in ASCII.

Exit: To the problem program.

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

This routine sets the FEOV switch on in the DTF table and calls $$BCEOV1 to close the file and execute the user specified rewind option. If no rewind option is specified, a message is issued indicating the tape should be rewound and unloaded.

MTMOD: GET Macro, Charts DD-DE

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

Entry: From a GET macro expansion to the label:

IJFFGET for files containing fixed length records in EBCDIC.

IJFXGET for files containing fixed length records in ASCII.

IJFUGET for files containing undefined records in EBCDIC.

IJFNGET for files containing undefined records in ASCII.

IJFVGET for files containing variable length records in EBCDIC.

IJFSGET for files containing spanned records in EBCDIC.

IJFRGET for files containing variable length records in ASCII.

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

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

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

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

MTMOD: NOTE Macro, Work File, Chart DF

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

Entry: From NOTE macro expansion to the label IJFWNOTE.

Exit: To the problem program.

Method: The NOTE macro instruction obtains the identification of the last physical record that was read or written in the specified file. The user must ensure that the last operation was completed satisfactorily by issuing a CHECK macro before issuing a NOTE. The record identification is put in register 1. This identification is a 3-byte block count. The user is responsible for saving the block count to be used with a POINT macro at some later time.

MTMOD: POINTR Macro, Work File, Chart DF

Objective: To position the tape to read a record previously identified by a NOTE macro.

Entry: From a POINTR macro expansion to the label IJFWPNTR.

Exit: To the problem program.

Method: This macro instruction repositions the file to read a record previously identified by a NOTE macro. The block count previously supplied by the NOTE macro is used by the POINTR macro to position the tape to the specified record on the file.

The block count is updated in the DTF and compared with a count that was saved by the user from a previous NOTE macro. Then, the tape is spaced backward or forward, and the block count in the DTF is correspondingly decreased or increased until it is equal to the NOTE block count.

When the equal condition is sensed, and if read backward is not specified in the DTF, the tape is backspaced one block. The tape is now positioned to read the desired record.

MTMOD: POINTS Macro, Work File, Chart DF

Objective: To position the tape at the beginning of the file for the next READ or WRITE macro.

Entry: From a POINTS macro expansion to the label IJFWPNTS.

Exit: To the problem program.

Method: The POINTS macro instruction repositions a tape file to the beginning of the file. When a POINTS is issued to the tape file, the forward space file switch, bit 7 of byte 21 in the DTF, is set on and the tape is rewound. If header labels are present, they are bypassed when the next READ, WRITE, POINTR, or POINTW macro is issued.

MTMOD: POINTW Macro, Work File, Chart DF

Objective: To position the tape to write a record following a record identified by a previous NOTE macro.

Entry: From a POINTW macro expansion to the label IJFWPNTW.

Exit: To the problem program.

Method: The POINTW macro repositions the file to write a record after the record identified by a NOTE macro. The block count supplied by a previous NOTE macro is used by the POINTW to position the tape to write a record immediately after the one identified by the NOTE macro.

The block count is updated in the DTF and compared with a count that was saved by the user from a previous NOTE macro. Then the tape is spaced backward or forward, and the block count in the DTF is correspondingly decreased or increased until it is equal to the NOTE block count. The tape is then in position to WRITE the desired record.

MTMOD: PUT Macro, Charts DG-DH

Objective: To write a logical record on a magnetic tape file.

Entry: From a PUT macro expansion to the label:

IJFFPUT for files containing fixed length records in EBCDIC.

IJFXPUT for files containing fixed length records in ASCII.

IJFUPUT for files containing undefined records in EBCDIC.

IJFNPUT for files containing undefined records in ASCII.

IJFVPUT for files containing variable length records in EBCDIC.

IJFSPUT for files containing spanned records in EBCDIC.

IJFRPUT for files containing variable length records in ASCII.

Exit: To the problem program.

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

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

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

MTMOD: READ Macro, Work Files, Chart DJ

Objective: To read a physical record from tape.

Entry: From a READ macro expansion IJFWREAD.

Exit: To the problem program.

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

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

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

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

MTMOD: RELSE Macro, Chart DJ

Objective: To skip the remaining records in a block.

Entry: From a RELSE macro expansion to the label:

IJFFREL for files containing fixed length records in EBCDIC.

IJFXREL for files containing fixed length records in ASCII.

IJFVREL for files containing variable length records in EBCDIC.

IJFSREL for files containing spanned records in EBCDIC.

IJFRREL for files containing variable length records in ASCII.

Exit: To the problem program.

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

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

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

MTMOD: TRUNC Macro, Chart DJ

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

Entry: From a TRUNC macro expansion to the label:

IJFFTRU for files containing fixed length records in EBCDIC.

IJFXTRU for files containing fixed length records in ASCII.

IJFVTRU for files containing variable length records in EBCDIC.

IJFSTRU for files containing spanned records in EBCDIC.

IJFRTRU for files containing variable length records in ASCII.

Exit: To the problem program.

Method: This routine causes the current block to be considered full, and then writes it on the file. It sets up conditions that cause the next PUT macro instruction to begin placing records at the beginning of a new block.

The TRUNC macro can only be issued to a blocked output file.

The routine gets the current blocksize, and checks to determine if it is 0. If yes, control returns to the problem program. If no, a new blocksize is set in the CCW, and control branches to the Read/Write subroutine to write the truncated record. Then the CCW is restored, and control returns to the problem program.

MTMOD: WRITE Macro Work File, Chart DJ

Objective: To write a physical record on tape.

Entry: From a WRITE macro expansion to IJFWRIT.

Exit: To the problem program.

Method: The WRITE macro instruction causes a record to be written from an indicated area to the output file. This routine requires the user to specify the area from which the record is written, and allows the user to specify the actual number of bytes to be written. If the length is not specified, the user specifies RECFORM=UNDEF in the DTF. Blocking of records is not provided for in the WRITE macro instruction. If blocking is desired, it is the user's responsibility to block the records before issuing a WRITE.

The WRITE macro routine branches to the Workarea Subroutine to determine the record length and set up the proper controls. Before returning control, the subroutine checks bit 7 of byte 21 in the DTF to determine if a POINTS macro was issued before the WRITE. If the bit is on, indicating that POINTS was issued, the tape is at loadpoint. It is, therefore, necessary to execute a forward space file operation to bypass any labels and position the tape to write the first data record. When this operation is complete, control is returned to the WRITE macro routine.

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

MTMOD: Bypass Checkpoint Records Routine, Charts DN and DV

Objective: To bypass checkpoint records within an input EBCDIC tape data file.

Entry: From the Read/Write subroutine when CHKPTREC=YES is specified in the DTFMT entry and there are checkpoint records in the file.

Exit: To the Read/Write subroutine.

Method: For any mode input tape, this routine checks the first 12 bytes of each record in the input area for /// CHKPT //, indicating a checkpoint header record has been read. For read backwards, the checkpoint header record is located in the 20 lower-order bytes of the input area.

The method for bypassing checkpoint records is:

Extract bytes 14-15 of the checkpoint header label (or trailer label if reading backwards), pack the field (convert to binary) and forward space (or backspace) that number of records.

INITIALIZATION AND TERMINATION

Magnetic tape files processed by logical IOCS must conform to certain standards. These standards concern labels, placement of tapemarks, and the grouping (blocking) of tape records.

Tape files, with or without labels, can be processed by logical IOCS. It is the function of the open and close logical transients to process (check or create) the labels as required. Refer to VSE/Advanced Functions Tape Labels for the organization of tape file labels.

When standard labels are specified for a tape file, logical IOCS processes only the standard volume label, VOl1, and the standard file labels, HDR1, EOF1, and EOV1. (Refer to VSE/Advanced Functions Tape Labels for the format of these labels.) User labels and nonstandard labels must be processed by the user.

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

Tape Open, Close and EOF/EOV Routines, Charts 01 and 02

Note: Tape files are identified by a single job control statement, // TLBL. Where reference is made to the // TLBL job control statement in the text and flowcharts in this manual, the reference applies to the collective // VOL and // TPLAB statements as well. Also, the decision block in some of the flowcharts containing the question "TLBL INFORMATION IN DTF" determines if the DTF table contains tape label information updated by the DTFMT macro.

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

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

Entry:

- From the Open Monitor, $$BOPEN1.
- From $$BCMT02.
- From a message writer phase or $$BOMLTA to the label RELOCATE.

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

Exits:

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

2. To $$BCEOV1 if end-of-volume reached.

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

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

Upon entry into the $$BOMT01 open phase, a test is made to determine if the phase was entered after issuing a message or after performing a tape rewind operation. If this is the case, a branch is made to the appropriate return point. This address is represented in the flowcharts by the exit(s) from the decision block following the message block.

The DTF table should contain tape label information. A test is made for the presence of this label information. If the DTF contains such information, it must then be used to update the label information area.

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

If the file is not already open, the volume serial number in the tape VOL1 label is compared to the file serial number (EBCDIC) or set identifier (ASCII) in the // TLBL card image. If the numbers are not the same, error message 4112A is printed. If the response to this message is IGNORE, the // TLBL file serial number (EBCDIC) or set identifier (ASCII) is made equal to the volume serial number. If access control is present in the system, the operator is not given the option to ignore the message because IGNORE is an invalid response. If the response is NEWTAP, indicating that the operator has mounted a new tape, checking of the standard volume label is repeated. If the response is BYPASS, or if the serial numbers are equal, the next record on the tape is read.

Any volume labels written after the VOL1 label are bypassed. When a standard tape file label, HDR1, is read, the following label entries are checked against the corresponding entries in the // TLBL card image as specified: file serial number (EBCDIC) or set identifier (ASCII), volume sequence number (EBCDIC) or file section number (ASCII), file sequence number, generation number, version number, creation date, and file identifier (FILEID). Any discrepancies are noted by error messages.

After the standard file label (HDR1) is read and checking is complete, all additional file labels are bypassed and the phase proceeds to check for user labels. If the address of the user's label routine is specified in the DTF table (LABADDR), any user labels on the tape are checked. If the user's label routine address is not specified, checking of user labels is bypassed.

The reading of a tapemark indicates the end of the label area on the tape.

$$BOMT02: Open Input Standard Labels, Backward, Chart EC

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

Entry: From the Open Monitor, $$BOPEN1, or from a message writer phase or $$BOMLTA to the label RELOCATE.

Input: Input is from the label information area (// TLBL card image) and the tape currently being processed.

Exits:

1. To the Open Monitor, $$BOPEN, if an open statement was issued.

2. To CANCEL.

Method: This routine opens input tape files in a backward direction by reading the trailer labels instead of header labels.

Upon entry into the $$BOMT02 open phase, a test is made to determine if the phase was entered after issuing a message or after performing a tape rewind operation. If this is the case, a branch is made to the appropriate return point. This address is represented in the flowchart by the exit(s) from the decision block following the message blcck.

The $$BOMT02 routine then checks for a tapemark to determine that the tape is positioned correctly. If a tapemark is not read, error message 4117D is printed. After reading the tapemark, the routine reads the trailer labels.

If user trailer labels are present and the address of the user's label routine, LABADDR, is specified in the DTF table, the routine exits tc the user's label routine to process the user trailer labels. If LABADCR is not specified, the user trailer labels are bypassed.

After processing the user labels, if required, the $$BOMT02 routine reads the file trailer labels until either a standard end-of-file or end-of-volume, EOF1 or EOV1, label is encountered. All intervening file trailer labels are ignored. When EOF1 or EOV1 label is read, the file serial number from the label is saved in VOLSAVE. Only the file identifier (FILEID) portion of the label is checked against its corresponding entry in the // TLBL card image in main storage.

Note: If the FILEID in the // TLBL card image is blank, no checking of the standard file trailer label is performed.

If the FIIEIC numbers are not equal, errcr message 4118D(I) is printed.

After the file trailer label is processed, the rcutine reads the tapemark that precedes the trailer label area, posts the file open in the DTF table and returns to $$BOPEN.

A successful cpen positions the tape at the IRG that follows the last data record.

$$BOMT03: Open Output Standard Labels, Forward (Phase 1), Charts ED-EE

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

Entry: From the Open Monitor, $$BOPEN1, or from a message writer phase, or from $$BOMLTA.

Input: From the label information area (// TLBI card image) and from the tape currently being processed.

Exits:

1. To the second phase of Open Output Standard Labels, Forward, $$BOMT04.
2. To the Standard Volume Label Rewriter, $$BONVOL.

Method: The $$BOMT03 routine checks and builds labels for output tape files. The routine:

1. Checks the information in the // TLBL card image and inserts the proper data when blank entries are encountered.
2. Reads the standard volume label from the tape and compares it to the information in the // TLBL card image.
3. Determines the availability of the tape from the standard file label.
4. Writes new standard file and user labels as required.

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

Upon entry into the $$BOMT03 phase, a check is made to determine if the phase was entered after issuing a message or after performing a tape rewind operation. If this is the case, a branch is made to the appropriate return point. This address is represented in the flowcharts by the exit(s) from the decision block following a message block.

A test is made to determine if tape label information is contained in the DTF table. If so, it is necessary to update the // TLBL card image in main storage with informaticn contained in the DTF table.

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

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

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

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

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

When the standard file label, HDR1, is read, the expiration date is checked to determine the availability of the tape. If the tape is available (that is, if the retention period has expired or the expiration date has been reached) or if no HDR1 label was found, the tape is backspaced and $$BOMT04 is fetched.

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

$$BOMT04: Open Output Standard Labels, Forward (Phase 2), Chart EF

Objective: To write user labels and open output tape data files with standard labels.

Entry: From the first phase of the Open Output Standard Labels, $$BOMT03.

Input: From the DTF table and the user's label routine.

Exits:

1. To the Open Monitor, $$BOPEN, if the file is not already posted open.
2. To the logical IOCS module if the file is already posted open.

Method: If phase $$BOMT04 is reentered from a message writer phase, the return bit in byte 89 of the communications region is reset. Control is then passed to the address in register 14.

When phase $$BOMT04 is entered normally, a new standard file label is built and written on the tape.

To build the new standard file label, information is obtained from the // TLBL card image. If the file identifier (FILEID) field of the // TLBL card is blank or zero, the file name is obtained from the DTF table and inserted into the card image. The information from the // TLBL card image used to build the label includes:

- File identifier.
- File serial number (EBCDIC) or set identifier (ASCII).
- Volume sequence number (EBCDIC) or file section number (ASCII).
- File sequence number.
- Generation number.
- Version number.
- Creation date.
- Expiration date.
- Block count.

A test is then made to determine the version level of the DTF table. If the table was generated by a Version 2.1 or Version 3 DTFMT macro, the file serial number (EBCDIC) or set identifier (ASCII), volume sequence number (EBCDIC) or file section number (ASCII), and file sequence numbers from the // TLBL card image (bytes 26 through 39) are inserted into the corresponding entries of the DTF table (bytes 42 through 55). Otherwise, the new // TLBL card image is written back onto the label information area.

The $$BOMT04 phase then exits to the user's label routine if specified, to process and write up to eight user labels for an EBCDIC file and an unlimited number of user labels fcr an ASCII file. When the user's label routine is complete, a tapemark is written to indicate the end of the tape label area and the open indicator in the CTF table is checked. If the open indicatcr is already on, control returns to the logical IOCS module. If the open indicator is nct on, it is turned on and $$BOPEN is fetched.

A successful cpen posts the file open in the DTF table, and positions the tape at the IRG preceding the area of the first data record.

$$BCMT05: Cpen Input or Cutput, Nonstandard or Unlabeled, Chart EG

Objective: To open nonstandard and unlabeled files, bcth input and output, in either a forward or a backward direction.

Entry: From the Open Monitor, $$BOPEN1, $$BOMLTA, or frcm $$BCMT02.

Input: From the tape being processed.

Exits:

1. To the Open Monitor, $$BOPEN, after the file is posted open in the DTF table.
2. To the logical IOCS module if the file is already posted open in the DTF table(multireel file).

Method: This rcutine checks for 7-track or 9-track tape and sets the correct mode. If nonstandard labels are specified, an exit is made to the user label address. After the return from the user, a tapemark may or may not be written for an output file, according to the opticn specified in the DTF. For input files, a check is made to determine if the tapemark was read by the user.

For unlabeled files, tests for options such as tapemark, rewind, etc., are made to determine the various factors stated in the DTF parameters and appropriate action is taken. If the file has been previously opened (multireel file), control returns to logical IOCS. If the file has not been opened, the routine opens the file and calls the Open Mcnitor.

Upon entry tc the $$BOMT05 phase, a test is made to determine if the phase was entered after issuing a message or after a tape rewind operation. If this is the case, a branch is made to the appropriate return point. This address is represented in the flowcharts by the exit(s) from the decision block following the message block.

The CTF table is examined to determine the directicn cf the file to be opened and the appropriate action is taken.

1. For nonstandard labeled input files opened in a forward direction, the labels are processed by the user if the address of the user's label routine is specified in the DTF table. Detection of a tapemark indicates the end of the label area, and the file is posted open.
2. For unlabeled input files opened in a forward direction, the tape is read in search of a tapemark. If a tapemark is not read, the tape is backspaced one record to return it to its original position. In either case, the file is posted open in the DTF table.
3. For nonstandard labeled output files opened in a forward direction, the labels are processed by the user if the address of the user's label routine is specified in the DTF table. If the tapemark cption is specified in the DTF table, a tapemark is written after the last label. The file is then posted open.
4. For unlabeled output files opened in a forward direction (and input files if access control is present in the system), the tape is read to determine if the tape contains any labels. If a label is detected, error message 4125D(I) is printed and the operator can elect either to ignore the label or tc mount a new tape. If access control is present in the system, the operator is not given the option to ignore the label because IGNORE is an invalid response. If the operator chooses to ignore the label and the tape is a 9-track tape, it is backspaced, written, and again backspaced. This sets the mode and erases the label. If the tapemark option is specified in the DTF table, a tapemark is written. The file is then posted open in the DTF table.
5. For nonstandard labeled input files opened in a backward direction, the labels are processed by the user if the user's label routine address is specified in the DTF table and the file is posted open in the DTF table.
6. For unlabeled input files opened in a backward direction, the only check is for the existence of a tapemark. This test positions the tape properly. The file is then posted open in the DTF table.
7. For nonstandard labeled output files opened in a backward direction, the same action as in item 5 occurs.
8. For unlabeled output files opened in a backward direction, the same action as in item 6 occurs.
9. If access control is present in the

system, any attempt to open an unlabeled tape file with REWIND=NO specified in the DTF results in the message 4125I being issued and the job is canceled.

$$BCMT06: Open Work Files, Magnetic Tape, Chart EH

Objective: To open a tape work file.

Entry: From $$BOPEN1 (the Open Monitor) and $$BOMLTA to the label TRANSENT.

Input: From the tape being processed.

Exits:

1. To the Open Monitor, $$BOPEN.

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

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

If this phase is reentered after issuing a message or a tape rewind operation, a branch is made to the appropriate return point. This address is represented in the flowcharts by the exit(s) from the decision block that follows a message block.

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

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

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

If a standard volume label is read, the volume serial number is saved in location VOLSAVE. Additional read commands are executed to bypass any additional volume labels. If a standard file label (HDR1) is read that is neither blank nor all zeros, the expiration date is checked to determine the availability of the tape. If the tape is available or if no HDR1 label is read, or if this is the first time the file is opened as an output file, a check is made to see if density support is needed. If it is, $$BONVOL (Standard Volume Label Rewriter) is fetched. Otherwise, the file is considered open, and control returns to $$BOPEN.

$$BOMT07: Open Input Standard Labels, Forward, Chart EJ

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

Entry: From phase 1 of Open Input Standard Labels, $$BOMT01.

Input: From the DTF table.

Exits:

1. To $$BOMTES to check for TES and, if supported, to post status.

2. To the Open Monitor, $$BOPEN.

3. To the user.

Method: If the DTF table contains TLBL information, the table is updated from the new // TLBL card image. If the DTF table does not contain TLBL information, the new // TLBL card image is stored back in the label information area. The file is posted open in the DTF table and the $$BOMT07 phase is terminated.

$$BJCOPT (Phase 1) and $$BJCOP1 (Phase 2): Job Control Tape Open, Charts EK-EL

Objective: To open SYSRDR, SYSIPT, SYSPCH, and SYSLST when these logical units are assigned to tape units.

Entries:

1. From Job Control to open SYSRDR, SYSIPT, SYSPCH, and SYSLST when these logical units are assigned to tape units.

2. From the Linkage Editor to open SYS001 if SYS001 is a tape.

3. From Phase $$BCMT07 to open an alternate assignment.

Exits:

1. To the problem program via SVC 11.

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

Method: This routine opens SYSRDR, SYSIPT, SYSPCH, and SYSLST when these logical units are assigned to tape units. Three attempts, using the following assumptions, are made to read the first record:

1. A label in standard mode and read first record; no retries.

2. Unlabeled file and read first record in user mode; no retries.

3. Read first record in standard mode; allow retries.

This is a table of actions taken by the routine depending upon the result of reading the first record:

Result	Input File	Output File
Unreadable	Rewind	Error
Unlabeled	Rewind	Rewind
Standard Label Set	Skip labels, and tapemark.	Skip volume labels, write HDR1 and tapemark.

For a 9-track dual density output tape, a comparison is made between the user-specified density (800, 1600, or 6250 BPI) and the VOL1 density of the mounted tape. If a discrepancy is found, and if the tape is at load point, the volume label(s) is rewritten according to the user-specified density.

Note: If SYSLST, SYSPCH, or SYSOUT is assigned to a tape file, Job Control Open preserves the volume label; and if the header label is expired, Job Control writes a new header label with the format HDR1 followed by 76 binary zeros.

$$BCEOV1: EOF/EOV Monitor, Chart EM

Objective: To initialize for end-of-file/ end-of-volume (EOF/EOV) routines, and to fetch the proper EOF/EOV routine. This phase is used for tape files only.

Entries:

1. From the problem program when an end-of-file or end-of-volume condition is sensed (SEOV). Forced EOV (FEOV macro) also calls this routine.

2. From $$BOMT01 when end-of-volume during open multifile multi-volume.

Exit: To the proper EOF/EOV routine.

Method: This phase determines the type and format of the file from the DTF table, and selects the name of the required tape EOF or EOV phase. It then reads label information for the file from the label area into the open table at the end of the logical transient area. This information is passed to the required tape EOF/EOV phase fetched by the EOF/EOV Monitor. The file open bit in the PUB2 table is then turned off.

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

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

Entry: From $$BCEOV1 (the EOF/EOV Monitor) and $$BOMLTA to the label TRANSENT.

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

Exits:

1. To the user's EOF routine if specified in the DTF table.

2. To phase $$BCMT02 for an EOV condition to switch to an alternate tape.

3. To CANCEL if an error condition occurs.

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

Upon entry into the $$BCMT01 phase, a test determines if the phase was entered after issuing a message or after performing a tape rewind operation. If this is the case, a branch is made to the appropriate return point. This address is represented, in the flowcharts, by the exit(s) from the decision block following a message block.

A check of the DTF table (byte 36) then determines if the file is being closed as a result of an FEOV (Force-End-of-Volume) macro being issued. If bit 3 of byte 36 is on, an FEOV macro was issued. The tape is rewound, unless the NORWD option is specified in the DTF. No label checking is performed. Phase $$BCMT02 is then fetched to determine if an alternate tape is available.

If an FEOV macro was not issued, the $$BCMT01 phase reads the tape in search of standard trailer labels if such are specified in the DTF table. Only the block count contained in these labels is checked against the block count contained in the DTF table. Any discrepancy is noted by error message 4131D. User labels are bypassed unless the user furnishes the name of his label processing routine. This phase checks DTF byte 16, bit 3, to determine if American National Standard COBOL has been specified. If it has been specified, a first time switch is set before yielding control to the user's label processing routine. The tape is rewound, unless the NORWD option is specified, and a check determines the type of standard trailer label read (that is, EOV1 or EOF1). An EOF1 label sets the EOF/EOV switch, bit 4 of byte 36 in the DTF table. If the switch is turned on, the file is posted closed in the DTF table and an exit is made to user's end-of-file address, EOFADDR. If the switch is off, indicating that an end-of-volume condition exists, a test determines if the file is a COBOL file that must be rewound and unloaded. If so, the rewind/unload operation is executed. Phase $$BCMT02 is fetched to determine if an alternate tape is available.

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

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

$$BCMT02: Tape Close, Alternate Switching for EOV, Chart FB

Objective: To switch tape drives when EOV is sensed.

Entry: From $$BCMT01, $$BCMT04, or $$BOMLTA.

Input: From the trailer label of the active file and from the label information area (// TLBL card image).

Exits: To $$BOMT01, $$BOMT05, $$BOPEN, or CANCEL.

Method: This routine switches from one drive to another, if an alternate drive is specified and if EOV is sensed in the trailer label of the tape file previously processed. It switches either input or output tape files and calls in the appropriate open phase to open the alternate file. If EOV is sensed and if no alternate drive is specified, a message is issued. The user can cancel the job or mount a new tape on the same drive and continue.

If the $$BCMT02 phase is reentered after issuing a message or after performing a tape rewind operation, a branch is made to the appropriate return point. This address is represented in the flowcharts by the exit(s) from the decision block following a message block.

If the $$BCMT02 phase was not entered from a message writer phase, the // TLBL card image in main storage must be updated with information contained in the DTF table.

An SVC 22 is issued to seize the system; that is, to disable the multiprogramming operation. The address of the logical unit block (LUB) entry for the tape being closed is obtained. From this entry, the addresses of the Physical Unit Block (PUB) entry and the first Job Information Block (JIB) entry, if any, are calculated. Any alternate assignment for the LUB is stored in the JIB table. After the alternate PUB is obtained and checked for availability, an SVC 22 is again issued to release the system, that is, to return to multiprogramming operation.

Note: Further description of the SVC 22 and of the functions of the LUB, PUB, and JIB tables is found in *VSE/Advanced Functions Diagnosis Reference: Supervisor*.

If the file being closed (the file for which an alternate assignment has been found) is an output file, the address of the DTF table is saved and $$BOPEN is fetched. If the file is an input file with nonstandard labels, phase $$BOMT05 is fetched to open the alternate assignment.

If the file being closed is a standard labeled input file, the volume sequence number in the // TLBL card image in main storage is updated by 1. If the DTF table contains tape label information, it is also necessary to update the volume sequence number in the DTF table. If the DTF table does not contain tape label information, the updated // TLBL card image is written back onto the label information area. In either case, phase $$BOMT01 is fetched to open the alternate assignment.

If no alternate assignment is available for the file being closed, message 4140A is printed. The operator can mount an alternate tape and give a NEWTAP response to the message. The phase then proceeds to reopen the file as if an alternate had been assigned.

$$BCMT03: Tape Close, EOF Input Backward, Chart FC

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

Entry: From the EOF/EOV Monitor, $$BCEOV1, to the label TRANSENT, or from $$BCMT08.

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

Exits:

1. To $$BCMT08.

2. To the user's end-of-file routine, EOFADDR.

3. To cancel, if an error condition is encountered.

Method: The $$BCMT03 phase reads the file header labels and processes the labels as trailer labels. The block count is checked and the file is then ready to be closed by the user's end-cf-file routine.

Upon entry into the $$BCMT03 phase, a test determines if the phase was entered on a return from a message writer phase. If this is the case, a branch is made to the address (in register 14) specified by the response received from the operator for the message printed. This address is represented in the flowchart by the exit(s) from the decision block that follows a message block.

If the phase was not reentered from a message writer phase, a check of the DTF table determines the type of labels specified. If the file is unlabeled, that is, if neither standard nor nonstandard labels are specified in the DTF table, the end-of-file indicator (bit 4 of byte 35) is turned on and the first time switch (bit 0 of byte 21) is turned off in the DTF table. An exit is made to the user's end-of-file routine, EOFADDR.

If nonstandard labels are specified in the DTF table, an exit is made to the user's label routine (LABADDR) to process the labels. After the user's label routine is complete, or if the address of the user's label routine is not specified, the $$BCMT03 phase proceeds as described for unlabeled files.

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

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

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

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

Entry: From $$BCEOV1 (the EOF/EOV monitor), and $$BOMLTA to the label TRANSENT.

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

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

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

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

- File identification (FILEID) number (default = filename from the DTF table).

- Generation number (default = 0001).

- Version number (default = 01 for EBCDIC files and 00 for ASCII files).

- Expiration date (default = retention period set to 0 days).

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

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

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

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

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

$$BCMT05 and $$ECMT08: Close Standard, Nonstandard, and Unlabeled Files, All Types Except Work Files, Charts FE-FF

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

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

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

Exit: To the Clcse Monitor, $$BCLOSE, or to $$BCMT02.

Method: The $$BCMT05 phase closes standard, nonstandard, and unlabeled input or output files in either directior, fcrward or backward. The phase writes a standard EOF1 trailer label on output files when standard labels are specified and/or exits to the user's label routine to process user labels or nonstandard labels. Nc label checking is performed cn input files.

Upon entry into the $$BCMT05 phase, a check is made of the open indicator (bit 5 of byte 36) in the DTF table. If the bit is not on (file not open), the block count in the DTF table is set to zero, the first time switch (bit 0 of byte 21) in the DTF table is turned off, and the Close Monitor, $$BCLOSE, is fetched. If the open indicator is on, a test determines if the file is input or output.

If the file is an input file and EOF is not present, or American National Standard COBOL is not specified in the DTF, the tape is rewound, rewound and unloaded, or left positioned as specified in the DTF table by the user. If the tape is unloaded, the PUB2 OPEN bit is reset. The file is then posted closed in the DTF table. The The $$BCMT05 phase is concluded by resetting the DTF block count to zero and turning off the DTF first time switch. The Close Monitor, $$BCLOSE, is then fetched. If the file is a standard or nonstandard input file and American National Standard COBOL indicates that labels are processed at CLOSE time rather time rather than at EOF time, the COBOL label routine is called and the ECF label is processed.

If the file is an output file, a tapemark is written to indicate the end of the data area. For nonstandard labeled output files, an exit is made to the user's label routine (LABADDR), if specified. When the user's label routine is complete, or if the address of the user's label routine is not specified for nonstandard labels, or if the file is unlabeled, a tapemark is written and the DTF table is checked for the NORWD option. If the NORWD option is not specified, the tape is either rewound, or, rewound and unloaded, as the user requires. If NORWD is specified, the tape is backspaced over the last tapemark written. The $$BCMT05 phase is then concluded by posting the file closed, resetting the DTF block count to zero, turning off the first time switch in the DTF, and fetching the Close Monitor, $$BCLCSE.

For standard labeled output files, the version level of the DTF table must be determined by examination of bit 5 of byte 31 in the DTF table. If this bit is on, the table was generated by a Version 2.1 onward DTFMT macro and contains information needed to update the // TIBL card image in main storage.

The first step in the updating procedure inserts the file serial, volume sequence, and file sequence numbers from the DTF table into the corresponding entries of the // TLBL card image. The second step checks the fcllowing entries of the // TLBL card image and inserts the proper default value where blank entries are encountered:

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

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

If the DTF table was not generated by a Version 2.1 onward DTFMT macro, no modification of the // TLBL card image is necessary and the update procedure is bypassed.

A standard ECF1 label is then built from the // TLBL card image information and written on the tape. If user labels are specified in the DTF table, an exit is made to the user's label routine (LABADDR) to process and write the user's labels.

When all labels have been written, two tapemarks are written to indicate the end of the label area and the last file on the reel. If the user has specified the NORWD option, the tape is backspaced over the last tapemark written. On the other hand, if the NORWD option is not specified, the tape is either rewound, or rewound and unloaded, as the user requires. The file is then posted closed, the block count in the DTF table reset to zero, the first time switch in the DTF table turned off, and the Close Monitor, $$BCLOSE, fetched.

$$BCMT06: Close Tape Work Files, Chart FG

Objective: To close a tape work file.

Entry: From the Close Monitor, $$BCLOSE, or $$BOMLTA to the label WORKFILE.

Input: From the DTF table.

Exit: To the Clcse Monitcr, $$BCLOSE.

Method: This routine closes any magnetic tape work file. It checks to determine if the last tape operation was a read or a write. If write, a tapemark is written, and the rewind cption is tested. If read, the rewind cpticn is tested. In either case, if NORWD was specified, the rewind function is bypassed. If NORWD was not specified, the tape is either rewound to the loadpoint, or, rewound and unloaded, as the user requires. If the tape is unloaded, the PUB2 OPEN bit is reset. The block count in the DTF table is reset to zero, and the Close Mcritor is fetched.

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

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

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

Input: From the Supervisor I/O tables. (Refer to VSE/Advanced Functions Diagnosis Reference: Supervisor.

Exits:

1. To the message writer phase, $$BMSGWR, if an alternate drive is not available.

2. To the Job Control open phase, $$BJCOPT, to open the alternate assignment.

3. To the logical IOCS module if a unit exception occurs while writing a tapemark.

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

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

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

Note: Additional information concerning SVC 22 and the functions of the LUB, PUB, and JIB tables is found in VSE/Advanced Functions Diagnosis Reference: Supervisor.

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

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

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

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

$$BOMTOM and $$BOMTOW: Tape Open Message Writers, Chart FK

Note: The tape cpen message writer phases $$BOMTOM and $$EOMTOW perform the same function. The two phases differ only in the messages printed and the calling phases.

Objective: To print informational messages to the operator. For messages requiring operatcr response, exit to $$BOMSVA which in turn transfers control to the SVA message writer.

Entries:

1. Phase $$BCMTOM -- from tape open phase $$BOMT02, $$BOMT03, $$BOMT04, $$BOMT06.
2. Phase $$BCMTOW -- from tape open phases $$BOMT02, $$BOMT03, $$BOMT04, $$BOMT05, $$BCMT06, $$BCMT01, $$BCMT02, $$BCMT03, $$BCMT04, $$BCMT05, $$BCMT06, $$BCMT07.

Input: Pointer, received from an open routine, to indicate the desired message.

Exits:

1. To the calling open routine or to the Cpen Monitor ($$BOPEN), if a legal response to the message is received, or
2. To CANCEL if appropriate or if a console is not available.
3. To $$BOMSVA for action messages.

$$BOMTOM Messages:

4118I FILE ID ERROR, READBK
4120I TAPE POSITIONED WRONG
4123I WRONG POSITN, READBK
4124I TCO MANY UHL'S
4151I HDR1 LBL INFORMATION

$$BOMTOW Messages:

4113I NO HDR1 LBL FOUND
4151I HDR1 LBL INFORMATION
4883I INVALID LCGICAL UNIT (see Note)

Note: Message 4883I is issued by any tape open/close phase in which this error is encountered.

Method: The calling phase passes the following information to the message writers:

- Register 0. The number of the desired message.
- Register 12. The BYPASS reply return address.
- Register 13. The NEWTAP reply return address.
- Register 14. The IGNORE, RETRY, VOLUME SERIAL NO. reply, or the information type message return address.
- The last byte in the name of the phase returned to.

Each message contained in the message writer phases consists of five sections:

1. Option byte 1. Indicates the action to be taken after printing the message or the valid responses.

   Bit 0 = Automatic terminate (cancel).
   Bit 1 = IGNORE.
   Bit 2 = NEWTAP.
   Bit 3 = RETRY.
   Bit 4 = BYPASS.
   Bit 5 = Information message (no response required).
   Bit 6 = Volume serial number.
   Bit 7 = Print TLBL = XXXXXX.

2. Displacement byte. Contains displacement, in bytes, from the start of the label to the field(s) to be printed.
3. Length byte. Contains the length, in bytes, of the label information to be printed.
4. Message.
5. Option byte 2. The following bits in Option byte 2 indicate:

   Bit 0 = IGNORE is an invalid reply if access control is present in the system.

   Bit 1 = Change the message action indicator to information only with cancel if access control is present in the system.

   Bits 2 - 7 = Unused.

For informational messages, the message writer assembles and writes the required message. The phase then passes control to the proper open phase or to the cancel routine. If control is returned to an open phase, bit 4 cr byte 89 in the communicaticns region is set to indicate to the to the open phase that entry is made on a return from a message writer phase.

For action (A- and D-type) messages, the registers are saved and $$BOMSVA is fetched in order to exit to the SVA.

$$BMSGWR: Tape Cpen/Close Message Writer, Chart FI

Objective: To print informational messages to the operator. For messages requiring operator response, exit to $$BOMSVA which in turn transfers control to the SVA message writer.

Entry: From tape Open/Close phases:

$$BOMT05	$$BCMT03
$$BCMT01	$$BCMT07
$$BCMT02	

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

Exits:

1. To CANCEL.
2. To $$BJCOPT if entered from tape close phase $$BCMT07.
3. To $$BOMSVA for action messages.

Messages:

4122I EOV ENCOUNTERED
4125I VCL1 IBI FCUND
4126I EOV ENCOUNTERED

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

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

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

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

Each message contained in this phase consists of four sections:

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

   Bit 0 = EOV or EOF
   Bit 1 = IGNORE
   Bit 2 = NEWTAP
   Bit 3 = Double message
   Bit 4 = RETRY
   Bit 5 = Information message (no response required)
   Bit 6 = System unit possible and CANCEL not valid response
   Bit 7 = Leave DTF blank.

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

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

For action (A- and D-type) messages, the registers are saved and $$BOMSVA is fetched in order to exit to the SVA.

It is also possible for other modules to call the message writer and pass messages to it that are then issued by the $$BMSGWR message writer.

$$BONVOL: Standard Volume Label Rewriter, Chart FP

Objective: To rewrite the standard volume labels in user-specified density on a dual density tape unit.

Entry: Fetched by $$BOMT03, $$BOMT06, $$BJCOP1, and $$BOCPT3. The entry is made after a density discrepancy is found between an expired, standard labeled output tape and the assigned density.

Exit: Returns to the calling phase. The tape is positioned at the end of HDR1.

Method: $$BONVOL rewinds the tape and begins reading volume labels. As soon as the HDR1 label has been read, the tape is rewound, and the volume labels are rewritten at the specified density (either

800, 1600, or 6250 BPI). When all volume labels have been rewritten, a dummy header label of blanks is written. Control is then returned to the calling phase.

$$BOMSVA and $$BOMSV2: Transfer Control to SVA Message Writer, Chart FQ

Objective: Free up the B-transient area while waiting for operator response or while rewinding a magnetic tape.

Entry: From all OPEN/CLOSE transients that require a tape rewind operation and from all message writers.

Exit: To IJJGMSGT in the SVA.

512 bytes are obtained from the partition's GETVIS area. The calling routine's registers and the OPEN table are saved and the input from the various calling modules is converted and saved in the message writer I/F table and the message writer parameter list (MPL). The user save area is then saved and the user's PSW is changed to point to the IJJGMSGT entry point in the SVA. SVC 11 transfers control to this phase and frees the B-transient area.

$$BOMSVA

Function: To handle input from various B-transients in order to issue action messages and initiate tape rewind operations.

Exit: To $$BOMSV2 for message writing and to IJJGMSGT for tape rewind.

Input:

R0	Message code
R2	Pointer to the DTF
R3	Pointer to the TLBL
R4	Second half of the calling phase's name
R6	Function to be performed
R8	Phase to be called by $$BOMLTA
R12,R13,R14	Return points after different operator responses

$$BOMSV2

Function: Exits to the SVA writer to issue action message and read in the operator response.

Input:

R0	LTA message code
R2	DTF pointer
R4	OPEN table pointer
R6	Indicates tape or ISAM message
R8	Pointer to message writer I/F table
R9	Message writer load point
R7,R13,R14	Return points after operator responses

Output:

R2	DTF pointer
R5	Pointer to problem program
R8	Pointer to message writer I/F table
R13	Pointer to message writer save area
R15	IJJGMSGT entry point

Messages:

4110A	NO VOL1 LBL FOUND
4111A	NO VOL1 LBL FOUND
4112A	VOL SERIAL NO. ERROR
4113D	NO HDR1 LBL FOUND
4114A	FILE SEQ NO. ERROR
4115A	FILE SER NO. ERROR
4116A	VOLUME SEQ NO. ERROR
4117D	NO TM FOUND ON READBK
4118D	FILE ID ERROR, READBK
4119A	FILE UNEXPIRED
4123D	WRONG POSITN, READBK
4132D	ERROR IN FILE ID
4133D	ERROR IN HDR LBL

IJJGMSGT: Issue Error Message, Chart FR

Objective: To issue an error message, read operator response, and rewind tape if required.

Entry: From $$BOMSVA and $$BOMSV2

Exit: To $$BOMLTA or as designated by operator's response.

Input:

R2	DTF pointer
R5	Pointer to problem program save area
R8	Pointer to message writer I/F table (0 if entered from the SVA)
R10	Pointer to the MNPL if entered from the SVA
R13	Pointer to the message writer save area
R15	Entry point

$$BOMLTA: Return to Logical Transient Area, Chart FS

Objective: To return to the LTA after an action message has been issued or after a tape rewind command has been executed in the SVA; to issue a FREEVIS for space obtained by a GETVIS from $$BOMSVA; to restore the user save area; to set up return points, and then exit to the proper

B-transient phase.

Entry: From IJJGMSGT.

Exit: To next B-transient or Job Control.

Input:

R0 Return code from SVA
R8 Pointer to the message writer I/F table

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

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

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

COMPILER FILES

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

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

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

DTFCP Macro

A DTFCP macro instruction can be used for each file that has fixed, unblocked records, and limited device independence. When the file is opened, a channel program for reading/ writing on a particular device is built in the DTFCP table area (Figures 31, 32, and 33). Only the following devices can be accessed by DTFCP: IBM 1442, 1443, 1403, 3211, 3800, 3289-4, 3262, 2501, 2520, 2540, 2560, 3203, 3504, 3505, 3525, 2400/3400 series, 2311, 2314, 2319, 3310, 3330, 3340, 3350, 3370, 3540, 5203, and 5424/5425.

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

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

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

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

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

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

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

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

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

RECSIZE=n: This parameter specifies the number of characters in the record. I/O routines use this factor for checking record length. If this entry is not specified, the maximum record length of 81 bytes is assumed.

TYPEFLE=INPUT or OUTPUT: This parameter defines whether the file is an input or output file. From this specification, IOCS creates the appropriate channel program for this file. If this entry is not specified, an input file type is assumed.

EOFADDR=NAME: This parameter defines a problem program address to which IOCS branches on an end-of-file condition. This entry must be supplied if TYPEFLE=INPUT. The user must perform any necessary end-of-file processing; for example, issue a CLOSE macro.

DISK=YES, NO or omitted: If the DISK= parameter is omitted, a tape system is assumed, and the same DTF table (see Figure 33) is generated as in the tape system. If the DISK= parameter is present, the disk resident system is assumed. If DISK=YES, a DTF table (see Figure 31) is generated that supports any device including the IBM 2311, 2314, 2319, 3330, FBA devices, 3340, 3350, and 3540. If DISK=NO, a DTF table (see Figure 32) is generated for a disk resident system that supports a device other than DASD or diskette.

IOPTR=YES or omitted: If YES is specified, the I/O area address must be put into register 0 before a GET or a PUT is issued. The module inserts the contents of the register into the data address portion of the CCW and proceeds as in the case of one I/O area. The parameter must be included if a variable I/O area is required.

RDONLY=YES or omitted: If YES or blank, the file uses a reentrant CP logic module. Unless RDONLY is omitted, the problem program must provide the logic module with the address of the unique 72-byte, doubleword aligned, save area for the file. This address is supplied in general register 13 before a logical IOCS imperative macro is issued to the file.

CISIZE=n: If the logical unit is assigned to an FBA device, this parameter specifies the size of the CI.

ASCII=YES: If ASCII=YES, ASCII records are supported. DISK=YES must be specified; the IOAREA2, IOPTR, and RDONLY parameters must be omitted; and system logical units must not be specified in the DEVADDR parameter. The macro generates the module name IJJCPA1N. This module is used for FORTRAN ASCII files. (Refer to CPMOD: PUT Macro, one I/O area detail charts.)

CHARACTERISTICS OF DTFCP FILES

1. Only fixed unblocked records formats are supported.

2. Forward read only is supported. No read backwards is permitted.

3. Employs standard physical IOCS error recovery procedures.

4. American National Standard printer control character is always required for printer and punch files.

5. GET and PUT are supported.

6. Multivolume output of SYSLST and SYSPCH on tape is supported.

7. Variable I/O area is supported if IOPTR is specified. For example, any area pointed to by register 0 when a GET or PUT is issued is considered the I/O area.

8. Multivolume input and output on diskettes is supported.

9. If special records (deleted or sequentially relocated records) are read, they are skipped (not passed to the user).

10. For a multivolume diskette input file processing continues until all extents are exhausted. No volume sequence number or multivolume indicator checking is done.

Bytes*	Bits	Function
8-byte header preceding the DTF		
0-3		Reserved
4-7		Address of the Version 3 area of the DTF[1]
0-15 (00-0F)		CCB. If the file is on a DASD device, the CCW address in bytes 9-11 (09-0B) is changed by OPEN to point to a DTF extension in the user virtual save area. CLOSE restores it. If 3800 extended buffering is selected, the CCW address is changed by OPEN to point to a DTF extension work area in the user virtual area. CLOSE restores it.
16 (10)	0	Not used.
	1	Set by MAINT; indicates that LIOCS must retrieve extents from the VTOC instead of the label cylinder.
	2	COBOL open; ignore option.
	3	X'10' indicates an unlabeled FORTRAN tape.
	4	DTF table address constants relocated by OPENR.
	5	Used by FORTRAN (Sequential Disk Backspace and Rewind).
	6	1 = ASCII, 0 = EBCDIC.
	7	FORTRAN is calling DTFCP.
17-19 (11-13)		Logic module address. If the file is on a DASD device, OPEN changes this address to point to the logic module residing in the system virtual area (SVA). CLOSE restores it. If 3800 extended buffering is selected, OPEN changes this address to point to the extended buffering logic module IJDPR3 in the system virtual area. CLOSE restores it.
20 (14)		DTF type X'32', except in the case of disk assigned to units other than SYSLNK. In this case, DTFCP open changes it to X'20'.
21 (15)		Cpen indicators: X'02' input, X'00' output, except for tapes assigned to SYS000 to SYSnnn when X'00' = input and X'08' = output. X'08' DISK=YES indicator.
	0	1 = no rewind, 0 = rewind.
22-28 (16-1C)		Filename (see byte 29).
29 (1D)		Device type code: X'00' = 2311 X'08' = 3340 general X'01' = 2314, 2319 X'09' = 3340 35MB X'04' = 3330-1,-2 X'0A' = 3340 70MB X'05' = 3330-11 X'45' = 3800 with TRC X'07' = 3350 X'90' = 3310, 3370
30-35 (1E-23)		File address for disk; block count if bit 7 of byte 16 is on.
36-37 (24-25)		Volume sequence number or work area.
38 (26)		Open switch.
39 (27)		Sequence number of current extent.
40 (28)		Sequence number of last extent, or X'80' if 1442 punch.
41 (29)		X'80' indicates request for standard label tape OPEN.

[1] The Versicn 3 area is described in VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 4, SAM for DASD.

Figure 31. DTFCP: DISK=YES (Part 1 of 3)

Bytes*	Eits	Function
42 (2A)		X'80' device is a 2560. X'40' DTF has been extended into the user virtual save area. X'20' device is a DASD. X'10' device is a tape. X'08' device is a printer X'04' device is a punch X'02' device is a reader X'01' RPS is supported
43 (2B)		X'F3' device is a DASD. X'F1' device is a reader. X'F0' device is other type.
44 (2C)	0	1 = input, 0 = output.
	1	1 = eject needed for a reader punch; 0 = no eject.
	2	0 = first pass, 1 = not first pass.
	3	1 = two I/O areas, 0 = one I/O area.
	4	1 = 2540 punch.
	5	1 = SYSLST or SYSPCH.
	6	1 = SYSLST or SYSPCH on output tape.
	7	1 = TLBL is present and tape is labeled.
45-47 (2D-2F)		IOAREA2 address.
48 (30)	0	1 = Always on.
	1	Reserved for future use.
	2	1 = Version 3 DTF.
	3-7	Reserved for future use.
49-51 (31-33)		Reserved for future use.
52-53 (34-35)		Lower head limit.
54-57 (36-39)		Extent upper limit.
58-64 (3A-40)		BBCCHHR seek address or physical block number for FBA DASD.
65-67 (41-43)		EOF address.
68-71 (44-47)		Control bucket CCHH; not used for FBA DASD.
72 (48)		Number of records per track for output, number of record per track +1 for input.
73 (49)		X'00' for output, X'01' for input.
74-75 (4A-4B)		X'0020' for output, X'0018' for input for DASD. X'0008' for 2560 and 5424/5425 output. X'0000' for nondisk device.
76-80 (4C-50)		CCHHR for count field; not used for FBA DASD.
81 (51)		Key length.
82-83 (52-53)		Data length.
84-87 (54-57)		Instruction to load user I/O area address to I/O register.

Figure 31. DTFCP: DISK=YES (Part 2 of 3)

Bytes*	Bits	Function
End-of-table if DTF is defined for an input file.		
88-111 (58-6F)		Seek, search, TIC CCWs; not used for FBA DASD.
112-119 (70-76)		CCW for DASD input and first CCW for DASD output; not used for FBA DASD. This CCW can be used for other devices if unit is not a DASD.
120-127 (77-7F)		Second CCW for output.
128-151 (80-97)		Verify CCWs for output.
End-of-table if DTF is defined for output file and DEVADDR does not equal SYSPCH.		
152-159 (98-9F)		2540 punch error recovery CCW 1.
160-167 (A0-A7)		2540 punch error recovery CCW 2.
168-231 (A8-E7)		Reserved.
When the CP open initializes the table and determines that the device is a 2540 punch, the following bytes in the table are changed:		
30 (1F)		X'FF' indicator to DTFCP open phases and logic module.
32-35 (20-23)		Instruction to load user I/O area to I/O register.
48-55 (30-37)		CCW.
56-63 (38-3F)		2540 punch error recovery CCW 1.
64-71 (40-47)		2540 punch error recovery CCW 2.
72-151 (48-97)		80-byte card image, savearea 1.
152-231 (98-E7)		80-byte card image, savearea 2.
When the CP open initializes the table and determines that the device is a 2560 or 5425, the following bytes in the table are changed:		
32-35 (20-23)		Instruction to load user I/O area to I/O register.
48-55 (30-37)		First output CCW.
56-63 (38-3F)		Second output CCW.
64 (40)		Stacker select character V for ASCII.
65 (41)		Stacker select character W for EBCDIC.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 31. DTFCP: DISK=YES (Part 3 of 3)

Bytes*	Bits	Function
0-15 (00-0F)		CCB.
16 (10)	0-1	Not used.
	2	COBOL open; ignore option.
	3	Not used.
	4	DTF table address constants relocated by OPENR.
	5	Not used.
	6	1 = ASCII (used only if DISK=YES), 0 = EBCDIC (used only if DISK=YES).
	7	FORTRAN is calling DTFCP.
17-19 (11-13)		Logic module address.
20 (14)		DTF type X'32' except in the case of tape assigned to units SYS000 to SYSnnn. In this case, a DTFCP open phase changes it to X'10'.[1]
21 (15)		Open indicators X'02' input, X'00' output (except for tapes assigned to SYS000 to SYSnnn when it is X'00' input, X'08' output).
22-28 (16-1C)		Filename (see byte 29).
29 (1D)		Device type code: X'45' = 3800 with TRC.
30 (1E)		Indicator to DTFCP open phases and logic module. X'FF' for input files. X'00' for output files.
31 (1F)		Reserved for future use.
32-35 (20-23)		Instruction to load user's I/O area address into I/O register.
36-37 (24-25)		Volume sequence number or work area.
38 (26)		Open switch.
39 (27)		Sequence number of current extent.
40 (28)		Sequence number of last extent, or X'80' if 1442 punch.
41 (29)		X'20'
42 (2A)		X'80' device is a 2560. X'40' device is a 5424/5425. X'10' device is a tape. X'08' device is a printer. X'04' device is a punch. X'02' device is a reader.
43 (2B)		X'F1' device is a reader or tape. X'F0' device is other type.

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

Figure 32. DTFCP: DISK=NO (Part 1 of 2)

Bytes*	Bits	Function
44 (2C)	0	1 = input, 0 = output.
	1	1 = eject needed for a reader-punch, 0 = no eject.
	2	1 = not first pass, 0 = first pass.
	3	1 = two I/O areas, 0 = one I/O area.
	4	1 = 2540 punch.
	5	1 = SYSLST or SYSPCH.
	6	1 = SYSLST or SYSPCH on output tape.
	7	Reserved for future use.
45-47 (2D-2F)		IOAREA2 address.
48-55 (30-37)		CCW.
End-of-table is DTF is defined as output file and DEVADDR is not equal to SYSPCH.		
56-63 (38-3F)		2540 punch error recovery CCW 1.
64-71 (40-47)		2540 punch error recovery CCW 2.
65-67 (41-43)		EOF address, input only.
End-of-table if DTF is defined as input file.		
72-151 (48-97)		80-byte card image, save area 1.
152-231 (98-E7)		80-byte card image, save area 2.
If device is a 2560 or 5424/5425, bytes 56 onward contain the following information:		
56-63 (38-3F)		Second output CCW.
64 (40)		Stacker select character V for ASCII.
65 (41)		Stacker select character W for EBCDIC.
66-75 (42-4B)		Reserved for future use.
76-235 (4C-EB)		First I/O area.
236-237 (EC-ED)		Reserved.
238-317 (EF-13D)		Second I/O area.
318-319 (13E-13F)		Reserved.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 32. DTFCP: DISK=NO (Part 2 of 2)

Input cr output table generated when the parameter DISK= is omitted from the DTFCP macrc (Tape Resident System Type).

Bytes*	Bits	Function
0-15 (00-0F)		CCE.
16 (10)	0-1	Not used
	2	COBOL open; ignore option.
	3	Not used.
	4	DTF table address constants relocated by OPENR.
	5	Not used.
	6	1 = ASCII (used only if DISK=YES), 0 = EBCDIC (used only if DISK=YES).
	7	Used by FCRTRAN.
17-19 (11-13)		Logic module address.
20 (14)		DTF type X'31' except in the case of tape assigned to units SYS000 to SYSnnn. In this case DTFCP open phase changes it to X'10'.
21 (15)		Open indicators X'02' input, X'00' output (except for tapes assigned to SYS000 to SYSnnn when it is X'00' input, X'08' output).
22-28 (16-1C)		Filename.
29 (1D)		Device type code: X'45' = 3800 with TRC.
30 (1E)		X'00' indicator to DTFCP open phases and logic module.
31 (1F)	0	1 = input, 0 = output.
	1	1 = eject needed for a read punch, 0 = no eject.
	2	1 = not first pass, 0 = first pass.
	3	1 = two I/O areas, 0 = one I/O area.
	4	1 = 2540 punch.
	5	1 = SYSLST or SYSPCH
	6	1 = SYSLST or SYSPCH on output tape.
	7	1 = TLBL specified and tape is labeled
32 (20)		Cpen indicators.
33-35 (21-23)		IOARFA2 address.
36-39 (24-27)		Instruction to load user's I/O area address into I/O register.
40-47 (28-2F)		CCW.

Figure 33. DTFCP: DISK= Parameter Omitted (Part 1 of 2)

Bytes*	Bits	Function
End of table if DTF is defined as output file and DEVADDR is not equal to SYSPCH.		
48-55 (30-37)		2540 punch error recovery CCW 1.
56-63 (38-3F)		2540 punch error recovery CCW 2.
57-59 (39-3B)		EOF address, input only.
End of table if DTF is defined as input file.		
64-143 (40-8F)		80-byte card image, save area 1.
144-223 (90-13F)		80-byte card image, save area 2.
For 2560 and 5424/5425 bytes 48 onwards contain the following information:		
48-207 (30-CF)		IOAREA1.
208-209 (D0-D1)		Reserved.
210-369 (D2-171)		IOAREA2.
370-371 (172-173)		Reserved.
372-451 (174-1C3)		Compare area.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 33. DTFCP: DISK= Parameter Omitted (Part 2 of 2)

CPMOD Macro

The CPMOD macro is used with IBM internal programs only; and is not documented in System Library publications, (see the section "Compiler Files"). VSE/Advanced Functions, Release 2, provides a preassembled superset logic module that is loaded in the SVA during IPL; therefore, CPMOD is not used for DASD files because OPEN selects the appropriate logic module in the SVA and links it with the DTF via the DTF extension.

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

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

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

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

NAME: If NAME is specified, it is generated as the CSECT name of the module. No other entry points are generated.

TYPEFLE=INPUT or OUTPUT: If INPUT is specified, IOCS generates a module to handle input only. If OUTPUT is specified, IOCS generates a module that handles both input and output. If the TYPEFLE parameter is omitted, OUTPUT is assumed.

IOAREA2=YES or Omitted: If YES is specified, the generated logic module provides automatic switching between two

I/O areas. If this parameter is omitted, one I/O area is assumed.

IOPTR=YES or Omitted: If YES is specified, the I/O area address is assumed to be in register 0. This parameter allows the user to change the I/O area address used with 1 I/O area. If IOPTR is specified in CPMOD, it must also be specified in the DTFCP macro.

RETRY=NO or Omitted: If NO is specified, the IOCS error recovery procedures for the IBM 2520 and the IBM 2540 punch devices are eliminated from the generated module. If the parameter is omitted, the 2520 and the 2540 punch error recovery procedures are included on the generated module. When the file is defined for input, this parameter is ignored.

DISK=YES or Omitted: If YES is specified, it is ignored; OPEN selects the appropriate logic module in the SVA and automatically links it to the DTF (via the DTF extension). If omitted, the module generated is the same as in the tape resident system.

RDONLY=YES or Omitted: If omitted, the module generated by the CPMOD macro is not reenterable. To be reenterable, general register 13 is used to point to a 72-byte, doubleword aligned, save area unique to the particular task using the module. In this way several tasks can asynchronously use the same module if each task provides its own save area.

ASCII=YES or Omitted: If ASCII=YES, both EBCDIC and ASCII records are supported, and the macro generates the module name IJJCPA1N. DISK=YES and RETRY=NO must be specified, and the IOAREA2, IOPTR, and RDONLY parameters must be omitted. If TYPFLE= is specified, TYPEFLE must equal OUTPUT. If, however, TYPEFLE is omitted, output is assumed as a default value. If ASCII= is omitted or equals any other value except YES, ASCII records are not supported. For FORTRAN ASCII files, refer to CPMOD: PUT Macro, one I/O area detail charts.

TRC={YES|NO}: This operand applies to the IBM 3800 Printing Subsystem. YES specifies that a table reference character may be included as the first byte of each output data line (following the optional print control character). The printer uses the table reference character (0, 1, 2, or 3) to select the character arrangement table corresponding to the order in which the table names have been specified with the CHARS parameter on the SETPRT job control statement (or SETPRT macro instruction).

For a DTFCP file, the presence of table reference characters is specified with the TRC=Y parameter on the SETPRT job control statement (or SETPRT macro instruction).

If the TRC parameter is specified, TYPEFLE=INPUT must not be specified.

The IBM-supplied preassembled logic modules do not have TRC=YES. The system programmer can reassemble them with TRC=YES to support 3800 table reference characters. If some, but not all CPMOD logic modules are reassembled this way, it may interfere with subsetting or supersetting.

If extended printer buffering is used for all DTFCP files printed, then the CPMOD module is not needed.

Note: When using CPMOD be sure to issue a new LTORG for each CSECT of your program which uses literals because CPMOD may have issued a different LTORG for its own literals.

CPMOD: GET Macro, Two I/O Areas, Chart NA

Objective: To cause two I/O areas to be loaded from an input device and to allow overlap.

Entry: From a GET macro expansion issued by the problem program.

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

Method: This routine is similar to the PUT two I/O areas routine, except it does not contain the logic to support a printer or punch. It contains only the logic necessary to retrieve records from an input file.

During the first pass, both I/O areas are read into. Overlap is made possible by allowing the second I/O area to be filled while the first one is being processed. A wait for I/O complete is issued to allow the previous read to be completed before processing the data of that area.

When an end of file is sensed, the user's registers are restored and a branch to the user's EOF routine is made.

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

All user's registers are saved upon entry into CPMOD and are restored when control is returned to the problem program.

CPMOD: GET Macro, One I/O Area, Chart NB

Objective: To read a logical record.

Entry: From a GET macro issued by the problem program.

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

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

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

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

CPMOD: GET Macro, IOPTR=YES, Chart NB

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

Entry: From a GET macro expansion.

Exit: To the problem program or EOF address.

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

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

Objective: To write a logical record with overlap.

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

Exits:

1. To the problem program.
2. To the user's EOF address, if EOF occurs on an output device other than a printer.
3. To $$BCMT07 if EOF occurs on an output tape assigned as SYSLST or SYSPCH.
4. To $$BERRTN error recovery routine, if an error occurs on a 2540 or 2520 punch.
5. To $$BERRTN if the upper extent is exceeded on a DASD file.
6. To $$BOPEN to get a new extent if the upper extent is exceeded on a diskette file.

Method: If entered for the first time, the PUT routine performs an I/O operation immediately. (After the initial entry, this I/O operation is bypassed.) When the I/O operation is complete, the routine checks for an EOF condition. If an end of file is detected on a unit record device (other than a printer), control is passed to the user's end-of-file routine. If an end of file is detected on a magnetic tape device assigned to SYSPCH or SYSLST, the PUT routine fetches phase $$BCMT07 to determine if an alternate device is available. If it is not necessary to handle an EOF condition, the addresses of the two I/O areas are exchanged, another I/O operation is performed, and control is returned to the problem program.

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

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

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

CPMOD: PUT Macro, One I/O Area, Chart NE-NF

Objectives:

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

Entry: From PUT macro expansion, or from an SD work file close.

Exits:

1. To the problem program.

2. To the user's EOF address if EOF occurs cn an input device.

3. To $$BCMT07 if EOF occurs on an output tape assigned as SYSLST or SYSPCH.

4. To $$BEPRTN error recovery routine if an error cccurs on a 2540 or 2520 punch.

5. To $$BERRTN if the upper extent is exceeded on a DASD file.

6. To $$BOPEN tc get a new extent if the upper extent is exceeded on a diskette file.

Methcd: The PUT routine saves the user's registers and then checks for various devices. If the device is a tape unit, the I/O operation is performed immediately. If the device is a DASD or diskette, the specified limits are checked and the seek address and count are updated.

When an end-of-extent condition is found on a diskette and no more extents are available, $$BOPEN is fetched to get another extent, and processing continues.

If the device is a printer or punch, the American National Standard printer control character is picked up and converted to EBCDIC coding (refer to VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 1 for ASCII).

Next, the specified control operation is performed. TRC processing will occur at this time if indicated in the DTFCP. When the control operation is complete, the I/O operation indicated by the PUT macro instruction is performed. The user's registers are restored and the routine returns control to the problem program.

Error and EOF conditions are handled in the same manner as described in the PUT routine for two I/O areas.

CPMCD: PUT Macro, IOPTR=YES, Chart NG-NH

Objective: To write a logical record from the I/O area pointed to by register 0 (IOPTR).

Entry: From a PUT macro expansion or from $$BOSDC1 - Close SD work file.

Exits:

1. To the problem program.

2. To the user's EOF routine if EOF occurs cn punch or DASD device.

3. To $$BCMT07 if EOF occurs on an output tape assigned as SYSLST or SYSPCH.

4. To $$BERRTN error recovery routine if an error cccurs on a 2540 or 2520 punch.

5. To $$BERRTN if the upper extent is exceeded on a DASD file.

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

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

DEVICE INDEPENDENT SYSTEM FILES

Device Independent (DI) system files are files assigned to the system logical units SYSRDR, SYSIPT, SYSPCH, and SYSLST. The files are defined by a DTFDI macro for fixed unblocked records on the following IBM devices:

- 1403 Printer
- 1404 Printer
- 1442 Read punch
- 1443 Printer
- 2311 Disk Storage Drive
- 2314 Direct Access Storage Facility
- 2319 Disk Storage
- 3203 Printer
- 3211 Printer
- 3262 Printer
- 3289-4 Printer
- 3310 Disk Storage Device
- 3330 Disk Storage
- 3340 Direct Access Storage Device
- 3350 Direct Access Storage
- 3370 Disk Storage Device
- 2400-series or 3400-series Magnetic Tape Unit - forward read only.
- 2501 Card Reader
- 2520 Card Read Punch
- 2560 MFCM
- 3504 Card Reader
- 3505 Card Reader
- 3525 Card Punch
- 3800 Printer
- 5203 Printer
- 5424/5425 Multifunction Card Unit.

DTFDI Macro

The DTFDI macro defines the file for device independent system units and generates a table containing the information necessary to describe the file for processing by the DIMOD logic module (see Figure 34).

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

Bytes*	Bits	Function
0-15 (00-0F)		CCB. If the file is on a DASD, the CCW address in bytes 9-11 (09-0B) is changed by OPEN to point to the DTF extension. CLOSE restores it. If 3800 extended buffering is selected, the CCW address is changed by OPEN to point to a DTF extension work area in the user virtual area. CLOSE restores it.
16 (10)	0-1	Not used.
	2	COBOL open; ignore option.
	3	Not used.
	4	DTF table address constants relocated by OPENR.
	5-7	Not used.
17-19 (11-13)		Address of logic module. If the file is on a DASD, OPEN changes this address to point to the device independent logic module in the system virtual area. CLOSE restores it. If 3800 extended buffering is selected, OPEN changes the address to point to the extended buffering logic module IJDPR3 in the system virtual area. CLOSE restores it.
20 (14)		DTF Type = X'33'
21		Open/Close indicators - X'82' = input, X'80' = output.
(15)	0	Always set on for no rewind.
22-28 (16-1C)		Symbolic filename.
29 (1D)		DASD or diskette device indicators X'00' = 2311 X'01' = 2314, 2319 X'04' = 3330-1,-2 X'05' = 3330-11 X'07' = 3350 X'08' = 3340 general X'09' = 3340 35MB X'0A' = 3340 70MB. X'90' = 3310, 3370
30-35 (1E-23)		DASD address of format-1 label.
36-37 (24-25)		DASD or diskette volume sequence number.
38		Cpen communications switch.
(26)	0	1 = Nc more extents -- diskettes
	1-3	Not used
	4	Always 1
	5-7	Not used.

Figure 34. DTFDI (Part 1 of 3)

Bytes*	Bits	Function
39 (27)		Sequence number of current extent.
40 (28)		Sequence number of last extent, or X'80' for 1442 reader punch.
41 (29)		Open indicator = X'20'.
42 (2A)		Device type indicators:
	0	Unused
	1	1 = DTF has been extended into the partition GETVIS area
	2	1 = DASD
	3	1 = tape
	4	1 = printer
	5	1 = punch
	6	1 = reader
	7	1 = RPS supported.
43 (2B)		Logic module device indicators: X'F3' = DASD or diskette device X'F1' = reader or tape device X'F0' = other type device
44 (2C)		Logic module option switches
	0	1 = input; 0 = output.
	1	1 = eject for RDR-PCH; 0 = no eject.
	2	1 = not first pass; 0 = first pass.
	3	1 = two I/O areas; 0 = one I/O area.
	4	1 = 2540 Punch.
	5	1 = SYSLST/SYSPCH.
	6	1 = Tape SYSLST/SYSPCH.
	7	1 = ASCII; 0 = EBCDIC code.
45-47 (2D-2F)		Alternate I/O area address.
48 (30)		Logic flags:
	0-1	Reserved.
	2	1 = Version 3 DTF.
	3-4	Reserved.
	5	1 = TRC=YES specified on DTF
	6	1 = TRC in effect
	7	1 = 3800 printer
49-51 (31-33)		Reserved.
52-53 (34-35)		Extent lower head limit.
54-57 (36-39)		Extent upper head limit.
58-64 (3A-40)		DASD seek address. Diskette seek address at byte 60 (3C).
65-67 (41-43)		Users EOF address.
68-72 (44-48)		Control bucket CCHHR. Byte 72 (48) always X'01' for diskettes.

Figure 34. DTFDI (Part 2 of 3)

Bytes*	Bits	Function
73 (49)		Logic module switches X'01' = input X'00' = output X'00' = both input and output on diskettes.
74-75 (4A-4B)		Logic module constants X'0020' DASD output X'0018' DASD input X'0008' Diskette devices X'0000' Non-DASD devices
76-80 (4C-50)		Count field CCHHR (OCHRO for diskettes); not used for FBA DASD.
81 (51)		Key length.
82-83 (52-53)		Data length.
84-87 (54-57)		Instruction to load IOREG with correct I/O area address.
88-103 (58-67)		Seek, Search CCWs; not used for FBA DASD. Seek, Read/Write CCW for diskette files.
104-111 (68-6F)		TIC CCW. NOP CCW for diskette output files; unused for diskette input files.
112-119 (70-77)		Input/output CCW.
120-127 (78-7F)		Second output CCW.
128-151 (80-97)		Verify CCWs for output.
152-159 (98-9F)		Error CCW1.
160-167 (A0-A7)		Error CCW2.
168-231 (A8-E7)		Save area (64 bytes).
232-235 (E8-EB)		DC A(WLRERR) if WLRERR=Address. B 28(15) if ERROPT= omitted. B 25(15) if ERROPT=SKIP. B 28(15) if ERROPT=IGNORE.
236-239 (EC-EF)		DC A(ERROPT) if ERROPT=Address. B 0(15) if ERROPT= omitted. B 24(15) if ERROPT=SKIP. B 28(15) if ERROPT=IGNORE.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 34. DTFDI (Part 3 of 3)

DIMOD Macro

Device independent files provide users of the Assembler language with the same capabilities available to users of COBOL, FORTRAN, and PL/I in files defined by the DTFCP macro. Like the CP logic module, the device independent module generated by a DIMOD macro provides the logic to accomplish the logical IOCS functions of GET and PUT. For DASD, VSE/Advanced Function, Release 2, provides a preassembled superset logic module that is loaded into the SVA during IPL; therefore, this macro is ignored by OPEN in the case of DASD files. Instead, the appropriate logic module is selected by OPEN and linked to the DTFDI via the DTF extension in the user virtual area.

DIMOD: GET Macrc, One I/C Area, Chart PA

Objective: To read the next sequential logical record into an input area specified by the user.

Entry: From a GET macrc expansion.

Exits:

1. To the next sequential instruction following the GET macro expansion in the problem program after a record is read intc the input area.
2. Tc the user's EOF routine if an end-of-file condition occurs.
3. To cancel via an SVC 50 if an error cccurs and the ERROPT parameter is not specified in the DTFDI macro.
4. Tc $$BOPEN tc get a new extent if the upper extent limit is exceeded on a diskette file.

Method: The GET routine first determines the device type specified in the DTFDI table. If the device is DASD or diskette, the routine validates the record number and updates the CCHHR seek address if necessary. The routine then issues an SVC 0 and waits for the I/O operation to finish. If the device is DASD or diskette, the record number is increased by one at this time in preparation for the next GET.

A check is made for error and end-of-file conditions. Errors are handled in the manner specified by the user in the DTFDI macro parameters. An end-of-file conditicn causes the GET routine to exit to the address cf user's EOF routine instead of to the address of the next sequential instruction in the problem program.

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

DIMOD: GET Macrc, Two I/C Areas, Chart PB

Objectives: To read the next sequential logical record into an input area specified by the user, and to provide processing overlap by alternately reading records into one of the specified input areas and then into the other.

Entry: From a GET macro expansion.

Exits:

1. To the next sequential instruction in the problem program following the GET macro expansion.
2. To the user's EOF routine if an end-of-file condition occurs.
3. To cancel via an SVC 50 if an error occurs and the ERROPT parameter is not specified in the DTFDI table.
4. To $$BOPEN to get a new extent if the upper extent limit is exceeded on a diskette file.

Method: The GET routine, when two I/O areas are employed, functions in the same manner as a GET with one I/O area. The difference between the two routines is the utilization of two I/O areas by alternately exchanging the addresses of the two areas each time a GET macro is issued. Therefore, to the user it appears that the record is available in the same input area for every GET.

DIMOD: PUT Macro, One I/C Area, Chart PC

Objective: To write the next sequential logical record to the file from a user specified output area.

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

Exits:

1. To the next sequential instruction in the problem program following the PUT macro expansion after the record is written to the file.
2. To phase $$BCMT07 if an EOF condition is reached on an output tape assigned to either SYSLST or SYSPCH.
3. To phase $$BERRTN if a punch error occurs.
4. To the user's EOF routine if an EOF condition is reached on devices other than diskette, or SYSLST or SYSPCH assigned to an output tape.
5. To the user's WLR or ERROPT routine, if specified in the DTFDI macro parameters, to handle other error conditions.
6. To $$BOPEN to get a new extent if the upper extent limit is exceeded on a diskette file.

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

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

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

3. If the device is a printer or punch, the contrcl code is determined, converted to EBCDIC code if ASCII is specified (refer tc "Appendix C"), and the control function is performed. When the control function is complete, TRC processing will occur if indicated in the DTFDI. An SVC 0 is then issued to write the record.

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

1. Fcr printers, the unit exception is ignored.

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

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

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

DIMOD: PUT Macro, Two I/O Areas, Charts PE-PF

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

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

Exits:

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

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

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

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

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

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

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

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

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

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

Exits:

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

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

3. Job cancelation for CANCEL response.

4. To the calling routine for IGNORE response.

Method: Phase $$BERPTP first determines the type of paper tape code used.

1. If the code used is a shifted code and if there is no conflict between the delete and the shift characters, the character in error is repunched along with the remaining part of the record.

2. If the code used is a shifted code and if there is a conflict between the delete and the shift characters, the character in error is repunched along

with the remaining part of the record, preceded by its corresponding shift character. After three unsuccessful attempts, the job is automatically cancelled and message 4P01I is printed.

3. If the code used is a non-shifted code, the character in error is repunched along with the remaining part of the record. After three unsuccessful attempts, message 4P02D is printed, allowing the operator to CANCEL the job or IGNORE the error.

   Control returns to the calling routine via an SVC 11 (PTMOD) or an SVC 2 ($$BCLOSE).

4. In all cases, the character in error is overpunched with the user-specified DELETE character.

$$BERRTN: Punch Error Recovery, Chart PG

Objectives: To attempt a recovery from a punch error condition on either a 2520 or 2540 punch, or to cancel the job when an extent is exceeded while writing on a DASD.

Entry: From an output DIMOD, or an output CPMOD if RETRY=NO is not specified.

Exit: To the calling routine after the error is corrected or to the message writer phase $$BOMSG1 if an extent is exceeded.

Method: Phase $$BERRTN first determines the type of device on which the error occurred.

If a punch error occurred on a 2520, the message 4000I is printed to indicate that a retry is being made and the card in error is repunched.

If a punch error occurred on a 2540, the card in error is repunched, the retry message 4000I is printed, and the card following the error card is repunched. (It is necessary to repunch the card following the error card because it is in the process of being punched when the punch error is detected and both cards are stacker selected.)

Control returns to the calling routine via an SVC 11 when the punch error is corrected.

If an error occurred on a disk device, preparation is made to print message 4887I and the message writer phase $$BOMSG1 is fetched.

INITIALIZATION AND TERMINATION

Device independent files are opened by the same logical transients used to open compiler files. The close routine for a DTFDI file provides for reinitialization of the DTF table. This procedure removes from the DTFDI files the restriction on reopening imposed on DTFCP files.

$$BOCP01: Open Device Independent Files - Phase 1, Chart QA

Objective: To open device independent files defined by either DTFCP or DTFDI (Version 1 DTFCP excluded).

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

Exits:

1. To $$BOCP02 if the DTF is a tape resident system type DTFCP. (This DTF table type is generated if the DISK= parameter is omitted from the DTFCP macro.)

2. To $$BOCP03 if the device is a unit record device.

3. To $$BOCPT1 if the device type is tape and the file type is input.

4. To $$BOCPT2 if the device type is tape and the file type is output.

5. To $$BOPEN if the device type is DASD or additional files remain to be opened.

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

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

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

9. To $$BOESTV if TES processing for tape is required.

Method: This routine is called by the Open Monitor when the file is found to be a DTFCP or DTFDI type (Version 1 DTFCP excluded). Because the logic modules for both file types provide for device independence, this phase is needed to determine the device type associated with the logical unit. The PUB is located for the logical unit and the DTF is initialized according to the device type (see Figures 31, 32, and 33).

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

If the device is tape, a check

determines if Jcb Control has already opened the device. If the file is open and if there are more files tc open, $$BOPEN, is fetched. If no more files remain to be opened, control returns to the problem program. If the file has not been opened, $$BOCPT1 is called to open the input files, and $$BCCPT2 is called to open the output files.

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

$$BCCP02: Open Device Independent Files - Phase 2, Chart CB

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

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

Entry: From $$BCCP01, or from $$BOMSG1.

Exits:

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

2. To the prcblem prcgram, if no more files remain to be opened.

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

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

   4883I INVAIID IOGICAL UNIT
   4884D NEED FILE PROTECT RING

5. To $$BOPR3 if the device type is a 3800 printer.

Methcd: This phase functions in the same manner as $$BOCP01, except when it is entered from a message routine. If this is the case, a sense command is issued and a test is made fcr file prctect. If the file is not file protected, $$BOMT06 is fetched. If it is, a message is printed, via $$BOMSG1. If the file being opened is assigned to a 3800 printer, then this phase exits tc $$BOPR3 tc complete open processing for the 3800.

$$BOCP03: Open Device Independent Files - Phase 3, Chart QC

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

Entry: From $$BOCP01.

Exits:

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

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

3. To $$BOPR3 if the device type is a 3800 printer.

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

If the device is a 1403, 3203, or 5203 printer with the Universal Character Set (UCS), a set mode command is given to suppress data checks. If the file being opened is assigned to a 3800 printer, then this phase exits to $$BOPR3 to complete open processing for the 3800.

If the device is a 2560 or 5424/5425, the command code required for the specified hopper is inserted in the first CCW for input files. For output files, two CCWs are loaded; the first for stacker selection, the second for the punch operation.

This phase functions in the same manner as $$BOCP01 to open unit record files.

$$BOPR3: Open Processing for the 3800 Printer File, Charts QG-QH

Objective: To set up for the 3800 printer open initialization routine, IJDPR3 + 28.

Entry: From $$BOUR01, $$BOCP02, or $$BOCP03 when a 3800 printer file is being opened.

Exits: To the problem program if no files remain to be cpened. To phase 1 of Open processing ($$BOPEN) if the open parameter list contains more files to be opened.

Method: This phase issues SETPRT and QSETPRT macros that require a 512 byte work area. Part of the B-transient area itself is used for this work area.

The SETPRT macro is issued to either allow or to block data checks, depending on the DTF type and what the user has requested in his DTF. The QSETPRT macro is used tc query the currently set printer characteristics. These characteristics are passed to IJDFR3.

IJPDF3 is called by this phase to perfcrm open intialization processing for extended bufferirg mode. IJPDR3 is entered at offset 28. The CDLOAD function is used to obtain the main storage address of IJPDR3 as it resides on the SVA.

Return codes from CDLOAD or IJPDR3 may indicate that extended buffering mode is not possible. No message is issued for the normal situations of the partition being in virtual = real mcde or having no GETVIS area. If there is some GETVIS area, but not enough fcr extended buffering, a message is written to SYSLOG. CDLOAD or SETPRT failures cause messages to be written to SYSLCG. IN all cases, open processing continues.

For a DTFPR, the DTF open indicator is set. Fcr all DTF types, if the results of the QSETPRT operation indicate that TRC mode has been requested by the user's program or job control statements, the TRC bit is set in the DTF. (TRC mode means that the first data character in a data record is a table reference character that enables the user's program to dynamically select different character sets for different records of a 3800 output file.)

$$BOCP11: Open DTFCP (Version 1 Only) Phase 1, Charts QD-QE

Objective: To open Version 1 DTFCP tables for disk resident systems.

Entry: From the Open Monitor ($$BOPEN1), or from $$BOMSG1.

Exits:

1. To $$BOCP12, if DTFCP is Version 1 tape resident system type (this DTFCP table type is generated if the DISK= parameter is omitted from the DTFCP macro).

2. To $$BOPEN, if the device type is DASD or additional files remain to be opened.

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

4. To $$BOMT06, if the device type is tape.

5. To $$BOMSG1, to put out the following messages:

   4883I INVALID LOGICAL UNIT
   4884D FILE PROTECT RING NEEDED

Method: This phase is called by the Open Monitor when the file is found to be a Version 1 DTFCP table type. The phase performs the same functions as $$BOCP01, except that it applies only to Version 1.

$$BOCP12: Open DTFCP (Version 1 Only) Phase 2, Chart QF

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

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

Entry: From $$BOCP11, or $$BOMSG1.

Exits:

1. To $$BOMT06, if device type is tape.

2. To $$BOPEN, if additional files remain tc be opened.

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

4. To $$BOMSG1, to put out the following messages:

   4883I INVALID LOGICAL UNIT
   4884D NEED FILE PROTECT RING

Method: This phase is called by $$BOCP11 when the file is found to be a TOS Version 1 DTFCP. The phase performs the same functions as $$BOCP02, with the exception that it applies only to Version 1.

$$BOCPT1: Open DTFCP and DTFDI Input Tape, Charts RA-RC

Objective: To open input tape files.

Entries:

1. From $$BOCP01, if the file is defined by DTFCP or DTFDI and the device is a tape.

2. From $$BOCPM1, after a message has been issued and a non-cancel reply has been received.

3. From $$BOCPT4, if $$BOCPT4 was called by other than $$BOCPT1 or $$BOCPM1.

Exit: To $$BOCPT4, to continue processing label information.

Method: This routine performs various open procedures after determining:

1. If the user has provided a TLBL or TPLAB card.

2. The positicr of the tape with respect to its load point.

3. Whether the tape contains standard labels or a tapemark.

If an unlabeled input tape file is to be opened, this rcutine skips the tapemark, if present, and positions the tape at the first data record.

If a standard labeled input tape file is to be cpened and a TLBL or TPLAB card has been provided, this routine uses the label information supplied to check the HDR1 label. It skips the additional header, volume, and user labels and positions the tape at the first data record. Only single volume files or multifile volumes can be opened by this routine. After the proper procedure is determined, $$BOCPT4 is called.

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

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

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

Entries:

1. Frcm $$BOCP01 if the file is defined by DTFCP or DTFDI for output and the device is a tape.

2. From $$BOCPM2 after a message has been issued and a non-cancel reply has been received.

3. From $$BCCPT3 returning for retry.

Exits:

1. To $$BOCPT3, the open transient for labeled cutput tapes.

2. To the TES processor, $$BOPEN, if more files remain tc be opened.

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

4. To the message writer, $$BOCPM2, when an error condition occurs.

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

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

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

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

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

Objective: To open labeled output tape files.

Entries:

1. From $$BOCPT2 if output tape is already labeled or if label information was supplied by TLBL (TPLAB) job control statements.

2. By return from $$BOCPT2 after a message has been issued, and a non-cancel reply has been received.

Exits:

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

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

3. To the message writer, $$BOCPM2, when an error condition occurs.

4. To phase $$BOCPT2 for retry if entry to this phase was neither from $$BOCPT2 nor a return from the message writer.

5. To the Standard Volume Label Rewriter, $$BONVOL, if the volume label must be rewritten according tc the user specified density.

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

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

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

1. Expiration date is checked.

2. If label information is provided by TLBL (or TPLAB) cards, a new HDR1 label is written.

3. If label infcrmation is not provided, a dummy header is written.

4. No additional standard header or user header labels are written.

5. For a 9-track dual density output tape, a comparison is made between the user-specified density (either 800, 1600, or 6250 BPI) and the VOL1 density of the mcunted tape. If a discrepancy is found, and if the tape is at loadpoint, the volume label(s) is rewritten by $$BONVOL according to the user-specified density.

The last routine in each procedure group is an exit rcutine to fetch $$BOPEN, $$BOCPM2, or $$BOCPT2 as required or to return to the problem program. Note that this open phase can only create _single_ _file_ output tapes.

$$BCCPT4: _Cpen_ _DTFCP_ _and_ _DTFDI_ _Labeled_ _Input_ _Tape,_ _Charts_ _RK-RM_

Objective: To open labeled input tape files.

Entries:

1. From $$BOCPT1 to continue processing tape label information.

2. By return from $$BOCPT1 after a message has been issued, and a non-cancel reply has been received.

Exits:

1. To message writer, $$BOCPM1 to write the errcr message wher an error condition occurs.

2. To phase $$BOCPT1 for retry if entry to this phase was frcm neither $$BOCPT1 nor a return from the message writer.

Methcd: Upon entering from $$BOCPT1, the proper series of subroutines are executed as chcsen by the procedure pointer, which is passed to this phase in the register equated as PROCPTR.

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

1. If label information is provided by TLBL cards, a new HDR1 label is written.

2. A check is made to see if VOL1 and file serial number are blank.

3. A proper message is given for any error conditions that exist.

The last routine is an exit routine to fetch $$BOPEN, $$BOCPM1, or $$BOCPT1 as required or to return to the problem program.

$$BCCPT1: _Close_ _DTFCP_ _or_ _DTFDI_ _Tape_ _Files,_ _Charts_ _SA-SB_

Objective: To close DTFCP and DTFDI input and output tape files.

Entry: From the Close Monitor after it is determined that the file is a DTFCP or DTFDI type, and the device is a tape.

Exits:

- To the Close Monitor if additional files remain to be closed.

- To the problem program if no additional files remain to be closed.

- To the message writer, $$BOMSG1, when an incorrect TLBL (or TPLAB) card image record is found.

Method: This routine performs various close procedures after determining whether the rewind option has been specified, whether a TLBL (or TPLAB) card has been provided, whether the tape file is input or output, and whether the tape has an EOF1 trailer label.

If the file is an input tape, this routine checks to determine if the rewind option has been specified. If the rewind option has been specified, the tape is rewound. If it has not been specified and the tape has standard trailer labels, all standard labels and user trailer labels are bypassed and the tape is positioned to the first record after the tapemark which follows the set of trailer labels.

If the file is an unlabeled output tape, this routine writes a tapemark and rewinds the tape if the rewind option has been specified. If the file is a standard labeled output tape, this routine writes a tapemark. An EOF1 trailer label is built and written on tape followed by two tapemarks. If the rewind option has been specified, the tape is rewound. If it has not been specified, the tape is backspaced one record.

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

$$BCLOSP: Punch File Clcse, Charts SC-SD

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

Entries:

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

Exits:

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

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

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

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

$$BOCPM1 and $$BOCPM2: DTFCP/DTFDI Message Writers, Chart SE

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

Entry: From the DTFCP and DTFDI tape open routines, $$BOCPT1 and $$BOCPT2.

Exits:

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

$$BOCPM1 Messages:

4111A - NO VOL1 LBL FOUND

4112A - VOL SERIAL NO. ERROR

4113D(I) - NO HDR1 LABEL FOUND

4114A - FILE SEQ NO. ERROR

4115A - FILE SER. NO. ERROR

4116A - VOLUME SEQ. NO. ERROR

4132D - ERROR IN FILE ID

4133D - ERROR IN HDR LBL

$$BOCPM2 Messages:

4110A - NO VOL1 LBL FCUND

4112A - VOL SERIAL NC. ERROR

4119A - FILE EXPIRED

4184D - NEED FILE PROTECT RING

Method: Both $$BOCPM1 and $$BOCPM2 message writer phases perform the same function and differ only in the messages written. Each phase writes, on SYSLOG, the appropriate message for the information it receives from the calling phase. The calling phase supplies as input a message number, a return address for each legal response, and the name of the next phase to be fetched. After the correct message and label information is written on SYSLOG, the volume serial number is saved in the PUB2 table, then the message writers analyze and process the operator response and fetch the next appropriate phase.

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

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

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

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

Diskette files are opened and closed by logical transient routines that are fetched by the open and close monitors (refer to VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 1). The open routines provide procedures for checking each file before any records are processed. The close routines provide procedures for terminating each file after all records are processed.

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

Record Format

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

STORAGE AREAS

INPUT/OUTPUT AREAS

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

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

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

MODULE SAVE AREAS

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

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

For diskettes, the save area contains the user's general registers, switches, and other information needed by the module. Figures 35 and 36 illustrate the format of the save area for each logic module.

DUMODFI - Input

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

*Indicates to CPEN that no more DTFs are to be opened.
**If ERREXT=YES, bytes 32-39 contain the parameter list that includes the address of the DTF and the storage address of the block in error.

Figure 35. DUMCDFI Save Area

DUMODFO - Output

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

*Indicates to CPEN that no more DTFs are to be opened.
**If ERREXT=YES, bytes 32-39 contain the parameter list that includes the address of the DTF and the storage address of the block in error.

Figure 36. DUMCDFO Save Area

DTFDU Macro

To process a diskette file of data records, the file must first be defined by the declarative macro DTFDU (Define The File for Diskette Unit). This macro describes the characteristics of the logical file, indicates the type of function being performed, defines the format of the record being processed, and specifies the storage areas and routines used for the file. A DTF table is then generated according to the parameters specified in the operands of the DTFDU macro instruction. Figure 37 illustrates the DTF table generated for diskette files.

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

Figure 37. DTFDU Table (Part 1 of 3)

Bytes*	Bits	Contents	Function
39 (27)	0		1 = Extent bypassed before file opened (input).
	0-7		Sequence number of current extent opened (output).
40 (28)			Sequence number of last extent opened.
41-43 (29-2B)		X'000000'	Reserved.
44-47 (2C-2F)			Address of IOAREA1.
48-51 (30-33)			Address of last Read/Write CCW in chain.
52-53 (34-35)		X'0001'	Lower record limit.
54-57 (36-39)		X'00CC00RR'	End-of-data seek address (last record + 1).
58-59 (3A-3B)			Number of records in I/O area (used in short chain processing).
60-63 (3C-3F)		X'00FF0001'	Seek argument (0CHR).
64-67 (40-43)			End-of-file routine address (input); 4X'00' (output).
68-71 (44-47)		X'0049001A'	Seek argument control field.
72 (48)			Command chaining factor.
73 (49)			Switch byte 1.
	0		1 = Not first entry after open.
	1	B'0'	Not used.
	2		1 = In close routine (output).
	3		1 = Error chain to be skipped.
	4		1 = End of extent.
	5-7	B'000'	Not used.
74-75 (4A-4B)			(Record size multiplied by command chain factor) - 1
76-80 (4C-50)		X'FFFFFFFFFF'	Seek argument bucket.
81-83 (51-53)		X'000000'	Reserved.
84-87 (54-57)			Instruction to load user's I/O register (or NOP).
88-91 (58-5B)			Address of current I/O area.

Figure 37. DTFDU Table (Part 2 of 3)

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

144-X
(90-Y
		X=143+8*(no. of CCWs)
		Y=8F+8*(no. of CCWs)
			Read/Write data CCWs; 1, 2, 13, or 26 Read/Write CCWs.

X+1 (Y+1)			NOP CCW (output only).

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 37. DTFDU Table (Part 3 of 3)

DTFPH MACRO (DISKETTE)

When physical IOCS macro instructions are used to process a diskette file with standard labels, and the user wishes to have the labels checked, the file must be defined by a DTFPH macro. To define a diskette file, the parameters specified in the operands of the DTFPH macro instruction must include DEVICE=3540 and MOUNTED=SINGLE. Figure 38 illustrates the DTFPH table generated to define a diskette file for physical IOCS.

Bytes*	Bits	Function
0-15 (0-F)		CCB.
16 (10)	0	1 = Dequeue old volume extents
	1-3	Not used
	4	1 = DTF relocated by OPENR
	5-7	Not used.
17-19 (11-13)		3X'00'
20 (14)		DTF type (X'21').
21		Open/close indicators
	0-2	Not used
	3	1 = Work area
	4	1 = Not Version 1 DTF table type
	5	1 = Open; 0 = Closed
	6	1 = Input; 0 = Output
	7	Not used.
22-28 (16-1C)		Filename (see byte 29).
29 (1D)		Device type code (3540 = X'06').
30 (1E)		C'F' = EOF indicator for DTFPH.
30-35 (1E-23)		(OCHR00) Address of HDR1 label in VTOC (output).
36-37 (24-25)		Volume sequence number.
38 (26)		Open communications byte. Input:
	0	1 = No more extents
	1-3	Not used
	4	1 = New volume or new extent
	5-7	Not used.
		Output:
	0	1 = No more extents
	1	1 = Extents for LIOCS at close
	2-3	Not used
	4	1 = New volume on next extent
	5	1 = Extents entered via console
	6	Not used
	7	1 = Check extent for minimum of 2 tracks.

Figure 38. DTFPH Table for Diskette (Part 1 of 2)

Bytes*	Bits	Function
39 (27)		Sequence number of current extent being opened.
40 (28)		Sequence number of last extent opened (not a console extent entry).
41-43 (29-2B)		Not used.
44-47 (2C-2F)		Address of IOAREA1.
48-51 (30-33)		Not used.
52-53 (34-35)		X'0000'
54-57 (36-39)		Extent upper limit (OCHR).
58-59 (3A-3B)		Not used.
60-63 (3C-3F)		Extent lower limit (OCHR).
64 (40)		Record number. 1 = Input, 0 = Output.
65-67 (41-43)		Not used.
68-71 (44-47)		OCHR control bucket. OCHR = X'0049001A' for 3540 (output only).
72 (48)		Record number.
73 (49)		X'80' - verify requested. X'40' - last volume on multivolume file (input). X'10' - multivolume file (input).
74 (4A)		Record size (maximum of 128).
75 (4B)		Not used.
76-80 (4C-50)		OCHR bucket = extent lower limit and record number (output).
81-83 (51-53)		Not used.

*Numbers in parentheses are displacements in hexadecimal notation.

Figure 38. DTFPH Table for Diskette (Part 2 of 2)

MODULE GENERATION MACROS

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

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

MODULES

The basic modules are:

- DUMODFI - fixed-length input records.
- DUMODFO - fixed-length output records.

The modules are generalized routines that work with one cr more unique DTF tables to perform their various functions. Each module can perfcrm the required functions through the use of:

- A work area (optional)
- One or two I/O areas
- Errcr opticns (if specified at module generaticr time)
- ERET macro (if ERREXT=YES is specified at module generaticn time)
- Chained or urchained records
- RDONLY opticn (if specified at module generatior time).

The cptions listed must be defined by the DTFDU macro instruction.

DUMODFI: GET Processing, Chart UA

Objective: To read chained or unchained records from a diskette file.

Entry: From the GET macro expansion.

Exits:

1. To the problem program.
2. To $$BOPEN if end of data is reached on a diskette file.

Method: This module reads chained or unchained records and makes the logical records available to the problem program in a work area if cne is specified.

If special records, such as deleted or sequentially relocated records, are read, they are skipped (not passed to the user).

If the ERROPT parameter has been specified, errcrs are processed as each chain is read. If ERREXT has been specified, additional errors are returned to the problem program for further processing.

When end of data is reached on a diskette volume, a test is made to see if there are more extents. If so, the module addresses the next diskette volume and an SVC 2 is issued to fetch $$BOPEN (to open the next extent for the file). If there are no more extents, an end-of-file condition is assumed. This module feeds out the last diskette, based on the DTF FEED parameter, and issues an SVC 2 to fetch $$BOPEN (which in turn gives control to the user's EOF routine).

DUMODFO: PUT Processing, Chart UB

Objective: To write chained or unchained records in sequence on a diskette file.

Entry: From the PUT macro expansion.

Exits:

1. To the problem program.
2. To $$BOPEN, if end of extent is reached on a diskette file.

Method: This logic module writes chained or unchained records. If the user specifies a work area, this routine moves the logical record from the work area to the output area. It then updates the current I/O area address by the logical record size, and determines if the end of the I/O area has been reached; if not, control is returned to the user.

When the I/O area is full, a test is made to determine if the current seek address is beyond the end of extent. If so, and if there are more extents, the following action is taken:

1. The ERMAP record is updated if bad spot records were written on the diskette because of an ERET RETRY situation.
2. The current diskette is fed out, causing the next diskette to be fed in.
3. $$BOPEN is fetched to open the next extent.

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

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

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

DUMCDFC: Close Processing, Chart UB

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

Entry: From the close transient, $$BODIO4.

Exit: To $$BODIO4, via an SVC 9.

Method: This routine sets on the partial block switch and determines if there are any logical records to be written to the file. If so, the chain of CCWs to write the proper number of records is set up; then the PUT routine is given control to write the short chain to the file. After that the CLOSE routine regains control, the module switches are set off, and a branch is made to the PUT routine to exit to the close transient $$BODIO4.

INITIALIZATION AND TERMINATION PROCEDURES

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

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

The diskette open logical transients read the DASD label information into storage. The format of the DASD label information is illustrated in VSE/Advanced Functions DASD Labels. If the file is an input file, the open transients compare the file label information with the DASD label information in the label information area to determine if the logical file is correct and if the serial numbers are equal.

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

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

DISKETTE OPEN/CLOSE LOGIC

Open Diskette Files, General, Chart 03

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

1. The standard label(s) on the volume, or on the first volume of a multivolume file, is checked.
2. The first extent on the first volume is located and made available for processing.

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

OPEN (Input Diskette) General Flow, Chart 04

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

For multivolume diskette input files using DTFDU, the extent cards and the multivolume indicator are used in conjunction by the OPEN transients to determine when end of file has occurred. If three extents were provided by the user, the following multivolume indicator combination could occur:

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

For all other supported DTFs, processing continues until the number of extents is exhausted. Regardless of the DTF type, for system files processing continues until all extents are exhausted.

OPEN (Output Diskette) General Flow, Chart 05

Each EXTENT card provided for a diskette relates to a different volume. Therefore, each time a new extent is opened, both volume and file labels must be processed. The file labels are checked against the DLBL statements provided, to ensure that no duplicate or overlapping files are created on that volume. Labels are created and written as directed by the DLBL information, pertinent information is posted in the DTF and control returns to the user.

CLOSE Diskette General Flow, Chart 06

The close routine for diskette input and output files causes any additional records to be processed, any file labels to be updated, and the file to be indicated as being closed. A compiler file is closed in the same manner as an output file, except that processing of additional records is bypassed.

If the file is an input file, it is simply indicated as being closed. Feeding of diskettes at close time is indicated by the table in Figure 39.

$$B35400: Diskette Open, Initialization, Charts UE-UG

Objective: This phase determines which diskette open phase must be fetched to complete the open procedure. It also performs basic initialization functions and reads and examines the DLBL/EXTENT images from the label information area.

Entry: From $$BOPEN1.

Exits:

1. To $$B3540I or $$B35400 to complete the open procedure.
2. To $$BODMSG to communicate with the operator in the case of an error.

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

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

Figure 39. This Table Indicates Diskettes Being Fed at Close Time

Method: The phase first determines if the file to be opened is an input or output file by checking the DTF, and then initializes to fetch the proper transient. It initializes the Open Table with the device characteristics and the DIB (Disk Information Block) address if the file is a system file. It reads the DLBL/EXTENT images for the file, and examines them for validity.

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

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

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

Exits:

1. To $$BODIO1 to process volume labels.

2. To $$BOPEN if the last DLBL extent has been processed and another file is to be opened.

3. To the user's end-of-file routine.

4. To $$BODMSG for operator communication.

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

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

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

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

This routine also determines whether the assigned device is the correct device and ensures that only one file is open on the device. If this is not the case, the job is canceled; otherwise, this transient passes control to $$BODIO1.

$$BODIO1: Diskette Volume Label Processor, Charts VD-VE

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

Entries:

1. $$B3540I, $$B35400, and $$BODIO8.

2. Reentry from $$BODMSG and BODSMO.

Exits:

1. To $$BODIO2 if open output.

2. To $$BODIO5 if open input.

3. To $$BODSMO if a secured volume.

4. To $$BODMSG for operator communication.

Method: The volume label is read from the diskette and checked to ensure that it is standard. Then the volume serial number is compared against the one provided in the DLBL (if specified) to ensure that the correct volume is mounted. If the wrong diskette is mounted, a mount message is issued. If the volume is secured, the security message writer is fetched. If open is allowed to continue, the proper input or output transient is given control.

$$BODIO5: Diskette Open Input, Process HDR1 Label, Charts VF-VM

Objective: This phase is used to read and process the desired HDR1 label and to handle data set security.

Entry: From $$BODIO1 and reentry from $$BODSMO.

Exits:

1. To $$BODIO6 to initialize the DTF.

2. To $$BODSMO if a secured file is opened.

3. To $$BODMSG for operator communication.

Method: The desired HDR1 label is located and examined. If the file is secured, further qualification from the operator is required to allow the file to be opened. Volume sequence checking is performed if the user has requested that in his DTFDU. The extent limits are converted to binary and placed in the DTF table. This transient then exits to $$BODIO6.

$$BODIO6: Diskette Open Input, Initialize DTF, Charts VL-VM

Objective: To complete the initialization of the DTF table and to validate the extent limits.

Entry: From $$BCDIO5.

Exits:

1. To $$BOPEN to open the next file.

2. To the user's next sequential instruction.

3. To $$BODMSG for operator communication.

Method: The upper and lower extent limits in the DTF are checked tc verify that they are within the limits of the device and are correct relative to each other. The DTF open switch is set along with other switches indicating the last extent and new volume. For system files, the DIB is filled in from the DTF table.

$$B3540C: Diskette Open Output, DLBL Extents, Charts WA-WC

Objective: This phase controls the sequence of operations fcr opening each file extent. It provides for the entry of extents from the console, checks the COBOL Open/Ignore function, ensures that the assigned device is a 3540 on which only one file is open, and initializes the open procedure for system files.

Entry: From $$B35400.

Exits:

1. To $$BODIO1 to process volume labels.

2. To $$BODIO8 to allow additional extents to be entered from the console.

3. To $$BODMSG for operator communication.

4. To $$BOPEN if the open processing procedure is complete for this file and another file must be opened.

5. To the user's next sequential instruction.

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

$$B35400 tests for the availability of DLBL extents. If no more extents are available, the operator can enter more extents via the console. If the file has been opened previously, the next consecutive DLBL extent to be opened is read.

If the COBOL Open/Ignore option has been specified, and the device is unassigned or assigned to IGN, open is bypassed; if the device is assigned, open continues.

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

This routine also determines whether the assigned device is the correct device and ensures that only one file is open on that device. If not, the job is canceled; otherwise, this transient fetches $$BODIO1.

$$BODIO2: Diskette Open Output, Determine Extents and Delete HDR1 Labels, Charts WD-WF

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

Entry: From $$BODIO1.

Exits:

1. To $$BODIO3 to create a new HDR1 label.

2. To $$BODMSG for operator communication.

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

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

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

$$BODIO3: Diskette Open Output, Create/Write New HDR1 Label, Charts WG-WJ

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

Entry: From $$BCDIO2.

Exits:

1. To $$BODIC7 to initialize the DTF.
2. To $$BODMSG for operator communication.

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

$$BODIO7: Diskette Open Output, Initialize DTF Table, Charts WK-WL

Objective: To update the DTF table.

Entry: From $$BCDIO3.

Exits:

1. To $$BOPEN to open the next file.
2. To $$BODIC4 if Close needed an extent.
3. To $$BODMSG for operator communication.
4. To the problem program.

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

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

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

Entry: From $$BCCP01.

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

Method: Fields are initialized and CCW chains are moved into the DTF to make it usable for reading and writing diskette files. Exit is to the open monitor to reconstruct the common open table and to exit to the proper diskette open phase.

$$BODIO8: Diskette Open Output, Extents From Console, Charts WP-WQ

Objective: To allow the operator to enter extent information from the console.

Entry: From $$B35400.

Exits:

1. To $$BODIO1 to process a new volume.
2. To $$BODSPV to display the VTOC.
3. To $$BOVDMO to dump the VTOC.
4. To $$BODMSG for operator communication.

Method: This routine initiates a 'NO MORE AVAILABLE EXTENTS' message and reads the operator response from the console. If the operator does not cancel the job, a reply of '3540GO' is assumed, which indicates that a new diskette has been loaded. In that case control is passed to $$BODIO1.

$$BODIO4: Diskette Close, Charts WR-WU

Objective: To allow the last block of data to be written, to provide end-of-file feed control, and to indicate in the DTF that the file is closed.

Entry: From $$BCLOSE or reentry from $$BODIO7.

Exits:

1. To $$B35400 to open a new extent.
2. To $$BCLOSE to close the next file.
3. To $$BODMSG for operator communication.
4. To the problem program.

Method: If the DTF refers to a system logical input unit, the DTF is reset and no feeding occurs. If the DTF refers to a programmer logical input unit, the DTF is reset and feeding occurs if required. If the DTF is a DTFDU (either system or programmer unit) control is returned to LIOCS to allow the last block of data to be written.

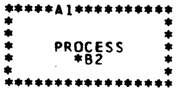
For all output files the HDR1 label for the file is updated to reflect proper end of extent, end of data, and correct multivolume indicator. Feeding, if required, occurs and the DTF is reset.

CHARTS

EXPLANATION OF FLOW CHART SYSMBOLS

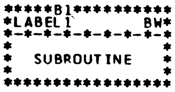
DESCRIPTION

EXAMPLE

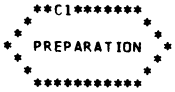


A group of program instructions that perform a processing function of the program. The label, if any, is shown above the block.

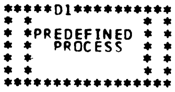
```
*B2
IF ANY ADDITIONAL EXPLANATION IS
REQUIRED, ITS LOCATION ON THE CHART
IS IDENTIFIED BY AN ASTERISK AND
THE BLOCK ID.
```



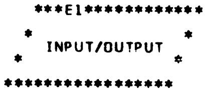
Description or title of a routine that is detailed on another flowchart. The starting label of the routine and the flowchart ID appear above the stripe.



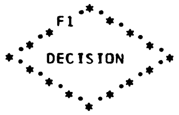
An instruction, or group of instructions, that changes portion of a routine or initializes a routine for a given condition.



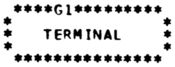
A group of operations not detailed in the flowcharts in this manual, such as user routines.



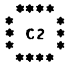
Any function of an input/output device or program, usually branching to an I/O routine to perform the function stated in the block.



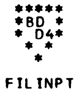
Points where the program branches to alternate processing, based upon variable conditions such as program switch settings and test results.



The beginning or end of a program or routine.



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



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

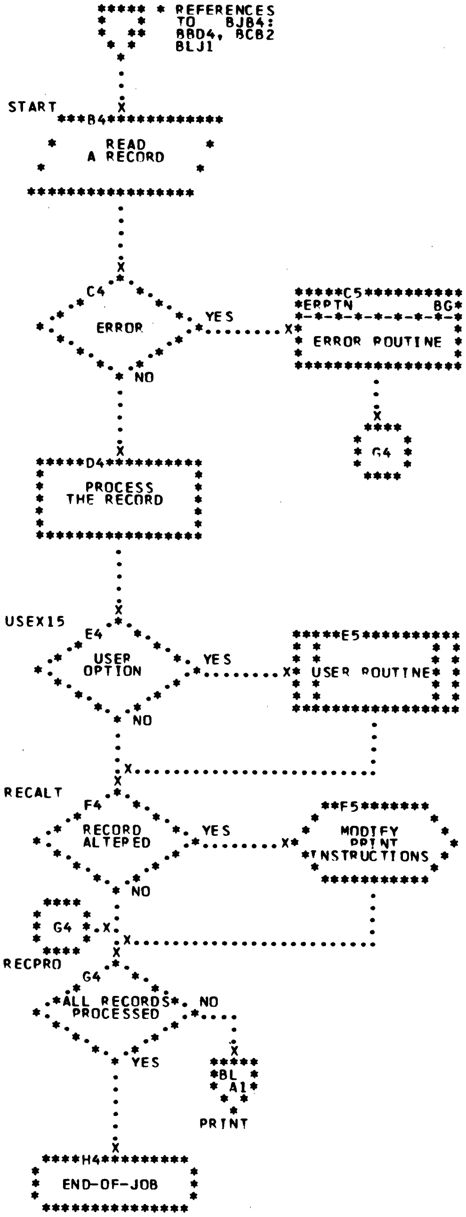
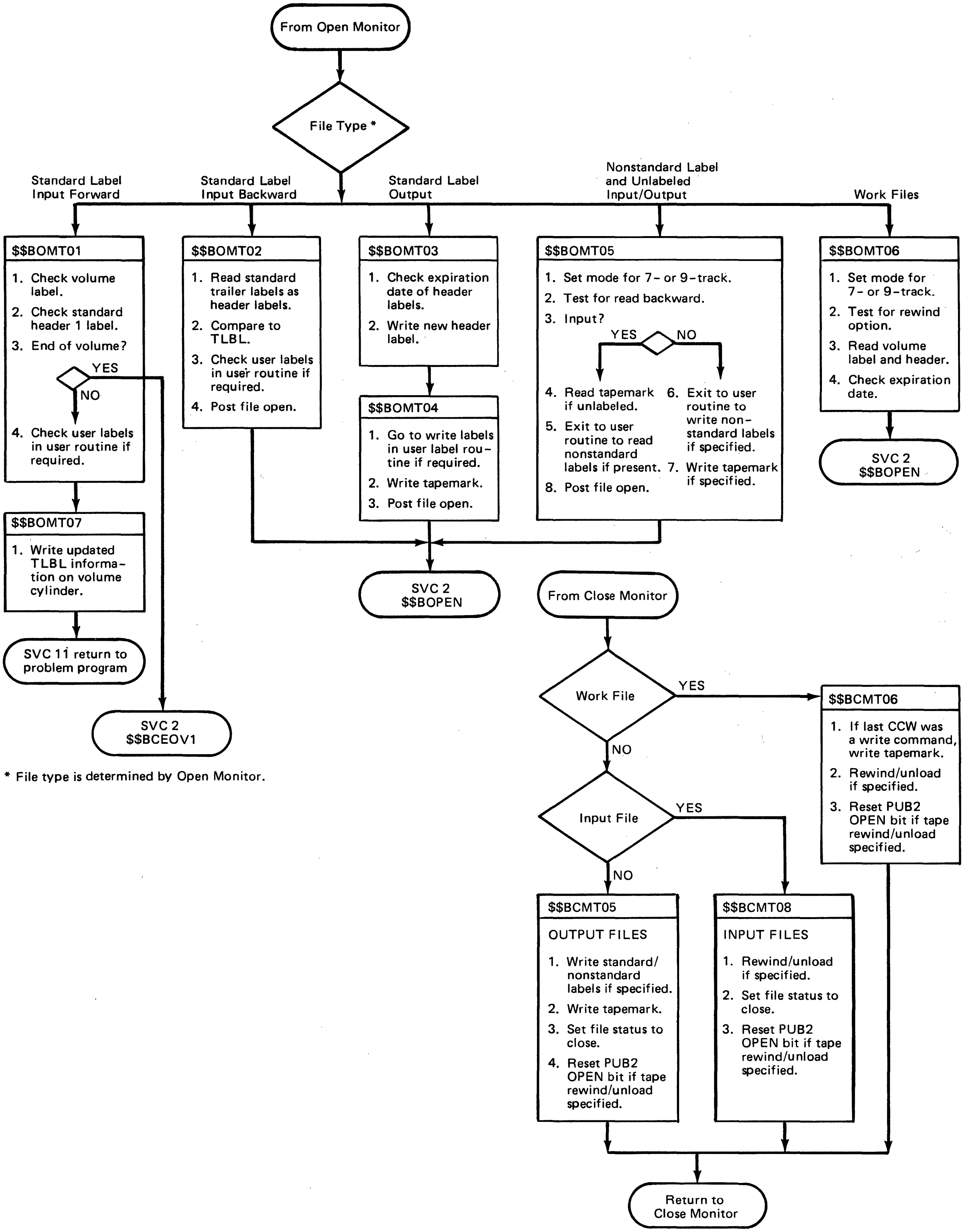


Chart 01. Magnetic Tape Open Routines


From Open Monitor
File Type *
Standard Label Input Forward
Standard Label Input Backward
Standard Label Output
Nonstandard Label and Unlabeled Input/Output
Work Files
$$BOMT01
1. Check volume label.
2. Check standard header 1 label.
3. End of volume?
YES
NO
4. Check user labels in user routine if required.
$$BOMT07
1. Write updated TLBL information on volume cylinder.
SVC 11 return to problem program
SVC 2 $$BCEOV1
$$BOMT02
1. Read standard trailer labels as header labels.
2. Compare to TLBL.
3. Check user labels in user routine if required.
4. Post file open.
$$BOMT03
1. Check expiration date of header labels.
2. Write new header label.
$$BOMT04
1. Go to write labels in user label routine if required.
2. Write tapemark.
3. Post file open.
SVC 2 $$BOPEN
$$BOMT05
1. Set mode for 7- or 9-track.
2. Test for read backward.
3. Input?
YES
NO
4. Read tapemark if unlabeled.
5. Exit to user routine to read nonstandard labels if present.
6. Exit to user routine to write nonstandard labels if specified.
7. Write tapemark if specified.
8. Post file open.
$$BOMT06
1. Set mode for 7- or 9-track.
2. Test for rewind option.
3. Read volume label and header.
4. Check expiration date.
SVC 2 $$BOPEN
From Close Monitor
Work File
YES
NO
$$BCMT06
1. If last CCW was a write command, write tapemark.
2. Rewind/unload if specified.
3. Reset PUB2 OPEN bit if tape rewind/unload specified.
Input File
YES
NO
$$BCMT05
OUTPUT FILES
1. Write standard/nonstandard labels if specified.
2. Write tapemark.
3. Set file status to close.
4. Reset PUB2 OPEN bit if tape rewind/unload specified.
$$BCMT08
INPUT FILES
1. Rewind/unload if specified.
2. Set file status to close.
3. Reset PUB2 OPEN bit if tape rewind/unload specified.
Return to Close Monitor

* File type is determined by Open Monitor.

Chart 02. Magnetic Tape Close and EOF/EOV Routine

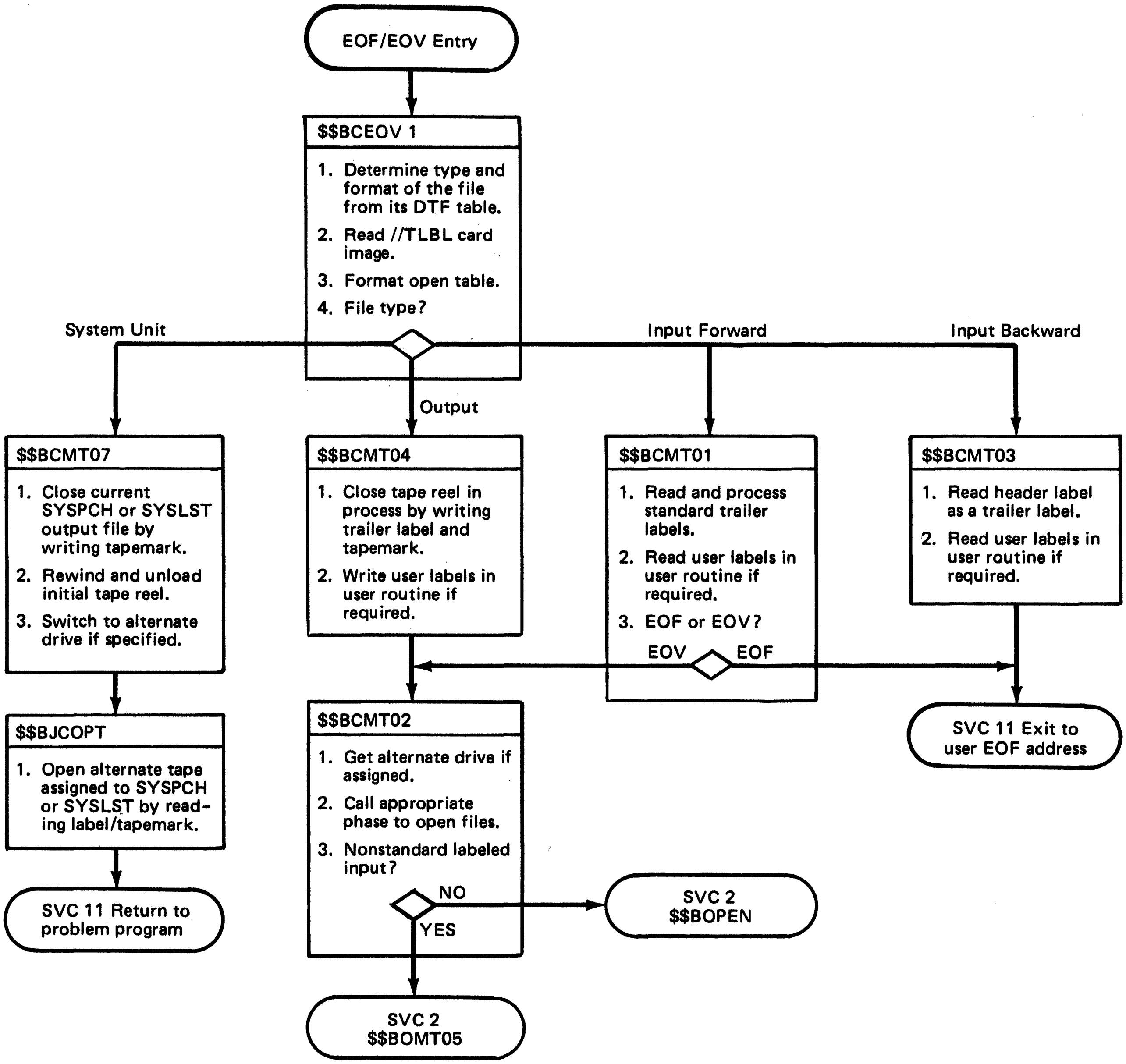

EOF/EOV Entry
$$BCEOV 1
1. Determine type and format of the file from its DTF table.
2. Read //TLBL card image.
3. Format open table.
4. File type?
System Unit
Input Forward
Input Backward
Output
$$BCMT07
1. Close current SYSPCH or SYSLST output file by writing tapemark.
2. Rewind and unload initial tape reel.
3. Switch to alternate drive if specified.
$$BCMT04
1. Close tape reel in process by writing trailer label and tapemark.
2. Write user labels in user routine if required.
$$BCMT01
1. Read and process standard trailer labels.
2. Read user labels in user routine if required.
3. EOF or EOV?
EOV
EOF
$$BCMT03
1. Read header label as a trailer label.
2. Read user labels in user routine if required.
$$BJCOPT
1. Open alternate tape assigned to SYSPCH or SYSLST by reading label/tapemark.
$$BCMT02
1. Get alternate drive if assigned.
2. Call appropriate phase to open files.
3. Nonstandard labeled input?
NO
YES
SVC 11 Exit to user EOF address
SVC 11 Return to problem program
SVC 2 $$BOPEN
SVC 2 $$BOMT05

Chart 03. Diskette Open, General Flow

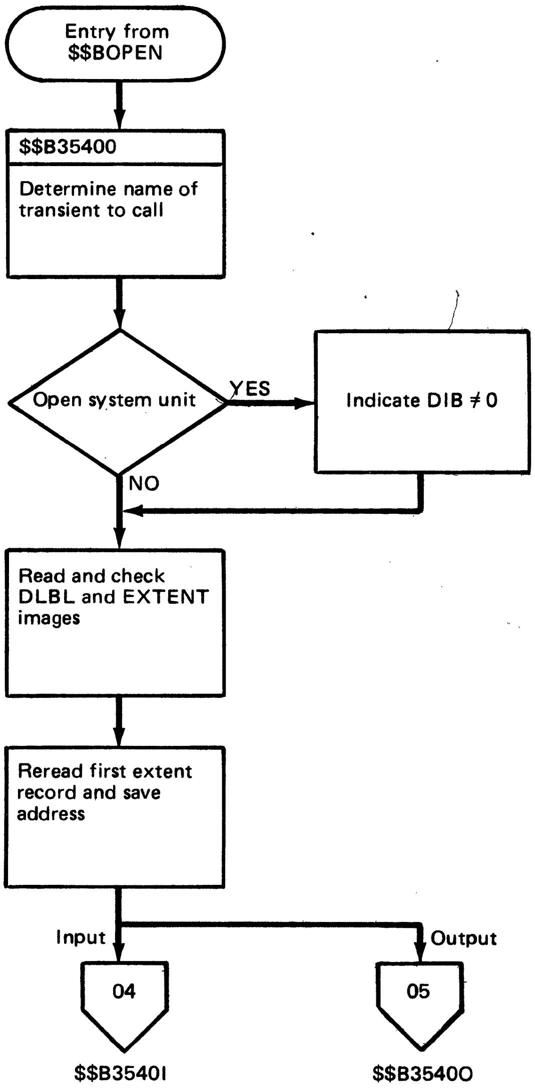

Entry from $$BOPEN
$$B35400
Determine name of transient to call
Open system unit
YES
Indicate DIB ≠ 0
NO
Read and check DLBL and EXTENT images
Reread first extent record and save address
Input
Output
04
05
$$B35401
$$B35400

Chart 04. Diskette Open, Input Files

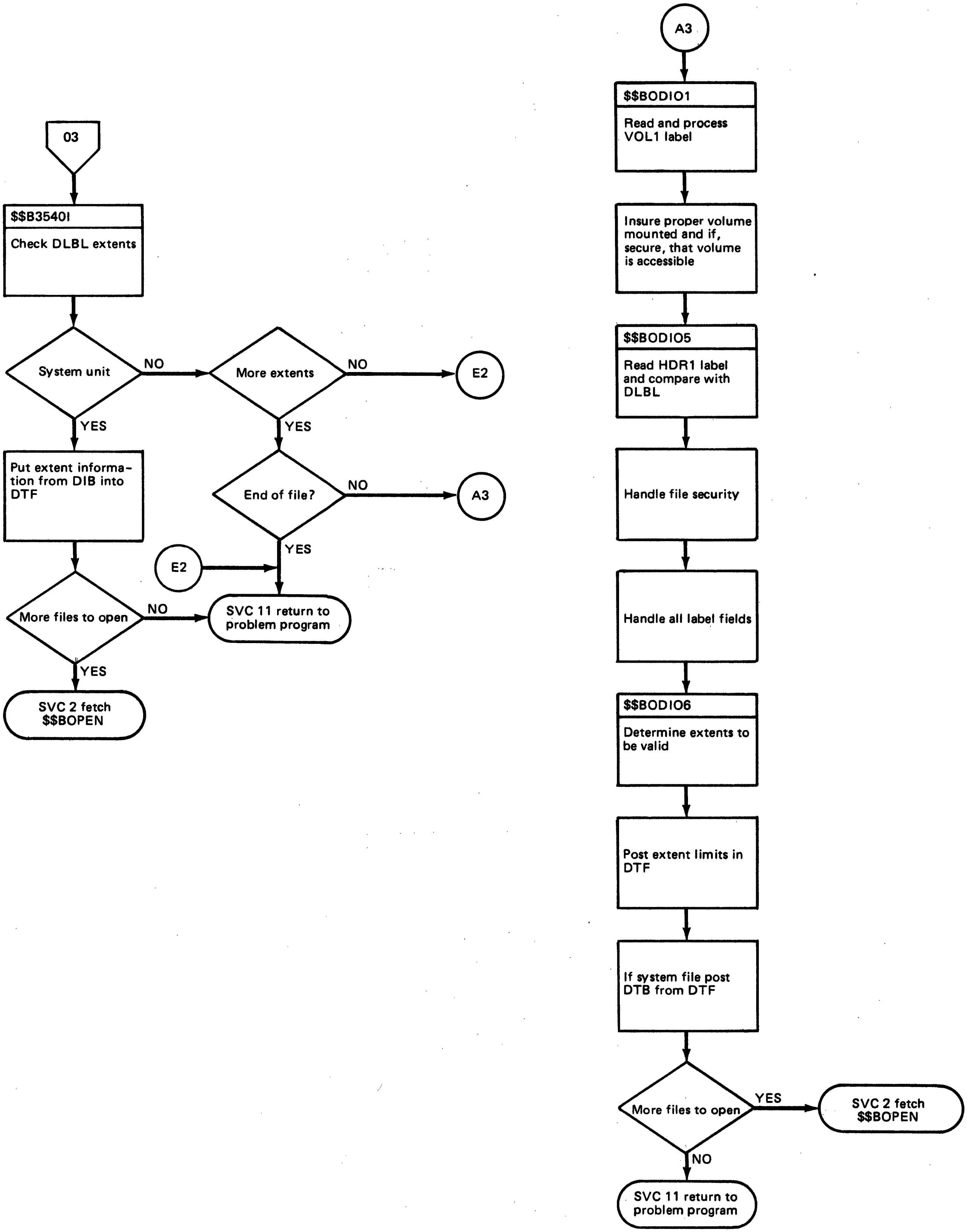

03
$$B35401
Check DLBL extents
System unit
NO
YES
Put extent information from DIB into DTF
More files to open
NO
YES
SVC 2 fetch $$BOPEN
More extents
NO
E2
YES
End of file?
NO
A3
YES
E2
SVC 11 return to problem program
A3
$$BODIO1
Read and process VOL1 label
Insure proper volume mounted and if, secure, that volume is accessible
$$BODIO5
Read HDR1 label and compare with DLBL
Handle file security
Handle all label fields
$$BODIO6
Determine extents to be valid
Post extent limits in DTF
If system file post DTB from DTF
More files to open
YES
SVC 2 fetch $$BOPEN
NO
SVC 11 return to problem program

Chart 05. Diskette Open, Output Files

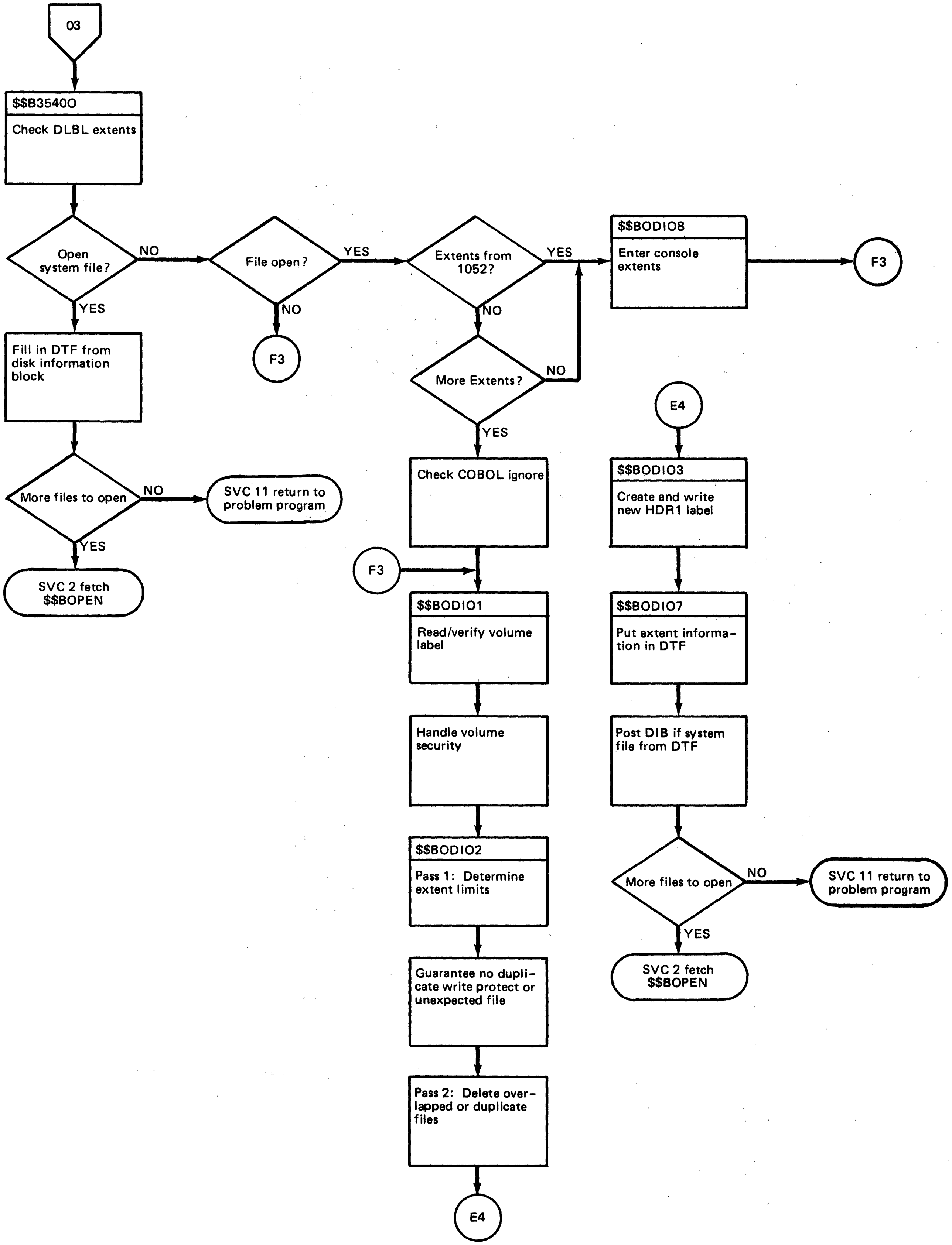

03
$$B35400
Check DLBL extents
Open system file?
NO
YES
Fill in DTF from disk information block
More files to open
NO
SVC 11 return to problem program
YES
SVC 2 fetch $$BOPEN
File open?
YES
NO
F3
Extents from 1052?
YES
NO
More Extents?
NO
YES
$$BODIO8
Enter console extents
F3
Check COBOL ignore
F3
$$BODIO1
Read/verify volume label
Handle volume security
$$BODIO2
Pass 1: Determine extent limits
Guarantee no duplicate write protect or unexpected file
Pass 2: Delete overlapped or duplicate files
E4
E4
$$BODIO3
Create and write new HDR1 label
$$BODIO7
Put extent information in DTF
Post DIB if system file from DTF
More files to open
NO
SVC 11 return to problem program
YES
SVC 2 fetch $$BOPEN

Chart 06. Diskette Close

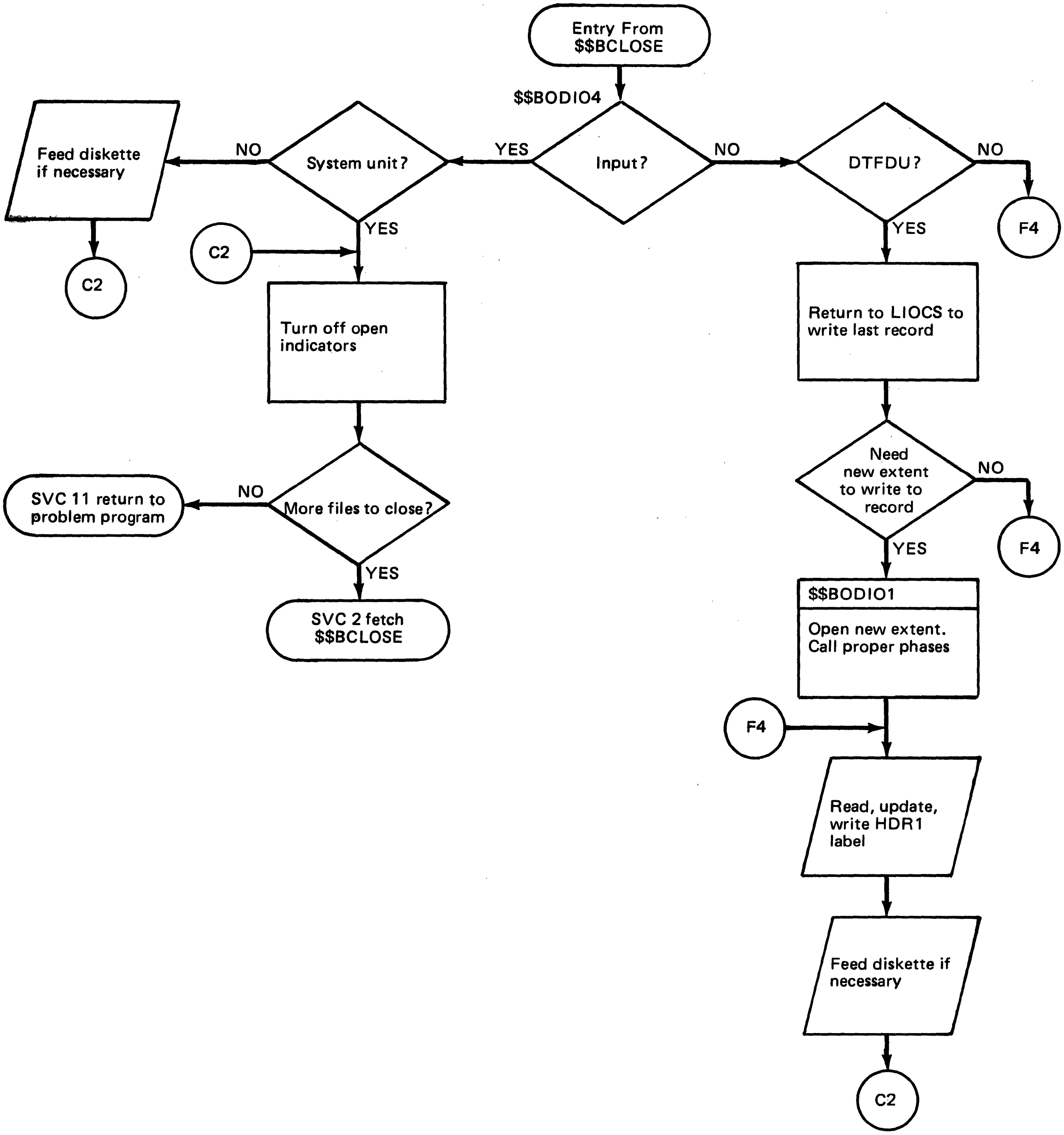

Entry From $$BCLOSE
$$BODIO4
Input?
YES
NO
System unit?
NO
Feed diskette if necessary
C2
YES
C2
Turn off open indicators
More files to close?
NO
SVC 11 return to problem program
YES
SVC 2 fetch $$BCLOSE
DTFDU?
NO
F4
YES
Return to LIOCS to write last record
Need new extent to write to record
NO
F4
YES
$$BODIO1
Open new extent. Call proper phases
F4
Read, update, write HDR1 label
Feed diskette if necessary
C2

Chart AA. $$BCUF01: Open Unit Record (Part 1 of 2)

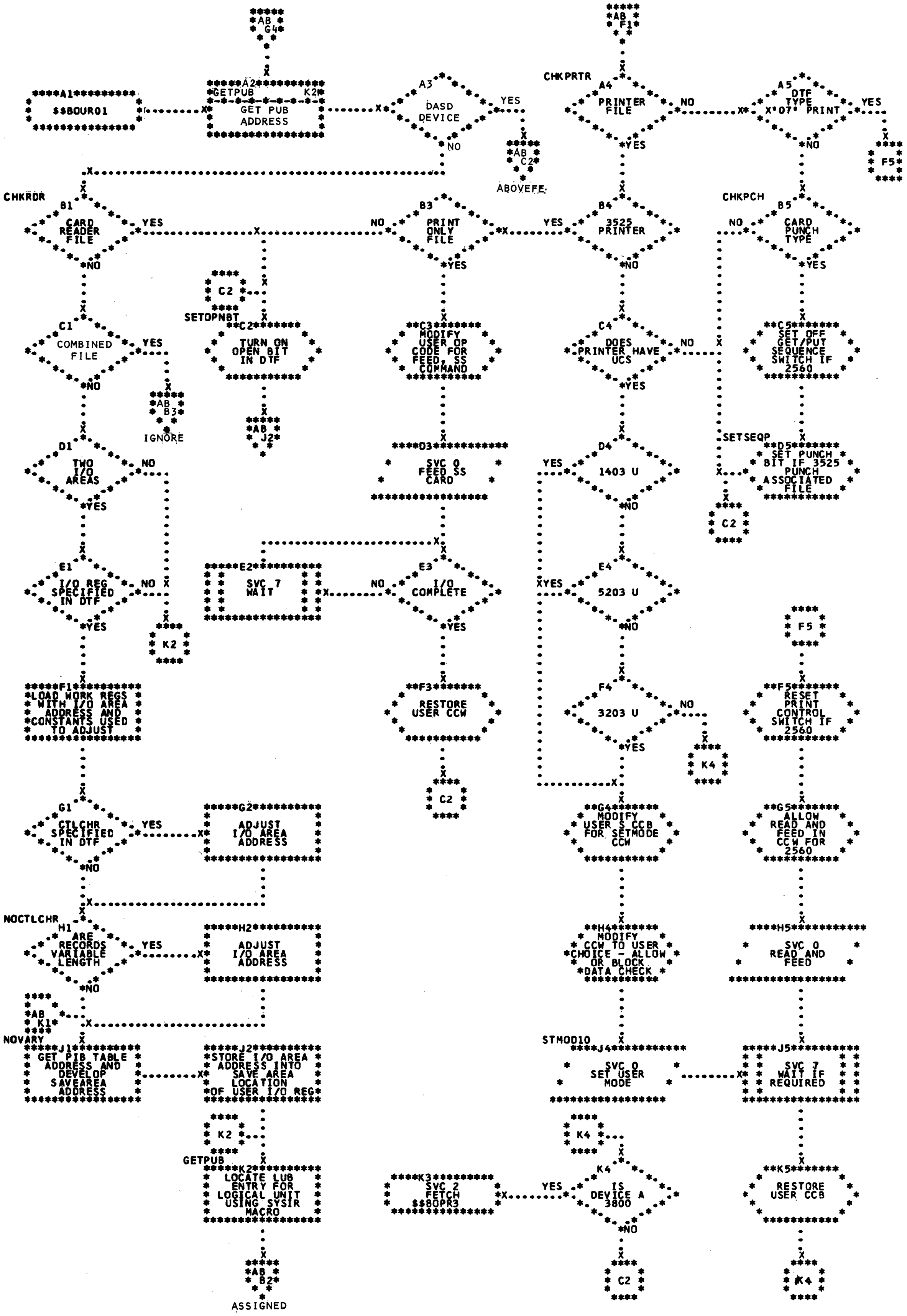

AB G4
A1 $$BOUR01
A2 GETPUB K2 GET PUB ADDRESS
A3 DASD DEVICE
YES
AB C2
ABOVEFE
NO
CHKPRTR
AB F1
A4 PRINTER FILE
NO
YES
A5 DTF TYPE X'07' PRINT
YES
F5
NO
CHKRDR
B1 CARD READER FILE
YES
NO
B3 PRINT ONLY FILE
YES
B4 3525 PRINTER
NO
CHKPCH
B5 CARD PUNCH TYPE
NO
YES
C2
SETOPNBT
C1 COMBINED FILE
YES
NO
AB B3
IGNORE
C2 TURN ON OPEN BIT IN DTF
AB J2
C3 MODIFY USER OP CODE FOR FEED, SS COMMAND
C4 DOES PRINTER HAVE UCS
NO
YES
C5 SET OFF GET/PUT SEQUENCE SWITCH IF 2560
D1 TWO I/O AREAS
NO
YES
D3 SVC 0 FEED SS CARD
D4 1403 U
YES
NO
SETSEQP
D5 SET PUNCH BIT IF 3525 PUNCH ASSOCIATED FILE
C2
E1 I/O REG SPECIFIED IN DTF
NO
YES
K2
E2 SVC 7 WAIT
E3 I/O COMPLETE
NO
YES
E4 5203 U
YES
NO
F5
F1 LOAD WORK REGS WITH I/O AREA ADDRESS AND CONSTANTS USED TO ADJUST
F3 RESTORE USER CCW
F4 3203 U
NO
YES
K4
F5 RESET PRINT CONTROL SWITCH IF 2560
C2
G1 CTLCHR SPECIFIED IN DTF
YES
NO
G2 ADJUST I/O AREA ADDRESS
G4 MODIFY USER'S CCB FOR SETMODE CCW
G5 ALLOW READ AND FEED IN CCW FOR 2560
NOCTLCHR
H1 ARE RECORDS VARIABLE LENGTH
YES
NO
H2 ADJUST I/O AREA ADDRESS
H4 MODIFY CCW TO USER CHOICE - ALLOW OR BLOCK DATA CHECK
H5 SVC 0 READ AND FEED
AB K1
NOVARY
J1 GET PIB TABLE ADDRESS AND DEVELOP SAVEAREA ADDRESS
J2 STORE I/O AREA ADDRESS INTO SAVE AREA LOCATION OF USER I/O REG
STMOD10
J4 SVC 0 SET USER MODE
J5 SVC 7 WAIT IF REQUIRED
K2
GETPUB
K2 LOCATE LUB ENTRY FOR LOGICAL UNIT USING SYSIR MACRO
K3 SVC 2 FETCH $$BOPR3
YES
K4
K4 IS DEVICE A 3800
NO
K5 RESTORE USER CCB
AB B2
ASSIGNED
C2
K4

Chart AB. $$BOUR01: Open Unit Record (Part 2 of 2)

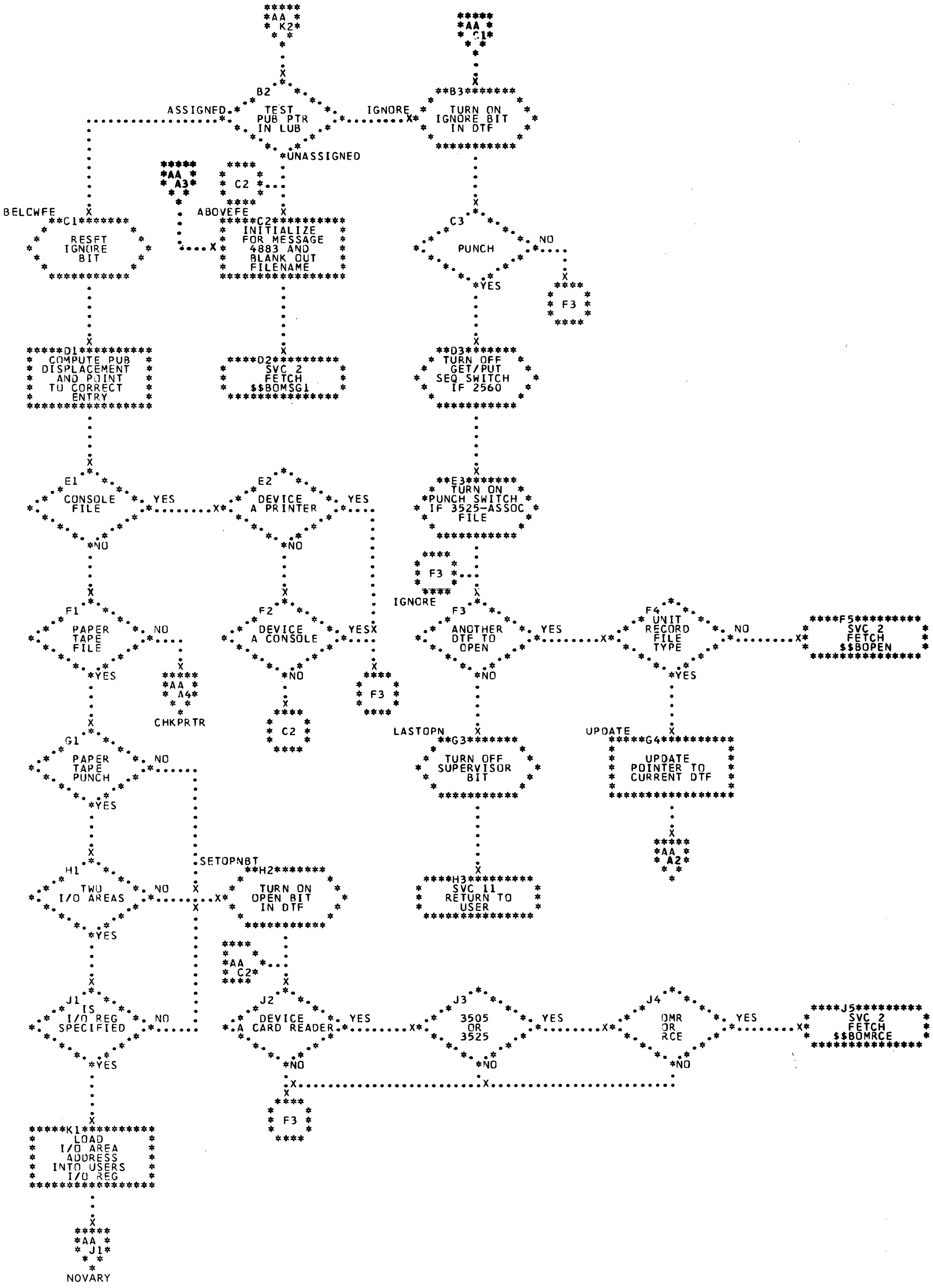


Chart AC. $$BOMRCE: OMR and RCE Open Routine (Part 1 of 2)

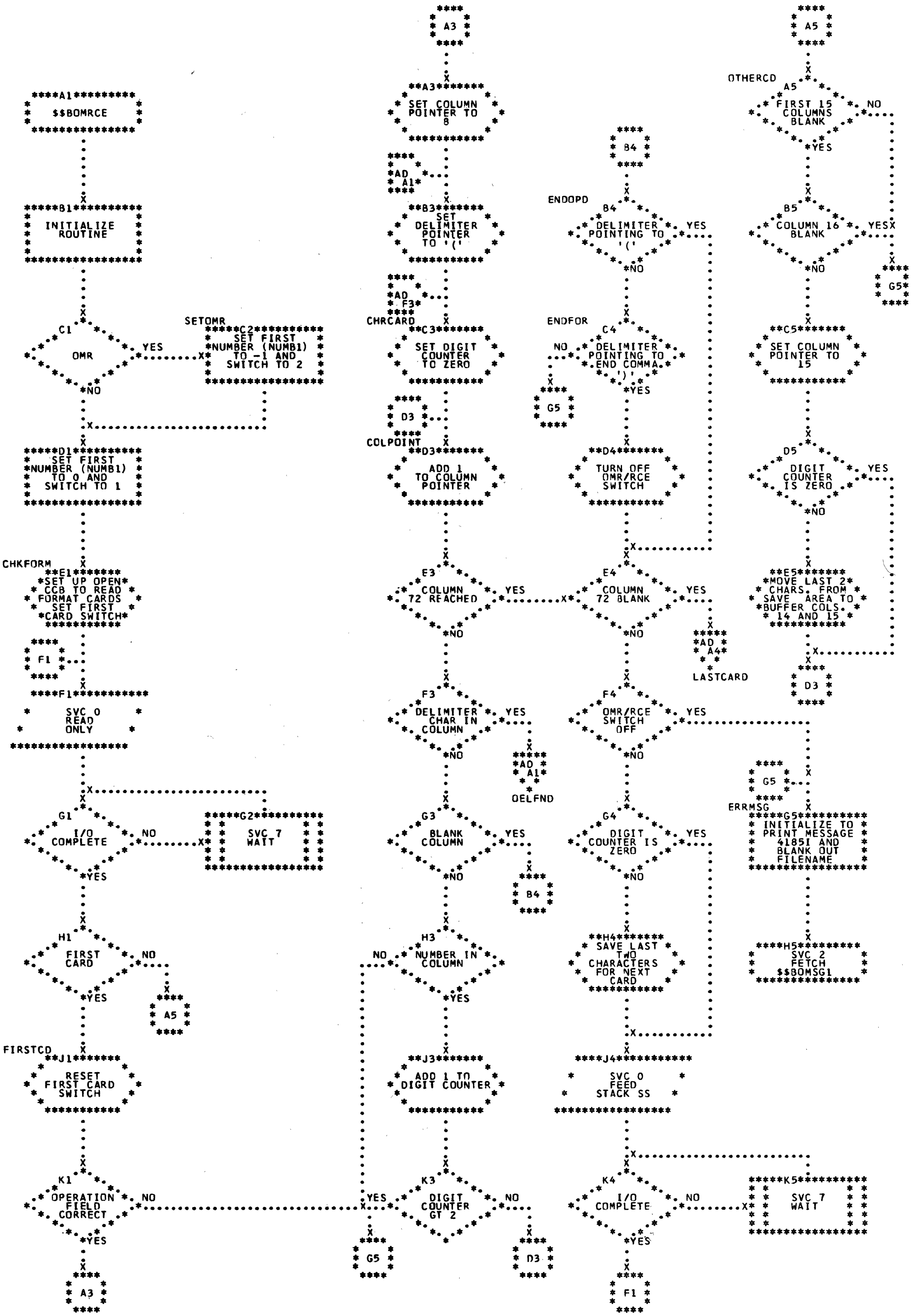


Chart AD. $$BCMRCE: OMR and RCE Open Routine (Part 2 of 2)

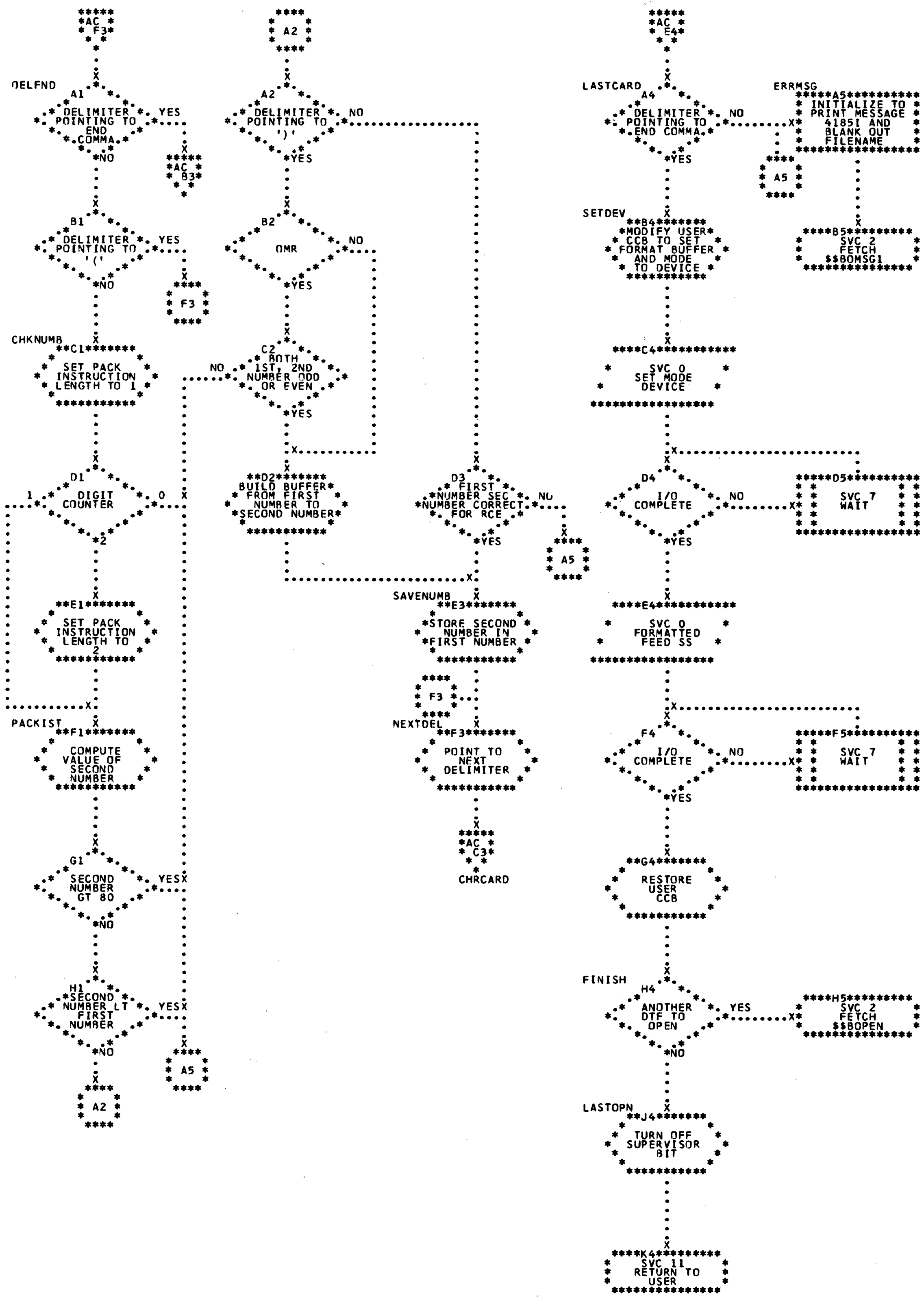
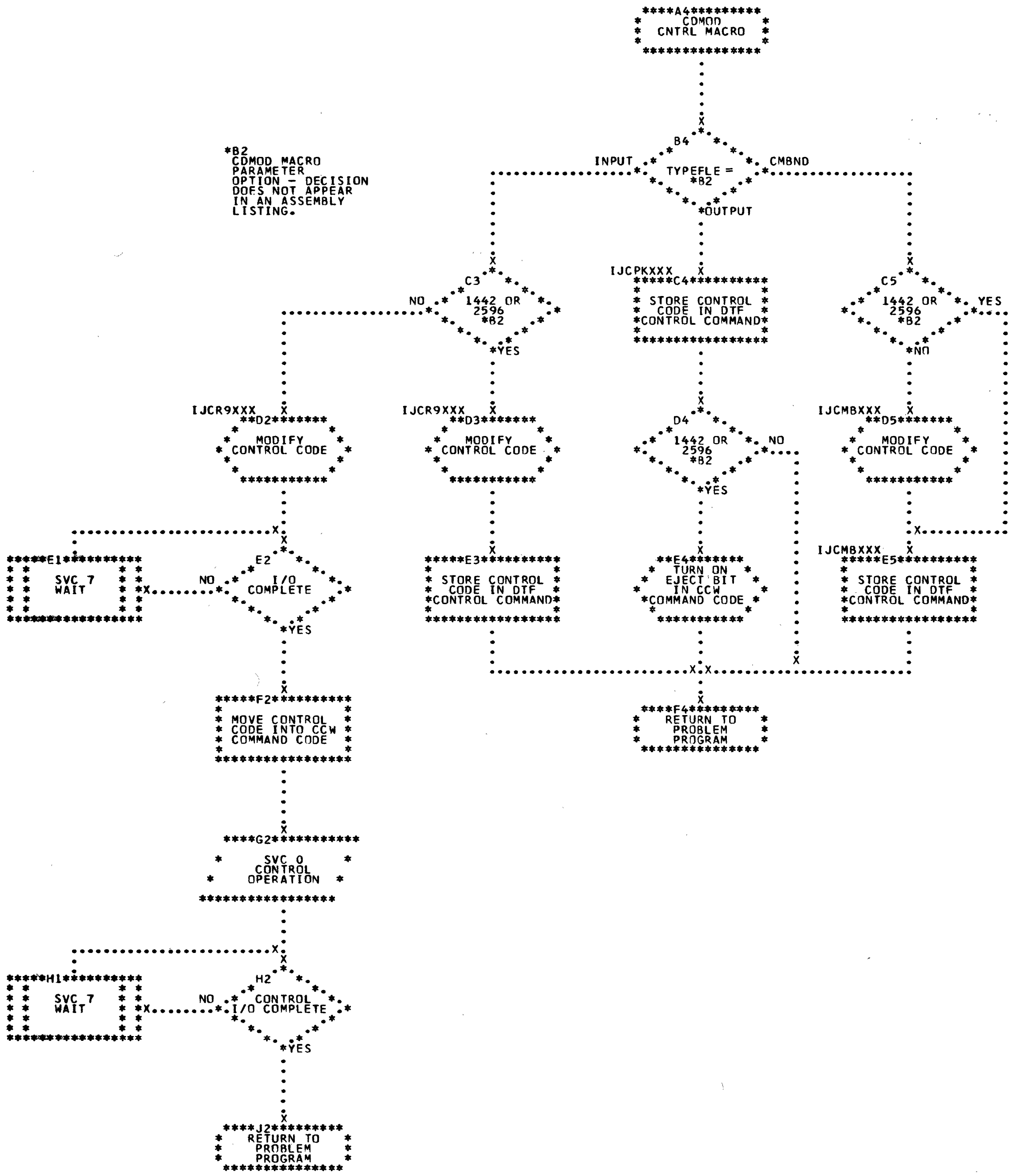


Chart AE. CDMOD: CNTRL Macro



A4 CDMOD CNTRL MACRO
*B2 CDMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
B4 TYPEFLE = *B2
INPUT
CMBND
OUTPUT
C3 1442 OR 2596 *B2
NO
YES
IJCPKXXX
C4 STORE CONTROL CODE IN DTF CONTROL COMMAND
C5 1442 OR 2596 *B2
YES
NO
IJCR9XXX
D2 MODIFY CONTROL CODE
IJCR9XXX
D3 MODIFY CONTROL CODE
D4 1442 OR 2596 *B2
NO
YES
IJCMBXXX
D5 MODIFY CONTROL CODE
E1 SVC 7 WAIT
E2 I/O COMPLETE
NO
YES
E3 STORE CONTROL CODE IN DTF CONTROL COMMAND
E4 TURN ON EJECT BIT IN CCW COMMAND CODE
IJCMBXXX
E5 STORE CONTROL CODE IN DTF CONTROL COMMAND
F2 MOVE CONTROL CODE INTO CCW COMMAND CODE
F4 RETURN TO PROBLEM PROGRAM
G2 SVC 0 CONTROL OPERATION
H1 SVC 7 WAIT
H2 CONTROL I/O COMPLETE
NO
YES
J2 RETURN TO PROBLEM PROGRAM

Chart AF. CDMOD: GET Macro (Part 1 of 4)

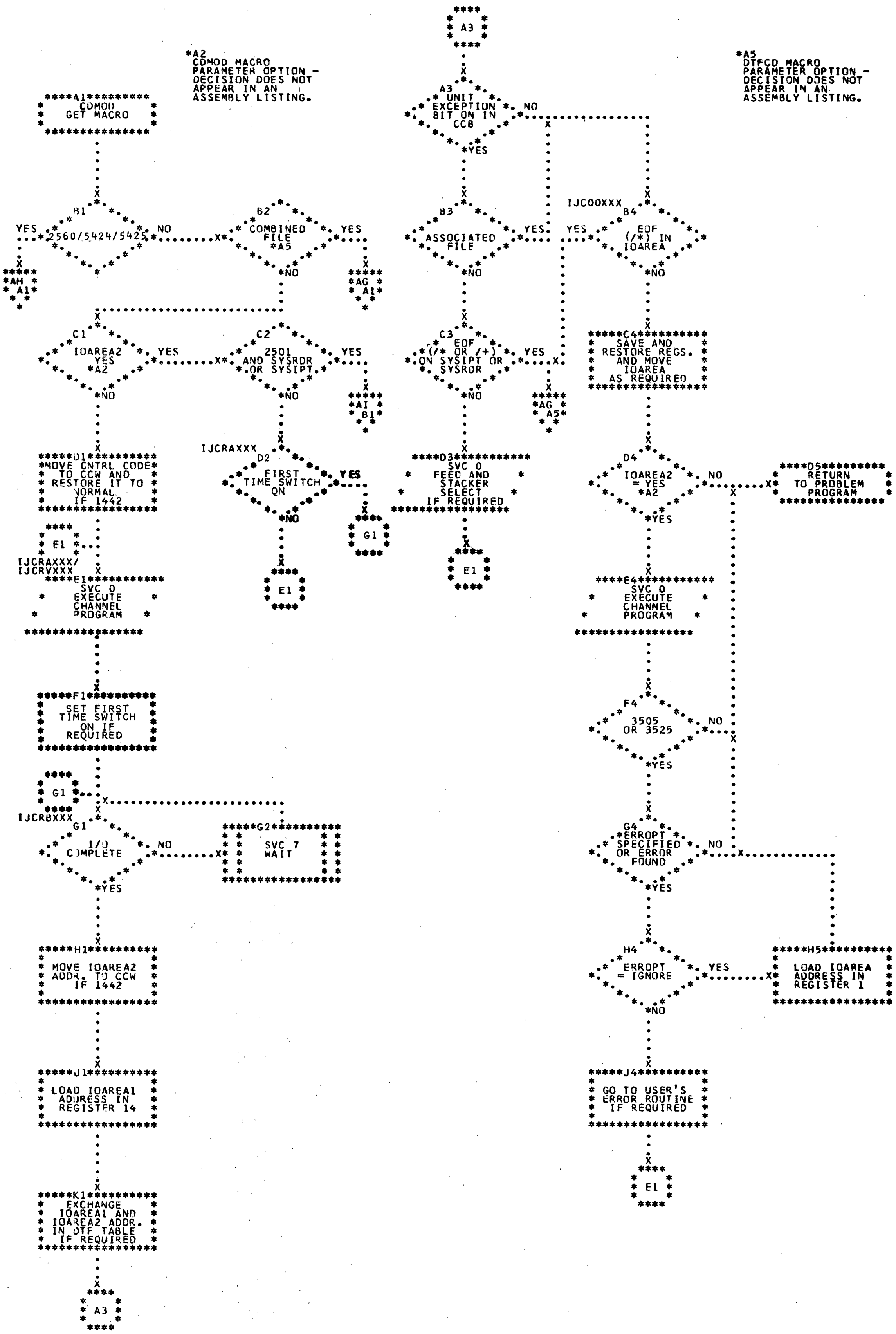

A2 CDMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
A5 DTFCD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
A3
A1 CDMOD GET MACRO
A3 UNIT EXCEPTION BIT ON IN CCB
NO
YES
B1 2560/5424/5425
YES
NO
AH A1
B2 COMBINED FILE A5
YES
NO
AG A1
B3 ASSOCIATED FILE
YES
NO
IJCOOXXX
B4 EOF (/*) IN IOAREA
YES
NO
C1 IOAREA2 YES A2
YES
NO
C2 2501 AND SYSRDR OR SYSIPT
YES
NO
AI B1
C3 EOF (/* OR /+) ON SYSIPT OR SYSRDR
YES
NO
AG A5
C4 SAVE AND RESTORE REGS. AND MOVE IOAREA AS REQUIRED
D1 MOVE CNTRL CODE TO CCW AND RESTORE IT TO NORMAL IF 1442
IJCRAXXX
D2 FIRST TIME SWITCH ON
YES
NO
G1
D3 SVC 0 FEED AND STACKER SELECT IF REQUIRED
D4 IOAREA2 = YES A2
NO
YES
D5 RETURN TO PROBLEM PROGRAM
E1
IJCRAXXX/ IJCRVXXX
E1 SVC 0 EXECUTE CHANNEL PROGRAM
E1
E1
E4 SVC 0 EXECUTE CHANNEL PROGRAM
F1 SET FIRST TIME SWITCH ON IF REQUIRED
F4 3505 OR 3525
NO
YES
G1
IJCRBXXX
G1 I/O COMPLETE
NO
YES
G2 SVC 7 WAIT
G4 ERROPT SPECIFIED OR ERROR FOUND
NO
YES
H1 MOVE IOAREA2 ADDR. TO CCW IF 1442
H4 ERROPT = IGNORE
YES
NO
H5 LOAD IOAREA ADDRESS IN REGISTER 1
J1 LOAD IOAREA1 ADDRESS IN REGISTER 14
J4 GO TO USER'S ERROR ROUTINE IF REQUIRED
E1
K1 EXCHANGE IOAREA1 AND IOAREA2 ADDR. IN DTF TABLE IF REQUIRED
A3

Chart AG. CDMOD: GET Macro (Part 2 of 4)

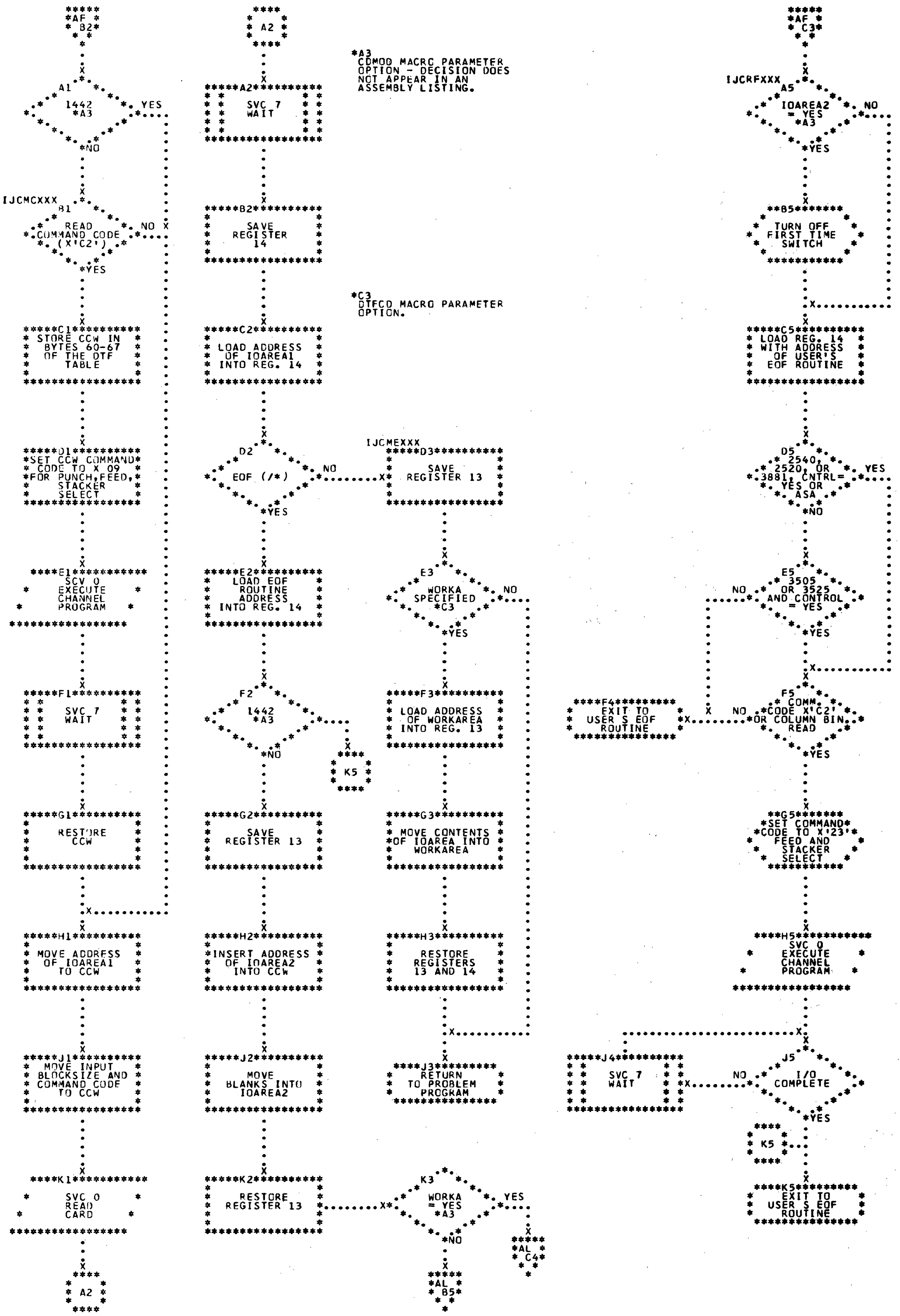

AF B2
A1
1442 *A3
YES
NO
IJCMCXXX
B1
READ COMMAND CODE (X'C2')
NO
YES
C1
STORE CCW IN BYTES 60-67 OF THE DTF TABLE
D1
SET CCW COMMAND CODE TO X 09 FOR PUNCH,FEED, STACKER SELECT
E1
SCV 0 EXECUTE CHANNEL PROGRAM
F1
SVC 7 WAIT
G1
RESTORE CCW
H1
MOVE ADDRESS OF IOAREA1 TO CCW
J1
MOVE INPUT BLOCKSIZE AND COMMAND CODE TO CCW
K1
SVC 0 READ CARD
A2
A2
SVC 7 WAIT
B2
SAVE REGISTER 14
C2
LOAD ADDRESS OF IOAREA1 INTO REG. 14
D2
EOF (/*)
NO
YES
E2
LOAD EOF ROUTINE ADDRESS INTO REG. 14
F2
1442 *A3
NO
K5
G2
SAVE REGISTER 13
H2
INSERT ADDRESS OF IOAREA2 INTO CCW
J2
MOVE BLANKS INTO IOAREA2
K2
RESTORE REGISTER 13
*A3
CDMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
*C3
DTFCD MACRO PARAMETER OPTION.
IJCMEXXX
D3
SAVE REGISTER 13
E3
WORKA SPECIFIED *C3
NO
YES
F3
LOAD ADDRESS OF WORKAREA INTO REG. 13
G3
MOVE CONTENTS OF IOAREA INTO WORKAREA
H3
RESTORE REGISTERS 13 AND 14
J3
RETURN TO PROBLEM PROGRAM
K3
WORKA = YES *A3
YES
NO
AL C4
AL B5
F4
EXIT TO USER S EOF ROUTINE
J4
SVC 7 WAIT
AF C3
IJCRFXXX
A5
IOAREA2 = YES *A3
NO
YES
B5
TURN OFF FIRST TIME SWITCH
C5
LOAD REG. 14 WITH ADDRESS OF USER'S EOF ROUTINE
D5
2540, 2520, OR 3881, CNTRL= YES OR ASA
YES
NO
E5
3505 OR 3525 AND CONTROL = YES
NO
YES
F5
COMM. CODE X'C2' OR COLUMN BIN. READ
NO
YES
G5
SET COMMAND CODE TO X'23' FEED AND STACKER SELECT
H5
SVC 0 EXECUTE CHANNEL PROGRAM
J5
I/O COMPLETE
NO
YES
K5
K5
EXIT TO USER S EOF ROUTINE

Chart AH. CDMOD: GET Macro (Part 3 of 4)

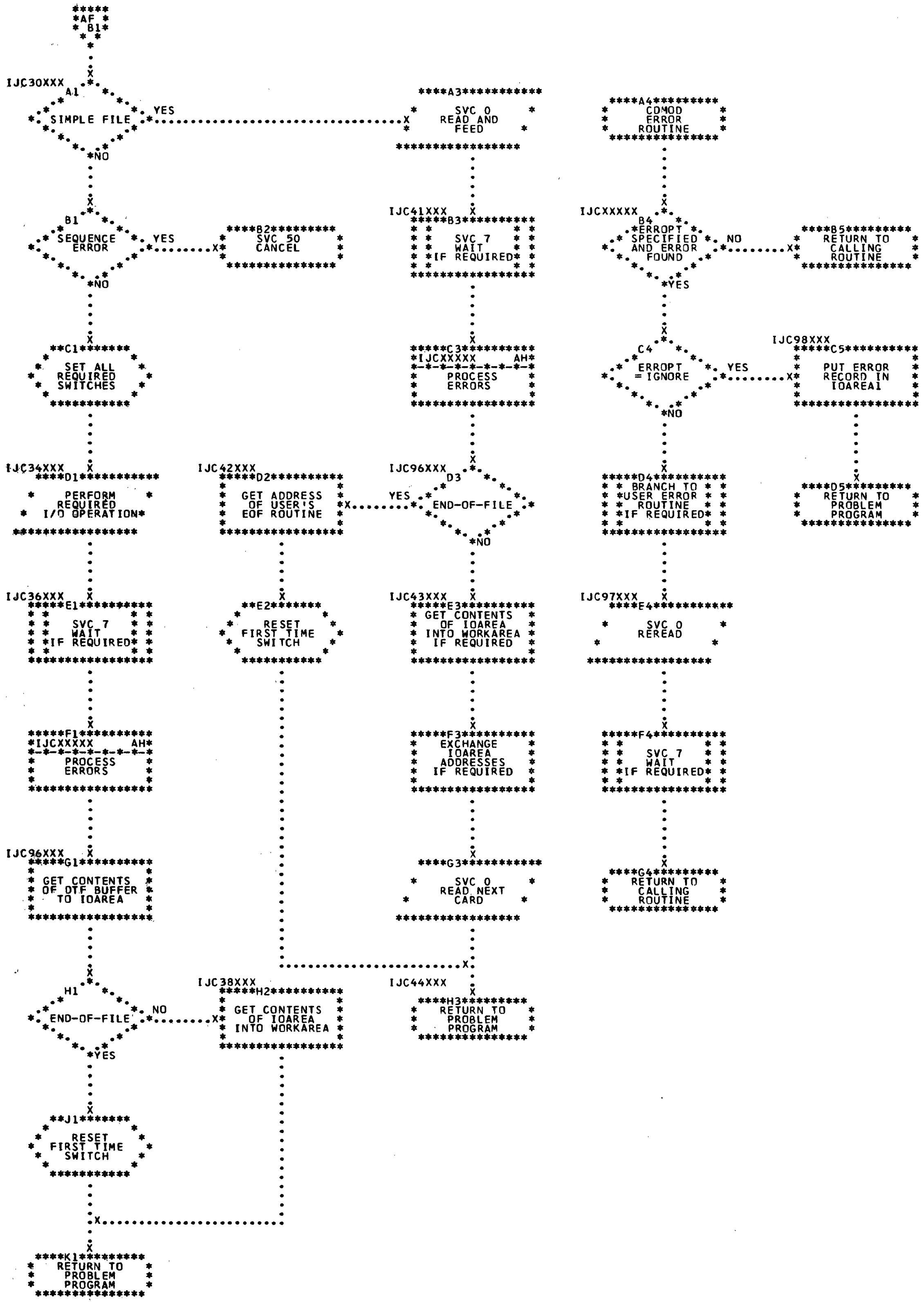

AF B1
IJC30XXX A1 SIMPLE FILE
YES
NO
A3 SVC 0 READ AND FEED
A4 CDMOD ERROR ROUTINE
B1 SEQUENCE ERROR
YES
NO
B2 SVC 50 CANCEL
IJC41XXX B3 SVC 7 WAIT IF REQUIRED
IJCXXXXX B4 ERROPT SPECIFIED AND ERROR FOUND
NO
YES
B5 RETURN TO CALLING ROUTINE
C1 SET ALL REQUIRED SWITCHES
C3 IJCXXXXX AH PROCESS ERRORS
C4 ERROPT = IGNORE
YES
NO
IJC98XXX C5 PUT ERROR RECORD IN IOAREA1
IJC34XXX D1 PERFORM REQUIRED I/O OPERATION
IJC42XXX D2 GET ADDRESS OF USER'S EOF ROUTINE
IJC96XXX D3 END-OF-FILE
YES
NO
D4 BRANCH TO USER ERROR ROUTINE IF REQUIRED
D5 RETURN TO PROBLEM PROGRAM
IJC36XXX E1 SVC 7 WAIT IF REQUIRED
E2 RESET FIRST TIME SWITCH
IJC43XXX E3 GET CONTENTS OF IOAREA INTO WORKAREA IF REQUIRED
IJC97XXX E4 SVC 0 REREAD
F1 IJCXXXXX AH PROCESS ERRORS
F3 EXCHANGE IOAREA ADDRESSES IF REQUIRED
F4 SVC 7 WAIT IF REQUIRED
IJC96XXX G1 GET CONTENTS OF DTF BUFFER TO IOAREA
G3 SVC 0 READ NEXT CARD
G4 RETURN TO CALLING ROUTINE
H1 END-OF-FILE
NO
YES
IJC38XXX H2 GET CONTENTS OF IOAREA INTO WORKAREA
IJC44XXX H3 RETURN TO PROBLEM PROGRAM
J1 RESET FIRST TIME SWITCH
K1 RETURN TO PROBLEM PROGRAM

Chart AI. CDMCI: GET Macro (Part 4 of 4)

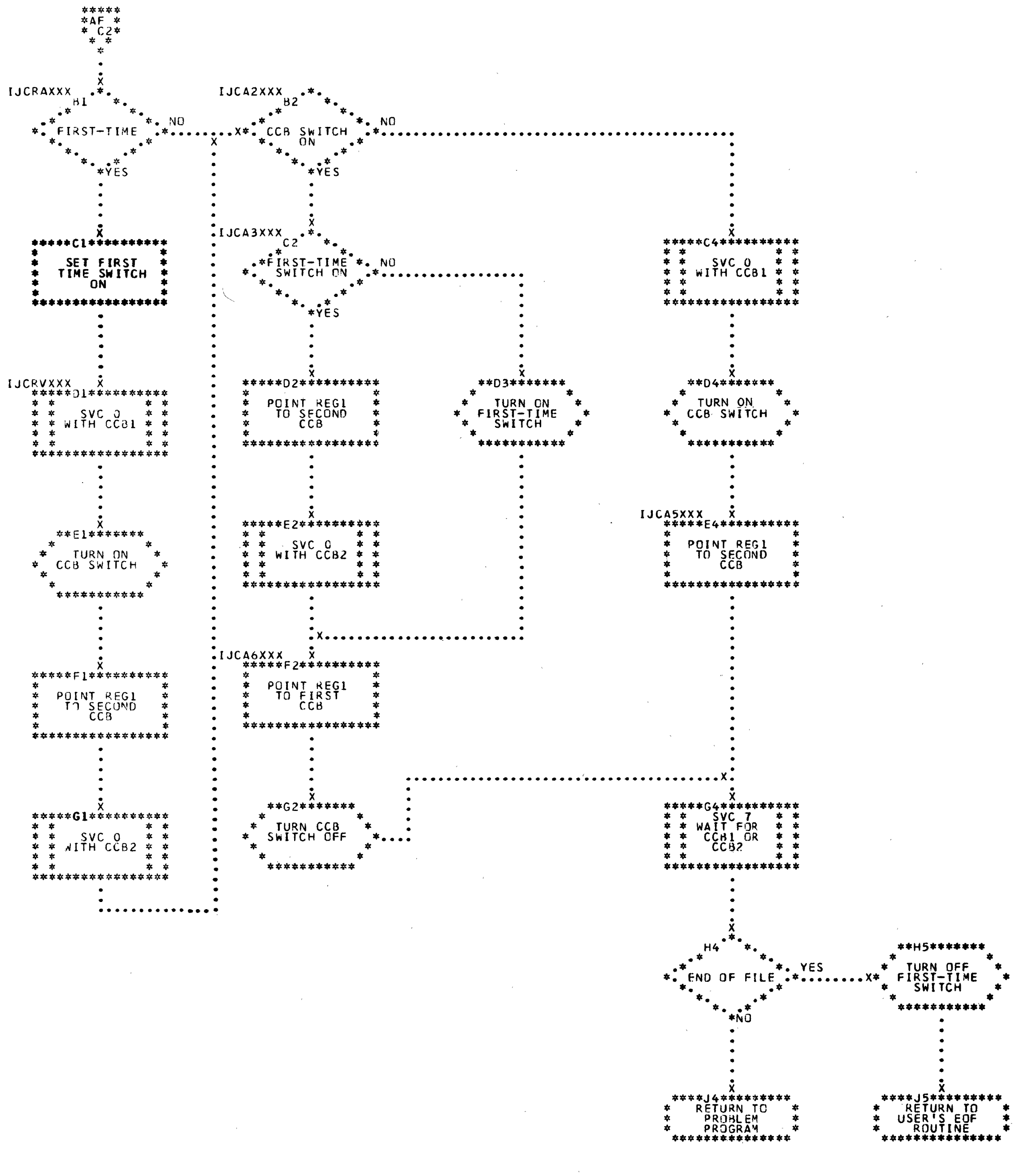


Chart AJ. CDMOD: PUT Macro (Part 1 of 4)

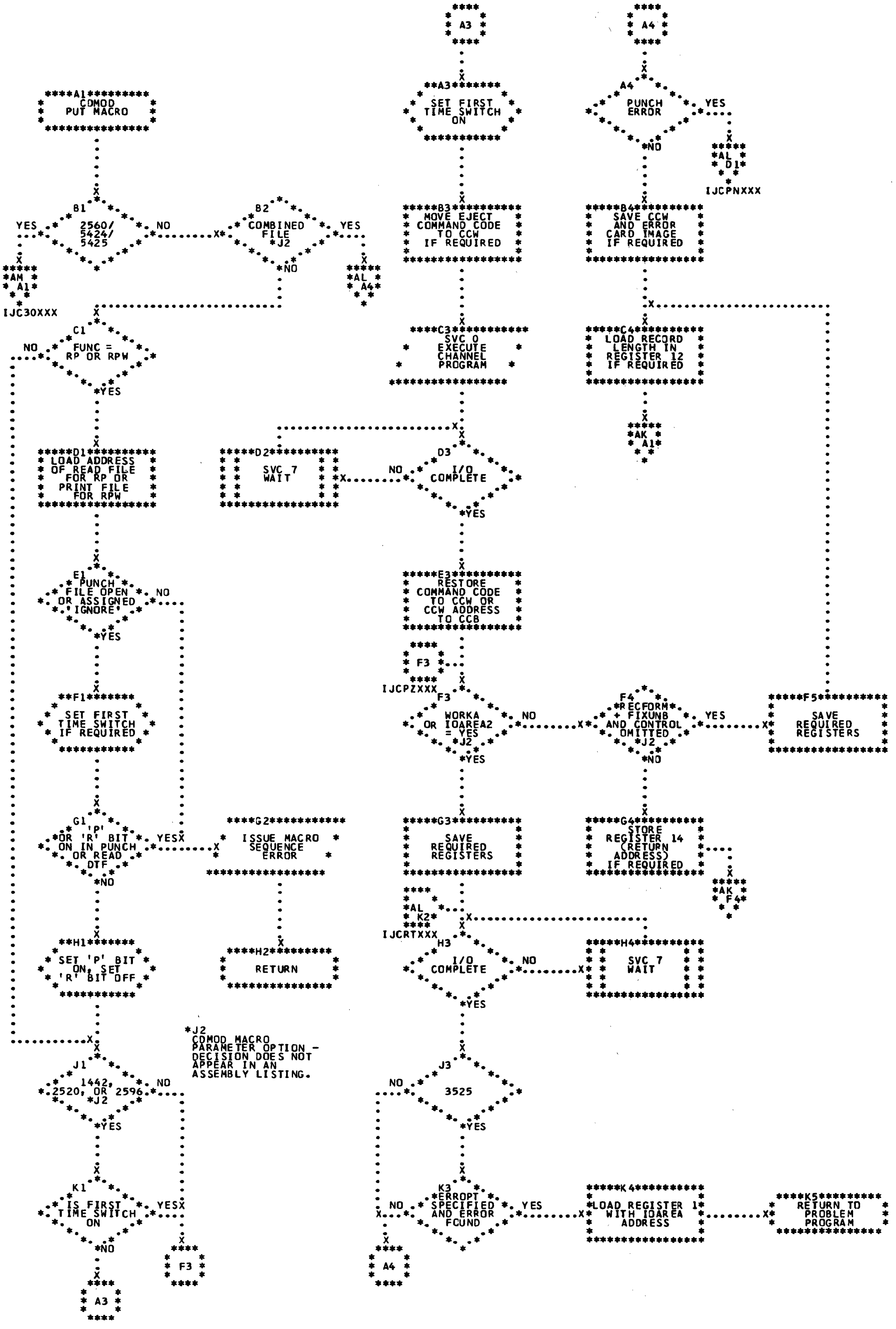
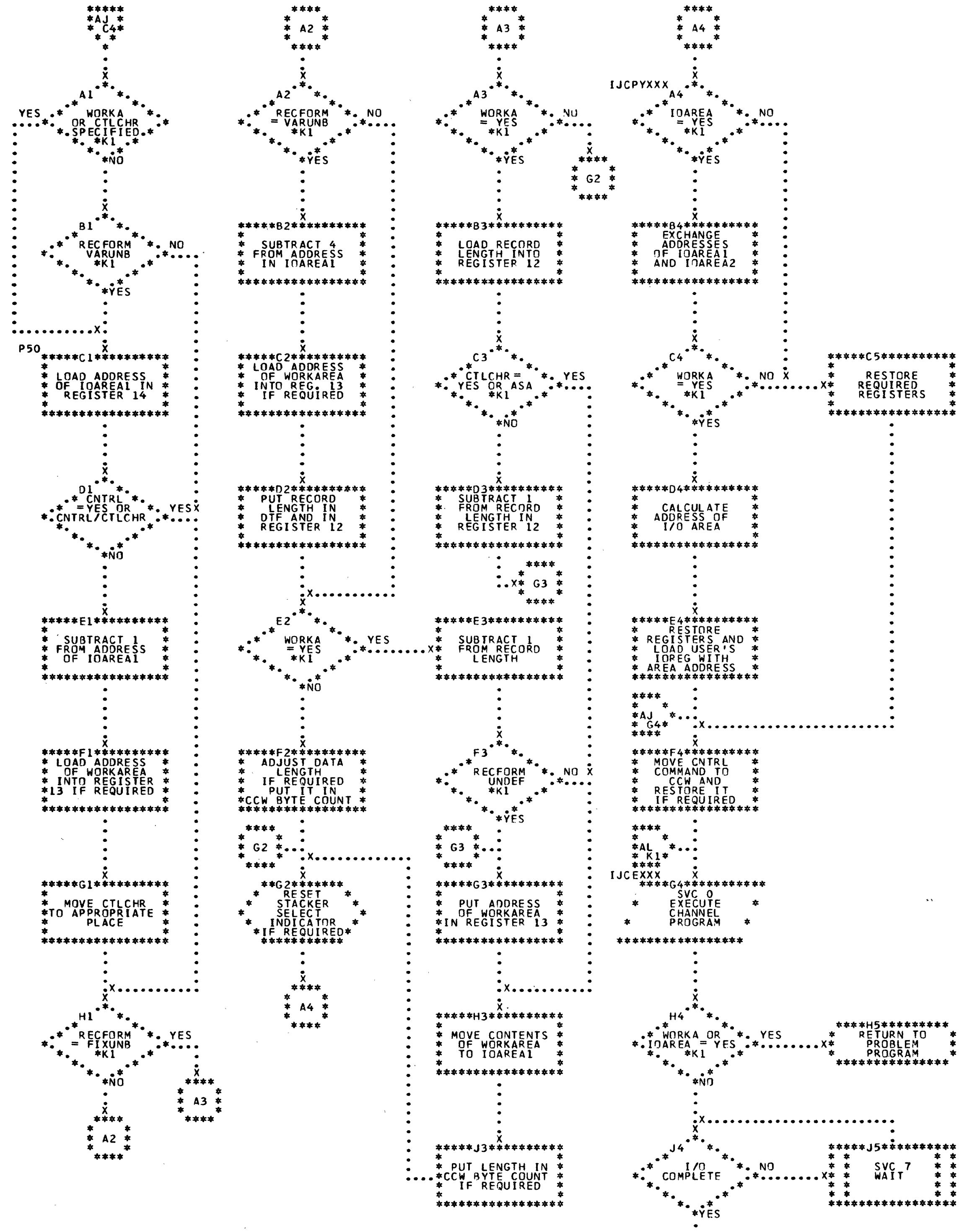

A3
A4
A1
CDMOD PUT MACRO
A3
SET FIRST TIME SWITCH ON
A4
PUNCH ERROR
YES
NO
AL D1
IJCPNXXX
B1
2560/ 5424/ 5425
YES
NO
AM A1
IJC30XXX
B2
COMBINED FILE *J2
YES
NO
AL A4
B3
MOVE EJECT COMMAND CODE TO CCW IF REQUIRED
B4
SAVE CCW AND ERROR CARD IMAGE IF REQUIRED
C1
FUNC = RP OR RPW
NO
YES
C3
SVC 0 EXECUTE CHANNEL PROGRAM
C4
LOAD RECORD LENGTH IN REGISTER 12 IF REQUIRED
AK A1
D1
LOAD ADDRESS OF READ FILE FOR RP OR PRINT FILE FOR RPW
D2
SVC 7 WAIT
D3
I/O COMPLETE
NO
YES
E1
PUNCH FILE OPEN OR ASSIGNED 'IGNORE'
NO
YES
E3
RESTORE COMMAND CODE TO CCW OR CCW ADDRESS TO CCB
F3
IJCPZXXX
F1
SET FIRST TIME SWITCH IF REQUIRED
F3
WORKA OR IOAREA2 = YES *J2
NO
YES
F4
RECFORM= FIXUNB AND CONTROL OMITTED *J2
YES
NO
F5
SAVE REQUIRED REGISTERS
G1
'P' OR 'R' BIT ON IN PUNCH OR READ DTF
YES
NO
G2
ISSUE MACRO SEQUENCE ERROR
G3
SAVE REQUIRED REGISTERS
G4
STORE REGISTER 14 (RETURN ADDRESS) IF REQUIRED
AK F4
AL K2
IJCRTXXX
H1
SET 'P' BIT ON, SET 'R' BIT OFF
H2
RETURN
H3
I/O COMPLETE
NO
YES
H4
SVC 7 WAIT
*J2 CDMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
J1
1442, 2520, OR 2596 *J2
NO
YES
J3
3525
NO
YES
K1
IS FIRST TIME SWITCH ON
YES
NO
K3
ERROPT SPECIFIED AND ERROR FCUND
NO
YES
K4
LOAD REGISTER 1 WITH IOAREA ADDRESS
K5
RETURN TO PROBLEM PROGRAM
F3
A3
A4

Chart AK. CDMOD: PUT Macro (Part 2 of 4)


AJ C4
A1 WORKA OR CTLCHR SPECIFIED *K1
YES
NO
B1 RECFORM VARUNB *K1
NO
YES
P50
C1 LOAD ADDRESS OF IOAREA1 IN REGISTER 14
D1 CNTRL =YES OR CNTRL/CTLCHR
YES
NO
E1 SUBTRACT 1 FROM ADDRESS OF IOAREA1
F1 LOAD ADDRESS OF WORKAREA INTO REGISTER 13 IF REQUIRED
G1 MOVE CTLCHR TO APPROPRIATE PLACE
H1 RECFORM = FIXUNB *K1
YES
NO
A3
A2
A2 RECFORM = VARUNB *K1
NO
YES
B2 SUBTRACT 4 FROM ADDRESS IN IOAREA1
C2 LOAD ADDRESS OF WORKAREA INTO REG. 13 IF REQUIRED
D2 PUT RECORD LENGTH IN DTF AND IN REGISTER 12
E2 WORKA = YES *K1
YES
NO
F2 ADJUST DATA LENGTH IF REQUIRED PUT IT IN CCW BYTE COUNT
G2
G2 RESET STACKER SELECT INDICATOR IF REQUIRED
A4
A3
A3 WORKA = YES *K1
NO
YES
G2
B3 LOAD RECORD LENGTH INTO REGISTER 12
C3 CTLCHR= YES OR ASA *K1
YES
NO
D3 SUBTRACT 1 FROM RECORD LENGTH IN REGISTER 12
G3
E3 SUBTRACT 1 FROM RECORD LENGTH
F3 RECFORM UNDEF *K1
NO
YES
G3
G3 PUT ADDRESS OF WORKAREA IN REGISTER 13
H3 MOVE CONTENTS OF WORKAREA TO IOAREA1
J3 PUT LENGTH IN CCW BYTE COUNT IF REQUIRED
A4
IJCPYXXX
A4 IOAREA = YES *K1
NO
YES
B4 EXCHANGE ADDRESSES OF IOAREA1 AND IOAREA2
C4 WORKA = YES *K1
NO
YES
C5 RESTORE REQUIRED REGISTERS
D4 CALCULATE ADDRESS OF I/O AREA
E4 RESTORE REGISTERS AND LOAD USER'S IOREG WITH AREA ADDRESS
AJ G4
F4 MOVE CNTRL COMMAND TO CCW AND RESTORE IT IF REQUIRED
AL K1
IJCEXXX
G4 SVC 0 EXECUTE CHANNEL PROGRAM
H4 WORKA OR IOAREA = YES *K1
YES
NO
H5 RETURN TO PROBLEM PROGRAM
J4 I/O COMPLETE
NO
YES
J5 SVC 7 WAIT
AL A1

*K1
CDMOD MACRO
PARAMETER OPTION -
DECISION DOES NOT
APPEAR IN AN
ASSEMBLY LISTING.

Chart AL. CDMOD: PUT Macro (Part 3 of 4)

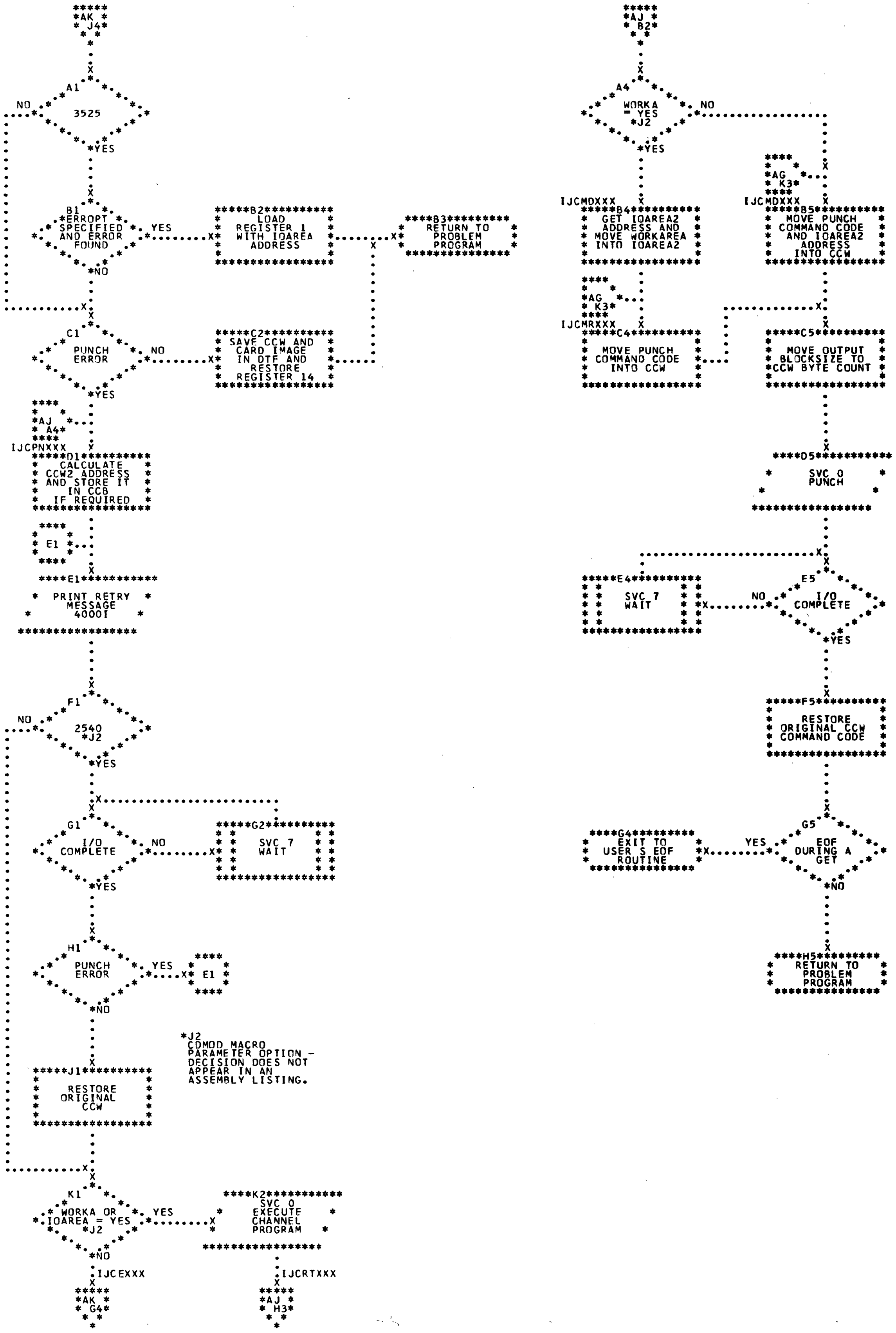

AK
J4
A1
3525
NO
YES
B1
ERROPT SPECIFIED AND ERROR FOUND
YES
NO
B2
LOAD REGISTER 1 WITH IOAREA ADDRESS
B3
RETURN TO PROBLEM PROGRAM
C1
PUNCH ERROR
NO
YES
C2
SAVE CCW AND CARD IMAGE IN DTF AND RESTORE REGISTER 14
AJ
A4
IJCPNXXX
D1
CALCULATE CCW2 ADDRESS AND STORE IT IN CCB IF REQUIRED
E1
E1
PRINT RETRY MESSAGE 4000I
F1
2540 *J2
NO
YES
G1
I/O COMPLETE
NO
YES
G2
SVC 7 WAIT
H1
PUNCH ERROR
YES
E1
NO
*J2
CDMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
J1
RESTORE ORIGINAL CCW
K1
WORKA OR IOAREA = YES *J2
YES
NO
K2
SVC 0 EXECUTE CHANNEL PROGRAM
IJCEXXX
AK
G4
IJCRTXXX
AJ
H3
AJ
B2
A4
WORKA = YES *J2
NO
YES
AG
K3
IJCMDXXX
IJCMDXXX
B4
GET IOAREA2 ADDRESS AND MOVE WORKAREA INTO IOAREA2
B5
MOVE PUNCH COMMAND CODE AND IOAREA2 ADDRESS INTO CCW
AG
K3
IJCMRXXX
C4
MOVE PUNCH COMMAND CODE INTO CCW
C5
MOVE OUTPUT BLOCKSIZE TO CCW BYTE COUNT
D5
SVC 0 PUNCH
E4
SVC 7 WAIT
E5
I/O COMPLETE
NO
YES
F5
RESTORE ORIGINAL CCW COMMAND CODE
G4
EXIT TO USER S EOF ROUTINE
G5
EOF DURING A GET
YES
NO
H5
RETURN TO PROBLEM PROGRAM

Chart AM. CDMCD: PUT Macro (Part 4 of 4)

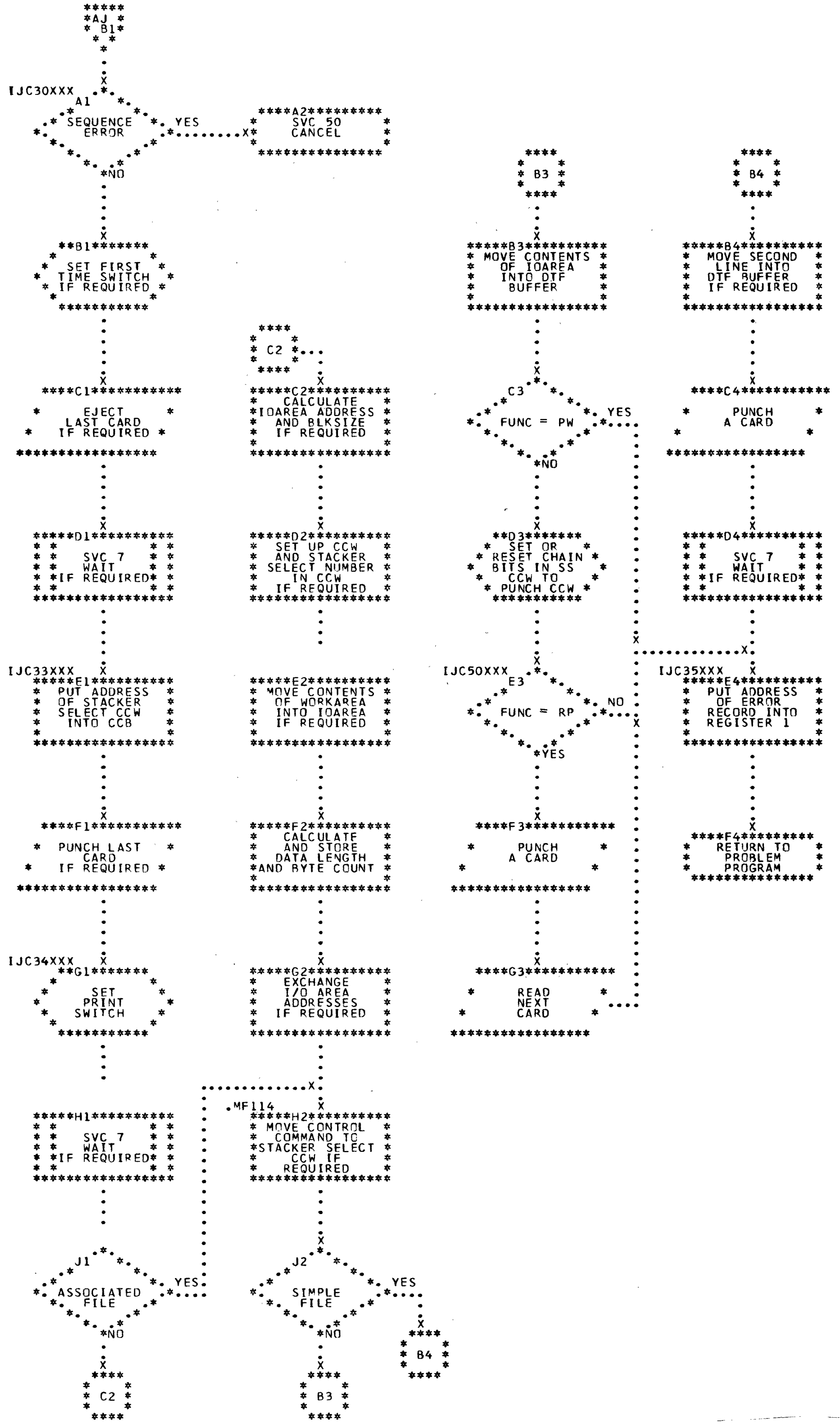

AJ B1
IJC30XXX
A1
SEQUENCE ERROR
YES
NO
A2
SVC 50 CANCEL
B3
B4
B1
SET FIRST TIME SWITCH IF REQUIRED
B3
MOVE CONTENTS OF IOAREA INTO DTF BUFFER
B4
MOVE SECOND LINE INTO DTF BUFFER IF REQUIRED
C2
C1
EJECT LAST CARD IF REQUIRED
C2
CALCULATE IOAREA ADDRESS AND BLKSIZE IF REQUIRED
C3
FUNC = PW
YES
NO
C4
PUNCH A CARD
D1
SVC 7 WAIT IF REQUIRED
D2
SET UP CCW AND STACKER SELECT NUMBER IN CCW IF REQUIRED
D3
SET OR RESET CHAIN BITS IN SS CCW TO PUNCH CCW
D4
SVC 7 WAIT IF REQUIRED
IJC33XXX
E1
PUT ADDRESS OF STACKER SELECT CCW INTO CCB
E2
MOVE CONTENTS OF WORKAREA INTO IOAREA IF REQUIRED
IJC50XXX
E3
FUNC = RP
NO
YES
IJC35XXX
E4
PUT ADDRESS OF ERROR RECORD INTO REGISTER 1
F1
PUNCH LAST CARD IF REQUIRED
F2
CALCULATE AND STORE DATA LENGTH AND BYTE COUNT
F3
PUNCH A CARD
F4
RETURN TO PROBLEM PROGRAM
IJC34XXX
G1
SET PRINT SWITCH
G2
EXCHANGE I/O AREA ADDRESSES IF REQUIRED
G3
READ NEXT CARD
H1
SVC 7 WAIT IF REQUIRED
.MF114
H2
MOVE CONTROL COMMAND TO STACKER SELECT CCW IF REQUIRED
J1
ASSOCIATED FILE
YES
NO
J2
SIMPLE FILE
YES
NO
B4
C2
B3

Chart AN. DTFCN: GET Macro

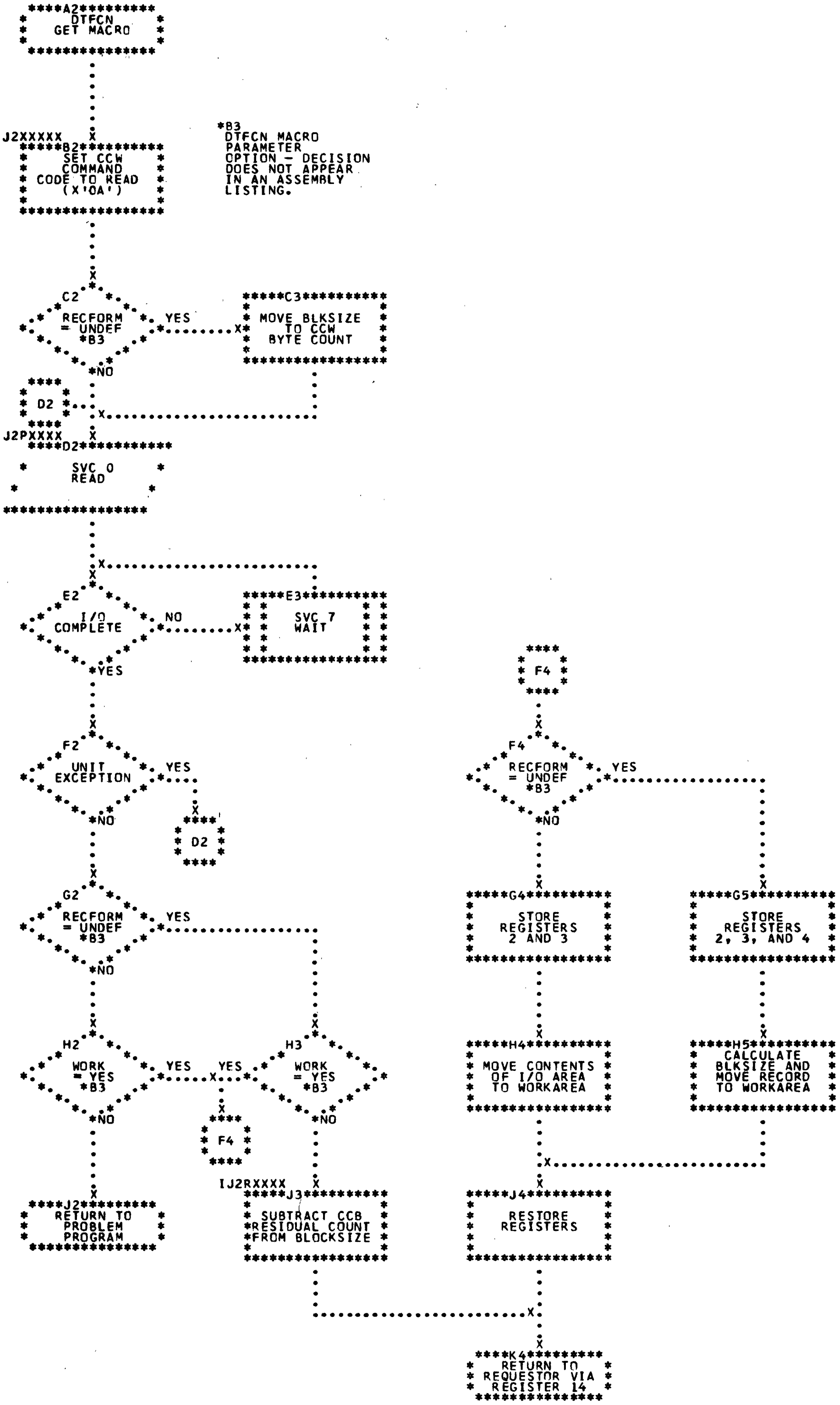

A2
DTFCN GET MACRO
IJ2XXXXX
B2
SET CCW COMMAND CODE TO READ (X'0A')
*B3
DTFCN MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
C2
RECFORM = UNDEF *B3
YES
NO
C3
MOVE BLKSIZE TO CCW BYTE COUNT
D2
IJ2PXXXX
D2
SVC 0 READ
E2
I/O COMPLETE
NO
YES
E3
SVC 7 WAIT
F2
UNIT EXCEPTION
YES
NO
D2
G2
RECFORM = UNDEF *B3
YES
NO
H2
WORK = YES *B3
YES
NO
H3
WORK = YES *B3
YES
NO
F4
J2
RETURN TO PROBLEM PROGRAM
IJ2RXXXX
J3
SUBTRACT CCB RESIDUAL COUNT FROM BLOCKSIZE
F4
F4
RECFORM = UNDEF *B3
YES
NO
G4
STORE REGISTERS 2 AND 3
G5
STORE REGISTERS 2, 3, AND 4
H4
MOVE CONTENTS OF I/O AREA TO WORKAREA
H5
CALCULATE BLKSIZE AND MOVE RECORD TO WORKAREA
J4
RESTORE REGISTERS
K4
RETURN TO REQUESTOR VIA REGISTER 14

Chart AP. DTFCN: PUT Macro

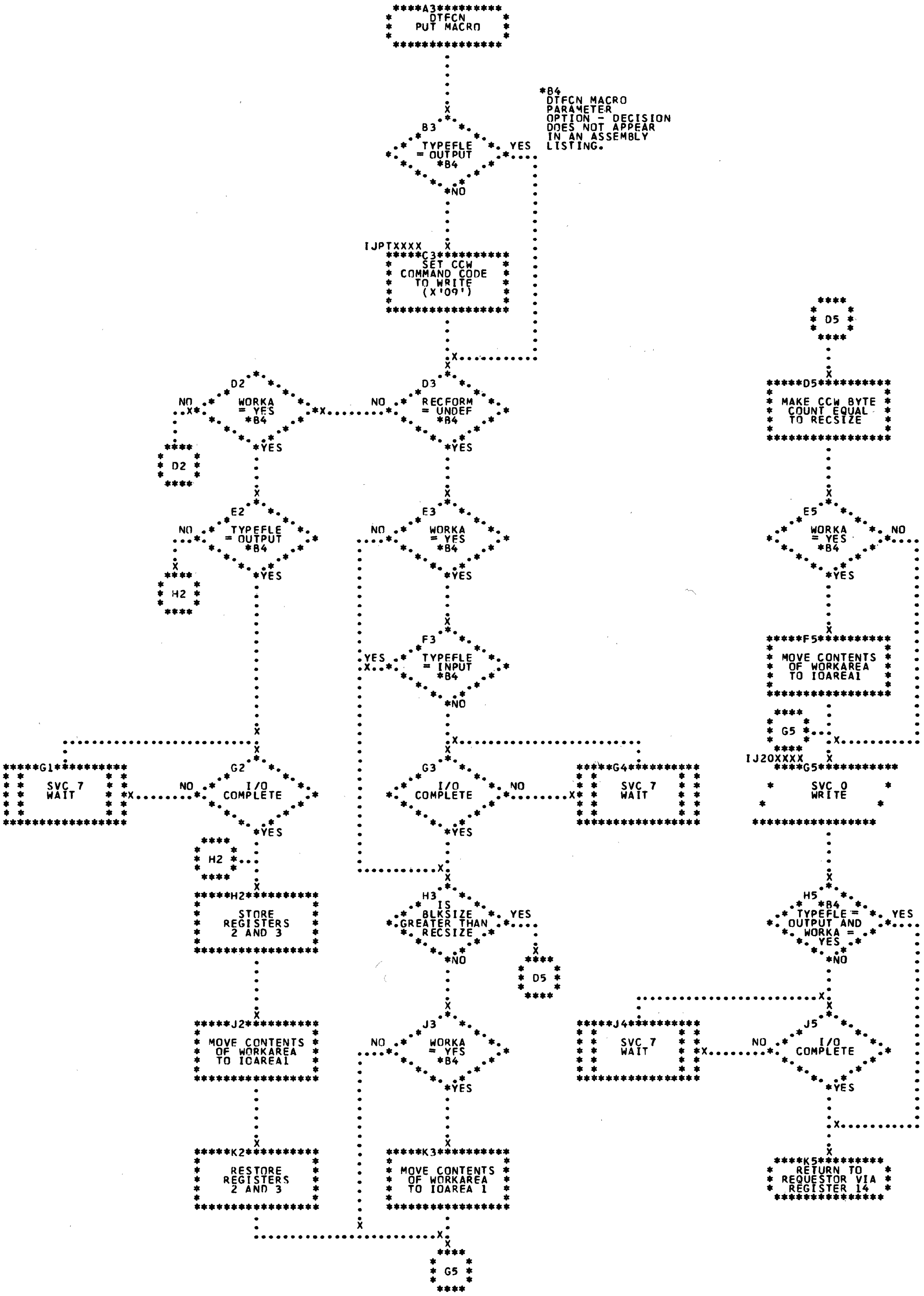


Chart AQ. DTFCN: PUTR Macro

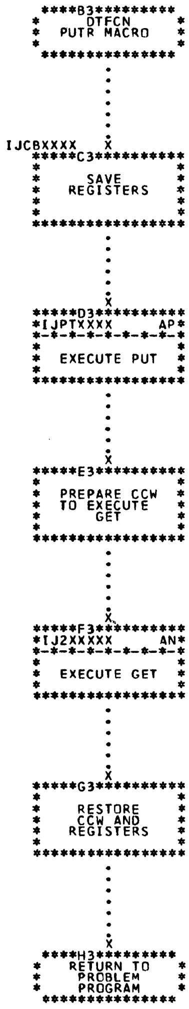


Chart BA. MRMOD: GET, READ, and CHECK Macros (Part 1 of 2)

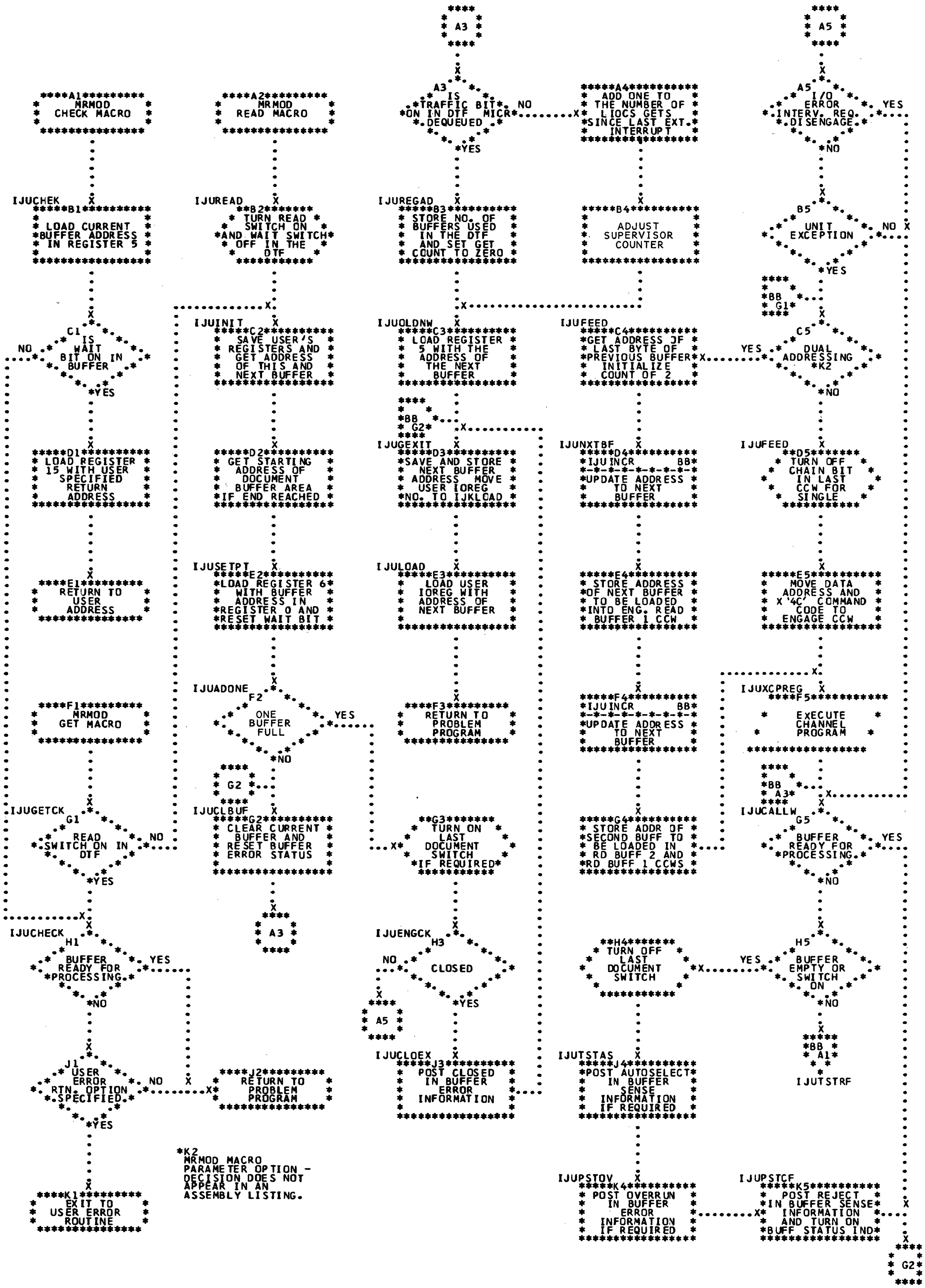

A3
A5
A1 MRMOD CHECK MACRO
A2 MRMOD READ MACRO
A3 IS TRAFFIC BIT ON IN DTF MICR DEQUEUED
NO
YES
A4 ADD ONE TO THE NUMBER OF LIOCS GETS SINCE LAST EXT. INTERRUPT
A5 I/O ERROR INTERV. REQ. DISENGAGE
YES
NO
IJUCHEK
B1 LOAD CURRENT BUFFER ADDRESS IN REGISTER 5
IJUREAD
B2 TURN READ SWITCH ON AND WAIT SWITCH OFF IN THE DTF
IJUREGAD
B3 STORE NO. OF BUFFERS USED IN THE DTF AND SET GET COUNT TO ZERO
B4 ADJUST SUPERVISOR COUNTER
B5 UNIT EXCEPTION
NO
YES
BB G1
C1 IS WAIT BIT ON IN BUFFER
NO
YES
IJUINIT
C2 SAVE USER'S REGISTERS AND GET ADDRESS OF THIS AND NEXT BUFFER
IJUOLDNW
C3 LOAD REGISTER 5 WITH THE ADDRESS OF THE NEXT BUFFER
IJUFEED
C4 GET ADDRESS OF LAST BYTE OF PREVIOUS BUFFER INITIALIZE COUNT OF 2
C5 DUAL ADDRESSING *K2
YES
NO
BB G2
D1 LOAD REGISTER 15 WITH USER SPECIFIED RETURN ADDRESS
D2 GET STARTING ADDRESS OF DOCUMENT BUFFER AREA IF END REACHED
IJUGEXIT
D3 SAVE AND STORE NEXT BUFFER ADDRESS MOVE USER IOREG NO. TO IJKLOAD
IJUNXTBF
D4 IJUINCR BB UPDATE ADDRESS TO NEXT BUFFER
IJUFEED
D5 TURN OFF CHAIN BIT IN LAST CCW FOR SINGLE
E1 RETURN TO USER ADDRESS
IJUSETPT
E2 LOAD REGISTER 6 WITH BUFFER ADDRESS IN REGISTER 0 AND RESET WAIT BIT
IJULOAD
E3 LOAD USER IOREG WITH ADDRESS OF NEXT BUFFER
E4 STORE ADDRESS OF NEXT BUFFER TO BE LOADED INTO ENG. READ BUFFER 1 CCW
E5 MOVE DATA ADDRESS AND X'4C' COMMAND CODE TO ENGAGE CCW
IJUADONE
F2 ONE BUFFER FULL
YES
NO
F1 MRMOD GET MACRO
F3 RETURN TO PROBLEM PROGRAM
F4 IJUINCR BB UPDATE ADDRESS TO NEXT BUFFER
IJUXCPREG
F5 EXECUTE CHANNEL PROGRAM
G2
BB A3
IJUGETCK
G1 READ SWITCH ON IN DTF
NO
YES
IJUCLBUF
G2 CLEAR CURRENT BUFFER AND RESET BUFFER ERROR STATUS
G3 TURN ON LAST DOCUMENT SWITCH IF REQUIRED
G4 STORE ADDR OF SECOND BUFF TO BE LOADED IN RD BUFF 2 AND RD BUFF 1 CCWS
IJUCALLW
G5 BUFFER READY FOR PROCESSING
YES
NO
A3
IJUCHECK
H1 BUFFER READY FOR PROCESSING
YES
NO
IJUENGCK
H3 CLOSED
NO
YES
A5
H4 TURN OFF LAST DOCUMENT SWITCH
H5 BUFFER EMPTY OR SWITCH ON
YES
NO
BB A1
IJUTSTRF
J1 USER ERROR RTN. OPTION SPECIFIED
NO
YES
J2 RETURN TO PROBLEM PROGRAM
IJUCLOEX
J3 POST CLOSED IN BUFFER ERROR INFORMATION
IJUTSTAS
J4 POST AUTOSELECT IN BUFFER SENSE INFORMATION IF REQUIRED
*K2 MRMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
K1 EXIT TO USER ERROR ROUTINE
IJUPSTOV
K4 POST OVERRUN IN BUFFER ERROR INFORMATION IF REQUIRED
IJUPSTCF
K5 POST REJECT IN BUFFER SENSE INFORMATION AND TURN ON BUFF STATUS IND
G2

Chart BB. MRMOD: GET, READ, and CHECK Macros (Part 2 of 2)

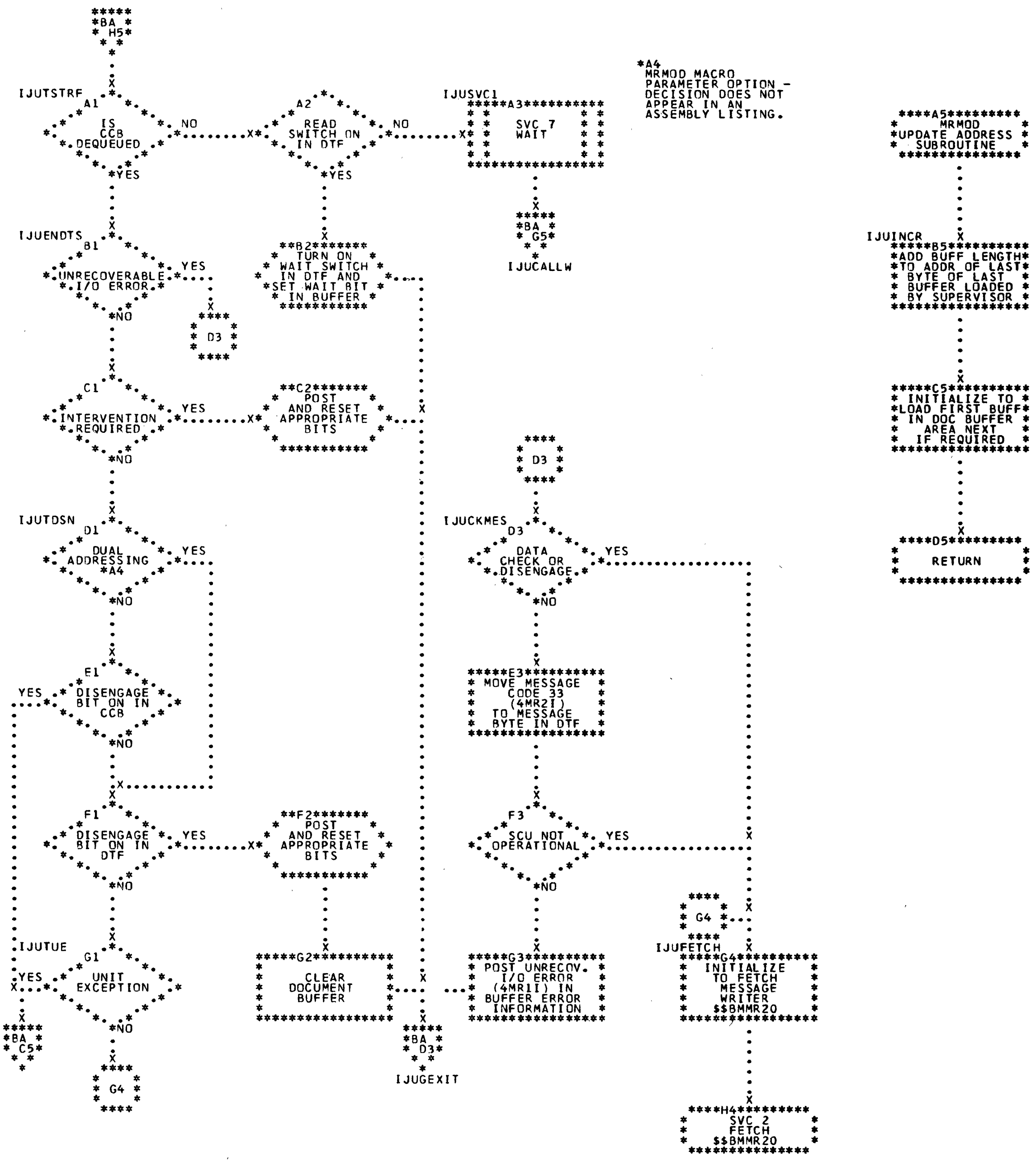
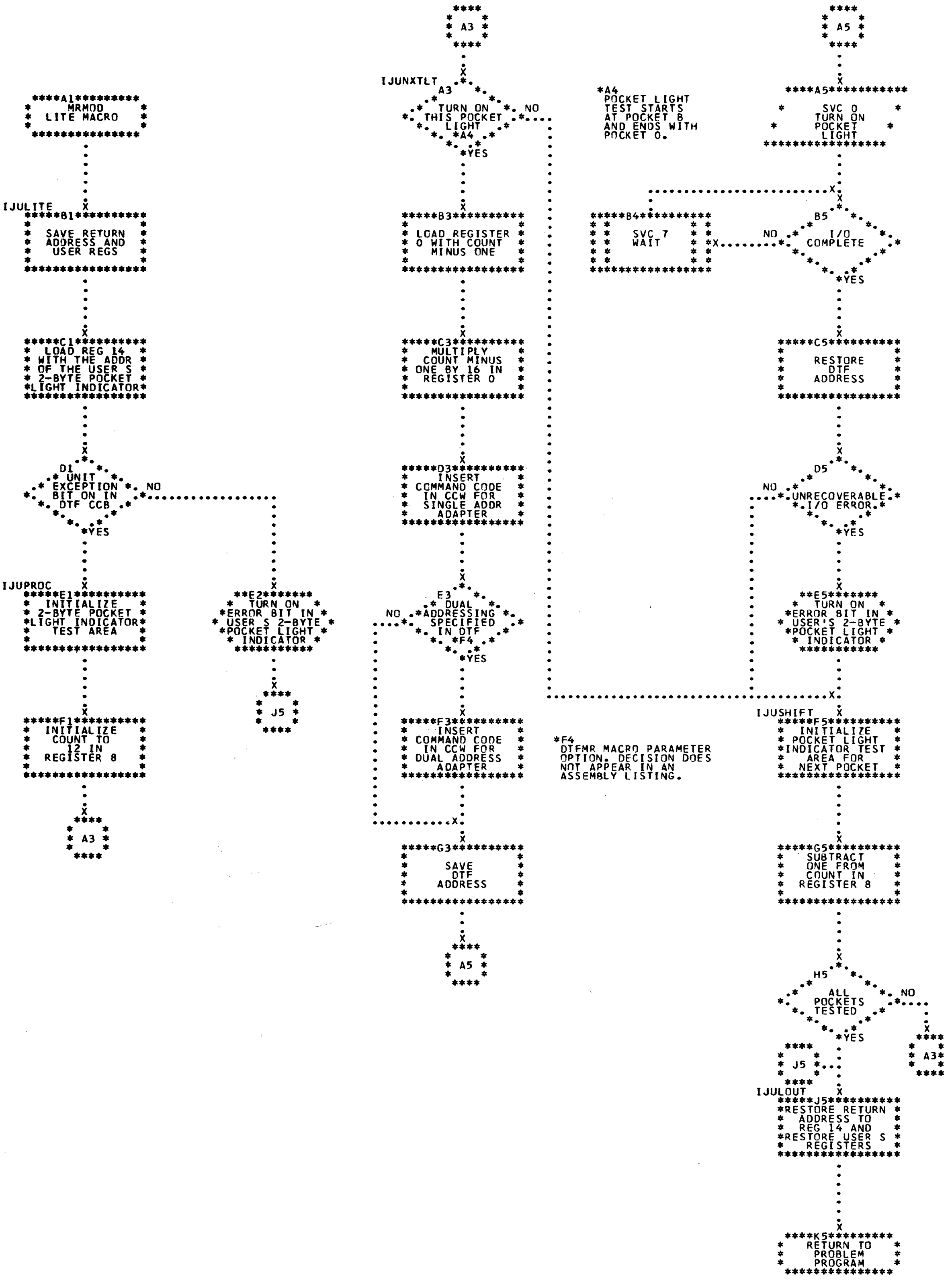

BA H5
IJUTSTRF
A1
IS CCB DEQUEUED
NO
YES
A2
READ SWITCH ON IN DTF
NO
YES
IJUSVC1
A3
SVC 7 WAIT
BA G5
IJUCALLW
*A4
MRMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
A5
MRMOD UPDATE ADDRESS SUBROUTINE
IJUENDTS
B1
UNRECOVERABLE I/O ERROR
YES
NO
D3
B2
TURN ON WAIT SWITCH IN DTF AND SET WAIT BIT IN BUFFER
IJUINCR
B5
ADD BUFF LENGTH TO ADDR OF LAST BYTE OF LAST BUFFER LOADED BY SUPERVISOR
C1
INTERVENTION REQUIRED
YES
NO
C2
POST AND RESET APPROPRIATE BITS
C5
INITIALIZE TO LOAD FIRST BUFF IN DOC BUFFER AREA NEXT IF REQUIRED
D3
IJUTDSN
D1
DUAL ADDRESSING *A4
YES
NO
IJUCKMES
D3
DATA CHECK OR DISENGAGE
YES
NO
D5
RETURN
E1
DISENGAGE BIT ON IN CCB
YES
NO
E3
MOVE MESSAGE CODE 33 (4MR2I) TO MESSAGE BYTE IN DTF
F1
DISENGAGE BIT ON IN DTF
YES
NO
F2
POST AND RESET APPROPRIATE BITS
F3
SCU NOT OPERATIONAL
YES
NO
G4
IJUTUE
G1
UNIT EXCEPTION
YES
NO
BA C5
G4
G2
CLEAR DOCUMENT BUFFER
BA D3
IJUGEXIT
G3
POST UNRECOV. I/O ERROR (4MR1I) IN BUFFER ERROR INFORMATION
IJUFETCH
G4
INITIALIZE TO FETCH MESSAGE WRITER $$BMMR20
H4
SVC 2 FETCH $$BMMR20

Chart EC. MRMOD: LITE Macro


A1
MRMOD LITE MACRO
IJULITE
B1
SAVE RETURN ADDRESS AND USER REGS
C1
LOAD REG 14 WITH THE ADDR OF THE USER S 2-BYTE POCKET LIGHT INDICATOR
D1
UNIT EXCEPTION BIT ON IN DTF CCB
NO
YES
IJUPROC
E1
INITIALIZE 2-BYTE POCKET LIGHT INDICATOR TEST AREA
F1
INITIALIZE COUNT TO 12 IN REGISTER 8
A3
E2
TURN ON ERROR BIT IN USER S 2-BYTE POCKET LIGHT INDICATOR
J5
A3
IJUNXTLT
A3
TURN ON THIS POCKET LIGHT
A4
NO
YES
B3
LOAD REGISTER 0 WITH COUNT MINUS ONE
C3
MULTIPLY COUNT MINUS ONE BY 16 IN REGISTER 0
D3
INSERT COMMAND CODE IN CCW FOR SINGLE ADDR ADAPTER
E3
DUAL ADDRESSING SPECIFIED IN DTF
F4
NO
YES
F3
INSERT COMMAND CODE IN CCW FOR DUAL ADDRESS ADAPTER
G3
SAVE DTF ADDRESS
A5
A5
A5
SVC 0 TURN ON POCKET LIGHT
B4
SVC 7 WAIT
B5
I/O COMPLETE
NO
YES
C5
RESTORE DTF ADDRESS
D5
UNRECOVERABLE I/O ERROR
NO
YES
E5
TURN ON ERROR BIT IN USER'S 2-BYTE POCKET LIGHT INDICATOR
IJUSHIFT
F5
INITIALIZE POCKET LIGHT INDICATOR TEST AREA FOR NEXT POCKET
G5
SUBTRACT ONE FROM COUNT IN REGISTER 8
H5
ALL POCKETS TESTED
NO
YES
A3
J5
IJULOUT
J5
RESTORE RETURN ADDRESS TO REG 14 AND RESTORE USER S REGISTERS
K5
RETURN TO PROBLEM PROGRAM

*A4
POCKET LIGHT
TEST STARTS
AT POCKET 8
AND ENDS WITH
POCKET 0.

*F4
DTFMR MACRO PARAMETER
OPTION. DECISION DOES
NOT APPEAR IN AN
ASSEMBLY LISTING.

Chart BD. MRMOD: DISEN and WAITF Macros

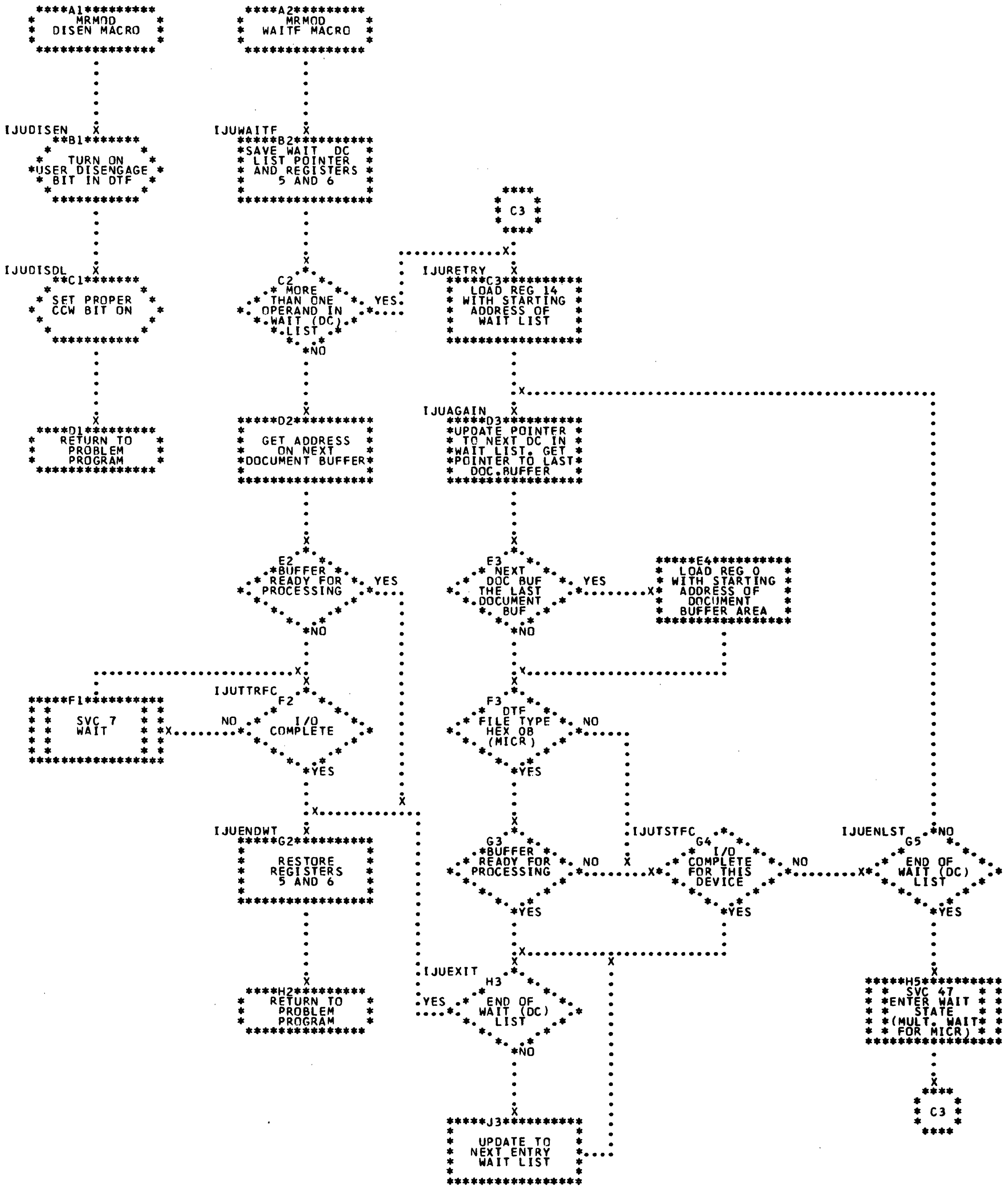


Chart BE. $$BOMR01 and $$BCMR01: Open and Close MICR

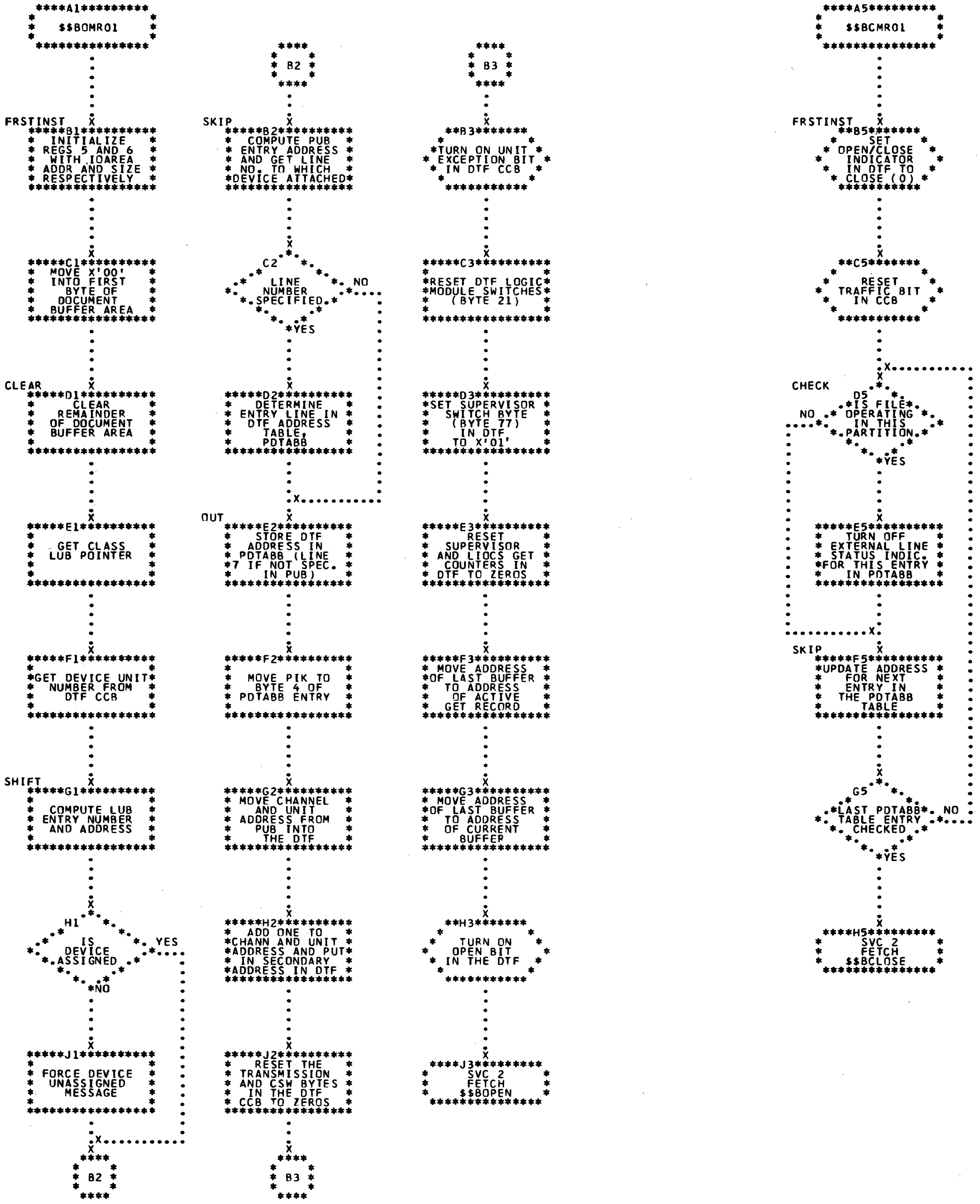

A1
$$BOMR01
FRSTINST
B1
INITIALIZE REGS 5 AND 6 WITH IOAREA ADDR AND SIZE RESPECTIVELY
C1
MOVE X'00' INTO FIRST BYTE OF DOCUMENT BUFFER AREA
CLEAR
D1
CLEAR REMAINDER OF DOCUMENT BUFFER AREA
E1
GET CLASS LUB POINTER
F1
GET DEVICE UNIT NUMBER FROM DTF CCB
SHIFT
G1
COMPUTE LUB ENTRY NUMBER AND ADDRESS
H1
IS DEVICE ASSIGNED
YES
NO
J1
FORCE DEVICE UNASSIGNED MESSAGE
B2
B2
SKIP
B2
COMPUTE PUB ENTRY ADDRESS AND GET LINE NO. TO WHICH DEVICE ATTACHED
C2
LINE NUMBER SPECIFIED
NO
YES
D2
DETERMINE ENTRY LINE IN DTF ADDRESS TABLE, PDTABB
OUT
E2
STORE DTF ADDRESS IN PDTABB (LINE 7 IF NOT SPEC. IN PUB)
F2
MOVE PIK TO BYTE 4 OF PDTABB ENTRY
G2
MOVE CHANNEL AND UNIT ADDRESS FROM PUB INTO THE DTF
H2
ADD ONE TO CHANN AND UNIT ADDRESS AND PUT IN SECONDARY ADDRESS IN DTF
J2
RESET THE TRANSMISSION AND CSW BYTES IN THE DTF CCB TO ZEROS
B3
B3
B3
TURN ON UNIT EXCEPTION BIT IN DTF CCB
C3
RESET DTF LOGIC MODULE SWITCHES (BYTE 21)
D3
SET SUPERVISOR SWITCH BYTE (BYTE 77) IN DTF TO X'01'
E3
RESET SUPERVISOR AND LIOCS GET COUNTERS IN DTF TO ZEROS
F3
MOVE ADDRESS OF LAST BUFFER TO ADDRESS OF ACTIVE GET RECORD
G3
MOVE ADDRESS OF LAST BUFFER TO ADDRESS OF CURRENT BUFFER
H3
TURN ON OPEN BIT IN THE DTF
J3
SVC 2 FETCH $$BOPEN
A5
$$BCMR01
FRSTINST
B5
SET OPEN/CLOSE INDICATOR IN DTF TO CLOSE (0)
C5
RESET TRAFFIC BIT IN CCB
CHECK
D5
IS FILE OPERATING IN THIS PARTITION
NO
YES
E5
TURN OFF EXTERNAL LINE STATUS INDIC. FOR THIS ENTRY IN PDTABB
SKIP
F5
UPDATE ADDRESS FOR NEXT ENTRY IN THE PDTABB TABLE
G5
LAST PDTABB TABLE ENTRY CHECKED
NO
YES
H5
SVC 2 FETCH $$BCLOSE

Chart BF. $$BMMR20: MICR Message Writer

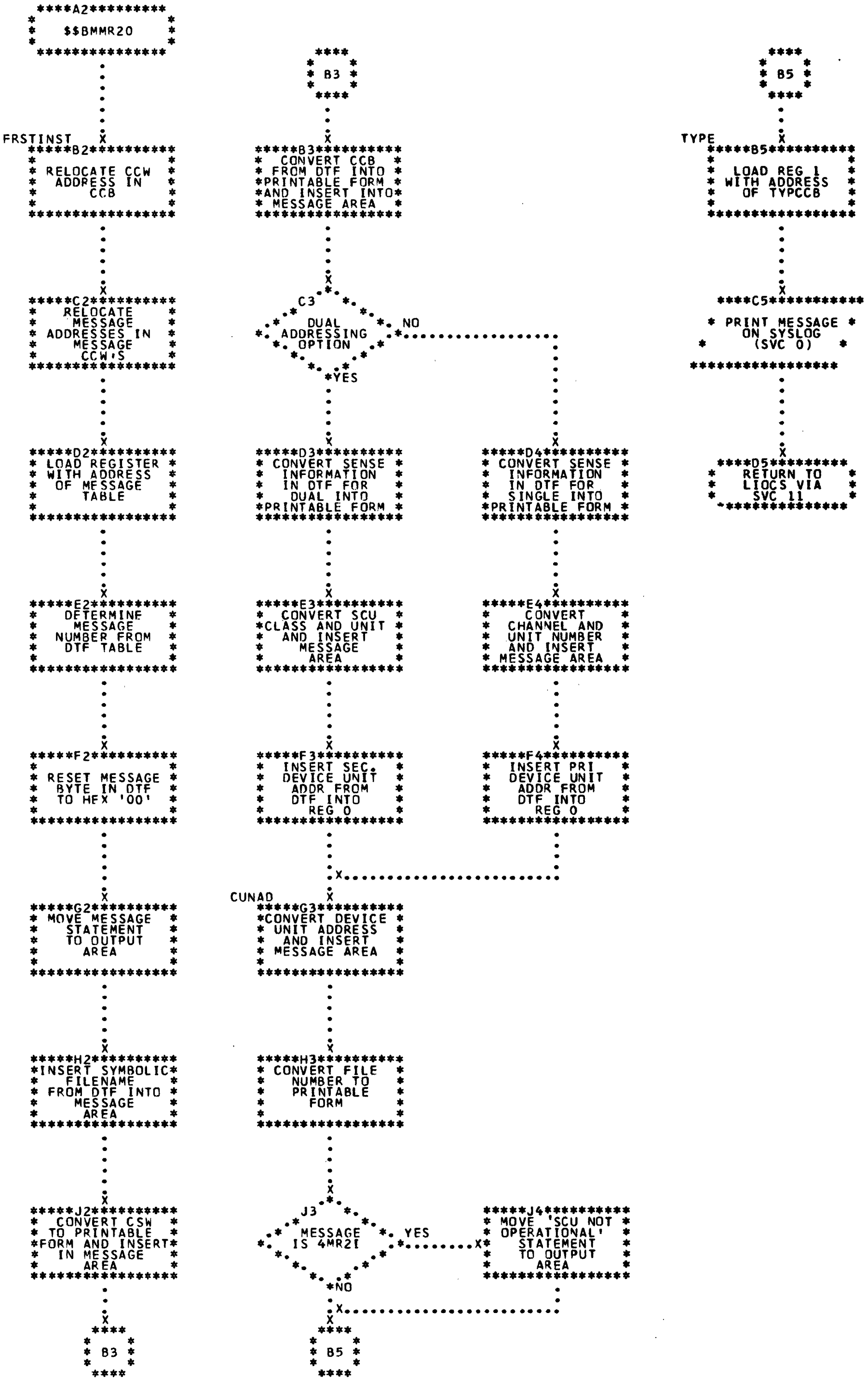

A2
$$BMMR20
B3
B5
FRSTINST
B2
RELOCATE CCW ADDRESS IN CCB
B3
CONVERT CCB FROM DTF INTO PRINTABLE FORM AND INSERT INTO MESSAGE AREA
TYPE
B5
LOAD REG 1 WITH ADDRESS OF TYPCCB
C2
RELOCATE MESSAGE ADDRESSES IN MESSAGE CCW'S
C3
DUAL ADDRESSING OPTION
NO
YES
C5
PRINT MESSAGE ON SYSLOG (SVC 0)
D2
LOAD REGISTER WITH ADDRESS OF MESSAGE TABLE
D3
CONVERT SENSE INFORMATION IN DTF FOR DUAL INTO PRINTABLE FORM
D4
CONVERT SENSE INFORMATION IN DTF FOR SINGLE INTO PRINTABLE FORM
D5
RETURN TO LIOCS VIA SVC 11
E2
DETERMINE MESSAGE NUMBER FROM DTF TABLE
E3
CONVERT SCU CLASS AND UNIT AND INSERT MESSAGE AREA
E4
CONVERT CHANNEL AND UNIT NUMBER AND INSERT MESSAGE AREA
F2
RESET MESSAGE BYTE IN DTF TO HEX '00'
F3
INSERT SEC. DEVICE UNIT ADDR FROM DTF INTO REG 0
F4
INSERT PRI DEVICE UNIT ADDR FROM DTF INTO REG 0
G2
MOVE MESSAGE STATEMENT TO OUTPUT AREA
CUNAD
G3
CONVERT DEVICE UNIT ADDRESS AND INSERT MESSAGE AREA
H2
INSERT SYMBOLIC FILENAME FROM DTF INTO MESSAGE AREA
H3
CONVERT FILE NUMBER TO PRINTABLE FORM
J2
CONVERT CSW TO PRINTABLE FORM AND INSERT IN MESSAGE AREA
J3
MESSAGE IS 4MR2I
YES
NO
J4
MOVE 'SCU NOT OPERATIONAL' STATEMENT TO OUTPUT AREA
B3
B5

Chart BG. ORMOD: CNTRL Macro (Part 1 of 2)

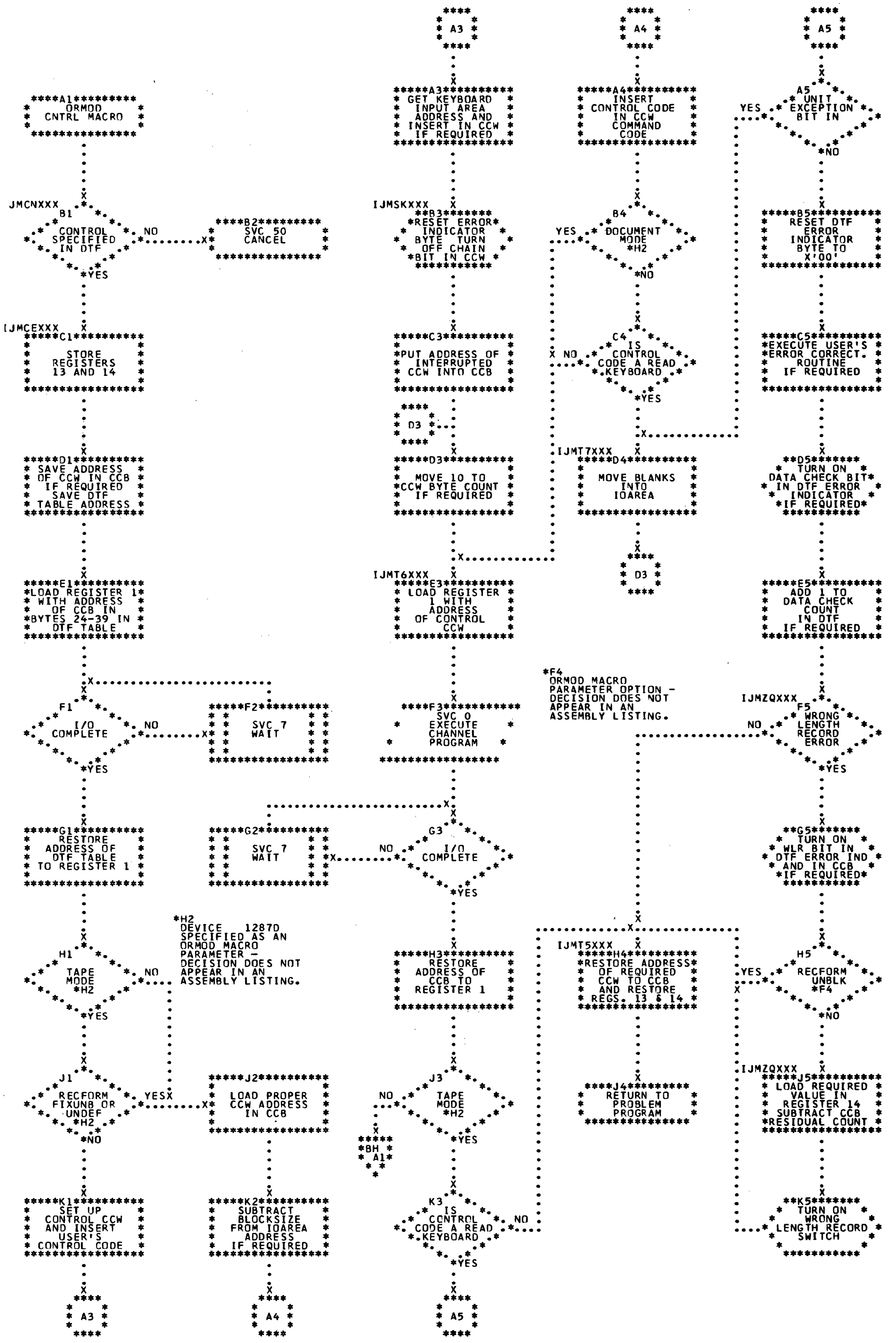

A1 ORMOD CNTRL MACRO
IJMCNXXX
B1 CONTROL SPECIFIED IN DTF
NO
YES
B2 SVC 50 CANCEL
IJMCEXXX
C1 STORE REGISTERS 13 AND 14
D1 SAVE ADDRESS OF CCW IN CCB IF REQUIRED SAVE DTF TABLE ADDRESS
E1 LOAD REGISTER 1 WITH ADDRESS OF CCB IN BYTES 24-39 IN DTF TABLE
F1 I/O COMPLETE
F2 SVC 7 WAIT
G1 RESTORE ADDRESS OF DTF TABLE TO REGISTER 1
H1 TAPE MODE *H2
*H2 DEVICE 1287D SPECIFIED AS AN ORMOD MACRO PARAMETER - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
J1 RECFORM FIXUNB OR UNDEF *H2
J2 LOAD PROPER CCW ADDRESS IN CCB
K1 SET UP CONTROL CCW AND INSERT USER'S CONTROL CODE
K2 SUBTRACT BLOCKSIZE FROM IOAREA ADDRESS IF REQUIRED
A3
A3 GET KEYBOARD INPUT AREA ADDRESS AND INSERT IN CCW IF REQUIRED
IJMSKXXX
B3 RESET ERROR INDICATOR BYTE TURN OFF CHAIN BIT IN CCW
C3 PUT ADDRESS OF INTERRUPTED CCW INTO CCB
D3
D3 MOVE 10 TO CCW BYTE COUNT IF REQUIRED
IJMT6XXX
E3 LOAD REGISTER 1 WITH ADDRESS OF CONTROL CCW
F3 SVC 0 EXECUTE CHANNEL PROGRAM
G2 SVC 7 WAIT
G3 I/O COMPLETE
H3 RESTORE ADDRESS OF CCB TO REGISTER 1
J3 TAPE MODE *H2
BH A1
K3 IS CONTROL CODE A READ KEYBOARD
A4
A4 INSERT CONTROL CODE IN CCW COMMAND CODE
B4 DOCUMENT MODE *H2
C4 IS CONTROL CODE A READ KEYBOARD
IJMT7XXX
D4 MOVE BLANKS INTO IOAREA
D3
*F4 ORMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
IJMT5XXX
H4 RESTORE ADDRESS OF REQUIRED CCW TO CCB AND RESTORE REGS. 13 & 14
J4 RETURN TO PROBLEM PROGRAM
A5
A5 UNIT EXCEPTION BIT IN
B5 RESET DTF ERROR INDICATOR BYTE TO X'00'
C5 EXECUTE USER'S ERROR CORRECT. ROUTINE IF REQUIRED
D5 TURN ON DATA CHECK BIT IN DTF ERROR INDICATOR IF REQUIRED
E5 ADD 1 TO DATA CHECK COUNT IN DTF IF REQUIRED
IJMZQXXX
F5 WRONG LENGTH RECORD ERROR
G5 TURN ON WLR BIT IN DTF ERROR IND AND IN CCB IF REQUIRED
H5 RECFORM UNBLK *F4
IJMZQXXX
J5 LOAD REQUIRED VALUE IN REGISTER 14 SUBTRACT CCB RESIDUAL COUNT
K5 TURN ON WRONG LENGTH RECORD SWITCH
A3
A4
A5

Chart BH. ORMOD: CNTRL Macro (Part 2 of 2)

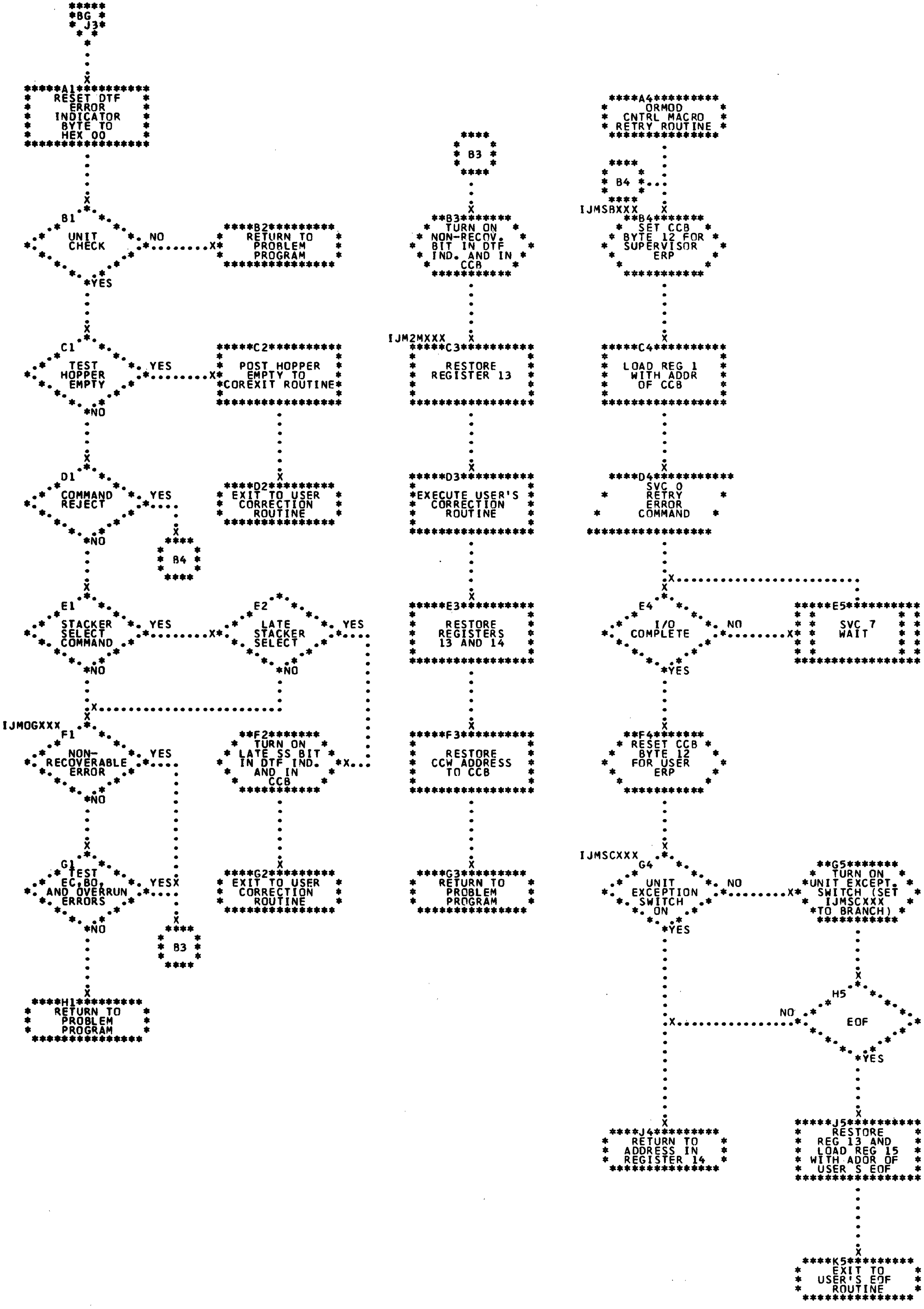

BG J3
A1 RESET DTF ERROR INDICATOR BYTE TO HEX 00
B1 UNIT CHECK
NO
YES
B2 RETURN TO PROBLEM PROGRAM
C1 TEST HOPPER EMPTY
YES
NO
C2 POST HOPPER EMPTY TO COREXIT ROUTINE
D1 COMMAND REJECT
YES
NO
B4
D2 EXIT TO USER CORRECTION ROUTINE
E1 STACKER SELECT COMMAND
YES
NO
E2 LATE STACKER SELECT
YES
NO
IJMOGXXX
F1 NON-RECOVERABLE ERROR
YES
NO
F2 TURN ON LATE SS BIT IN DTF IND. AND IN CCB
G1 TEST EC,BO, AND OVERRUN ERRORS
YES
NO
B3
G2 EXIT TO USER CORRECTION ROUTINE
H1 RETURN TO PROBLEM PROGRAM
B3
B3 TURN ON NON-RECOV. BIT IN DTF IND. AND IN CCB
IJM2MXXX
C3 RESTORE REGISTER 13
D3 EXECUTE USER'S CORRECTION ROUTINE
E3 RESTORE REGISTERS 13 AND 14
F3 RESTORE CCW ADDRESS TO CCB
G3 RETURN TO PROBLEM PROGRAM
A4 ORMOD CNTRL MACRO RETRY ROUTINE
B4
IJMSBXXX
B4 SET CCB BYTE 12 FOR SUPERVISOR ERP
C4 LOAD REG 1 WITH ADDR OF CCB
D4 SVC 0 RETRY ERROR COMMAND
E4 I/O COMPLETE
NO
YES
E5 SVC 7 WAIT
F4 RESET CCB BYTE 12 FOR USER ERP
IJMSCXXX
G4 UNIT EXCEPTION SWITCH ON
NO
YES
G5 TURN ON UNIT EXCEPT. SWITCH (SET IJMSCXXX TO BRANCH)
H5 EOF
NO
YES
J4 RETURN TO ADDRESS IN REGISTER 14
J5 RESTORE REG 13 AND LOAD REG 15 WITH ADDR OF USER S EOF
K5 EXIT TO USER'S EOF ROUTINE

Chart BJ. CRMOD: GET Macro, Unblocked Records (Part 1 of 3)

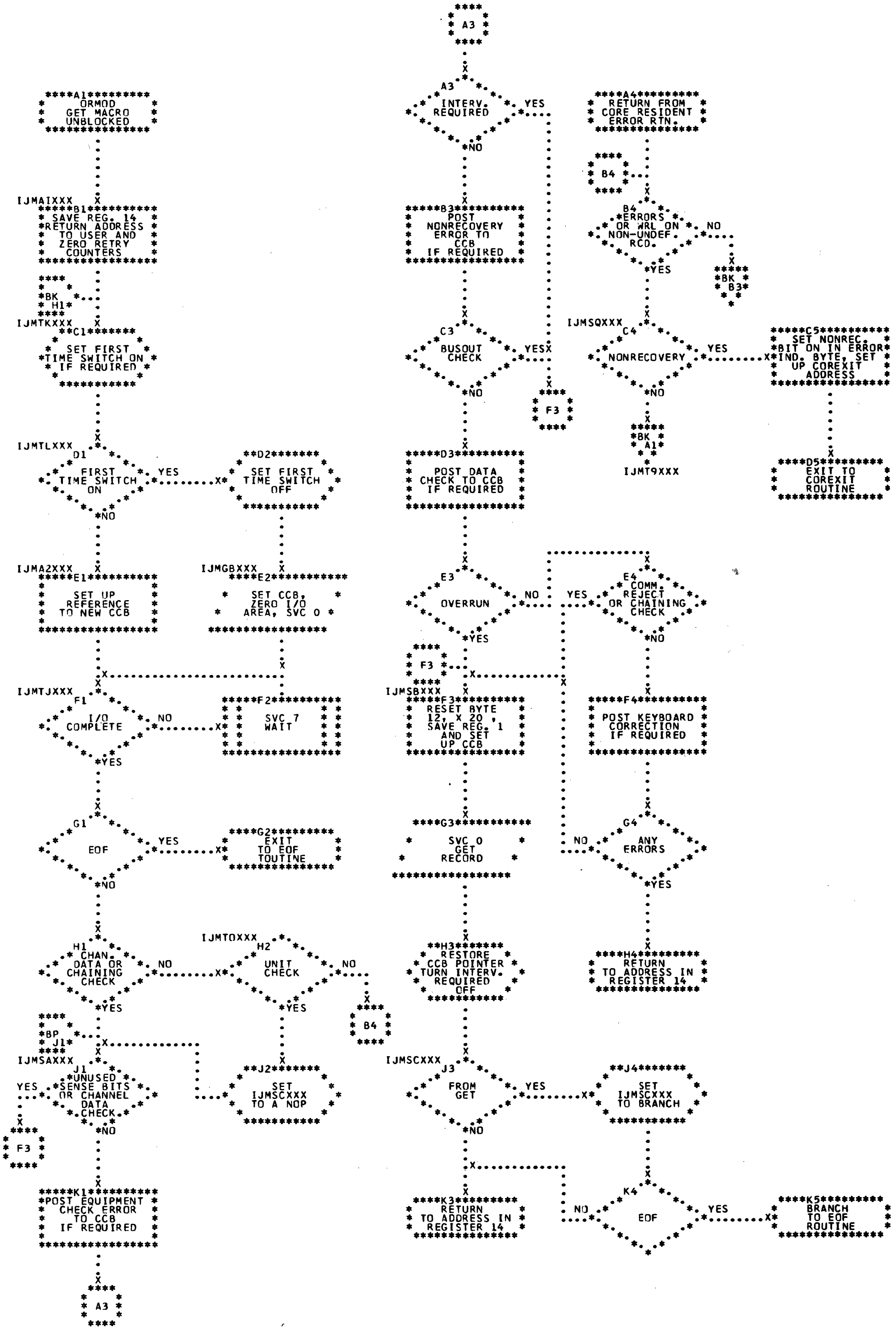


Chart BK. CRMCI: GET Macro, Unblocked Records (Part 2 of 3)

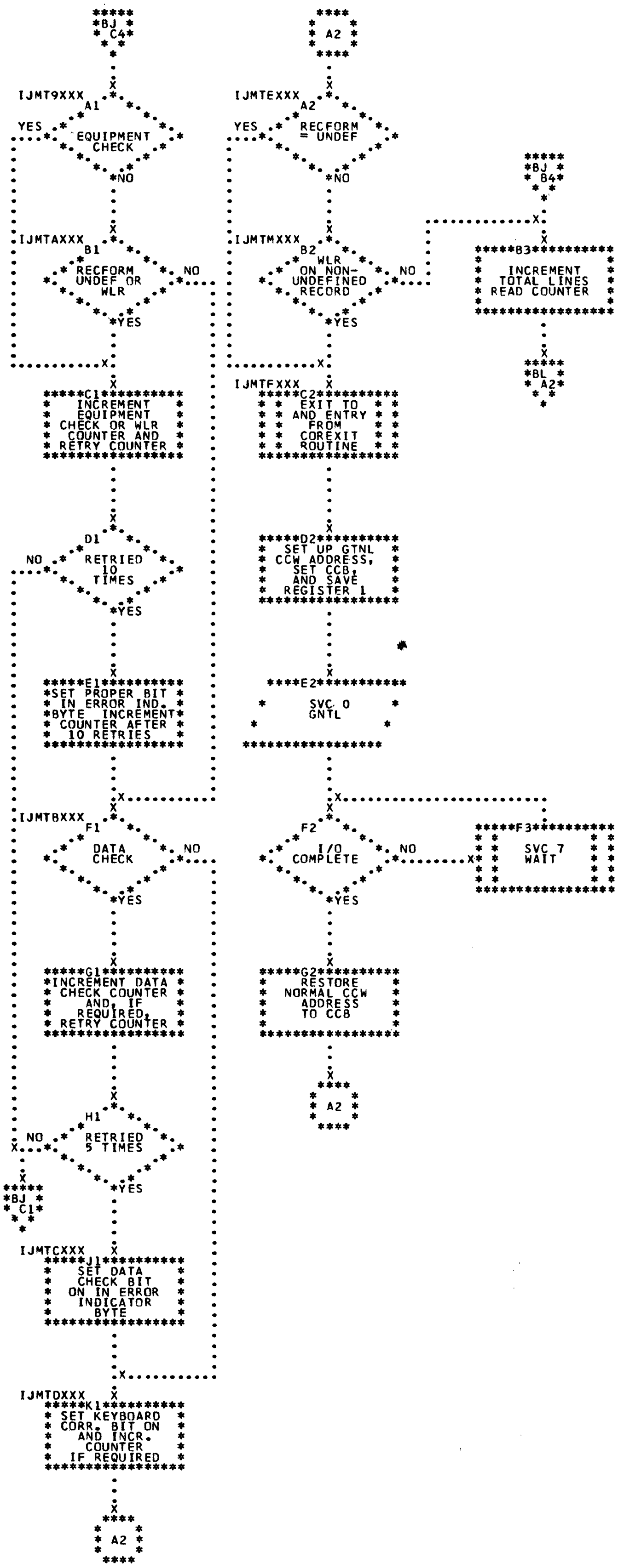

BJ C4
IJMT9XXX A1
EQUIPMENT CHECK
YES
NO
IJMTAXXX B1
RECFORM UNDEF OR WLR
NO
YES
C1
INCREMENT EQUIPMENT CHECK OR WLR COUNTER AND RETRY COUNTER
D1
RETRIED 10 TIMES
NO
YES
E1
SET PROPER BIT IN ERROR IND. BYTE INCREMENT COUNTER AFTER 10 RETRIES
IJMTBXXX F1
DATA CHECK
NO
YES
G1
INCREMENT DATA CHECK COUNTER AND, IF REQUIRED, RETRY COUNTER
H1
RETRIED 5 TIMES
NO
YES
BJ C1
IJMTCXXX J1
SET DATA CHECK BIT ON IN ERROR INDICATOR BYTE
IJMTDXXX K1
SET KEYBOARD CORR. BIT ON AND INCR. COUNTER IF REQUIRED
A2
A2
IJMTEXXX A2
RECFORM = UNDEF
YES
NO
IJMTMXXX B2
WLR ON NON-UNDEFINED RECORD
NO
YES
IJMTFXXX C2
EXIT TO AND ENTRY FROM COREXIT ROUTINE
D2
SET UP GTNL CCW ADDRESS, SET CCB, AND SAVE REGISTER 1
E2
SVC 0 GNTL
F2
I/O COMPLETE
NO
YES
G2
RESTORE NORMAL CCW ADDRESS TO CCB
A2
BJ B4
B3
INCREMENT TOTAL LINES READ COUNTER
BL A2
F3
SVC 7 WAIT

Chart BI. CFMOD: GET Macro, Unblocked Records (Part 3 of 3)

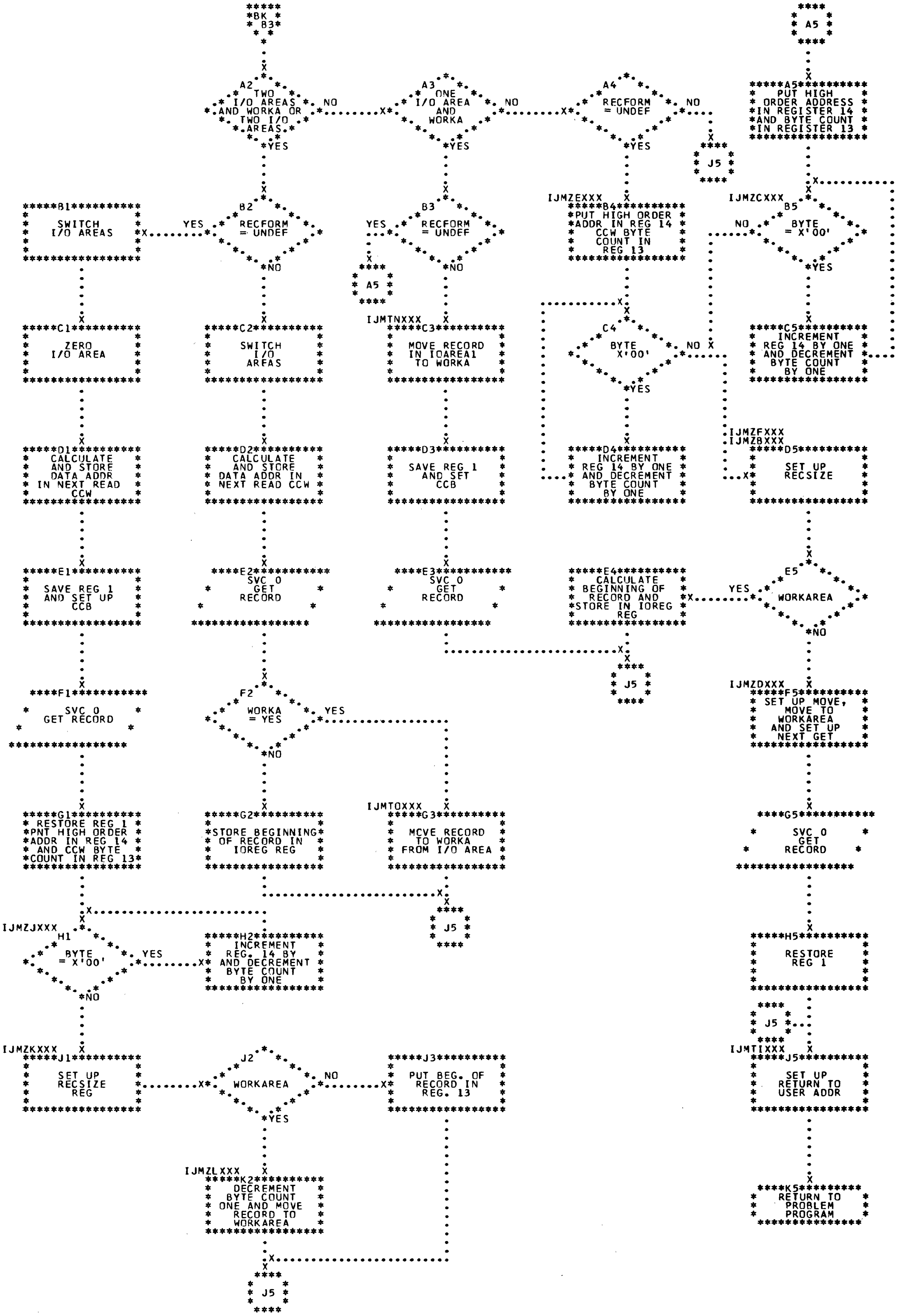

BK B3
A5
A2 TWO I/O AREAS AND WORKA OR TWO I/O AREAS
NO
YES
A3 ONE I/O AREA AND WORKA
NO
YES
A4 RECFORM = UNDEF
NO
YES
J5
A5 PUT HIGH ORDER ADDRESS IN REGISTER 14 AND BYTE COUNT IN REGISTER 13
B1 SWITCH I/O AREAS
YES
B2 RECFORM = UNDEF
NO
YES
B3 RECFORM = UNDEF
NO
A5
IJMZEXXX
B4 PUT HIGH ORDER ADDR IN REG 14 CCW BYTE COUNT IN REG 13
IJMZCXXX
NO
B5 BYTE = X'00'
YES
C1 ZERO I/O AREA
C2 SWITCH I/O AREAS
IJMTNXXX
C3 MOVE RECORD IN IOAREA1 TO WORKA
C4 BYTE X'00'
NO
YES
C5 INCREMENT REG 14 BY ONE AND DECREMENT BYTE COUNT BY ONE
D1 CALCULATE AND STORE DATA ADDR IN NEXT READ CCW
D2 CALCULATE AND STORE DATA ADDR IN NEXT READ CCW
D3 SAVE REG 1 AND SET CCB
D4 INCREMENT REG 14 BY ONE AND DECREMENT BYTE COUNT BY ONE
IJMZFXXX
IJMZBXXX
D5 SET UP RECSIZE
E1 SAVE REG 1 AND SET UP CCB
E2 SVC 0 GET RECORD
E3 SVC 0 GET RECORD
E4 CALCULATE BEGINNING OF RECORD AND STORE IN IOREG REG
YES
E5 WORKAREA
NO
J5
F1 SVC 0 GET RECORD
F2 WORKA = YES
YES
NO
IJMZDXXX
F5 SET UP MOVE, MOVE TO WORKAREA AND SET UP NEXT GET
G1 RESTORE REG 1 PNT HIGH ORDER ADDR IN REG 14 AND CCW BYTE COUNT IN REG 13
G2 STORE BEGINNING OF RECORD IN IOREG REG
IJMTOXXX
G3 MOVE RECORD TO WORKA FROM I/O AREA
G5 SVC 0 GET RECORD
J5
IJMZJXXX
H1 BYTE = X'00'
YES
NO
H2 INCREMENT REG. 14 BY AND DECREMENT BYTE COUNT BY ONE
H5 RESTORE REG 1
J5
IJMZKXXX
J1 SET UP RECSIZE REG
J2 WORKAREA
NO
YES
J3 PUT BEG. OF RECORD IN REG. 13
IJMTIXXX
J5 SET UP RETURN TO USER ADDR
IJMZLXXX
K2 DECREMENT BYTE COUNT ONE AND MOVE RECORD TO WORKAREA
K5 RETURN TO PROBLEM PROGRAM
J5

Chart BM. CFMOD: GET Macro, Blocked Records (Part 1 of 2)

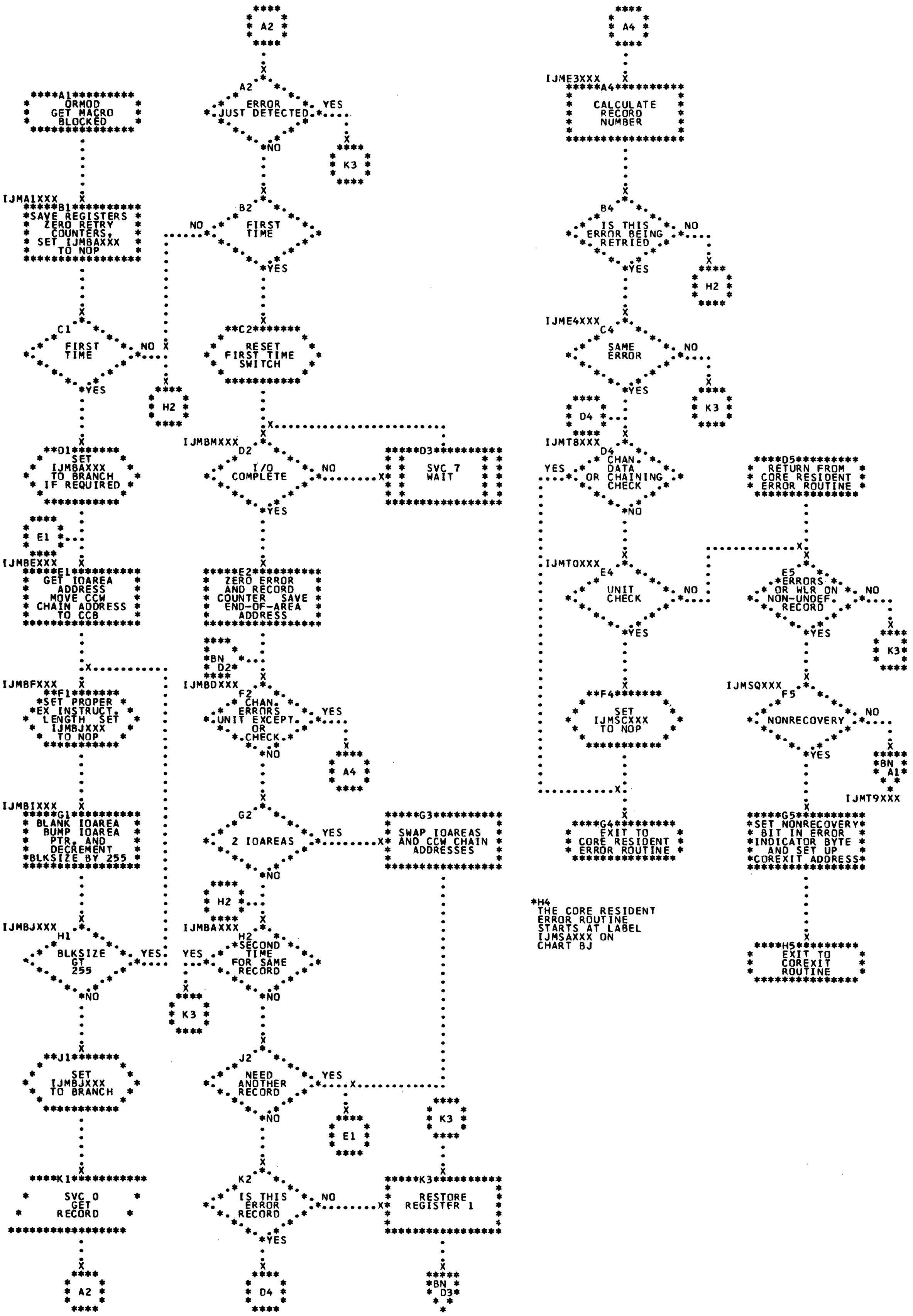


Chart BN. CFMCI: GET Macro, Blocked Records (Part 2 of 2)

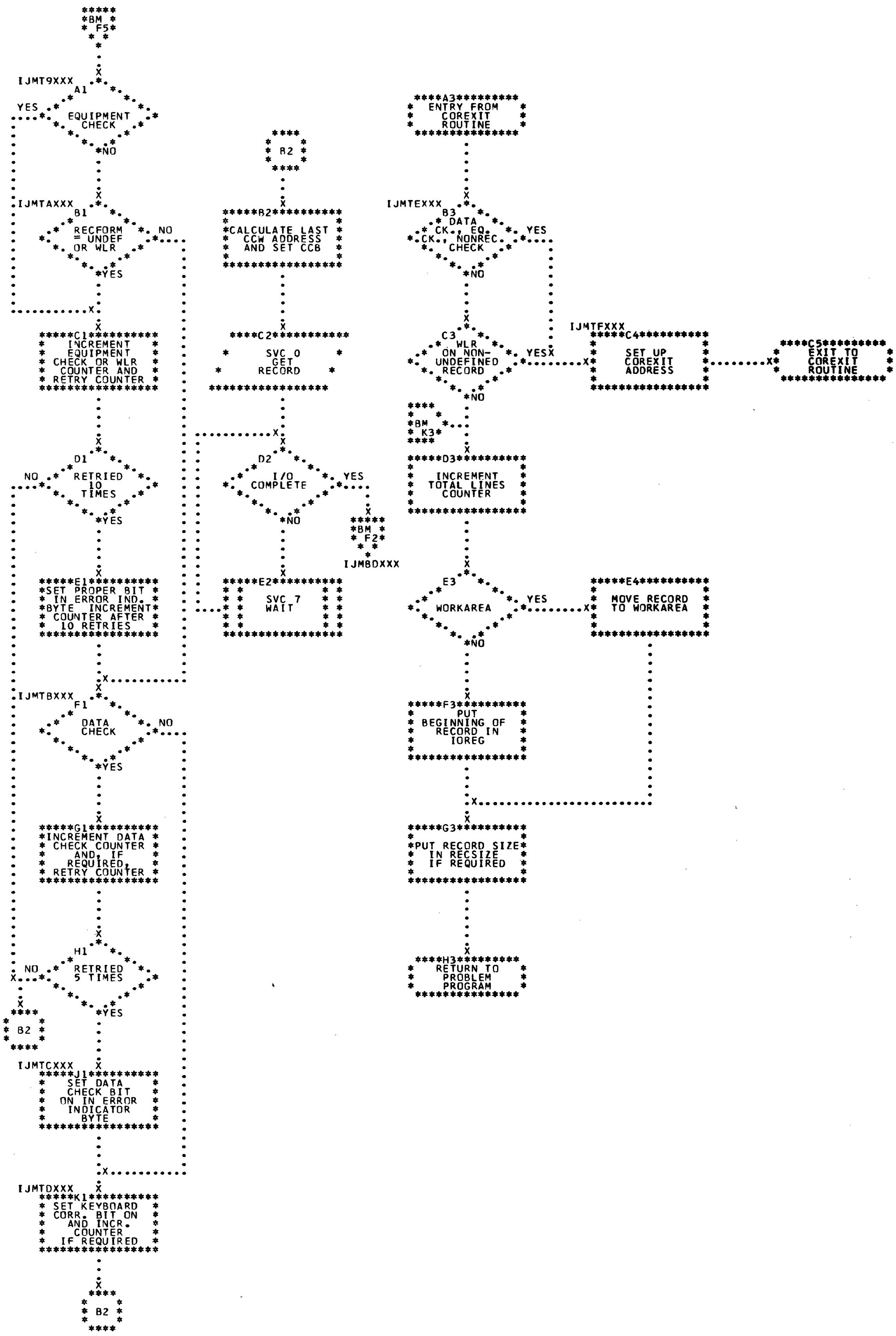

BM F5
IJMT9XXX
A1
EQUIPMENT CHECK
YES
NO
IJMTAXXX
B1
RECFORM = UNDEF OR WLR
NO
YES
C1
INCREMENT EQUIPMENT CHECK OR WLR COUNTER AND RETRY COUNTER
D1
RETRIED 10 TIMES
NO
YES
E1
SET PROPER BIT IN ERROR IND. BYTE INCREMENT COUNTER AFTER 10 RETRIES
IJMTBXXX
F1
DATA CHECK
NO
YES
G1
INCREMENT DATA CHECK COUNTER AND, IF REQUIRED, RETRY COUNTER
H1
RETRIED 5 TIMES
NO
YES
B2
IJMTCXXX
J1
SET DATA CHECK BIT ON IN ERROR INDICATOR BYTE
IJMTDXXX
K1
SET KEYBOARD CORR. BIT ON AND INCR. COUNTER IF REQUIRED
B2
B2
B2
CALCULATE LAST CCW ADDRESS AND SET CCB
C2
SVC 0 GET RECORD
D2
I/O COMPLETE
YES
NO
BM F2
IJMBDXXX
E2
SVC 7 WAIT
A3
ENTRY FROM COREXIT ROUTINE
IJMTEXXX
B3
DATA CK. EQ. CK., NONREC. CHECK
YES
NO
C3
WLR ON NON-UNDEFINED RECORD
YES
NO
IJMTFXXX
C4
SET UP COREXIT ADDRESS
C5
EXIT TO COREXIT ROUTINE
BM K3
D3
INCREMENT TOTAL LINES COUNTER
E3
WORKAREA
YES
NO
E4
MOVE RECORD TO WORKAREA
F3
PUT BEGINNING OF RECORD IN IOREG
G3
PUT RECORD SIZE IN RECSIZE IF REQUIRED
H3
RETURN TO PROBLEM PROGRAM

Chart BP. ORMOD: RDLINE Macro

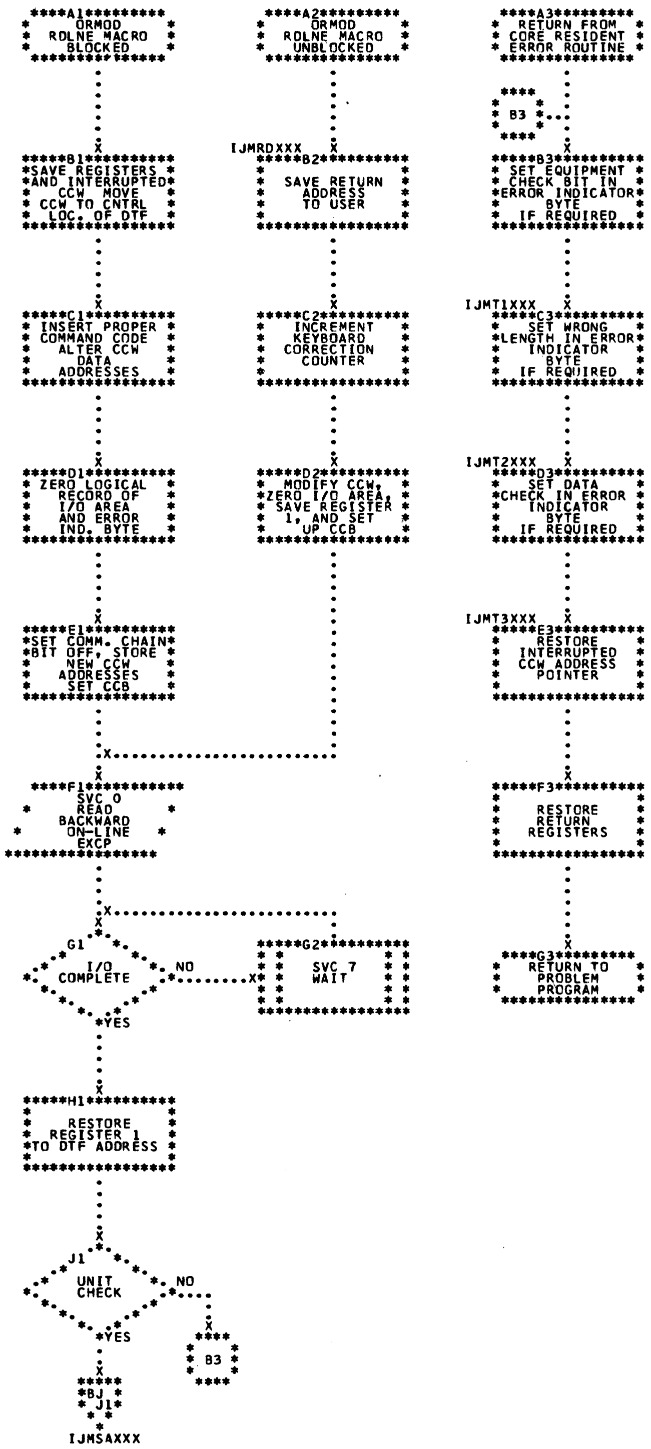

A1
ORMOD RDLNE MACRO BLOCKED
A2
ORMOD RDLNE MACRO UNBLOCKED
A3
RETURN FROM CORE RESIDENT ERROR ROUTINE
B3
B1
SAVE REGISTERS AND INTERRUPTED CCW MOVE CCW TO CNTRL LOC. OF DTF
IJMRDXXX
B2
SAVE RETURN ADDRESS TO USER
B3
SET EQUIPMENT CHECK BIT IN ERROR INDICATOR BYTE IF REQUIRED
C1
INSERT PROPER COMMAND CODE ALTER CCW DATA ADDRESSES
C2
INCREMENT KEYBOARD CORRECTION COUNTER
IJMT1XXX
C3
SET WRONG LENGTH IN ERROR INDICATOR BYTE IF REQUIRED
D1
ZERO LOGICAL RECORD OF I/O AREA AND ERROR IND. BYTE
D2
MODIFY CCW, ZERO I/O AREA, SAVE REGISTER 1, AND SET UP CCB
IJMT2XXX
D3
SET DATA CHECK IN ERROR INDICATOR BYTE IF REQUIRED
E1
SET COMM. CHAIN BIT OFF, STORE NEW CCW ADDRESSES SET CCB
IJMT3XXX
E3
RESTORE INTERRUPTED CCW ADDRESS POINTER
F1
SVC 0 READ BACKWARD ON-LINE EXCP
F3
RESTORE RETURN REGISTERS
G1
I/O COMPLETE
NO
YES
G2
SVC 7 WAIT
G3
RETURN TO PROBLEM PROGRAM
H1
RESTORE REGISTER 1 TO DTF ADDRESS
J1
UNIT CHECK
NO
YES
B3
BJ J1
IJMSAXXX

Chart BQ. ORMOD: DSPLY and READ Macros

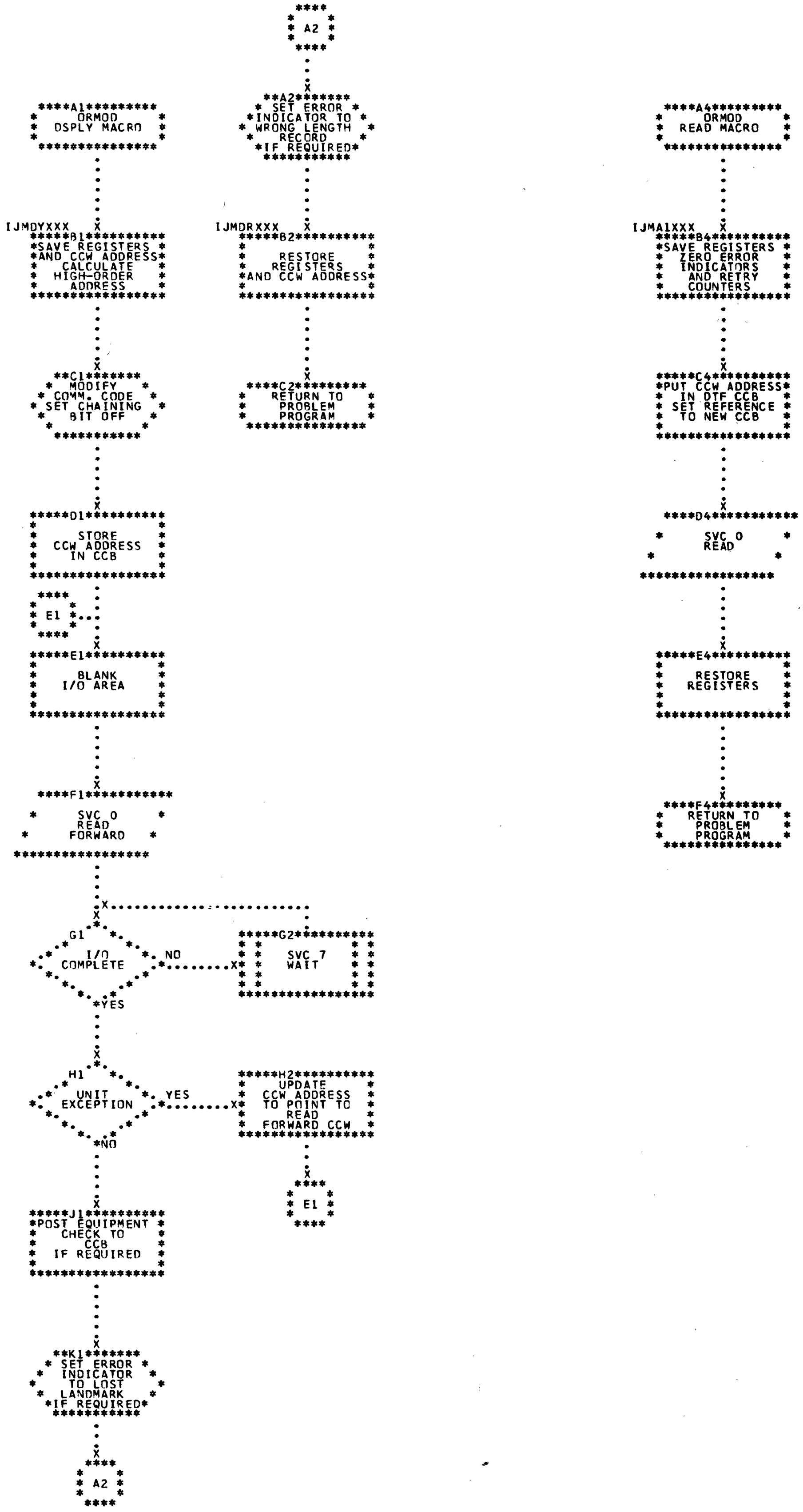


Chart BR. CRMOD: RESCN Macro

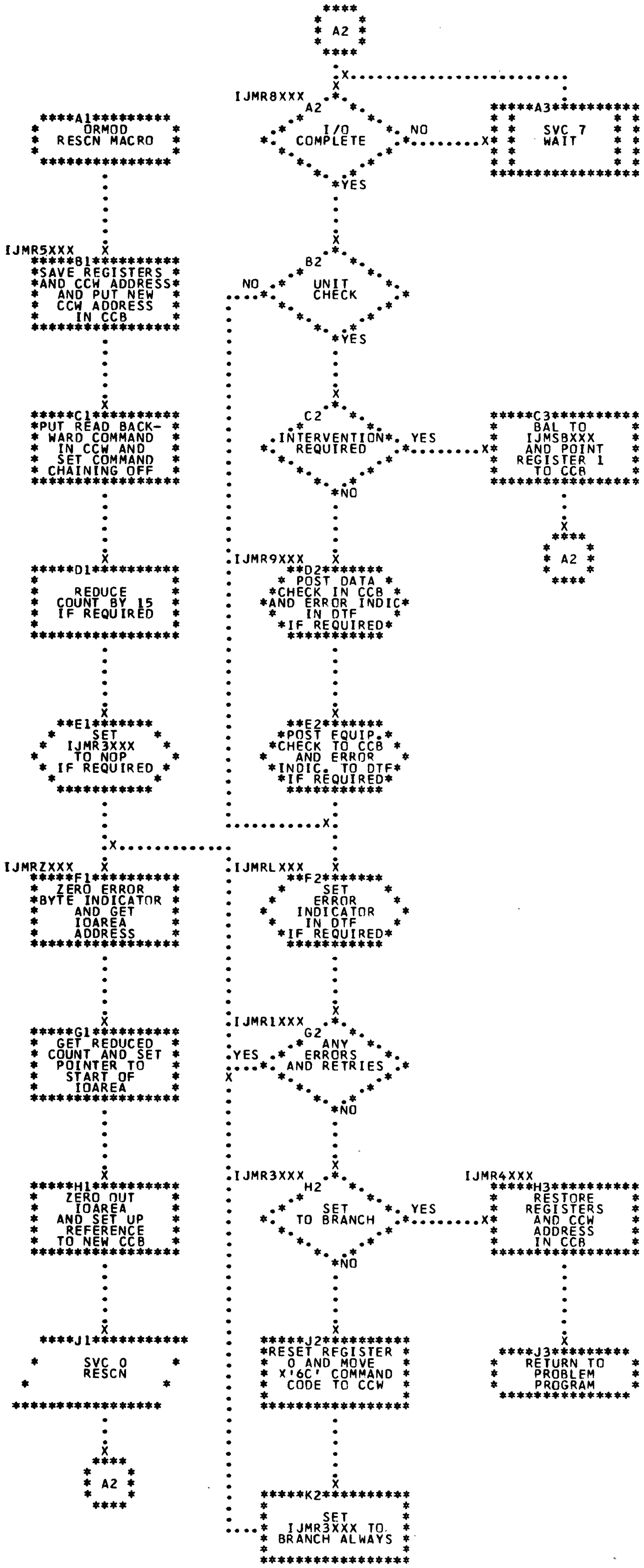

A2
IJMR8XXX
A2
I/O COMPLETE
NO
YES
A3
SVC 7 WAIT
A1
CRMOD RESCN MACRO
IJMR5XXX
B1
SAVE REGISTERS AND CCW ADDRESS AND PUT NEW CCW ADDRESS IN CCB
B2
UNIT CHECK
NO
YES
C1
PUT READ BACK-WARD COMMAND IN CCW AND SET COMMAND CHAINING OFF
C2
INTERVENTION REQUIRED
YES
NO
C3
BAL TO IJMSBXXX AND POINT REGISTER 1 TO CCB
A2
D1
REDUCE COUNT BY 15 IF REQUIRED
IJMR9XXX
D2
POST DATA CHECK IN CCB AND ERROR INDIC IN DTF IF REQUIRED
E1
SET IJMR3XXX TO NOP IF REQUIRED
E2
POST EQUIP. CHECK TO CCB AND ERROR INDIC. TO DTF IF REQUIRED
IJMRZXXX
F1
ZERO ERROR BYTE INDICATOR AND GET IOAREA ADDRESS
IJMRLXXX
F2
SET ERROR INDICATOR IN DTF IF REQUIRED
G1
GET REDUCED COUNT AND SET POINTER TO START OF IOAREA
IJMR1XXX
G2
ANY ERRORS AND RETRIES
YES
NO
H1
ZERO OUT IOAREA AND SET UP REFERENCE TO NEW CCB
IJMR3XXX
H2
SET TO BRANCH
YES
NO
IJMR4XXX
H3
RESTORE REGISTERS AND CCW ADDRESS IN CCB
J1
SVC 0 RESCN
A2
J2
RESET REGISTER 0 AND MOVE X'6C' COMMAND CODE TO CCW
J3
RETURN TO PROBLEM PROGRAM
K2
SET IJMR3XXX TO BRANCH ALWAYS

Chart BS. ORMOD: WAITF Macro (Part 1 of 3)

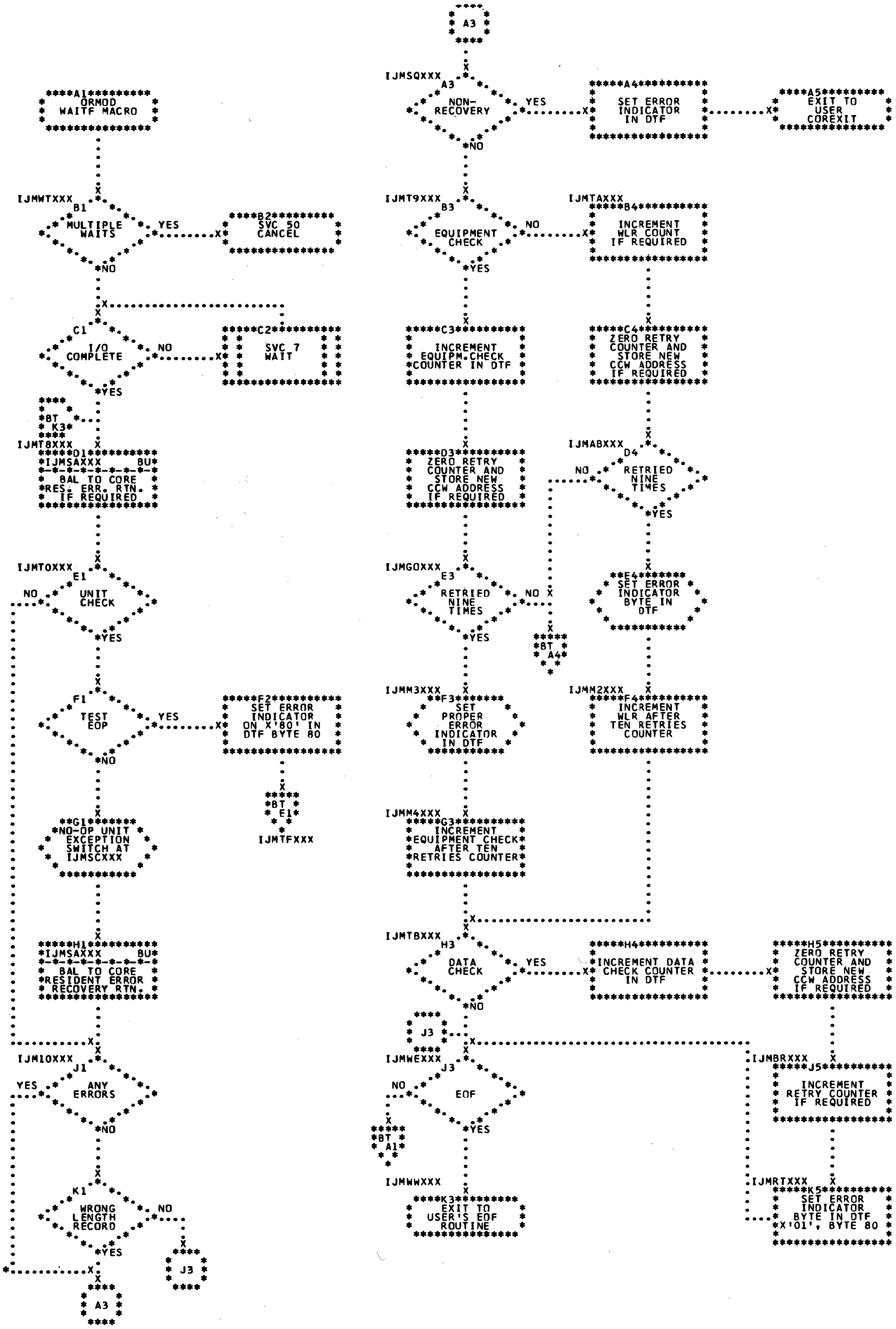

A3
A1
ORMOD WAITF MACRO
IJMWTXXX
B1
MULTIPLE WAITS
YES
NO
B2
SVC 50 CANCEL
C1
I/O COMPLETE
NO
YES
C2
SVC 7 WAIT
BT K3
IJMT8XXX
D1
IJMSAXXX BU
BAL TO CORE RES. ERR. RTN. IF REQUIRED
IJMTOXXX
E1
UNIT CHECK
NO
YES
F1
TEST EOP
YES
NO
F2
SET ERROR INDICATOR ON X'80' IN DTF BYTE 80
BT E1
IJMTFXXX
G1
NO-OP UNIT EXCEPTION SWITCH AT IJMSCXXX
H1
IJMSAXXX BU
BAL TO CORE RESIDENT ERROR RECOVERY RTN.
IJM10XXX
J1
ANY ERRORS
YES
NO
K1
WRONG LENGTH RECORD
NO
YES
J3
A3
A3
IJMSQXXX
A3
NON-RECOVERY
YES
NO
A4
SET ERROR INDICATOR IN DTF
A5
EXIT TO USER COREXIT
IJMT9XXX
B3
EQUIPMENT CHECK
NO
YES
IJMTAXXX
B4
INCREMENT WLR COUNT IF REQUIRED
C3
INCREMENT EQUIPM.CHECK COUNTER IN DTF
C4
ZERO RETRY COUNTER AND STORE NEW CCW ADDRESS IF REQUIRED
D3
ZERO RETRY COUNTER AND STORE NEW CCW ADDRESS IF REQUIRED
IJMABXXX
D4
RETRIED NINE TIMES
NO
YES
IJMGOXXX
E3
RETRIED NINE TIMES
NO
YES
BT A4
E4
SET ERROR INDICATOR BYTE IN DTF
IJMM3XXX
F3
SET PROPER ERROR INDICATOR IN DTF
IJMM2XXX
F4
INCREMENT WLR AFTER TEN RETRIES COUNTER
IJMM4XXX
G3
INCREMENT EQUIPMENT CHECK AFTER TEN RETRIES COUNTER
IJMTBXXX
H3
DATA CHECK
YES
NO
H4
INCREMENT DATA CHECK COUNTER IN DTF
H5
ZERO RETRY COUNTER AND STORE NEW CCW ADDRESS IF REQUIRED
J3
IJMWEXXX
J3
EOF
NO
YES
BT A1
IJMWWXXX
K3
EXIT TO USER'S EOF ROUTINE
IJMBRXXX
J5
INCREMENT RETRY COUNTER IF REQUIRED
IJMRTXXX
K5
SET ERROR INDICATOR BYTE IN DTF X'01', BYTE 80

Chart BT. CRMOD: WAITF Macro (Part 2 of 3)

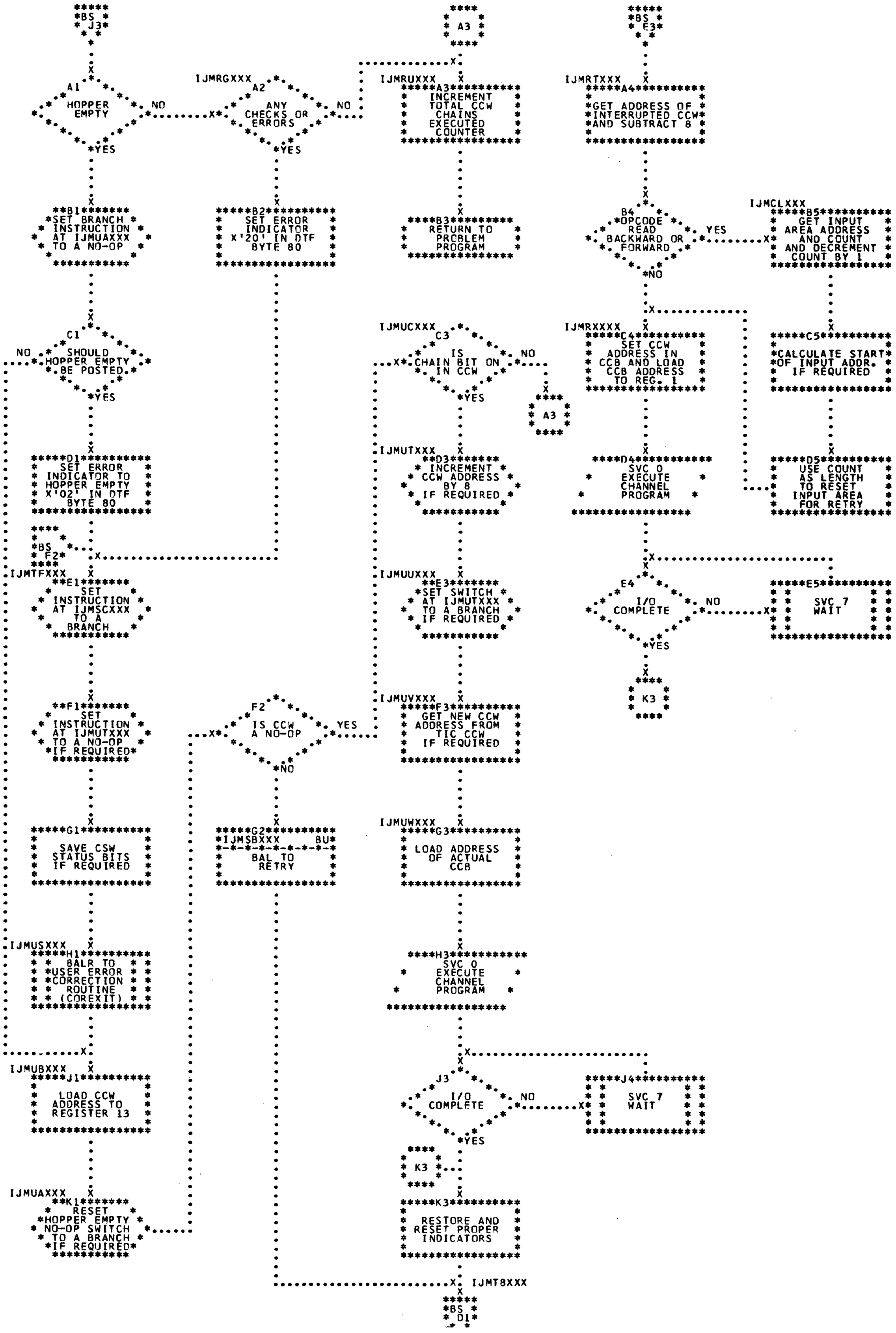

BS J3
A1 HOPPER EMPTY
NO
YES
IJMRGXXX
A2 ANY CHECKS OR ERRORS
NO
YES
A3
IJMRUXXX
A3 INCREMENT TOTAL CCW CHAINS EXECUTED COUNTER
BS E3
IJMRTXXX
A4 GET ADDRESS OF INTERRUPTED CCW AND SUBTRACT 8
B1 SET BRANCH INSTRUCTION AT IJMUAXXX TO A NO-OP
B2 SET ERROR INDICATOR X'20' IN DTF BYTE 80
B3 RETURN TO PROBLEM PROGRAM
B4 OPCODE READ BACKWARD OR FORWARD
YES
NO
IJMCLXXX
B5 GET INPUT AREA ADDRESS AND COUNT AND DECREMENT COUNT BY 1
C1 SHOULD HOPPER EMPTY BE POSTED
NO
YES
IJMUCXXX
C3 IS CHAIN BIT ON IN CCW
NO
YES
A3
IJMRXXXX
C4 SET CCW ADDRESS IN CCB AND LOAD CCB ADDRESS TO REG. 1
C5 CALCULATE START OF INPUT ADDR. IF REQUIRED
D1 SET ERROR INDICATOR TO HOPPER EMPTY X'02' IN DTF BYTE 80
IJMUTXXX
D3 INCREMENT CCW ADDRESS BY 8 IF REQUIRED
D4 SVC 0 EXECUTE CHANNEL PROGRAM
D5 USE COUNT AS LENGTH TO RESET INPUT AREA FOR RETRY
BS F2
IJMTFXXX
E1 SET INSTRUCTION AT IJMSCXXX TO A BRANCH
IJMUUXXX
E3 SET SWITCH AT IJMUTXXX TO A BRANCH IF REQUIRED
E4 I/O COMPLETE
NO
YES
E5 SVC 7 WAIT
K3
F1 SET INSTRUCTION AT IJMUTXXX TO A NO-OP IF REQUIRED
F2 IS CCW A NO-OP
YES
NO
IJMUVXXX
F3 GET NEW CCW ADDRESS FROM TIC CCW IF REQUIRED
G1 SAVE CSW STATUS BITS IF REQUIRED
G2 IJMSBXXX BU BAL TO RETRY
IJMUWXXX
G3 LOAD ADDRESS OF ACTUAL CCB
IJMUSXXX
H1 BALR TO USER ERROR CORRECTION ROUTINE (COREXIT)
H3 SVC 0 EXECUTE CHANNEL PROGRAM
IJMUBXXX
J1 LOAD CCW ADDRESS TO REGISTER 13
J3 I/O COMPLETE
NO
YES
J4 SVC 7 WAIT
K3
IJMUAXXX
K1 RESET HOPPER EMPTY NO-OP SWITCH TO A BRANCH IF REQUIRED
K3 RESTORE AND RESET PROPER INDICATORS
IJMT8XXX
BS D1

Chart EU. CFMCD: WAITF Macro (Part 3 of 3)

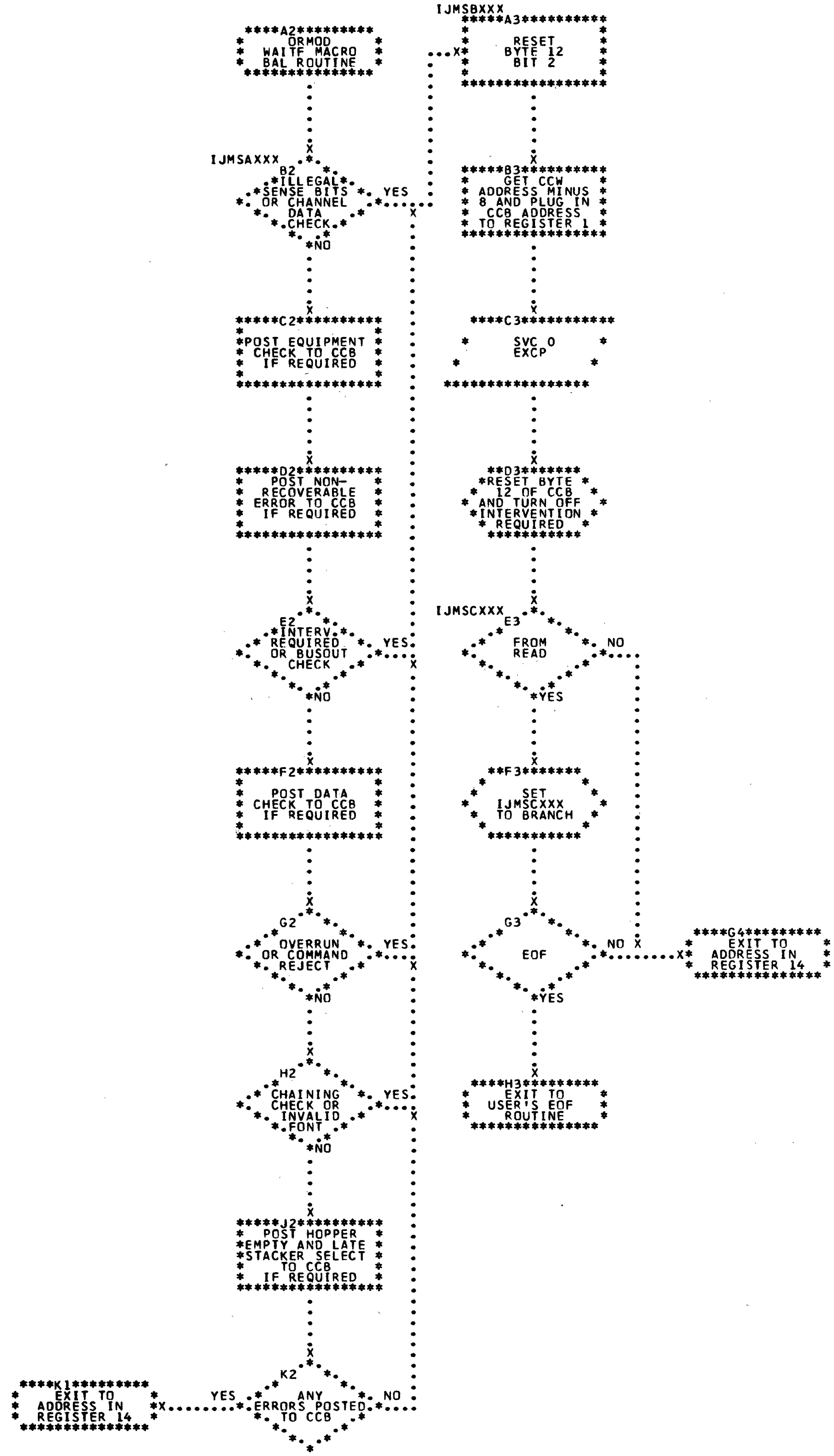


Chart CA. DRMOD: CNTRL and READ Macros

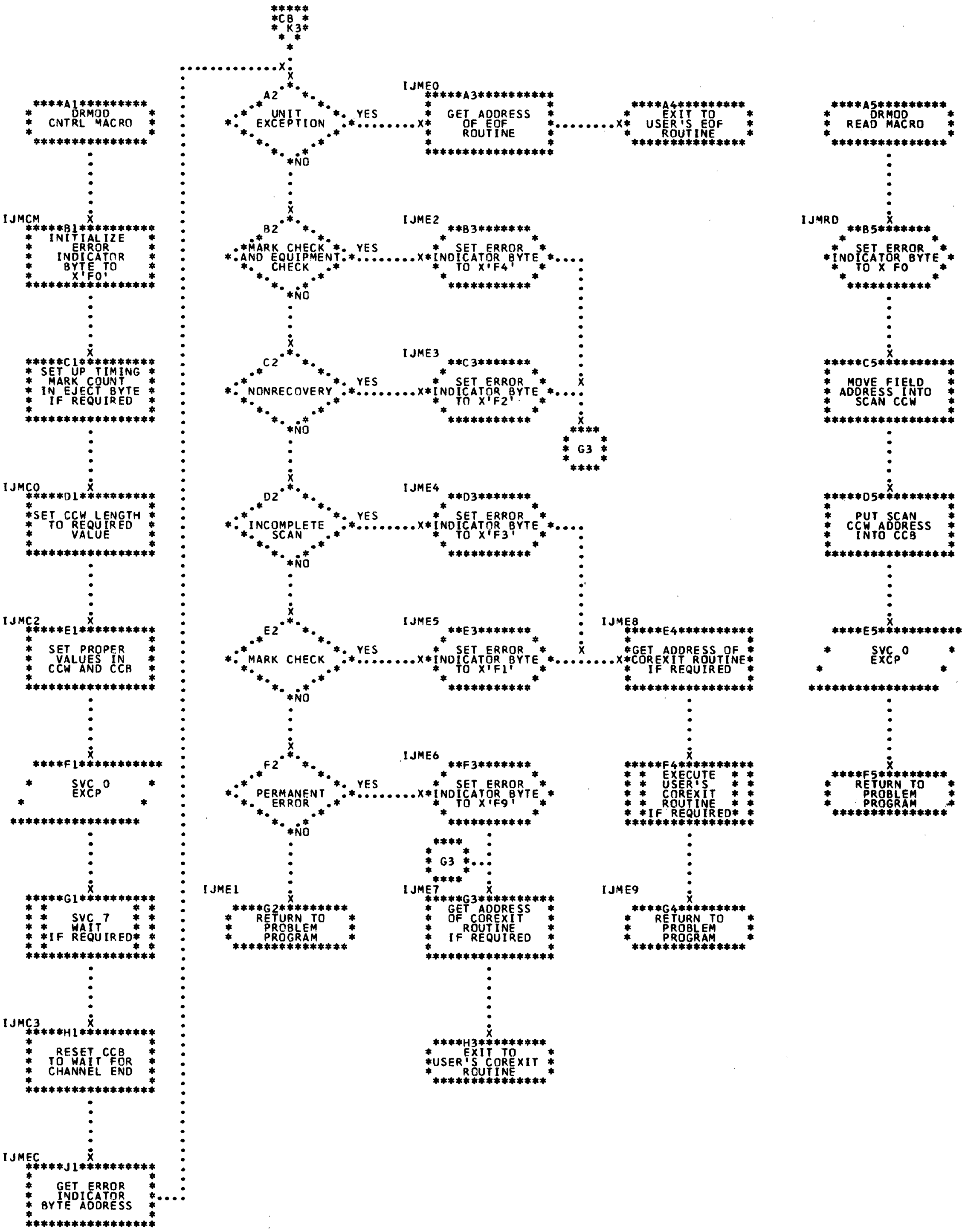

CB
K3
A1
DRMOD
CNTRL MACRO
IJMCM
B1
INITIALIZE ERROR INDICATOR BYTE TO X'F0'
C1
SET UP TIMING MARK COUNT IN EJECT BYTE IF REQUIRED
IJMCO
D1
SET CCW LENGTH TO REQUIRED VALUE
IJMC2
E1
SET PROPER VALUES IN CCW AND CCB
F1
SVC 0
EXCP
G1
SVC 7
WAIT
IF REQUIRED
IJMC3
H1
RESET CCB TO WAIT FOR CHANNEL END
IJMEC
J1
GET ERROR INDICATOR BYTE ADDRESS
A2
UNIT EXCEPTION
YES
NO
IJME0
A3
GET ADDRESS OF EOF ROUTINE
A4
EXIT TO USER'S EOF ROUTINE
B2
MARK CHECK AND EQUIPMENT CHECK
YES
NO
IJME2
B3
SET ERROR INDICATOR BYTE TO X'F4'
C2
NONRECOVERY
YES
NO
IJME3
C3
SET ERROR INDICATOR BYTE TO X'F2'
G3
D2
INCOMPLETE SCAN
YES
NO
IJME4
D3
SET ERROR INDICATOR BYTE TO X'F3'
E2
MARK CHECK
YES
NO
IJME5
E3
SET ERROR INDICATOR BYTE TO X'F1'
IJME8
E4
GET ADDRESS OF COREXIT ROUTINE IF REQUIRED
F2
PERMANENT ERROR
YES
NO
IJME6
F3
SET ERROR INDICATOR BYTE TO X'F9'
F4
EXECUTE USER'S COREXIT ROUTINE IF REQUIRED
G3
IJME1
G2
RETURN TO PROBLEM PROGRAM
IJME7
G3
GET ADDRESS OF COREXIT ROUTINE IF REQUIRED
IJME9
G4
RETURN TO PROBLEM PROGRAM
H3
EXIT TO USER'S COREXIT ROUTINE
A5
DRMOD
READ MACRO
IJMRD
B5
SET ERROR INDICATOR BYTE TO X F0
C5
MOVE FIELD ADDRESS INTO SCAN CCW
D5
PUT SCAN CCW ADDRESS INTO CCB
E5
SVC 0
EXCP
F5
RETURN TO PROBLEM PROGRAM

Chart CB. DRMOD: SETDEV and WAITF Macros

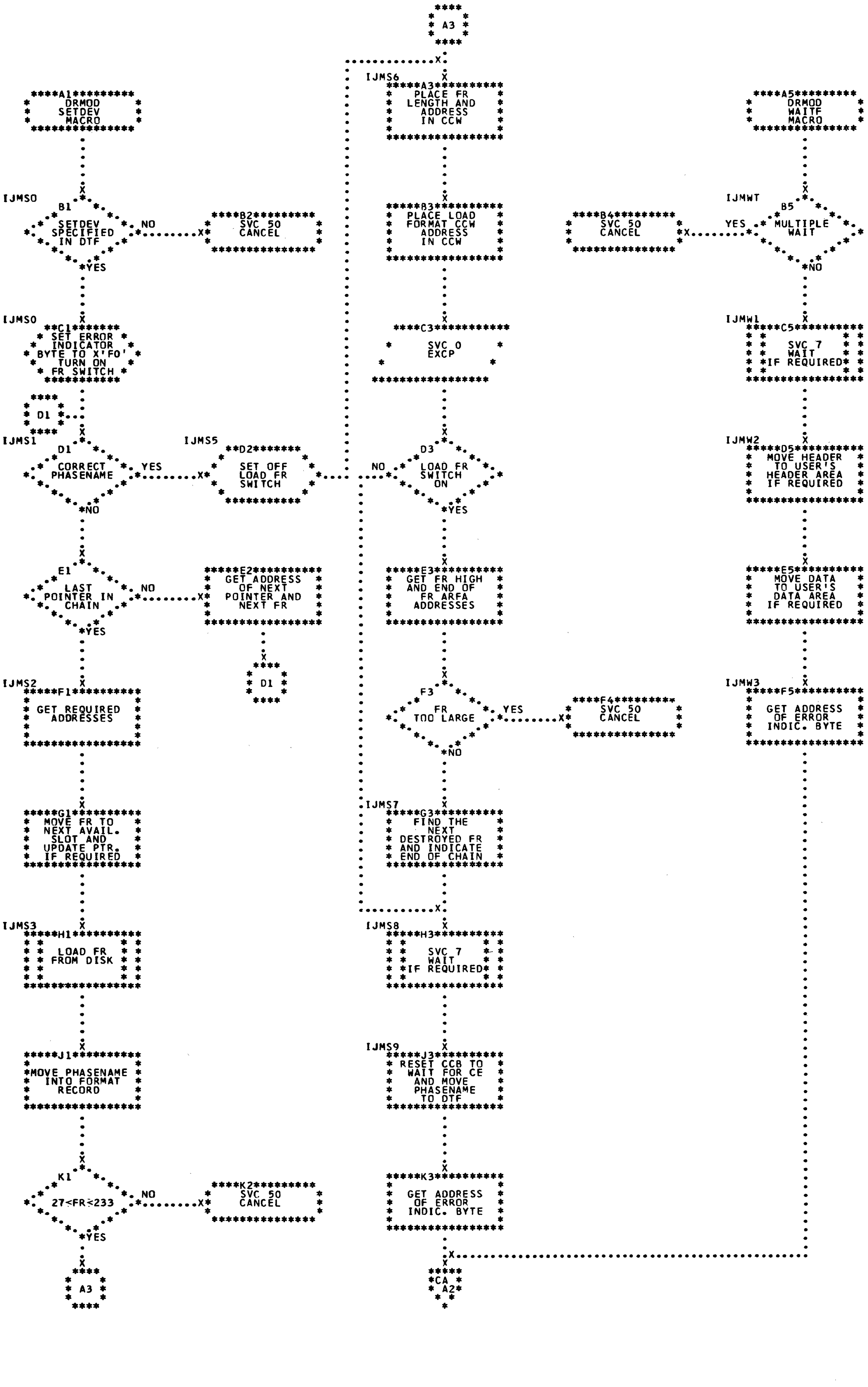

A3
IJMS6
A3
PLACE FR LENGTH AND ADDRESS IN CCW
A1
DRMOD SETDEV MACRO
A5
DRMOD WAITF MACRO
IJMS0
B1
SETDEV SPECIFIED IN DTF
NO
YES
B2
SVC 50 CANCEL
B3
PLACE LOAD FORMAT CCW ADDRESS IN CCW
B4
SVC 50 CANCEL
IJMWT
B5
MULTIPLE WAIT
YES
NO
IJMS0
C1
SET ERROR INDICATOR BYTE TO X'F0' TURN ON FR SWITCH
C3
SVC 0 EXCP
IJMW1
C5
SVC 7 WAIT IF REQUIRED
D1
IJMS1
D1
CORRECT PHASENAME
YES
NO
IJMS5
D2
SET OFF LOAD FR SWITCH
D3
LOAD FR SWITCH ON
NO
YES
IJMW2
D5
MOVE HEADER TO USER'S HEADER AREA IF REQUIRED
E1
LAST POINTER IN CHAIN
NO
YES
E2
GET ADDRESS OF NEXT POINTER AND NEXT FR
D1
E3
GET FR HIGH AND END OF FR AREA ADDRESSES
E5
MOVE DATA TO USER'S DATA AREA IF REQUIRED
IJMS2
F1
GET REQUIRED ADDRESSES
F3
FR TOO LARGE
YES
NO
F4
SVC 50 CANCEL
IJMW3
F5
GET ADDRESS OF ERROR INDIC. BYTE
G1
MOVE FR TO NEXT AVAIL. SLOT AND UPDATE PTR. IF REQUIRED
IJMS7
G3
FIND THE NEXT DESTROYED FR AND INDICATE END OF CHAIN
IJMS3
H1
LOAD FR FROM DISK
IJMS8
H3
SVC 7 WAIT IF REQUIRED
J1
MOVE PHASENAME INTO FORMAT RECORD
IJMS9
J3
RESET CCB TO WAIT FOR CE AND MOVE PHASENAME TO DTF
K1
27<FR≤233
NO
YES
K2
SVC 50 CANCEL
K3
GET ADDRESS OF ERROR INDIC. BYTE
A3
CA A2

Chart CC. $$BOOR01: Open Optical Reader

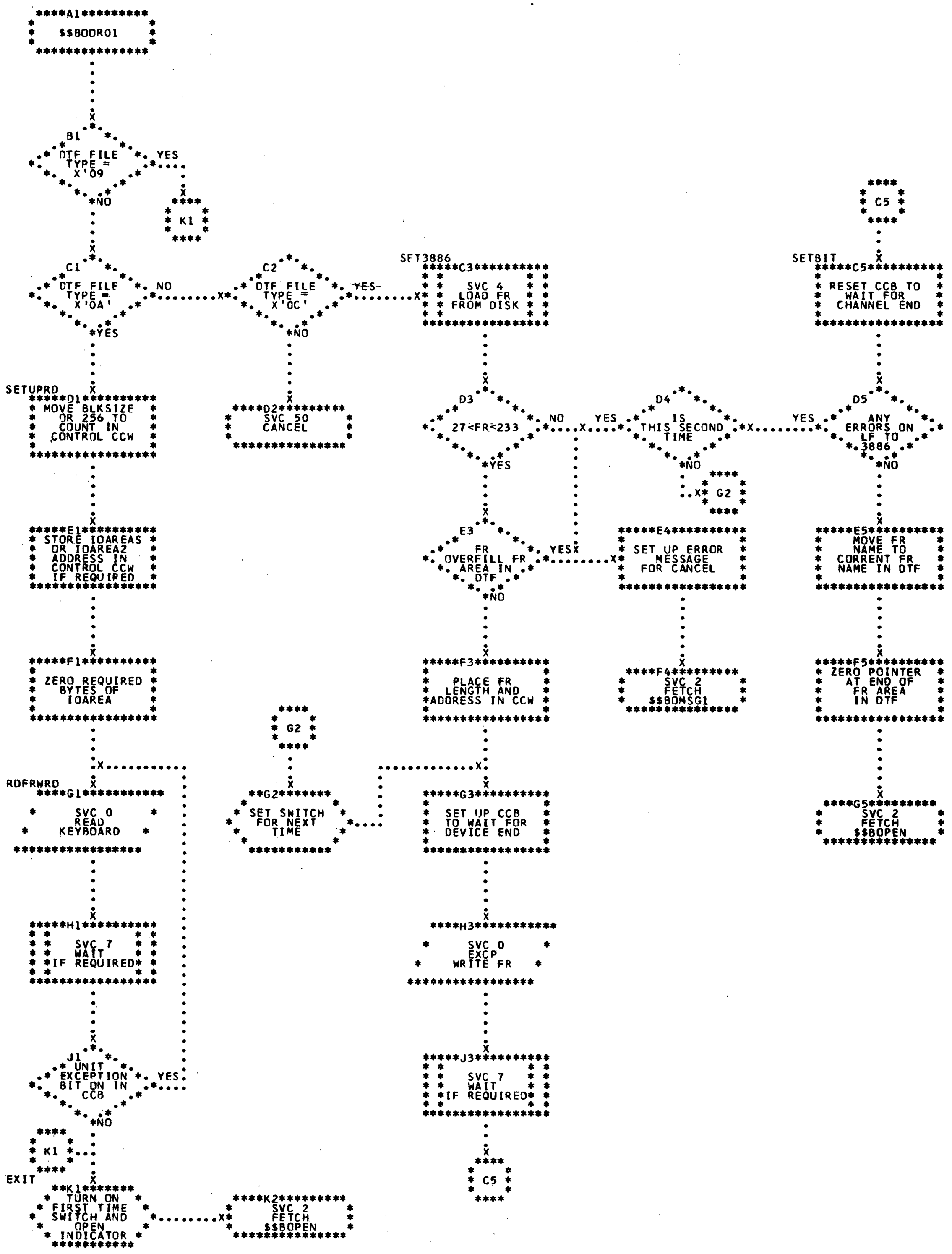


Chart CD. PRMOD: CNTRL Macro

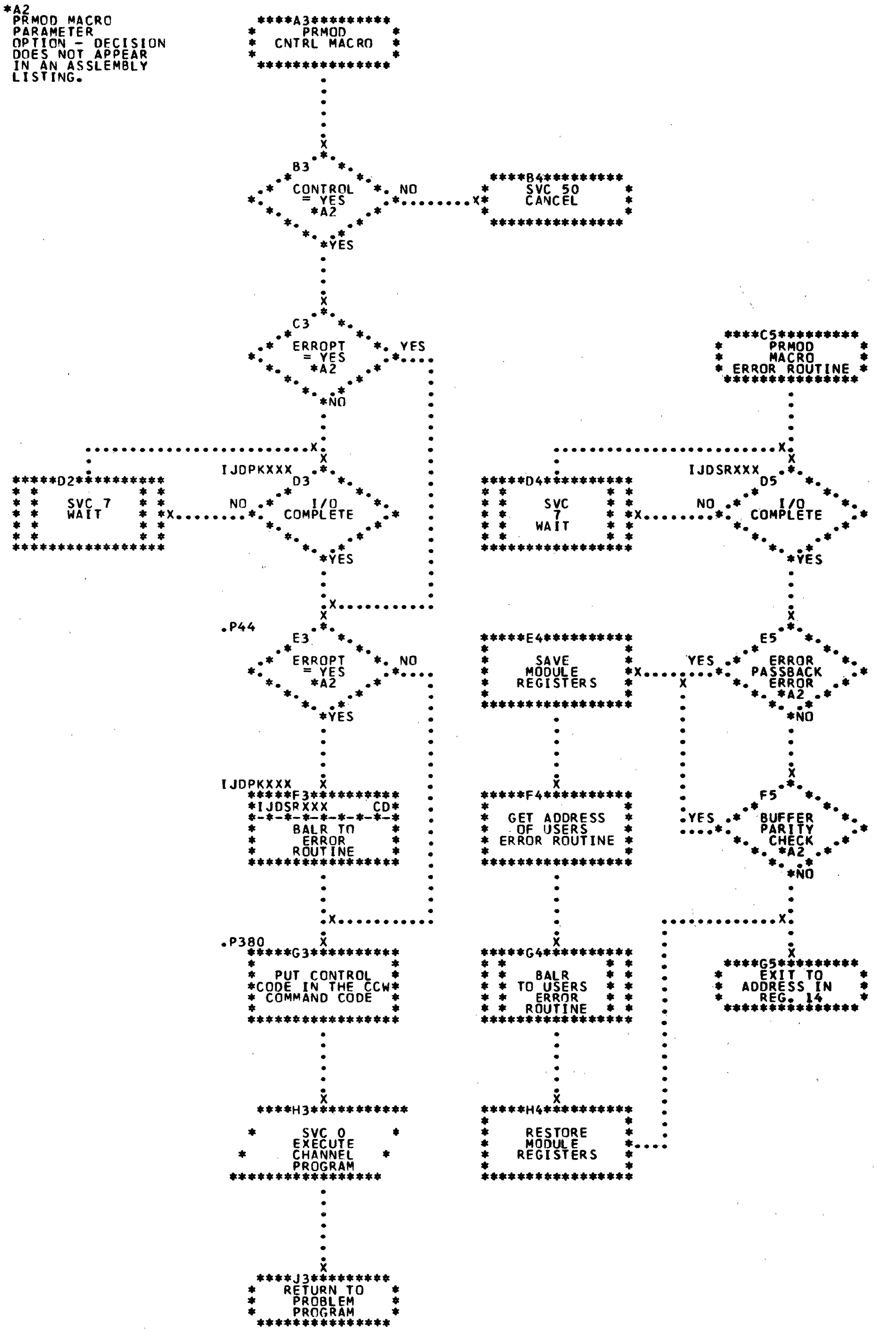


Chart CE. PRMOD: PRTOV Macro

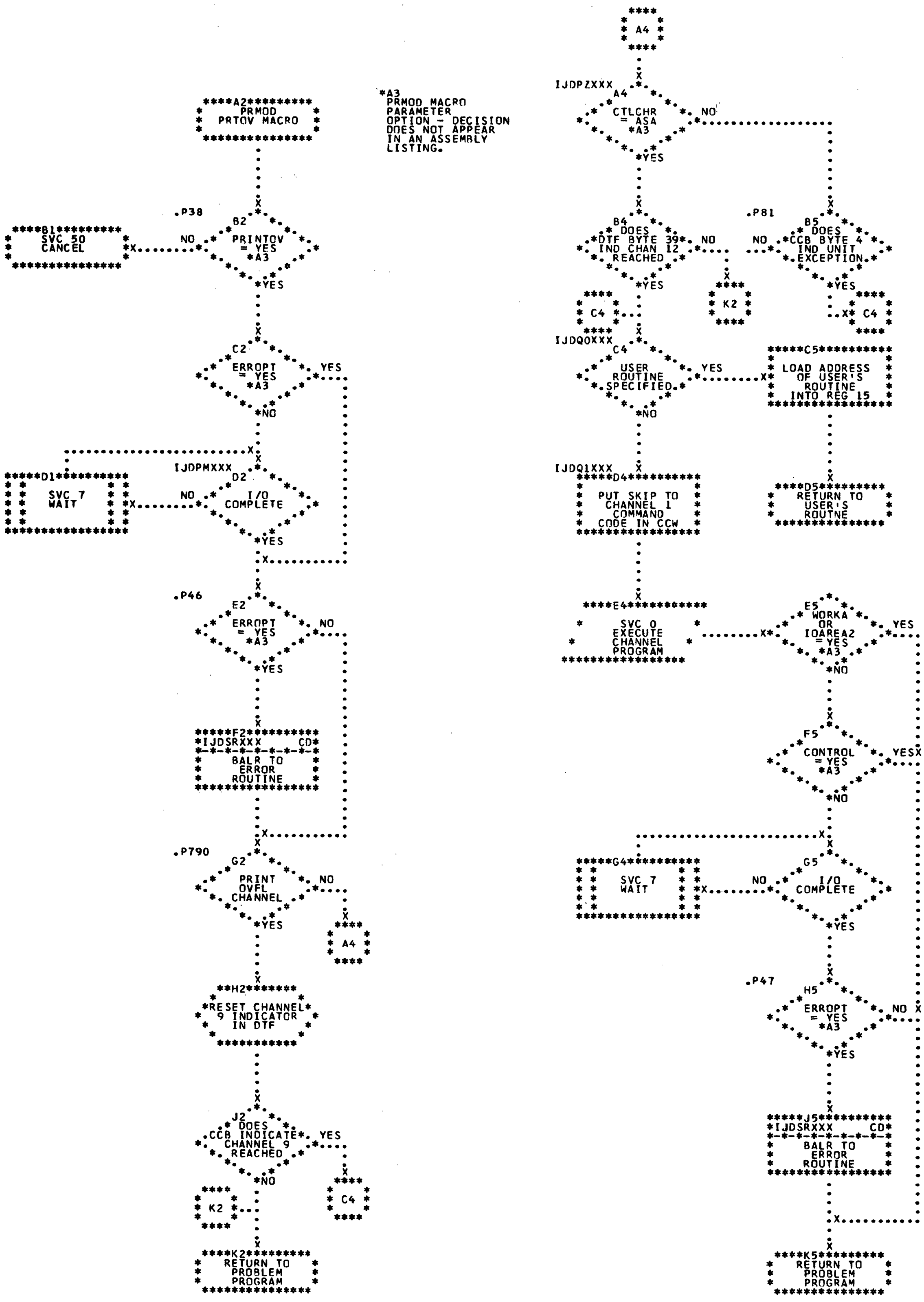

A2 PRMOD PRTOV MACRO
*A3 PRMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
.P38
B1 SVC 50 CANCEL
B2 PRINTOV = YES *A3
NO
YES
C2 ERROPT = YES *A3
YES
NO
IJDPMXXX
D1 SVC 7 WAIT
D2 I/O COMPLETE
NO
YES
.P46
E2 ERROPT = YES *A3
NO
YES
F2 IJDSRXXX CD
BALR TO ERROR ROUTINE
.P790
G2 PRINT OVFL CHANNEL
NO
A4
YES
H2 RESET CHANNEL 9 INDICATOR IN DTF
J2 DOES CCB INDICATE CHANNEL 9 REACHED
YES
C4
NO
K2
K2 RETURN TO PROBLEM PROGRAM
A4
IJDPZXXX
A4 CTLCHR = ASA *A3
NO
YES
B4 DOES DTF BYTE 39 IND CHAN 12 REACHED
NO
K2
YES
C4
.P81
B5 DOES CCB BYTE 4 IND UNIT EXCEPTION
NO
YES
C4
IJDQOXXX
C4 USER ROUTINE SPECIFIED
YES
C5 LOAD ADDRESS OF USER'S ROUTINE INTO REG 15
NO
IJDQ1XXX
D4 PUT SKIP TO CHANNEL 1 COMMAND CODE IN CCW
D5 RETURN TO USER'S ROUTNE
E4 SVC 0 EXECUTE CHANNEL PROGRAM
E5 WORKA OR IOAREA2 = YES *A3
YES
NO
F5 CONTROL = YES *A3
YES
NO
G4 SVC 7 WAIT
G5 I/O COMPLETE
NO
YES
.P47
H5 ERROPT = YES *A3
NO
YES
J5 IJDSRXXX CD
BALR TO ERROR ROUTINE
K5 RETURN TO PROBLEM PROGRAM

Chart CF. PRMOD: PUT Macro (Part 1 of 3)

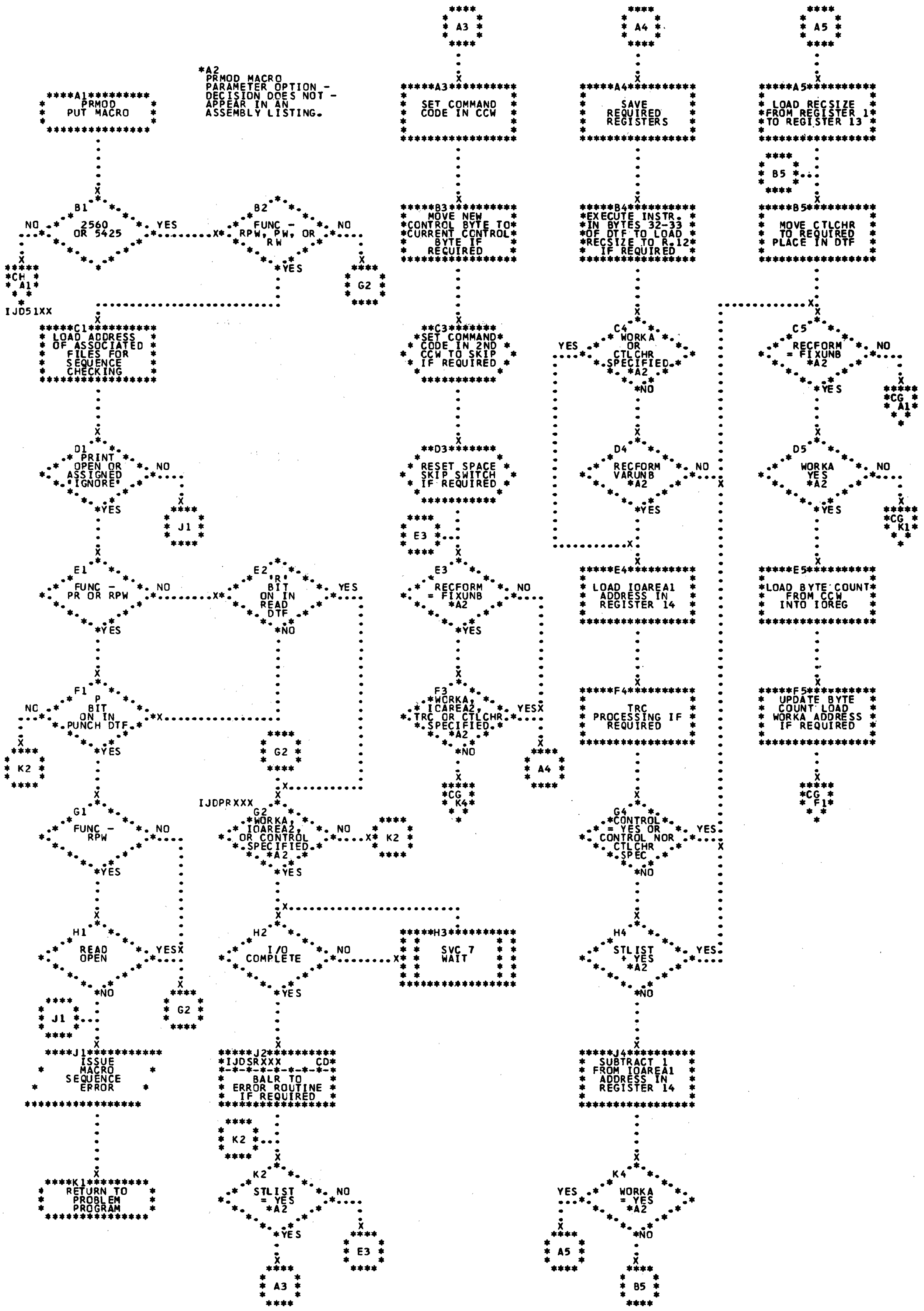

A1 PRMOD PUT MACRO
*A2 PRMOD MACRO PARAMETER OPTION - DECISION DOES NOT - APPEAR IN AN ASSEMBLY LISTING.
B1 2560 OR 5425
NO
YES
CH A1
IJD51XX
B2 FUNC - RPW, PW, OR RW
NO
YES
G2
C1 LOAD ADDRESS OF ASSOCIATED FILES FOR SEQUENCE CHECKING
D1 PRINT OPEN OR ASSIGNED 'IGNORE'
NO
YES
J1
E1 FUNC - PR OR RPW
NO
YES
E2 'R' BIT ON IN READ DTF
YES
NO
F1 P BIT ON IN PUNCH DTF
NO
YES
K2
G2
IJDPRXXX
G1 FUNC - RPW
NO
YES
G2 WORKA, IOAREA2, OR CONTROL SPECIFIED *A2
NO
K2
YES
H1 READ OPEN
YES
NO
J1
G2
H2 I/O COMPLETE
NO
YES
H3 SVC 7 WAIT
J1 ISSUE MACRO SEQUENCE ERROR
J2 IJDSRXXX CD
BALR TO ERROR ROUTINE IF REQUIRED
K2
K1 RETURN TO PROBLEM PROGRAM
K2 STLIST = YES *A2
NO
YES
E3
A3
A3
A3 SET COMMAND CODE IN CCW
B3 MOVE NEW CONTROL BYTE TO CURRENT CONTROL BYTE IF REQUIRED
C3 SET COMMAND CODE IN 2ND CCW TO SKIP IF REQUIRED
D3 RESET SPACE SKIP SWITCH IF REQUIRED
E3
E3 RECFORM = FIXUNB *A2
NO
YES
F3 WORKA, IOAREA2, TRC OR CTLCHR SPECIFIED *A2
YES
NO
A4
CG K4
A4
A4 SAVE REQUIRED REGISTERS
B4 EXECUTE INSTR. IN BYTES 32-33 OF DTF TO LOAD RECSIZE TO R.12 IF REQUIRED
C4 WORKA OR CTLCHR SPECIFIED *A2
YES
NO
D4 RECFORM VARUNB *A2
NO
YES
E4 LOAD IOAREA1 ADDRESS IN REGISTER 14
F4 TRC PROCESSING IF REQUIRED
G4 CONTROL = YES OR CONTROL NOR CTLCHR SPEC
YES
NO
H4 STLIST + YES *A2
YES
NO
J4 SUBTRACT 1 FROM IOAREA1 ADDRESS IN REGISTER 14
K4 WORKA = YES *A2
YES
NO
A5
B5
A5
A5 LOAD RECSIZE FROM REGISTER 1 TO REGISTER 13
B5
B5 MOVE CTLCHR TO REQUIRED PLACE IN DTF
C5 RECFORM = FIXUNB *A2
NO
YES
CG A1
D5 WORKA YES *A2
NO
YES
CG K1
E5 LOAD BYTE COUNT FROM CCW INTO IOREG
F5 UPDATE BYTE COUNT LOAD WORKA ADDRESS IF REQUIRED
CG F1

Chart CG. PRMOD: PUT Macro (Part 2 of 3)

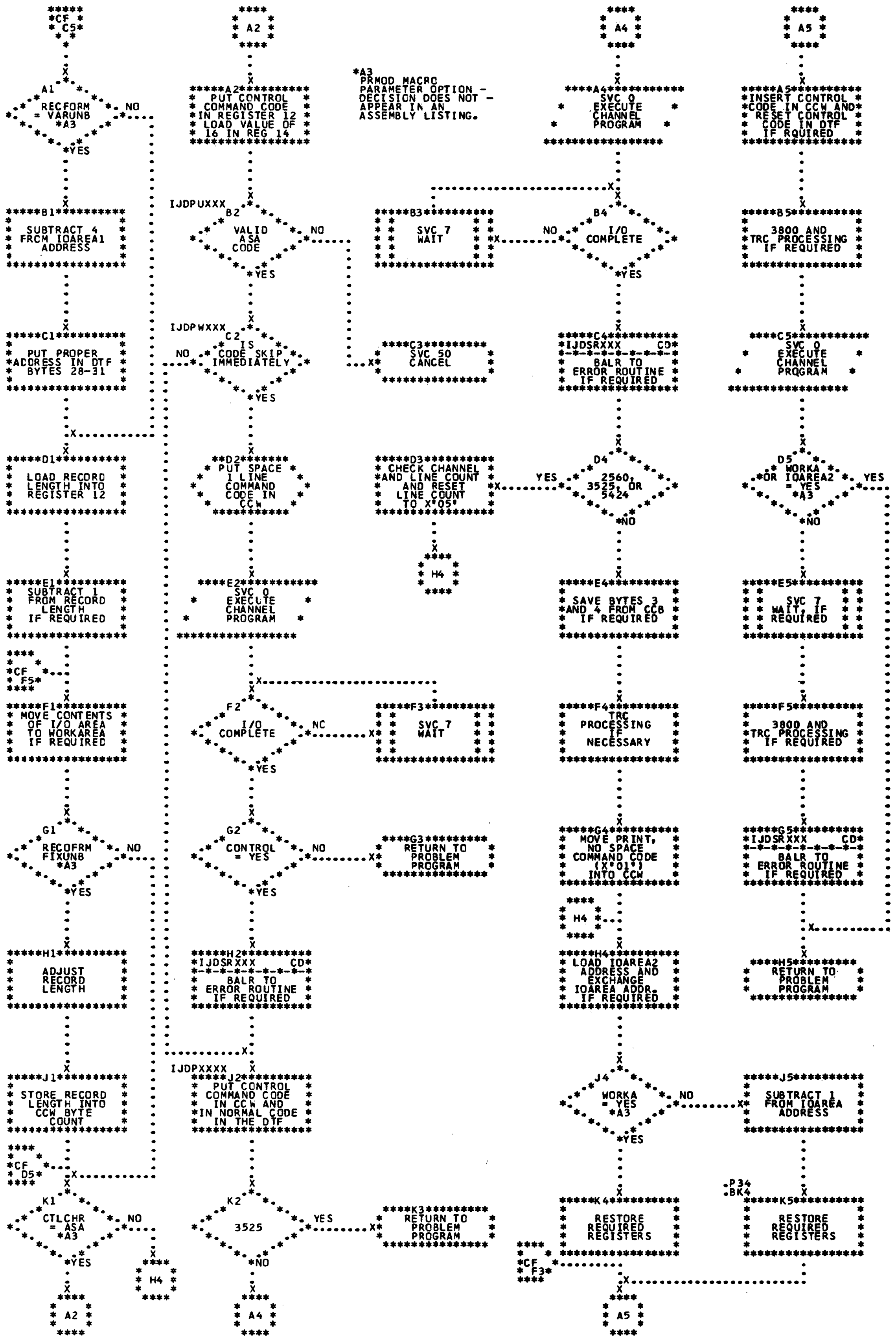

CF C5
A1
RECFORM = VARUNB *A3
NO
YES
B1
SUBTRACT 4 FROM IOAREA1 ADDRESS
C1
PUT PROPER ADDRESS IN DTF BYTES 28-31
D1
LOAD RECORD LENGTH INTO REGISTER 12
E1
SUBTRACT 1 FROM RECORD LENGTH IF REQUIRED
CF F5
F1
MOVE CONTENTS OF I/O AREA TO WORKAREA IF REQUIRED
G1
RECOFRM FIXUNB *A3
NO
YES
H1
ADJUST RECORD LENGTH
J1
STORE RECORD LENGTH INTO CCW BYTE COUNT
CF D5
K1
CTLCHR = ASA *A3
NO
YES
H4
A2
A2
A2
PUT CONTROL COMMAND CODE IN REGISTER 12 LOAD VALUE OF 16 IN REG 14
IJDPUXXX
B2
VALID ASA CODE
NO
YES
IJDPWXXX
C2
IS CODE SKIP IMMEDIATELY
NO
YES
D2
PUT SPACE 1 LINE COMMAND CODE IN CCW
E2
SVC 0 EXECUTE CHANNEL PROGRAM
F2
I/O COMPLETE
NC
YES
G2
CONTROL = YES
NO
YES
H2
IJDSRXXX CD
BALR TO ERROR ROUTINE IF REQUIRED
IJDPXXXX
J2
PUT CONTROL COMMAND CODE IN CCW AND IN NORMAL CODE IN THE DTF
K2
3525
YES
NO
A4
*A3
PRMOD MACRO PARAMETER OPTION - DECISION DOES NOT - APPEAR IN AN ASSEMBLY LISTING.
B3
SVC 7 WAIT
C3
SVC 50 CANCEL
D3
CHECK CHANNEL AND LINE COUNT AND RESET LINE COUNT TO X'05'
H4
F3
SVC 7 WAIT
G3
RETURN TO PROBLEM PROGRAM
K3
RETURN TO PROBLEM PROGRAM
A4
A4
SVC 0 EXECUTE CHANNEL PROGRAM
B4
I/O COMPLETE
NO
YES
C4
IJDSRXXX CD
BALR TO ERROR ROUTINE IF REQUIRED
D4
2560, 3525, OR 5424
YES
NO
E4
SAVE BYTES 3 AND 4 FROM CCB IF REQUIRED
F4
TRC PROCESSING IF NECESSARY
G4
MOVE PRINT, NO SPACE COMMAND CODE (X'01') INTO CCW
H4
H4
LOAD IOAREA2 ADDRESS AND EXCHANGE IOAREA ADDR. IF REQUIRED
J4
WORKA = YES *A3
NO
YES
.P34
.BK4
K4
RESTORE REQUIRED REGISTERS
CF F3
A5
A5
A5
INSERT CONTROL CODE IN CCW AND RESET CONTROL CODE IN DTF IF RQUIRED
B5
3800 AND TRC PROCESSING IF REQUIRED
C5
SVC 0 EXECUTE CHANNEL PROGRAM
D5
WORKA OR IOAREA2 = YES *A3
YES
NO
E5
SVC 7 WAIT, IF REQUIRED
F5
3800 AND TRC PROCESSING IF REQUIRED
G5
IJDSRXXX CD
BALR TO ERROR ROUTINE IF REQUIRED
H5
RETURN TO PROBLEM PROGRAM
J5
SUBTRACT 1 FROM IOAREA ADDRESS
K5
RESTORE REQUIRED REGISTERS

Chart CH. PRMOD: PUT Macro (Part 3 of 3)

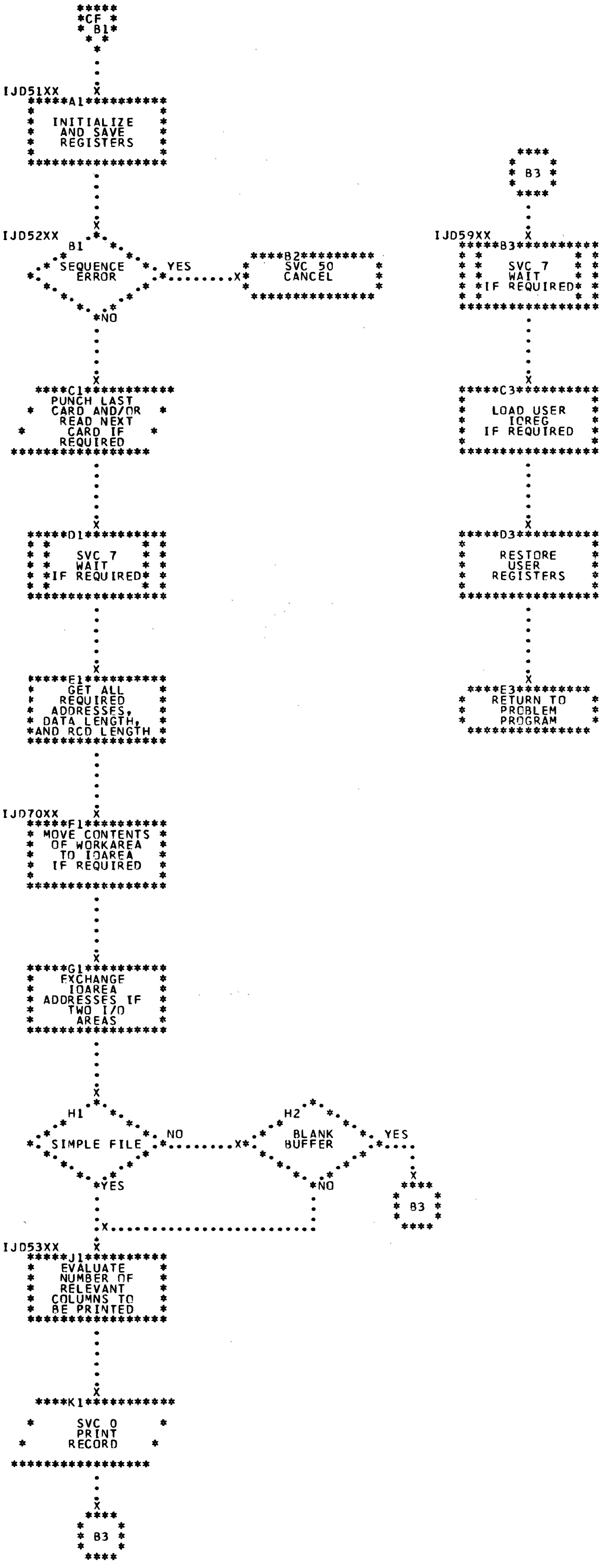

CF B1
IJD51XX
A1
INITIALIZE AND SAVE REGISTERS
IJD52XX
B1
SEQUENCE ERROR
YES
NO
B2
SVC 50 CANCEL
C1
PUNCH LAST CARD AND/OR READ NEXT CARD IF REQUIRED
D1
SVC 7 WAIT IF REQUIRED
E1
GET ALL REQUIRED ADDRESSES, DATA LENGTH, AND RCD LENGTH
IJD70XX
F1
MOVE CONTENTS OF WORKAREA TO IOAREA IF REQUIRED
G1
EXCHANGE IOAREA ADDRESSES IF TWO I/O AREAS
H1
SIMPLE FILE
NO
YES
H2
BLANK BUFFER
YES
NO
B3
IJD53XX
J1
EVALUATE NUMBER OF RELEVANT COLUMNS TO BE PRINTED
K1
SVC 0 PRINT RECORD
B3
B3
IJD59XX
B3
SVC 7 WAIT IF REQUIRED
C3
LOAD USER IOREG IF REQUIRED
D3
RESTORE USER REGISTERS
E3
RETURN TO PROBLEM PROGRAM

Chart CJ. PTMCD: GET Macro, no Translation, and GET Macro, Translation, no Shifted Code, Device=2671

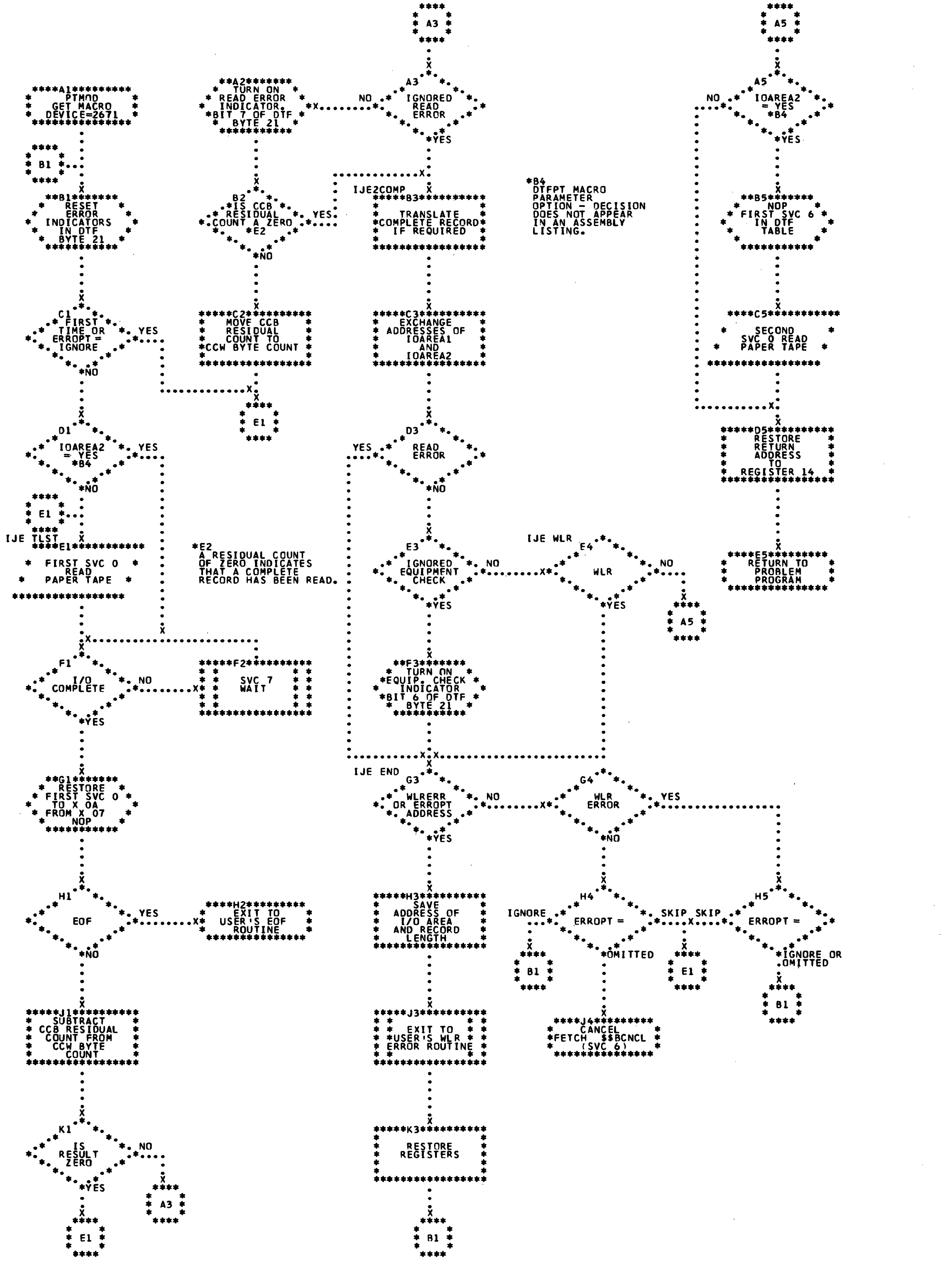


Chart CK. PTMOD: Get Macro, Translation, Shifted Code, Fixed Unblocked Records, Device=2671

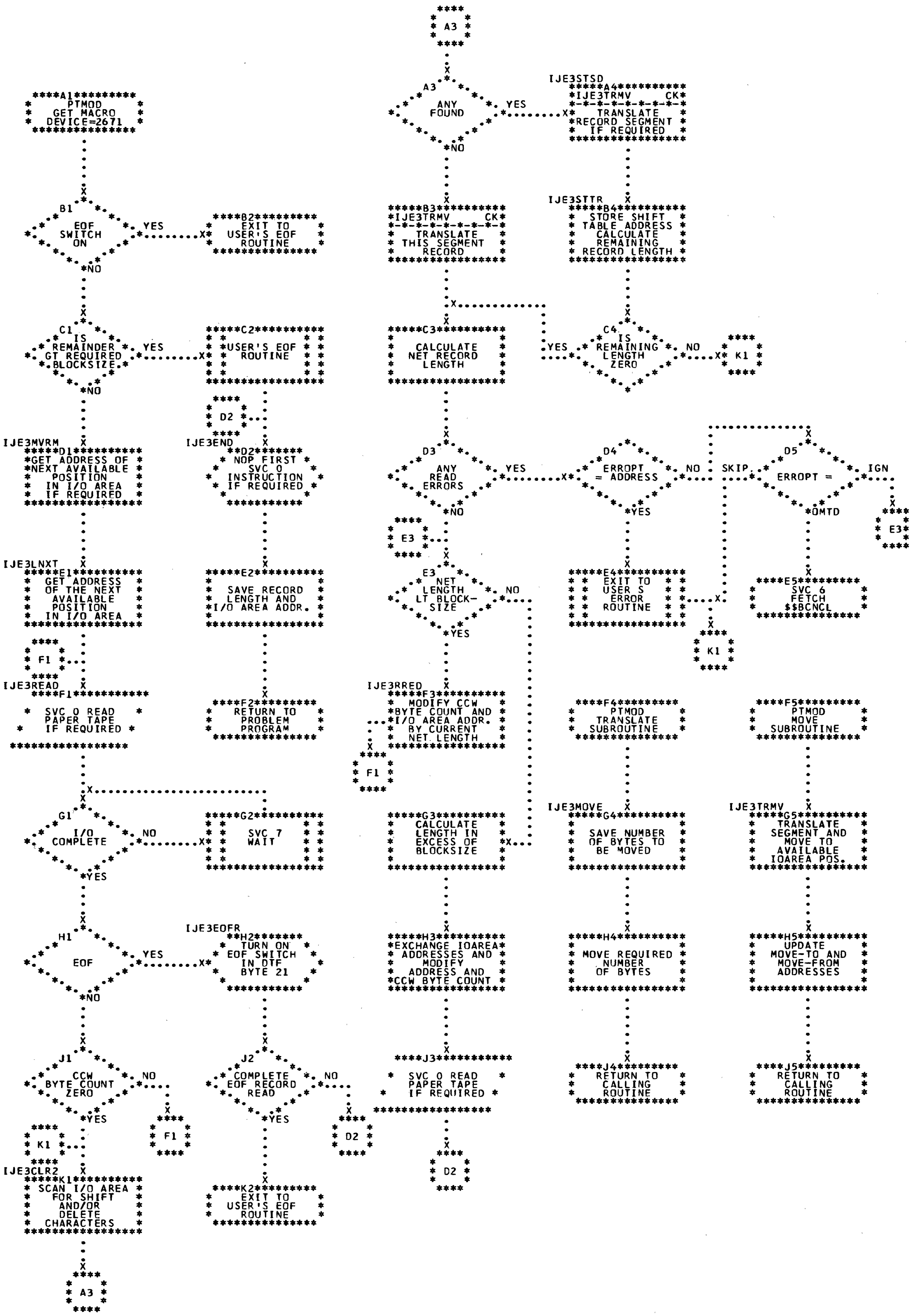

A1
PTMOD
GET MACRO
DEVICE=2671
B1
EOF SWITCH ON
YES
NO
B2
EXIT TO USER'S EOF ROUTINE
C1
IS REMAINDER GT REQUIRED BLOCKSIZE
YES
NO
C2
USER'S EOF ROUTINE
D2
IJE3MVRM
D1
GET ADDRESS OF NEXT AVAILABLE POSITION IN I/O AREA IF REQUIRED
IJE3END
D2
NOP FIRST SVC 0 INSTRUCTION IF REQUIRED
IJE3LNXT
E1
GET ADDRESS OF THE NEXT AVAILABLE POSITION IN I/O AREA
E2
SAVE RECORD LENGTH AND I/O AREA ADDR.
F1
IJE3READ
F1
SVC 0 READ PAPER TAPE IF REQUIRED
F2
RETURN TO PROBLEM PROGRAM
G1
I/O COMPLETE
NO
YES
G2
SVC 7 WAIT
H1
EOF
YES
NO
IJE3EOFR
H2
TURN ON EOF SWITCH IN DTF BYTE 21
J1
CCW BYTE COUNT ZERO
NO
YES
F1
K1
J2
COMPLETE EOF RECORD READ
NO
YES
D2
IJE3CLR2
K1
SCAN I/O AREA FOR SHIFT AND/OR DELETE CHARACTERS
A3
K2
EXIT TO USER'S EOF ROUTINE
A3
A3
ANY FOUND
YES
NO
IJE3STSD
A4
IJE3TRMV CK
TRANSLATE RECORD SEGMENT IF REQUIRED
B3
IJE3TRMV CK
TRANSLATE THIS SEGMENT RECORD
IJE3STTR
B4
STORE SHIFT TABLE ADDRESS CALCULATE REMAINING RECORD LENGTH
C3
CALCULATE NET RECORD LENGTH
C4
IS REMAINING LENGTH ZERO
YES
NO
K1
D3
ANY READ ERRORS
YES
NO
E3
D4
ERROPT = ADDRESS
NO
YES
D5
ERROPT =
SKIP
IGN
OMTD
E3
E3
NET LENGTH LT BLOCK-SIZE
NO
YES
E4
EXIT TO USER S ERROR ROUTINE
K1
E5
SVC 6 FETCH $$BCNCL
IJE3RRED
F3
MODIFY CCW BYTE COUNT AND I/O AREA ADDR. BY CURRENT NET LENGTH
F1
F4
PTMOD TRANSLATE SUBROUTINE
F5
PTMOD MOVE SUBROUTINE
G3
CALCULATE LENGTH IN EXCESS OF BLOCKSIZE
IJE3MOVE
G4
SAVE NUMBER OF BYTES TO BE MOVED
IJE3TRMV
G5
TRANSLATE SEGMENT AND MOVE TO AVAILABLE IOAREA POS.
H3
EXCHANGE IOAREA ADDRESSES AND MODIFY ADDRESS AND CCW BYTE COUNT
H4
MOVE REQUIRED NUMBER OF BYTES
H5
UPDATE MOVE-TO AND MOVE-FROM ADDRESSES
J3
SVC 0 READ PAPER TAPE IF REQUIRED
D2
J4
RETURN TO CALLING ROUTINE
J5
RETURN TO CALLING ROUTINE

Chart CL. PTMOD: GET Macro, Translation, Shifted Code, Undefined Records, Device=2671

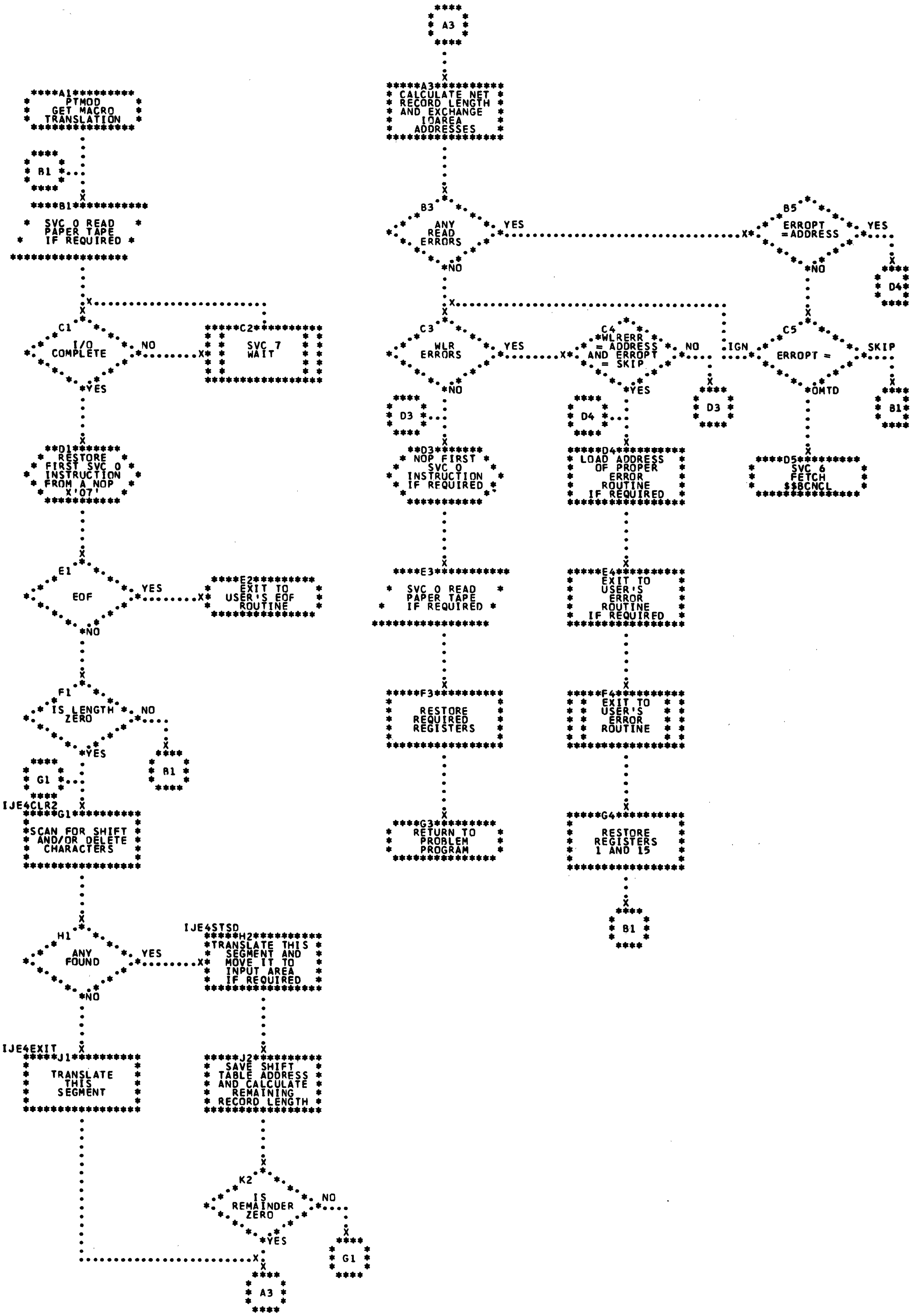

A3
A3 CALCULATE NET RECORD LENGTH AND EXCHANGE IOAREA ADDRESSES
A1 PTMOD GET MACRO TRANSLATION
B1
B1 SVC 0 READ PAPER TAPE IF REQUIRED
B3 ANY READ ERRORS
YES
NO
B5 ERROPT =ADDRESS
YES
NO
D4
C1 I/O COMPLETE
NO
YES
C2 SVC 7 WAIT
C3 WLR ERRORS
YES
NO
C4 WLRERR = ADDRESS AND ERROPT = SKIP
NO
YES
IGN
C5 ERROPT =
SKIP
OMTD
D3
D4
D3
B1
D1 RESTORE FIRST SVC 0 INSTRUCTION FROM A NOP X'07'
D3 NOP FIRST SVC 0 INSTRUCTION IF REQUIRED
D4 LOAD ADDRESS OF PROPER ERROR ROUTINE IF REQUIRED
D5 SVC 6 FETCH $$BCNCL
E1 EOF
YES
NO
E2 EXIT TO USER'S EOF ROUTINE
E3 SVC 0 READ PAPER TAPE IF REQUIRED
E4 EXIT TO USER'S ERROR ROUTINE IF REQUIRED
F1 IS LENGTH ZERO
NO
YES
B1
F3 RESTORE REQUIRED REGISTERS
F4 EXIT TO USER'S ERROR ROUTINE
G1
IJE4CLR2
G1 SCAN FOR SHIFT AND/OR DELETE CHARACTERS
G3 RETURN TO PROBLEM PROGRAM
G4 RESTORE REGISTERS 1 AND 15
B1
H1 ANY FOUND
YES
NO
IJE4STSD
H2 TRANSLATE THIS SEGMENT AND MOVE IT TO INPUT AREA IF REQUIRED
IJE4EXIT
J1 TRANSLATE THIS SEGMENT
J2 SAVE SHIFT TABLE ADDRESS AND CALCULATE REMAINING RECORD LENGTH
K2 IS REMAINDER ZERO
NO
YES
G1
A3

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

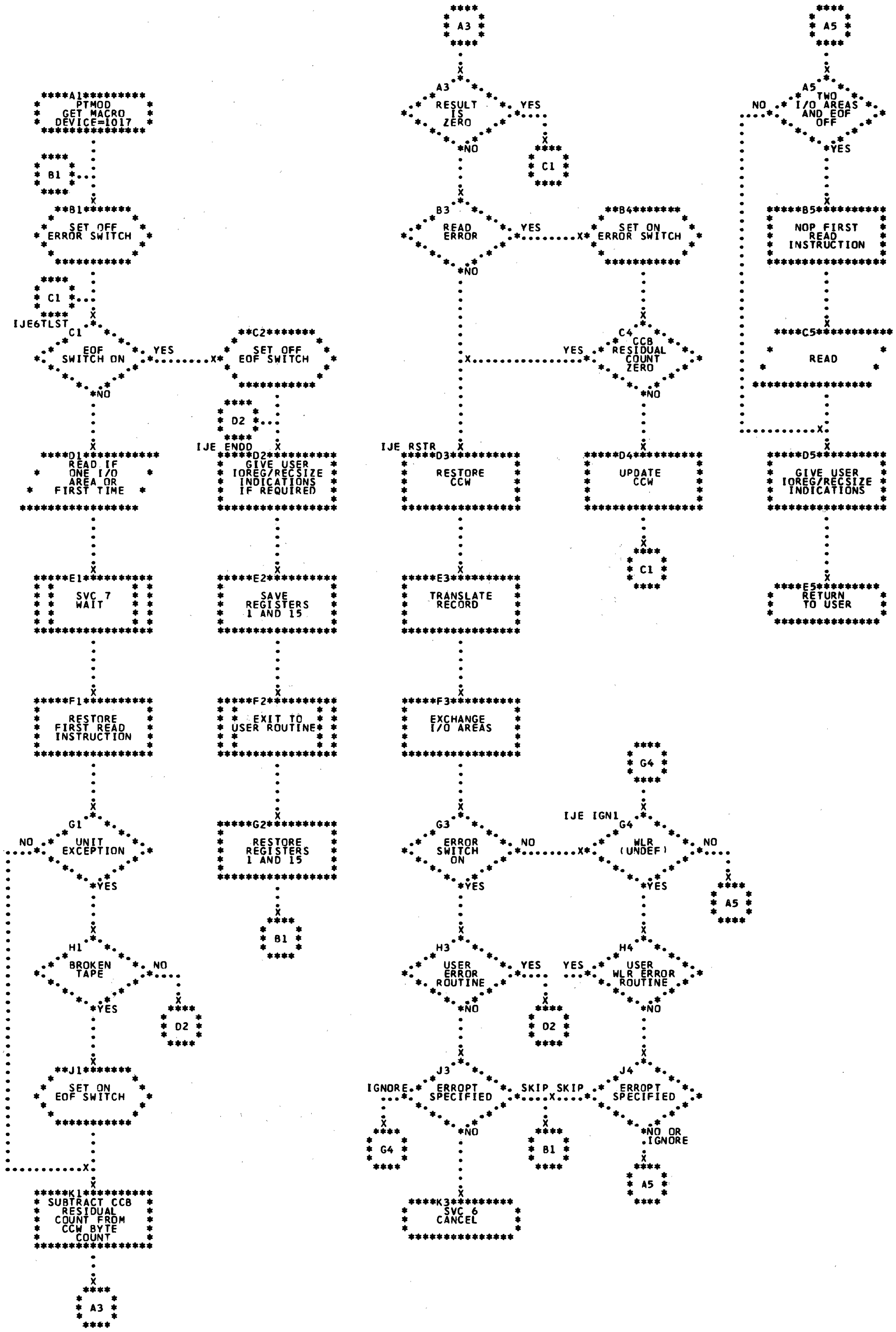


Chart CN. PTMOD: GET Macro, Translation, Shifted Code, Fixed Unblocked Records, Device=1017

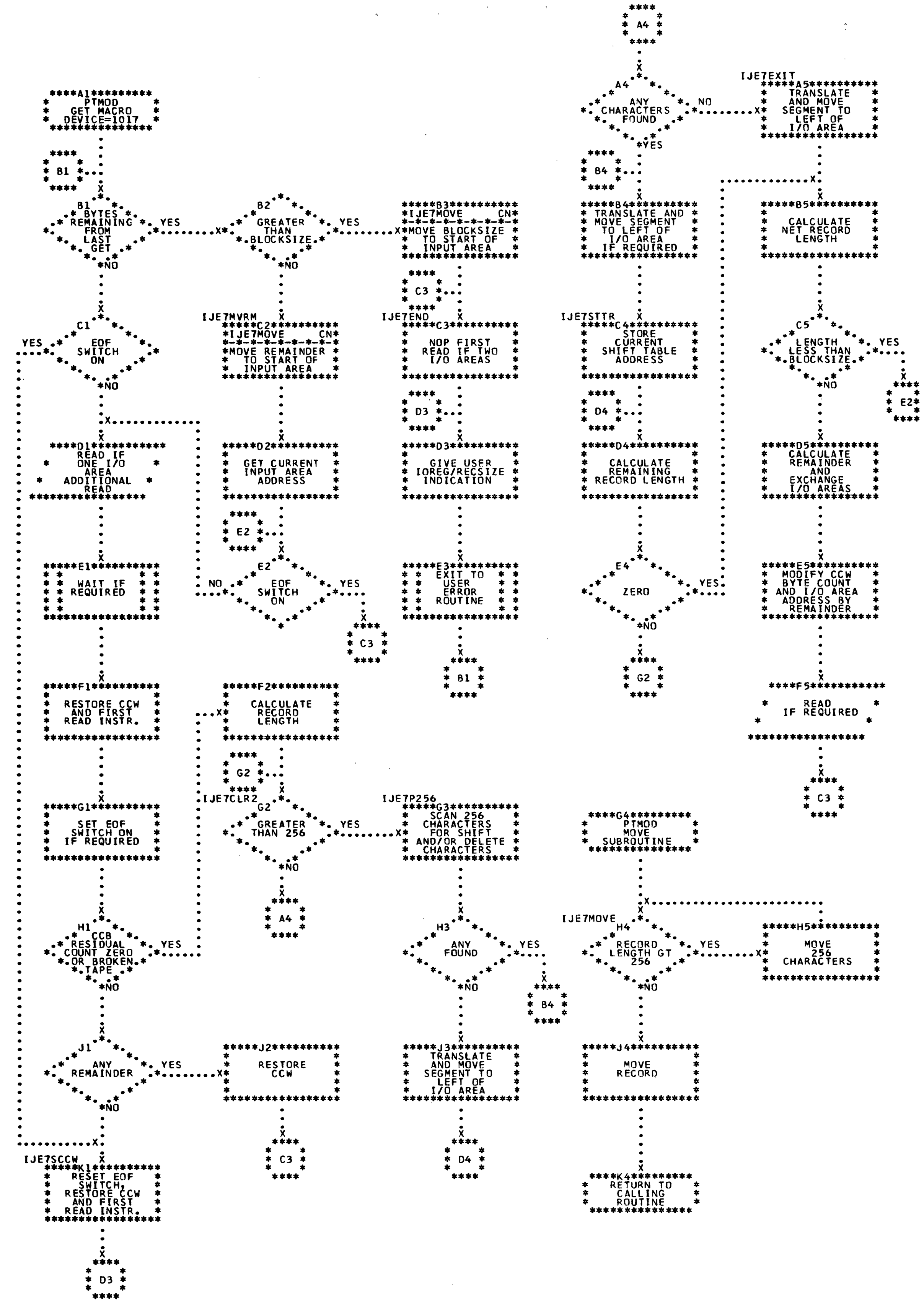

A4
A4
ANY CHARACTERS FOUND
NO
YES
IJE7EXIT
A5
TRANSLATE AND MOVE SEGMENT TO LEFT OF I/O AREA
A1
PTMOD GET MACRO DEVICE=1017
B1
B1
BYTES REMAINING FROM LAST GET
YES
NO
B2
GREATER THAN BLOCKSIZE
YES
NO
B3
IJE7MOVE CN
MOVE BLOCKSIZE TO START OF INPUT AREA
B4
B4
TRANSLATE AND MOVE SEGMENT TO LEFT OF I/O AREA IF REQUIRED
B5
CALCULATE NET RECORD LENGTH
C1
EOF SWITCH ON
YES
NO
IJE7MVRM
C2
IJE7MOVE CN
MOVE REMAINDER TO START OF INPUT AREA
C3
IJE7END
C3
NOP FIRST READ IF TWO I/O AREAS
IJE7STTR
C4
STORE CURRENT SHIFT TABLE ADDRESS
C5
LENGTH LESS THAN BLOCKSIZE
YES
NO
E2
D1
READ IF ONE I/O AREA ADDITIONAL READ
D2
GET CURRENT INPUT AREA ADDRESS
D3
D3
GIVE USER IOREG/RECSIZE INDICATION
D4
D4
CALCULATE REMAINING RECORD LENGTH
D5
CALCULATE REMAINDER AND EXCHANGE I/O AREAS
E2
E1
WAIT IF REQUIRED
E2
EOF SWITCH ON
NO
YES
C3
E3
EXIT TO USER ERROR ROUTINE
B1
E4
ZERO
YES
NO
G2
E5
MODIFY CCW BYTE COUNT AND I/O AREA ADDRESS BY REMAINDER
F1
RESTORE CCW AND FIRST READ INSTR.
F2
CALCULATE RECORD LENGTH
F5
READ IF REQUIRED
C3
G2
IJE7CLR2
G2
GREATER THAN 256
YES
NO
A4
IJE7P256
G3
SCAN 256 CHARACTERS FOR SHIFT AND/OR DELETE CHARACTERS
G1
SET EOF SWITCH ON IF REQUIRED
G4
PTMOD MOVE SUBROUTINE
H1
CCB RESIDUAL COUNT ZERO OR BROKEN TAPE
YES
NO
H3
ANY FOUND
YES
NO
B4
IJE7MOVE
H4
RECORD LENGTH GT 256
YES
NO
H5
MOVE 256 CHARACTERS
J1
ANY REMAINDER
YES
NO
J2
RESTORE CCW
C3
J3
TRANSLATE AND MOVE SEGMENT TO LEFT OF I/O AREA
D4
J4
MOVE RECORD
IJE7SCCW
K1
RESET EOF SWITCH, RESTORE CCW AND FIRST READ INSTR.
D3
K4
RETURN TO CALLING ROUTINE

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

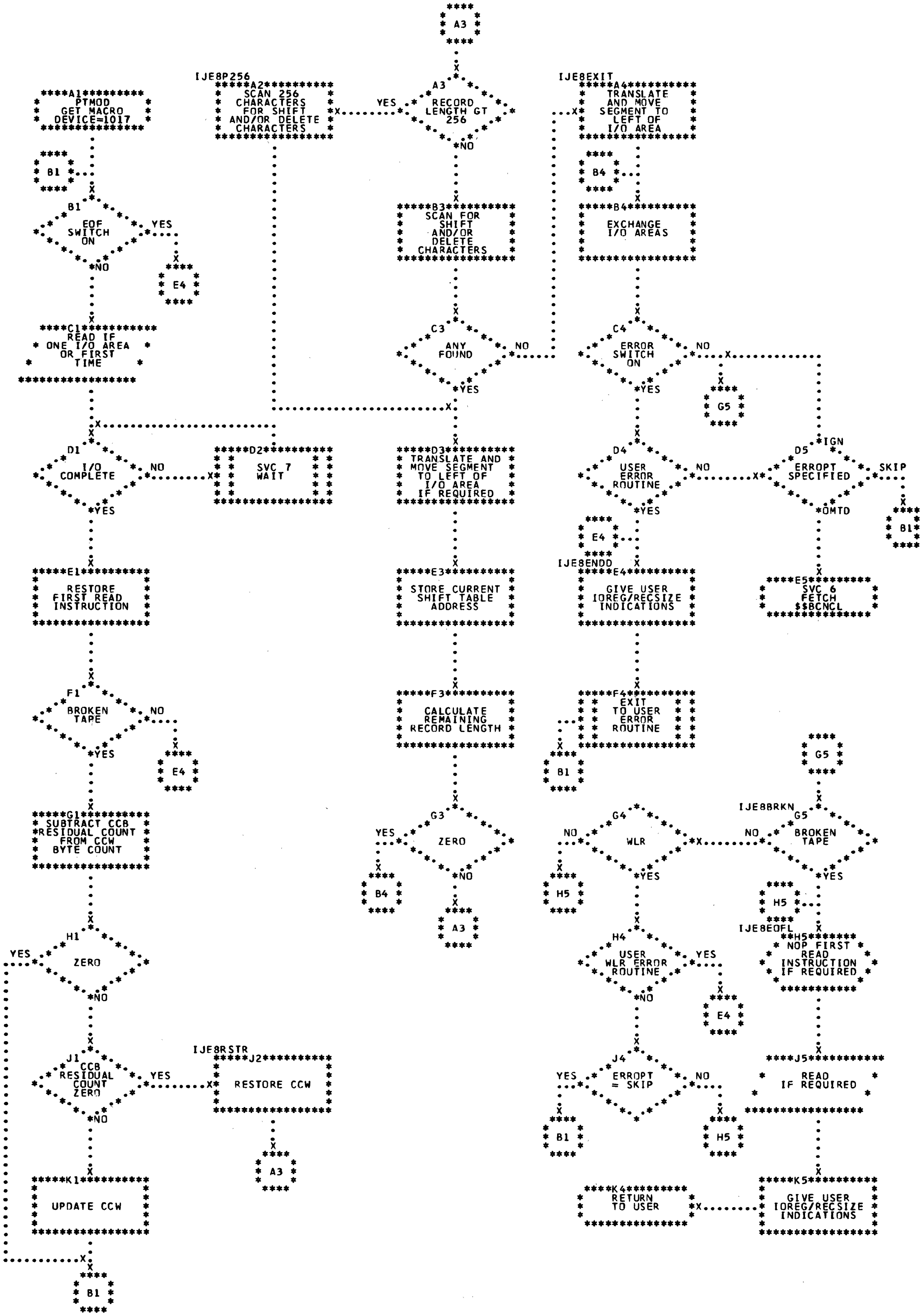

A3
A1 PTMOD GET MACRO DEVICE=1017
B1
B1 EOF SWITCH ON
YES
NO
E4
C1 READ IF ONE I/O AREA OR FIRST TIME
D1 I/O COMPLETE
NO
YES
D2 SVC 7 WAIT
E1 RESTORE FIRST READ INSTRUCTION
F1 BROKEN TAPE
NO
YES
E4
G1 SUBTRACT CCB RESIDUAL COUNT FROM CCW BYTE COUNT
H1 ZERO
YES
NO
J1 CCB RESIDUAL COUNT ZERO
YES
NO
IJE8RSTR
J2 RESTORE CCW
A3
K1 UPDATE CCW
B1
IJE8P256
A2 SCAN 256 CHARACTERS FOR SHIFT AND/OR DELETE CHARACTERS
A3 RECORD LENGTH GT 256
YES
NO
B3 SCAN FOR SHIFT AND/OR DELETE CHARACTERS
C3 ANY FOUND
NO
YES
D3 TRANSLATE AND MOVE SEGMENT TO LEFT OF I/O AREA IF REQUIRED
E3 STORE CURRENT SHIFT TABLE ADDRESS
F3 CALCULATE REMAINING RECORD LENGTH
G3 ZERO
YES
NO
B4
A3
IJE8EXIT
A4 TRANSLATE AND MOVE SEGMENT TO LEFT OF I/O AREA
B4
B4 EXCHANGE I/O AREAS
C4 ERROR SWITCH ON
NO
YES
G5
D4 USER ERROR ROUTINE
NO
YES
E4
IJE8ENDD
E4 GIVE USER IOREG/RECSIZE INDICATIONS
F4 EXIT TO USER ERROR ROUTINE
B1
G4 WLR
NO
YES
H5
H4 USER WLR ERROR ROUTINE
YES
NO
E4
J4 ERROPT = SKIP
YES
NO
B1
H5
K4 RETURN TO USER
IGN
D5 ERROPT SPECIFIED
SKIP
OMTD
B1
E5 SVC 6 FETCH $$BCNCL
G5
IJE8BRKN
G5 BROKEN TAPE
NO
YES
H5
IJE8EOFL
H5 NOP FIRST READ INSTRUCTION IF REQUIRED
J5 READ IF REQUIRED
K5 GIVE USER IOREG/RECSIZE INDICATIONS

Chart CQ. PTMOD: PUT Macro, No Shifted Code, Device=1018

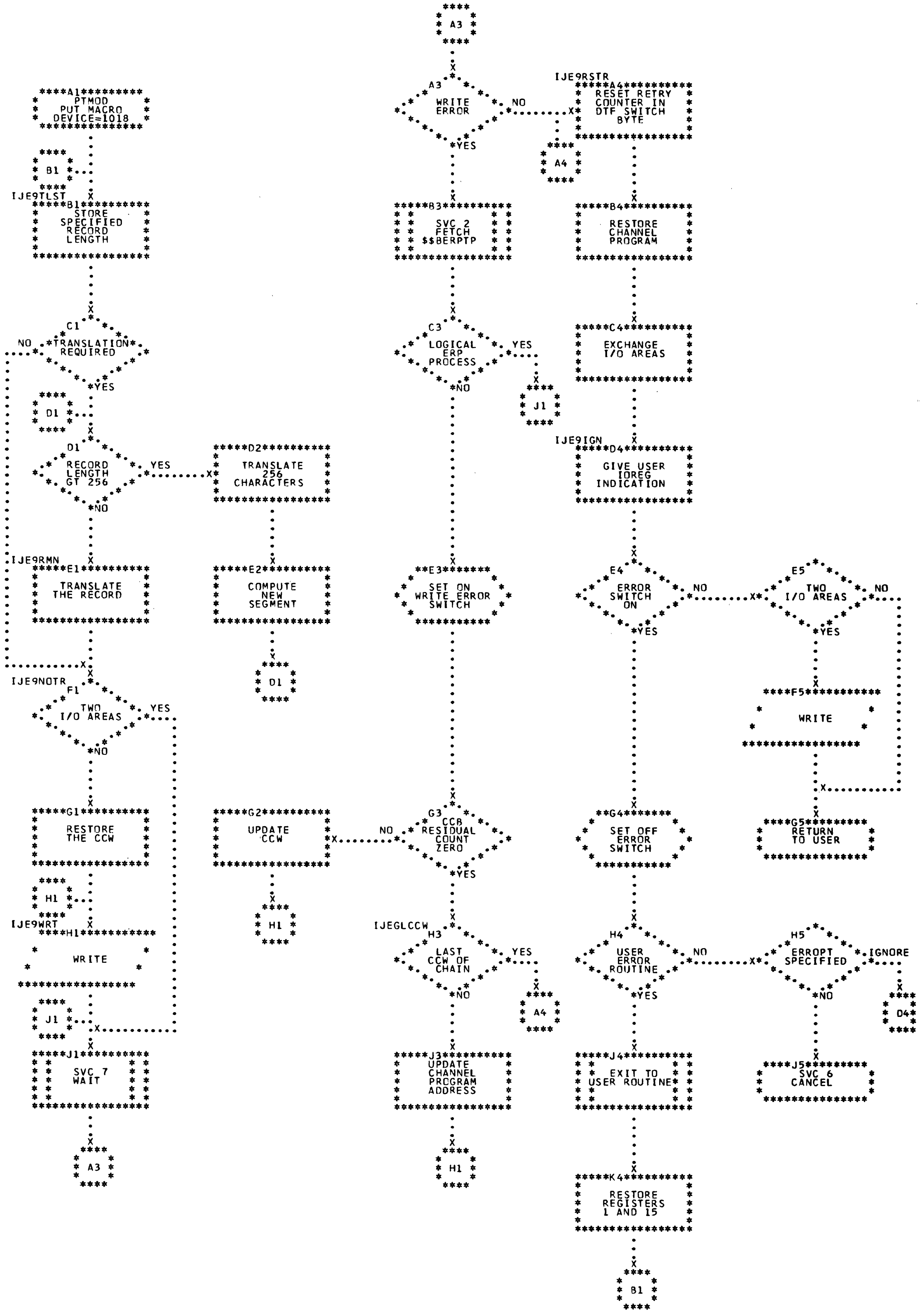


Chart CR. PTMOD: PUT Macro, Shifted Code, Device=1018

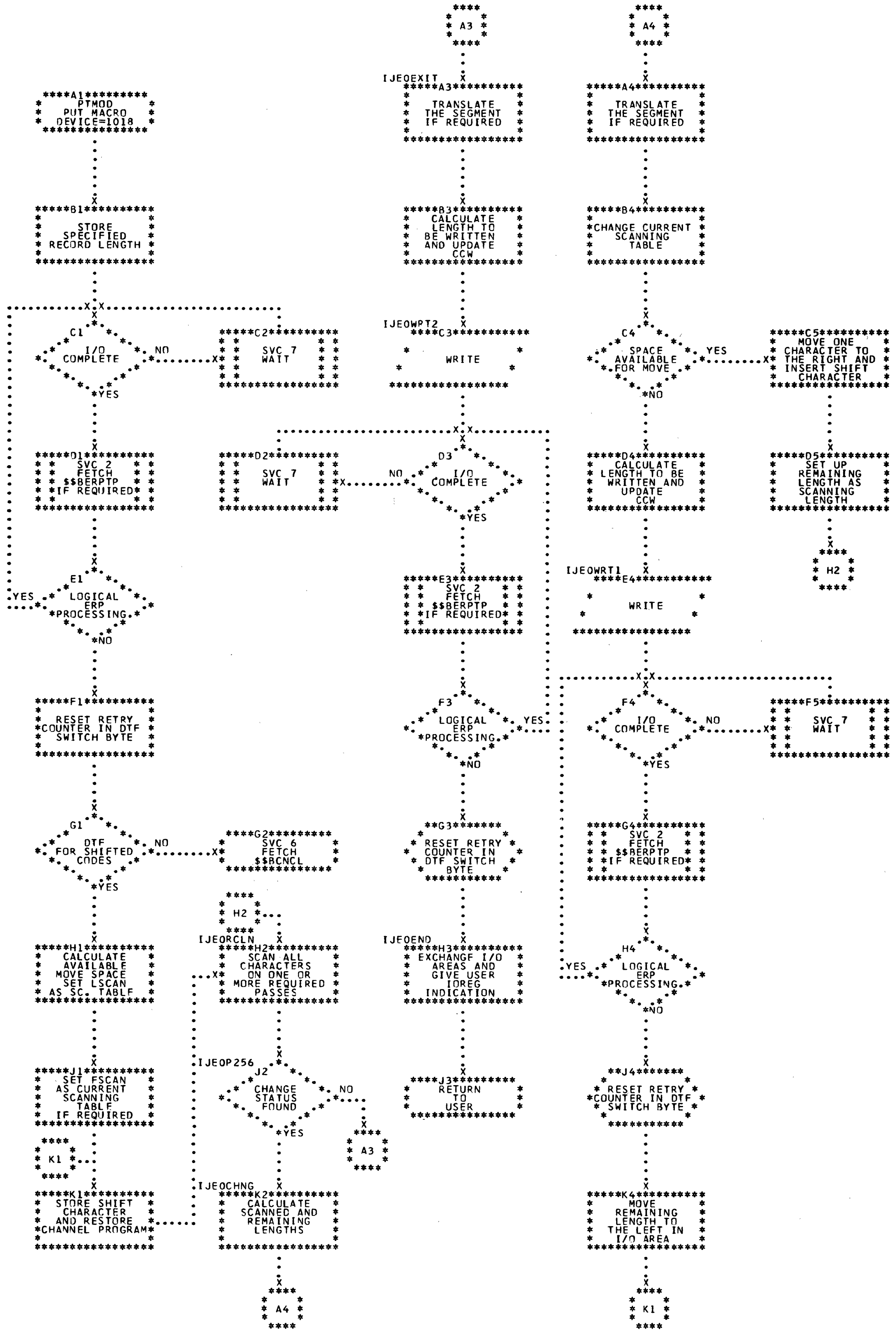

A1 PTMOD PUT MACRO DEVICE=1018
B1 STORE SPECIFIED RECORD LENGTH
C1 I/O COMPLETE
NO
YES
C2 SVC 7 WAIT
D1 SVC 2 FETCH $$BERPTP IF REQUIRED
E1 LOGICAL ERP PROCESSING
YES
NO
F1 RESET RETRY COUNTER IN DTF SWITCH BYTE
G1 DTF FOR SHIFTED CODES
NO
YES
G2 SVC 6 FETCH $$BCNCL
H1 CALCULATE AVAILABLE MOVE SPACE SET LSCAN AS SC. TABLE
J1 SET FSCAN AS CURRENT SCANNING TABLE IF REQUIRED
K1
K1 STORE SHIFT CHARACTER AND RESTORE CHANNEL PROGRAM
D2 SVC 7 WAIT
H2
IJEORCLN
H2 SCAN ALL CHARACTERS ON ONE OR MORE REQUIRED PASSES
IJEOP256
J2 CHANGE STATUS FOUND
NO
YES
A3
IJEOCHNG
K2 CALCULATE SCANNED AND REMAINING LENGTHS
A4
A3
IJEOEXIT
A3 TRANSLATE THE SEGMENT IF REQUIRED
B3 CALCULATE LENGTH TO BE WRITTEN AND UPDATE CCW
IJEOWPT2
C3 WRITE
D3 I/O COMPLETE
NO
YES
E3 SVC 2 FETCH $$BERPTP IF REQUIRED
F3 LOGICAL ERP PROCESSING
YES
NO
G3 RESET RETRY COUNTER IN DTF SWITCH BYTE
IJEOEND
H3 EXCHANGE I/O AREAS AND GIVE USER IOREG INDICATION
J3 RETURN TO USER
A4
A4 TRANSLATE THE SEGMENT IF REQUIRED
B4 CHANGE CURRENT SCANNING TABLE
C4 SPACE AVAILABLE FOR MOVE
YES
NO
C5 MOVE ONE CHARACTER TO THE RIGHT AND INSERT SHIFT CHARACTER
D4 CALCULATE LENGTH TO BE WRITTEN AND UPDATE CCW
D5 SET UP REMAINING LENGTH AS SCANNING LENGTH
H2
IJEOWRT1
E4 WRITE
F4 I/O COMPLETE
NO
YES
F5 SVC 7 WAIT
G4 SVC 2 FETCH $$BERPTP IF REQUIRED
H4 LOGICAL ERP PROCESSING
YES
NO
J4 RESET RETRY COUNTER IN DTF SWITCH BYTE
K4 MOVE REMAINING LENGTH TO THE LEFT IN I/O AREA
K1

Chart DA. MTMOD: CHECK Macro, Work File

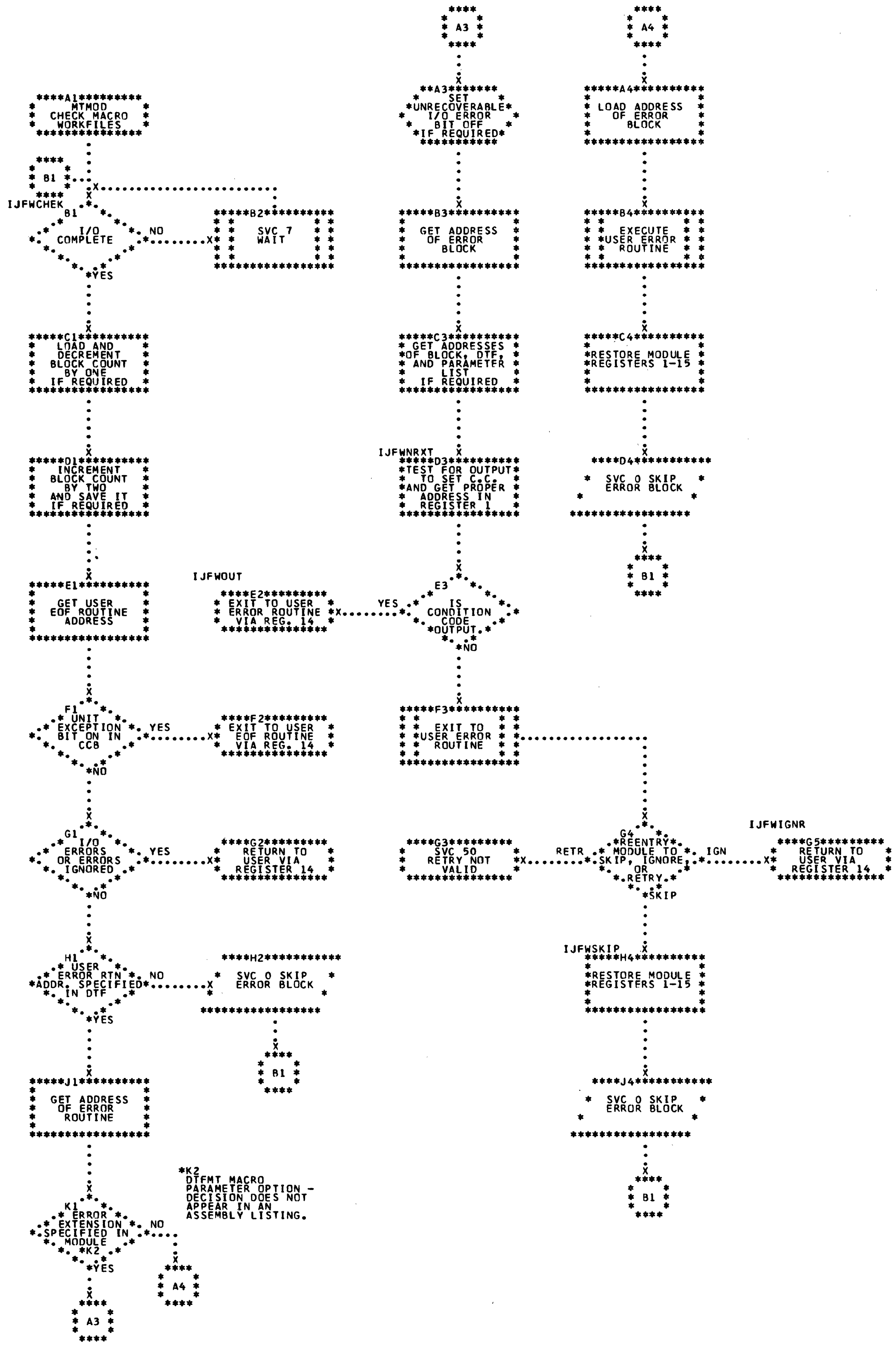
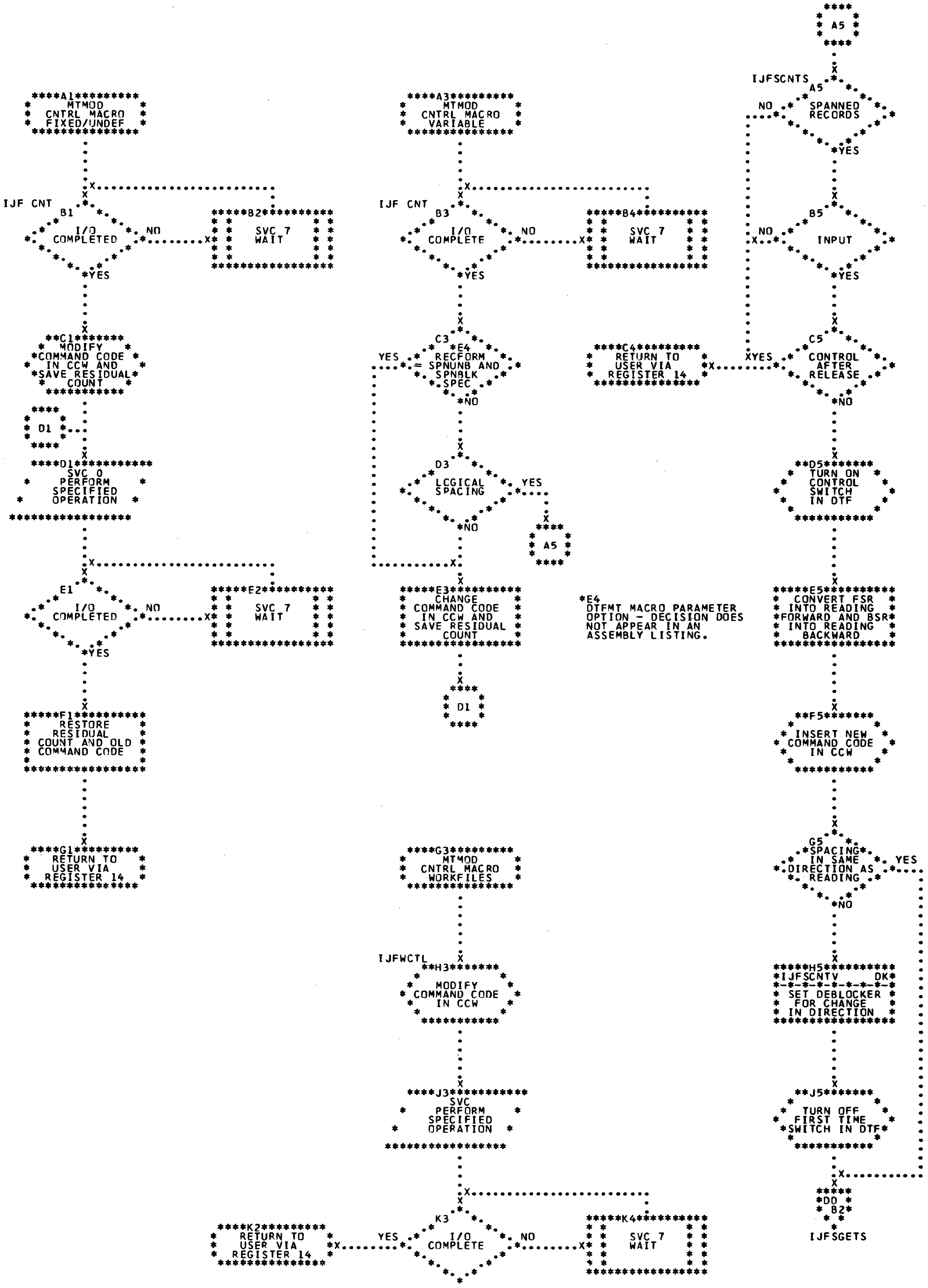

A1
MTMOD CHECK MACRO WORKFILES
B1
IJFWCHEK
B1
I/O COMPLETE
NO
YES
B2
SVC 7 WAIT
C1
LOAD AND DECREMENT BLOCK COUNT BY ONE IF REQUIRED
D1
INCREMENT BLOCK COUNT BY TWO AND SAVE IT IF REQUIRED
E1
GET USER EOF ROUTINE ADDRESS
F1
UNIT EXCEPTION BIT ON IN CCB
YES
NO
F2
EXIT TO USER EOF ROUTINE VIA REG. 14
G1
I/O ERRORS OR ERRORS IGNORED
YES
NO
G2
RETURN TO USER VIA REGISTER 14
H1
USER ERROR RTN ADDR. SPECIFIED IN DTF
NO
YES
H2
SVC 0 SKIP ERROR BLOCK
B1
J1
GET ADDRESS OF ERROR ROUTINE
K1
ERROR EXTENSION SPECIFIED IN MODULE
*K2
NO
YES
A4
A3
*K2
DTFMT MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
A3
A3
SET UNRECOVERABLE I/O ERROR BIT OFF IF REQUIRED
B3
GET ADDRESS OF ERROR BLOCK
C3
GET ADDRESSES OF BLOCK, DTF, AND PARAMETER LIST IF REQUIRED
IJFWNRXT
D3
TEST FOR OUTPUT TO SET C.C. AND GET PROPER ADDRESS IN REGISTER 1
IJFWOUT
E2
EXIT TO USER ERROR ROUTINE VIA REG. 14
YES
E3
IS CONDITION CODE OUTPUT.
NO
F3
EXIT TO USER ERROR ROUTINE
G3
SVC 50 RETRY NOT VALID
RETR
G4
REENTRY MODULE TO SKIP, IGNORE, OR RETRY.
IGN
SKIP
IJFWIGNR
G5
RETURN TO USER VIA REGISTER 14
IJFWSKIP
H4
RESTORE MODULE REGISTERS 1-15
J4
SVC 0 SKIP ERROR BLOCK
B1
A4
A4
LOAD ADDRESS OF ERROR BLOCK
B4
EXECUTE USER ERROR ROUTINE
C4
RESTORE MODULE REGISTERS 1-15
D4
SVC 0 SKIP ERROR BLOCK
B1

Chart DB. MTMOD: CNTRL Macro


A1 MTMOD CNTRL MACRO FIXED/UNDEF
IJF CNT
B1 I/O COMPLETED
NO
YES
B2 SVC 7 WAIT
C1 MODIFY COMMAND CODE IN CCW AND SAVE RESIDUAL COUNT
D1
D1 SVC 0 PERFORM SPECIFIED OPERATION
E1 I/O COMPLETED
NO
YES
E2 SVC 7 WAIT
F1 RESTORE RESIDUAL COUNT AND OLD COMMAND CODE
G1 RETURN TO USER VIA REGISTER 14
A3 MTMOD CNTRL MACRO VARIABLE
IJF CNT
B3 I/O COMPLETE
NO
YES
B4 SVC 7 WAIT
C3 *E4 RECFORM = SPNUNB AND SPNBLK SPEC
YES
NO
D3 LOGICAL SPACING
YES
A5
E3 CHANGE COMMAND CODE IN CCW AND SAVE RESIDUAL COUNT
D1
G3 MTMOD CNTRL MACRO WORKFILES
IJFWCTL
H3 MODIFY COMMAND CODE IN CCW
J3 SVC PERFORM SPECIFIED OPERATION
K2 RETURN TO USER VIA REGISTER 14
K3 I/O COMPLETE
YES
NO
K4 SVC 7 WAIT
A5
IJFSCNTS
A5 SPANNED RECORDS
NO
YES
B5 INPUT
NO
YES
C4 RETURN TO USER VIA REGISTER 14
C5 CONTROL AFTER RELEASE
YES
NO
D5 TURN ON CONTROL SWITCH IN DTF
E5 CONVERT FSR INTO READING FORWARD AND BSR INTO READING BACKWARD
F5 INSERT NEW COMMAND CODE IN CCW
G5 SPACING IN SAME DIRECTION AS READING
YES
NO
H5 IJFSCNTV DK SET DEBLOCKER FOR CHANGE IN DIRECTION
J5 TURN OFF FIRST TIME SWITCH IN DTF
DD B2
IJFSGETS

*E4 DTFMT MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.

Chart DC. MTMOD: FEOV Macro

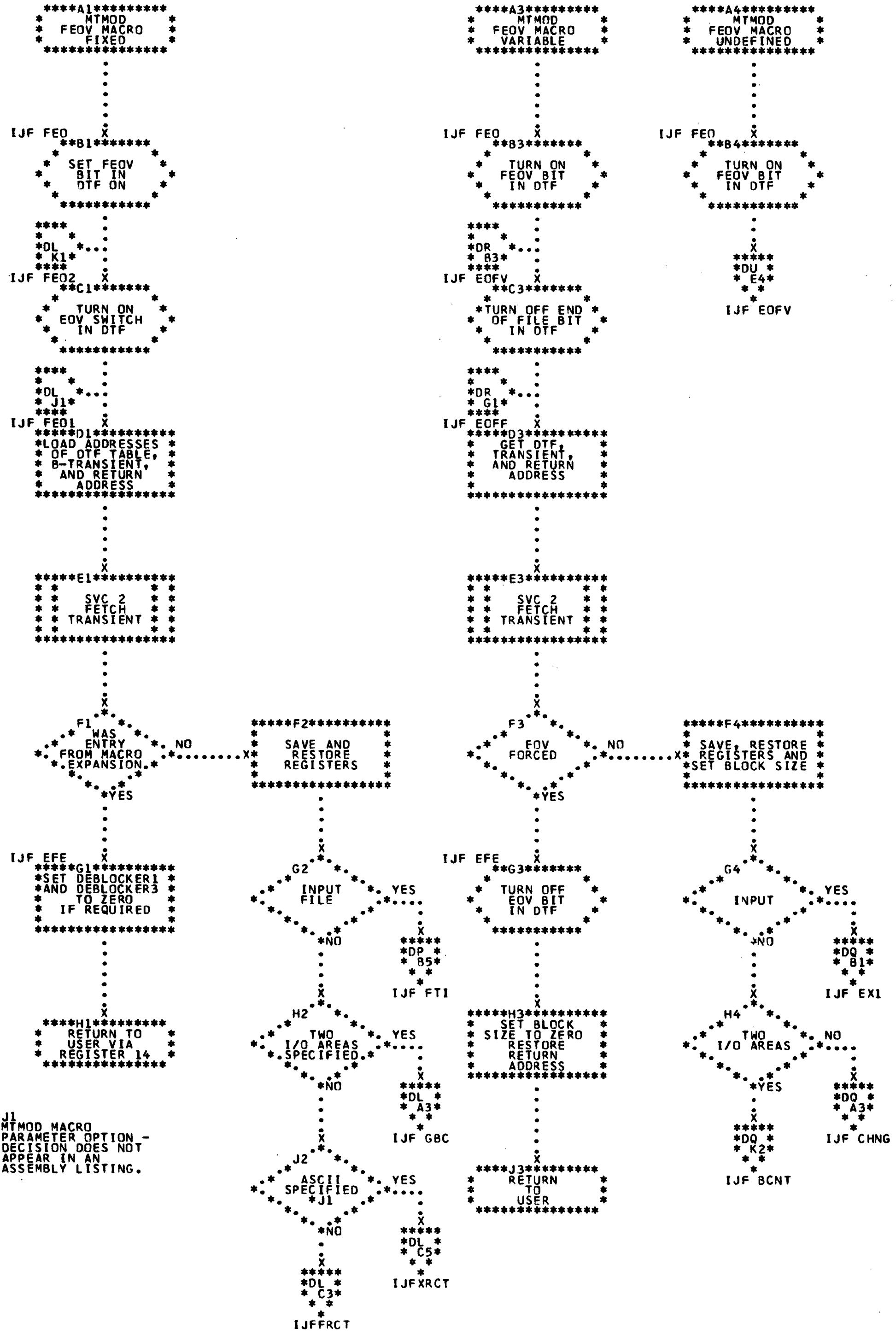

A1 MTMOD FEOV MACRO FIXED
IJF FEO
B1 SET FEOV BIT IN DTF ON
DL K1
IJF FEO2
C1 TURN ON EOV SWITCH IN DTF
DL J1
IJF FE01
D1 LOAD ADDRESSES OF DTF TABLE, B-TRANSIENT, AND RETURN ADDRESS
E1 SVC 2 FETCH TRANSIENT
F1 WAS ENTRY FROM MACRO EXPANSION
NO
YES
IJF EFE
G1 SET DEBLOCKER1 AND DEBLOCKER3 TO ZERO IF REQUIRED
H1 RETURN TO USER VIA REGISTER 14
*J1 MTMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
F2 SAVE AND RESTORE REGISTERS
G2 INPUT FILE
YES
NO
DP B5
IJF FTI
H2 TWO I/O AREAS SPECIFIED
YES
NO
DL A3
IJF GBC
J2 ASCII SPECIFIED *J1
YES
NO
DL C5
IJFXRCT
DL C3
IJFFRCT
A3 MTMOD FEOV MACRO VARIABLE
IJF FEO
B3 TURN ON FEOV BIT IN DTF
DR B3
IJF EOFV
C3 TURN OFF END OF FILE BIT IN DTF
DR G1
IJF EOFF
D3 GET DTF, TRANSIENT, AND RETURN ADDRESS
E3 SVC 2 FETCH TRANSIENT
F3 EOV FORCED
NO
YES
IJF EFE
G3 TURN OFF EOV BIT IN DTF
H3 SET BLOCK SIZE TO ZERO RESTORE RETURN ADDRESS
J3 RETURN TO USER
A4 MTMOD FEOV MACRO UNDEFINED
IJF FEO
B4 TURN ON FEOV BIT IN DTF
DU E4
IJF EOFV
F4 SAVE, RESTORE REGISTERS AND SET BLOCK SIZE
G4 INPUT
YES
NO
DQ B1
IJF EX1
H4 TWO I/O AREAS
NO
YES
DQ A3
IJF CHNG
DQ K2
IJF BCNT

Chart DD. MTMOD: GET Macro

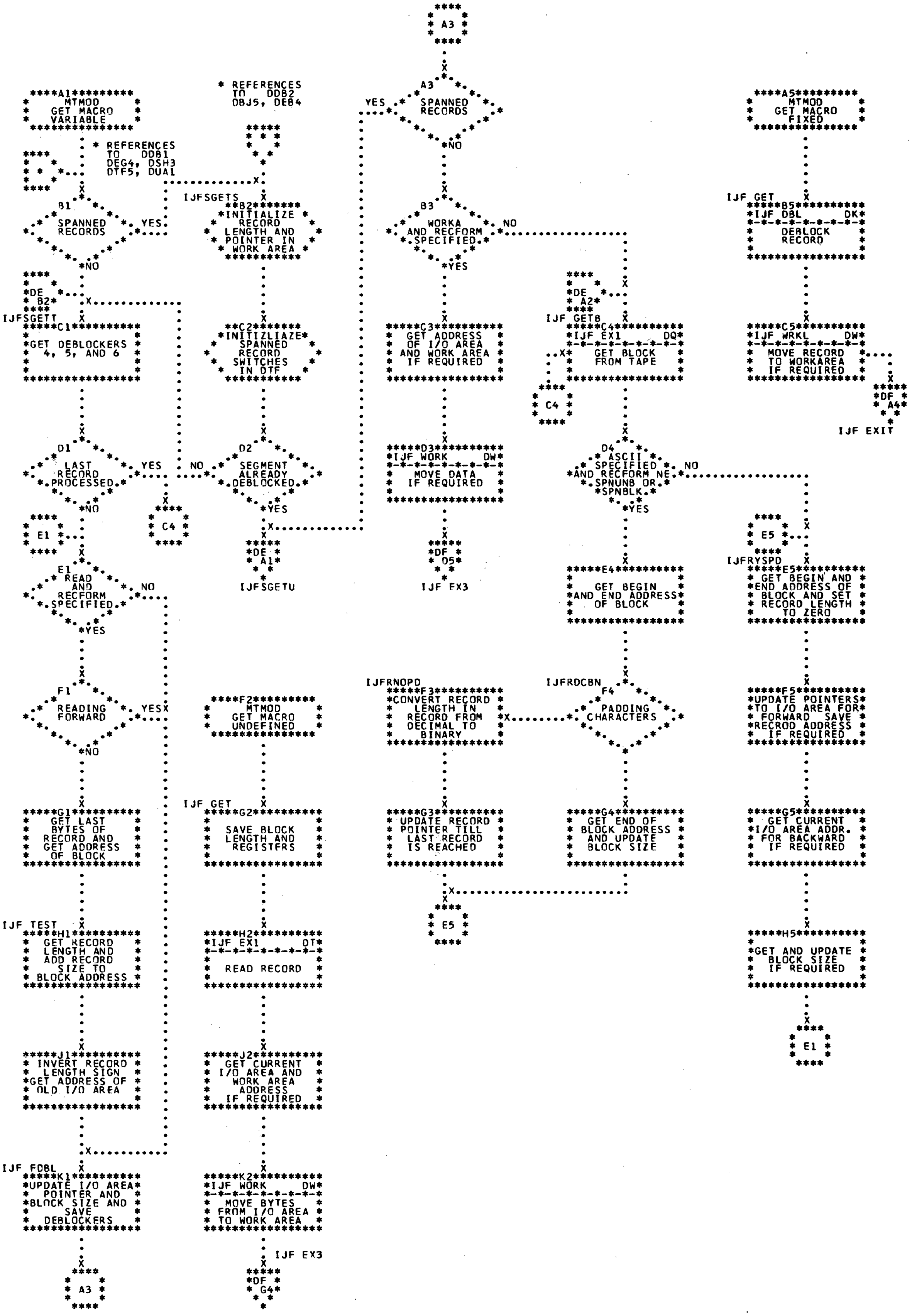

A1 MTMOD GET MACRO VARIABLE
* REFERENCES TO DDB1 DEG4, DSH3 DTF5, DUA1
B1 SPANNED RECORDS
YES
NO
DE B2
IJFSGETT
C1 GET DEBLOCKERS 4, 5, AND 6
D1 LAST RECORD PROCESSED
YES
NO
C4
E1
E1 READ AND RECFORM SPECIFIED
NO
YES
F1 READING FORWARD
YES
NO
G1 GET LAST BYTES OF RECORD AND GET ADDRESS OF BLOCK
IJF TEST
H1 GET RECORD LENGTH AND ADD RECORD SIZE TO BLOCK ADDRESS
J1 INVERT RECORD LENGTH SIGN GET ADDRESS OF OLD I/O AREA
IJF FDBL
K1 UPDATE I/O AREA POINTER AND BLOCK SIZE AND SAVE DEBLOCKERS
A3
* REFERENCES TO DDB2 DBJ5, DEB4
IJFSGETS
B2 INITIALIZE RECORD LENGTH AND POINTER IN WORK AREA
C2 INITIZLIAZE SPANNED RECORD SWITCHES IN DTF
D2 SEGMENT ALREADY DEBLOCKED
NO
YES
DE A1
IJFSGETU
F2 MTMOD GET MACRO UNDEFINED
IJF GET
G2 SAVE BLOCK LENGTH AND REGISTERS
H2 IJF EX1 DT
READ RECORD
J2 GET CURRENT I/O AREA AND WORK AREA ADDRESS IF REQUIRED
K2 IJF WORK DW
MOVE BYTES FROM I/O AREA TO WORK AREA
IJF EX3
DF G4
A3
A3 SPANNED RECORDS
YES
NO
B3 WORKA AND RECFORM SPECIFIED
NO
YES
C3 GET ADDRESS OF I/O AREA AND WORK AREA IF REQUIRED
D3 IJF WORK DW
MOVE DATA IF REQUIRED
DF D5
IJF EX3
IJFRNOPD
F3 CONVERT RECORD LENGTH IN RECORD FROM DECIMAL TO BINARY
G3 UPDATE RECORD POINTER TILL LAST RECORD IS REACHED
E5
DE A2
IJF GETB
C4 IJF EX1 DQ
GET BLOCK FROM TAPE
C4
D4 ASCII SPECIFIED AND RECFORM NE SPNUNB OR SPNBLK
NO
YES
E4 GET BEGIN AND END ADDRESS OF BLOCK
IJFRDCBN
F4 PADDING CHARACTERS
G4 GET END OF BLOCK ADDRESS AND UPDATE BLOCK SIZE
A5 MTMOD GET MACRO FIXED
IJF GET
B5 IJF DBL DK
DEBLOCK RECORD
C5 IJF WRKL DW
MOVE RECORD TO WORKAREA IF REQUIRED
DF A4
IJF EXIT
E5
IJFRYSPD
E5 GET BEGIN AND END ADDRESS OF BLOCK AND SET RECORD LENGTH TO ZERO
F5 UPDATE POINTERS TO I/O AREA FOR FORWARD SAVE RECROD ADDRESS IF REQUIRED
G5 GET CURRENT I/O AREA ADDR. FOR BACKWARD IF REQUIRED
H5 GET AND UPDATE BLOCK SIZE IF REQUIRED
E1

Chart DE. MTMCD: GET Macro, Spanned Records Routines

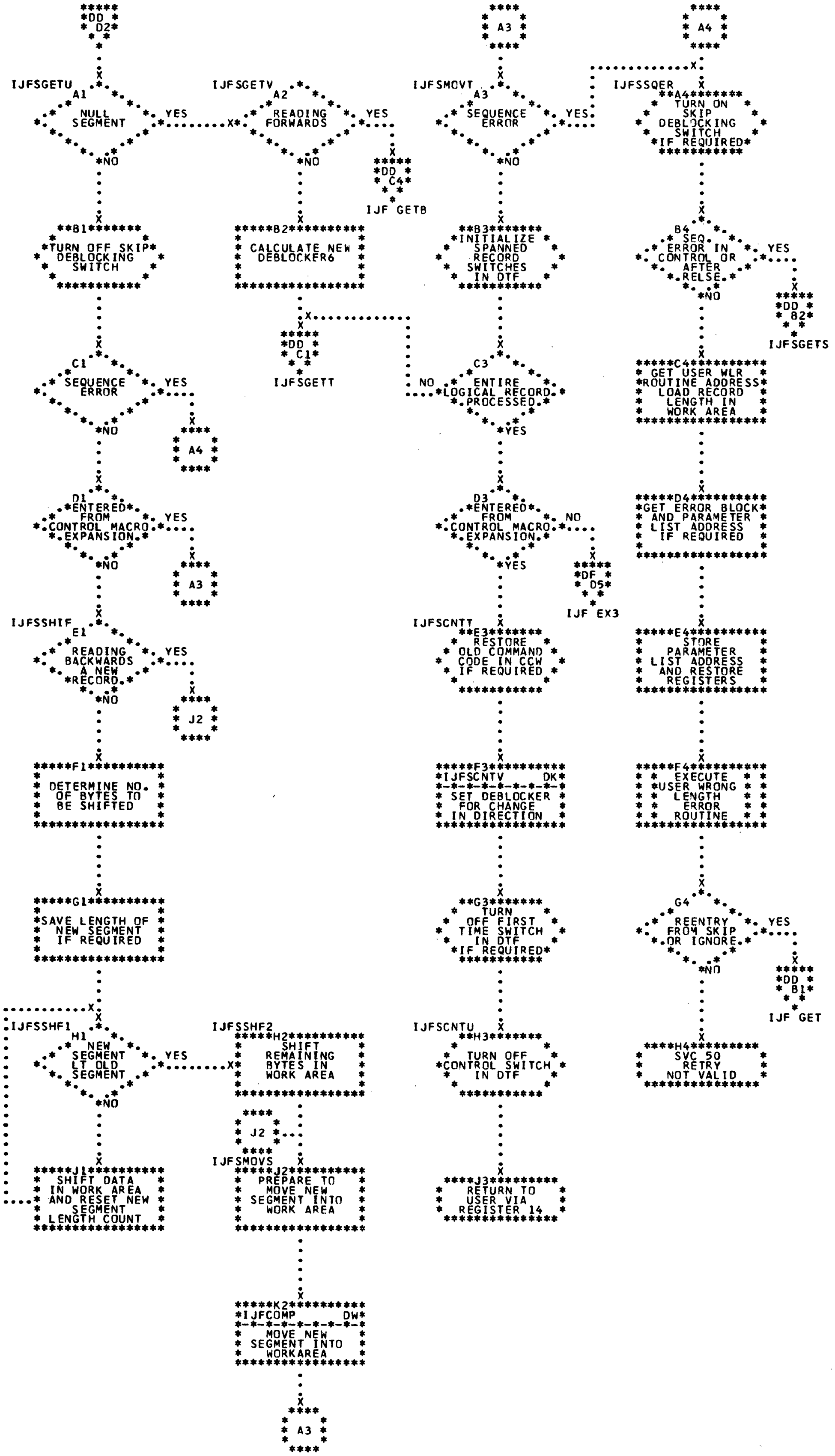

DD D2
IJFSGETU A1 NULL SEGMENT YES NO
IJFSGETV A2 READING FORWARDS YES NO
DD C4 IJF GETB
B1 TURN OFF SKIP DEBLOCKING SWITCH
B2 CALCULATE NEW DEBLOCKER6
DD C1 IJFSGETT
C1 SEQUENCE ERROR YES NO
A4
D1 ENTERED FROM CONTROL MACRO EXPANSION YES NO
A3
IJFSSHIF E1 READING BACKWARDS A NEW RECORD YES NO
J2
F1 DETERMINE NO. OF BYTES TO BE SHIFTED
G1 SAVE LENGTH OF NEW SEGMENT IF REQUIRED
IJFSSHF1 H1 NEW SEGMENT LT OLD SEGMENT YES NO
IJFSSHF2 H2 SHIFT REMAINING BYTES IN WORK AREA
J2
J1 SHIFT DATA IN WORK AREA AND RESET NEW SEGMENT LENGTH COUNT
IJFSMOVS J2 PREPARE TO MOVE NEW SEGMENT INTO WORK AREA
K2 IJFCOMP DW MOVE NEW SEGMENT INTO WORKAREA
A3
A3
IJFSMOVT A3 SEQUENCE ERROR YES NO
B3 INITIALIZE SPANNED RECORD SWITCHES IN DTF
C3 ENTIRE LOGICAL RECORD PROCESSED NO YES
D3 ENTERED FROM CONTROL MACRO EXPANSION NO YES
DF D5 IJF EX3
IJFSCNTT E3 RESTORE OLD COMMAND CODE IN CCW IF REQUIRED
F3 IJFSCNTV DK SET DEBLOCKER FOR CHANGE IN DIRECTION
G3 TURN OFF FIRST TIME SWITCH IN DTF IF REQUIRED
IJFSCNTU H3 TURN OFF CONTROL SWITCH IN DTF
J3 RETURN TO USER VIA REGISTER 14
A4
IJFSSQER A4 TURN ON SKIP DEBLOCKING SWITCH IF REQUIRED
B4 SEQ. ERROR IN CONTROL OR AFTER RELSE YES NO
DD B2 IJFSGETS
C4 GET USER WLR ROUTINE ADDRESS LOAD RECORD LENGTH IN WORK AREA
D4 GET ERROR BLOCK AND PARAMETER LIST ADDRESS IF REQUIRED
E4 STORE PARAMETER LIST ADDRESS AND RESTORE REGISTERS
F4 EXECUTE USER WRONG LENGTH ERROR ROUTINE
G4 REENTRY FROM SKIP OR IGNORE YES NO
DD B1 IJF GET
H4 SVC 50 RETRY NOT VALID

Chart DF. MTMOD: PCINTR, POINTW, POINTS, and NOTE Macros, Work Files, and GET/PUT Macrcs Commcn Routines

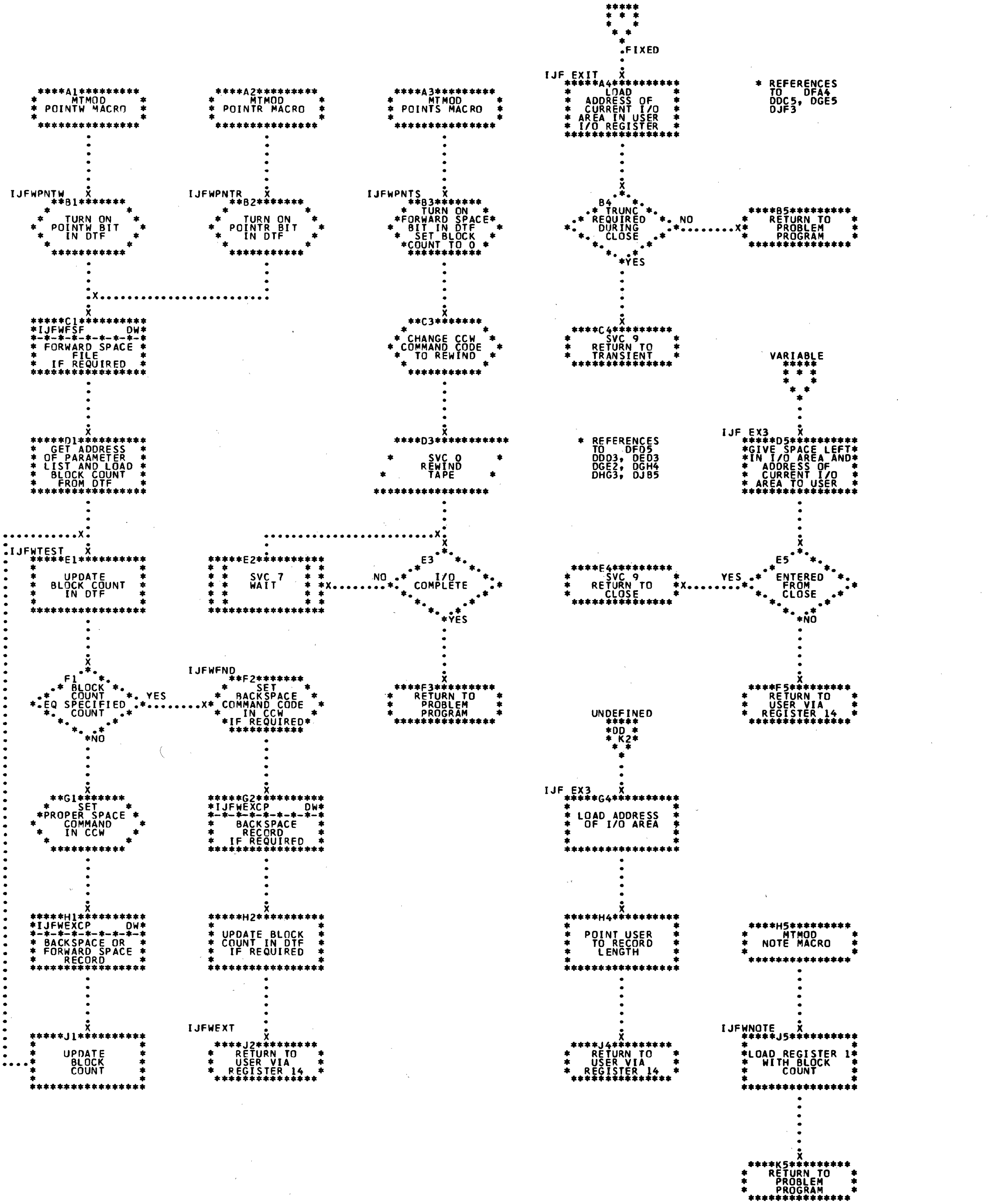


Chart DG. MTMOD: PUT Macro

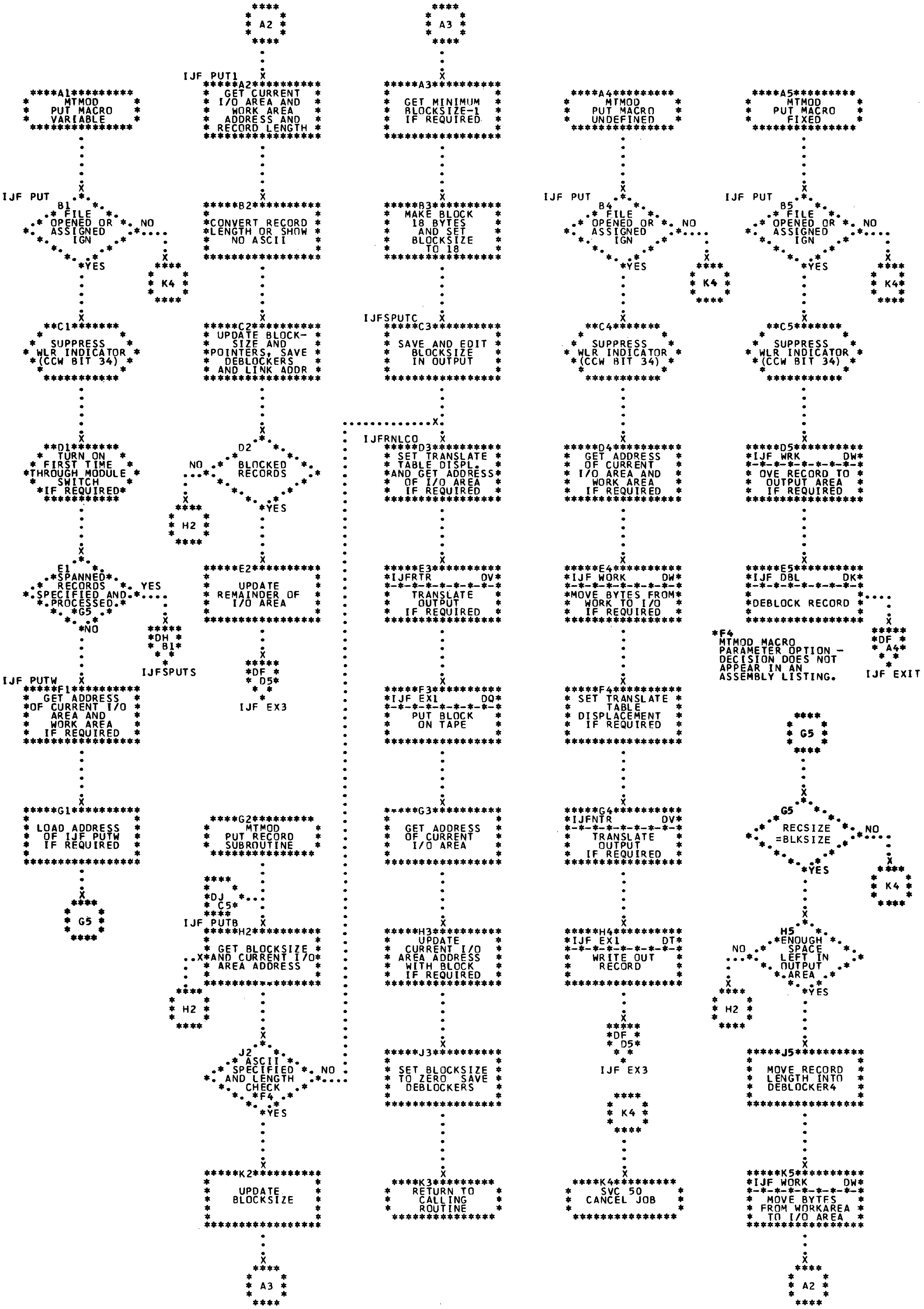

A2
A3
A1 MTMOD PUT MACRO VARIABLE
IJF PUT1
A2 GET CURRENT I/O AREA AND WORK AREA ADDRESS AND RECORD LENGTH
A3 GET MINIMUM BLOCKSIZE-1 IF REQUIRED
A4 MTMOD PUT MACRO UNDEFINED
A5 MTMOD PUT MACRO FIXED
IJF PUT
B1 FILE OPENED OR ASSIGNED IGN
NO
YES
K4
B2 CONVERT RECORD LENGTH OR SHOW NO ASCII
B3 MAKE BLOCK 18 BYTES AND SET BLOCKSIZE TO 18
IJF PUT
B4 FILE OPENED OR ASSIGNED IGN
NO
YES
K4
IJF PUT
B5 FILE OPENED OR ASSIGNED IGN
NO
YES
K4
C1 SUPPRESS WLR INDICATOR (CCW BIT 34)
C2 UPDATE BLOCK-SIZE AND POINTERS, SAVE DEBLOCKERS AND LINK ADDR
IJFSPUTC
C3 SAVE AND EDIT BLOCKSIZE IN OUTPUT
C4 SUPPRESS WLR INDICATOR (CCW BIT 34)
C5 SUPPRESS WLR INDICATOR (CCW BIT 34)
D1 TURN ON FIRST TIME THROUGH MODULE SWITCH IF REQUIRED
D2 BLOCKED RECORDS
NO
YES
H2
IJFRNLCO
D3 SET TRANSLATE TABLE DISPL. AND GET ADDRESS OF I/O AREA IF REQUIRED
D4 GET ADDRESS OF CURRENT I/O AREA AND WORK AREA IF REQUIRED
D5 IJF WRK DW
OVE RECORD TO OUTPUT AREA IF REQUIRED
E1 SPANNED RECORDS SPECIFIED AND PROCESSED. G5
YES
NO
DH B1
IJFSPUTS
E2 UPDATE REMAINDER OF I/O AREA
DF D5
IJF EX3
E3 IJFRTR DV
TRANSLATE OUTPUT IF REQUIRED
E4 IJF WORK DW
MOVE BYTES FROM WORK TO I/O IF REQUIRED
E5 IJF DBL DK
DEBLOCK RECORD
*F4 MTMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
DF A4
IJF EXIT
IJF PUTW
F1 GET ADDRESS OF CURRENT I/O AREA AND WORK AREA IF REQUIRED
F3 IJF EX1 DQ
PUT BLOCK ON TAPE
F4 SET TRANSLATE TABLE DISPLACEMENT IF REQUIRED
G5
G1 LOAD ADDRESS OF IJF PUTW IF REQUIRED
G2 MTMOD PUT RECORD SUBROUTINE
G3 GET ADDRESS OF CURRENT I/O AREA
G4 IJFNTR DV
TRANSLATE OUTPUT IF REQUIRED
G5 RECSIZE =BLKSIZE
NO
YES
K4
G5
DJ C5
IJF PUTB
H2 GET BLOCKSIZE AND CURRENT I/O AREA ADDRESS
H2
H3 UPDATE CURRENT I/O AREA ADDRESS WITH BLOCK IF REQUIRED
H4 IJF EX1 DT
WRITE OUT RECORD
H5 ENOUGH SPACE LEFT IN OUTPUT AREA
NO
YES
H2
DF D5
IJF EX3
J2 ASCII SPECIFIED AND LENGTH CHECK F4
NO
YES
J3 SET BLOCKSIZE TO ZERO SAVE DEBLOCKERS
K4
J5 MOVE RECORD LENGTH INTO DEBLOCKER4
K2 UPDATE BLOCKSIZE
K3 RETURN TO CALLING ROUTINE
K4 SVC 50 CANCEL JOB
K5 IJF WORK DW
MOVE BYTES FROM WORKAREA TO I/O AREA
A3
A2

Chart DH. MIMOD: PUT Macro, Spanned Records Routine

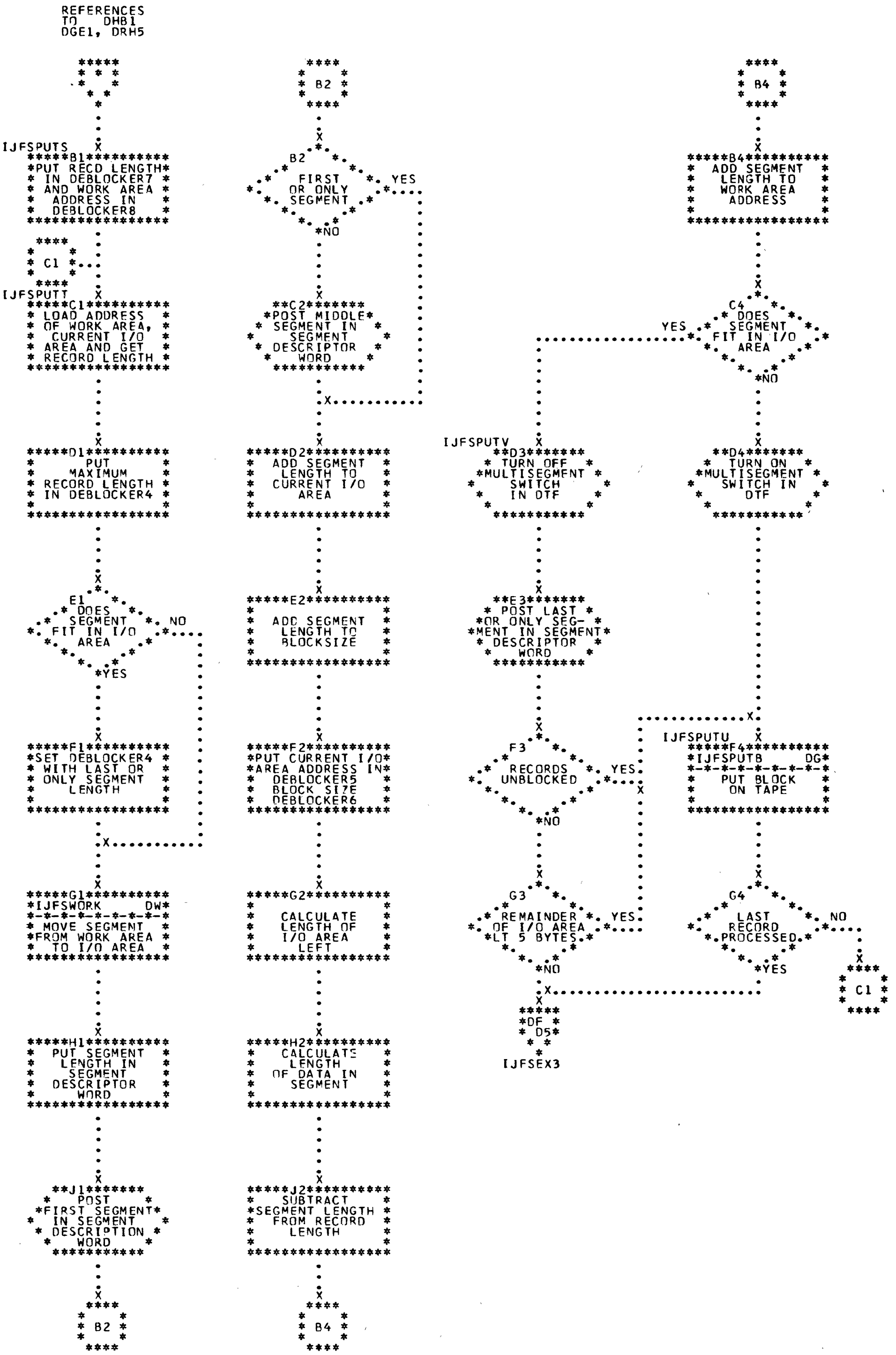

REFERENCES TO DHB1 DGE1, DRH5
IJFSPUTS
B1
PUT RECD LENGTH IN DEBLOCKER7 AND WORK AREA ADDRESS IN DEBLOCKER8
C1
IJFSPUTT
C1
LOAD ADDRESS OF WORK AREA, CURRENT I/O AREA AND GET RECORD LENGTH
D1
PUT MAXIMUM RECORD LENGTH IN DEBLOCKER4
E1
DOES SEGMENT FIT IN I/O AREA
NO
YES
F1
SET DEBLOCKER4 WITH LAST OR ONLY SEGMENT LENGTH
G1
IJFSWORK DW
MOVE SEGMENT FROM WORK AREA TO I/O AREA
H1
PUT SEGMENT LENGTH IN SEGMENT DESCRIPTOR WORD
J1
POST FIRST SEGMENT IN SEGMENT DESCRIPTION WORD
B2
B2
FIRST OR ONLY SEGMENT
YES
NO
C2
POST MIDDLE SEGMENT IN SEGMENT DESCRIPTOR WORD
D2
ADD SEGMENT LENGTH TO CURRENT I/O AREA
E2
ADD SEGMENT LENGTH TO BLOCKSIZE
F2
PUT CURRENT I/O AREA ADDRESS IN DEBLOCKER5 BLOCK SIZE DEBLOCKER6
G2
CALCULATE LENGTH OF I/O AREA LEFT
H2
CALCULATS LENGTH OF DATA IN SEGMENT
J2
SUBTRACT SEGMENT LENGTH FROM RECORD LENGTH
B4
B4
ADD SEGMENT LENGTH TO WORK AREA ADDRESS
C4
DOES SEGMENT FIT IN I/O AREA
YES
NO
IJFSPUTV
D3
TURN OFF MULTISEGMENT SWITCH IN DTF
D4
TURN ON MULTISEGMENT SWITCH IN DTF
E3
POST LAST OR ONLY SEG- MENT IN SEGMENT DESCRIPTOR WORD
F3
RECORDS UNBLOCKED
YES
NO
IJFSPUTU
F4
IJFSPUTB DG
PUT BLOCK ON TAPE
G3
REMAINDER OF I/O AREA LT 5 BYTES
YES
NO
G4
LAST RECORD PROCESSED
NO
YES
C1
DF D5
IJFSEX3
B2
B4

Chart DJ. MTMOD: READ and WRITE Macros, Work Files, and RELSE and TRUNC Macros

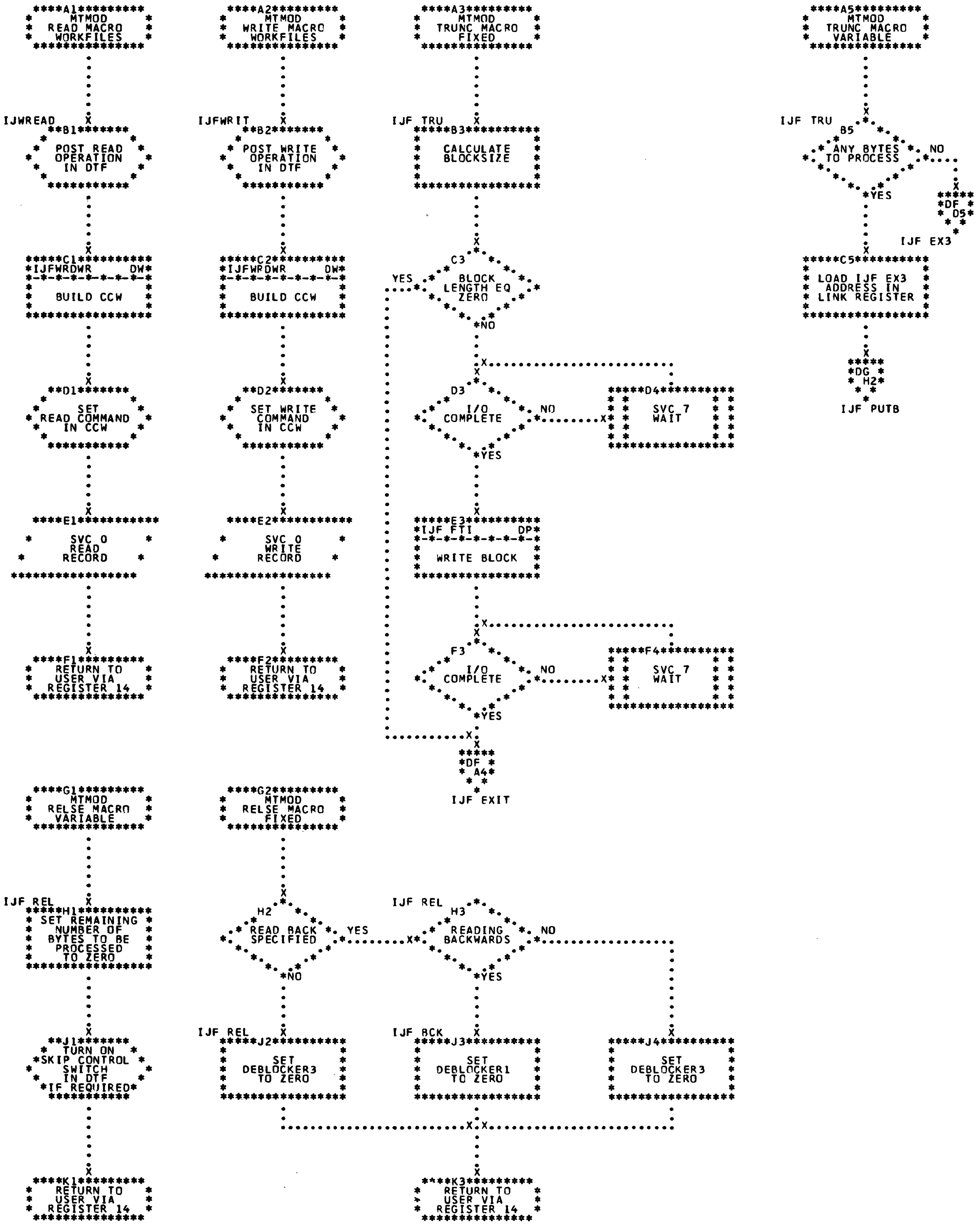


Chart DK. MTMOD: Logical Spacing and EOV Spanned /Records Routines, Deblocking Routines, and Translate Subroutine

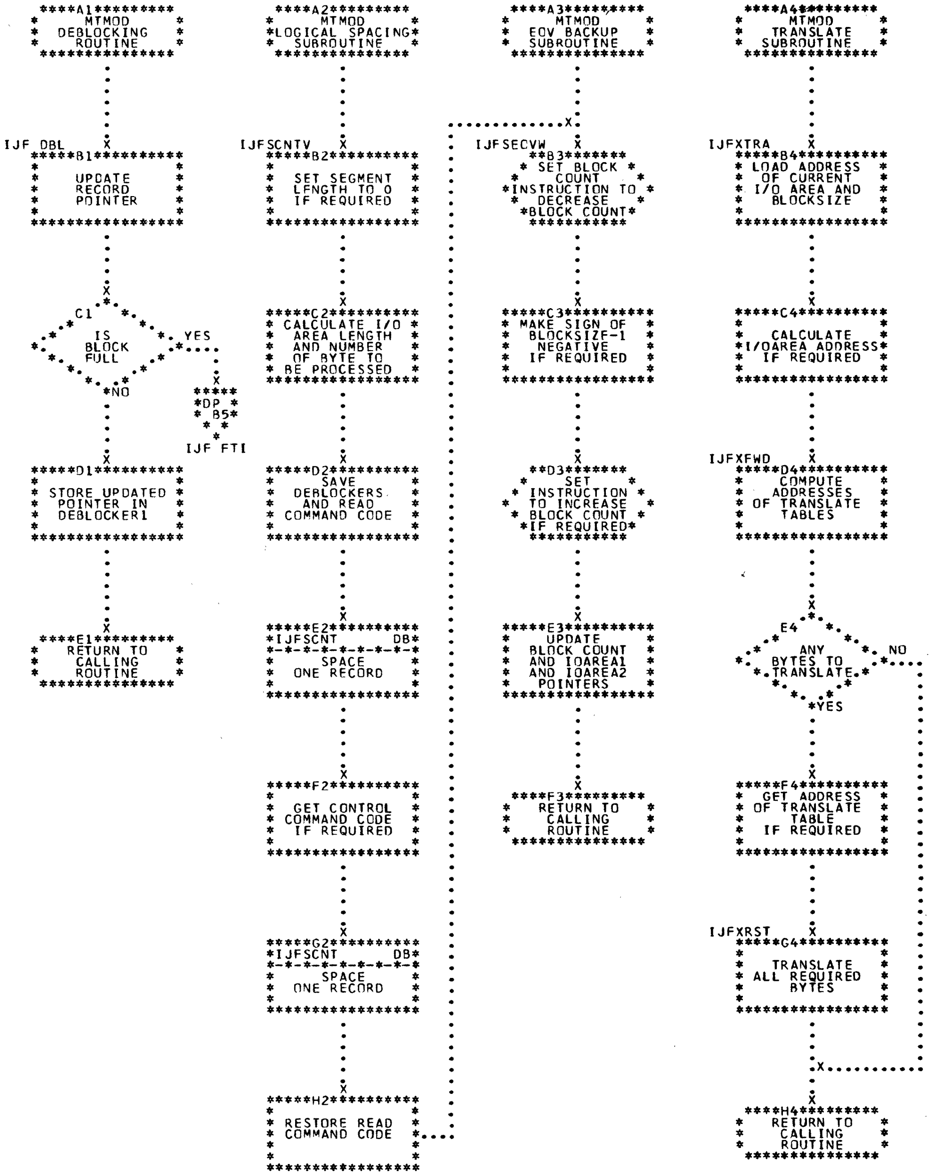

A1
MTMOD DEBLOCKING ROUTINE
IJF DBL
B1
UPDATE RECORD POINTER
C1
IS BLOCK FULL
YES
NO
DP B5
IJF FTI
D1
STORE UPDATED POINTER IN DEBLOCKER1
E1
RETURN TO CALLING ROUTINE
A2
MTMOD LOGICAL SPACING SUBROUTINE
IJFSCNTV
B2
SET SEGMENT LENGTH TO 0 IF REQUIRED
C2
CALCULATE I/O AREA LENGTH AND NUMBER OF BYTE TO BE PROCESSED
D2
SAVE DEBLOCKERS AND READ COMMAND CODE
E2
IJFSCNT DB
SPACE ONE RECORD
F2
GET CONTROL COMMAND CODE IF REQUIRED
G2
IJFSCNT DB
SPACE ONE RECORD
H2
RESTORE READ COMMAND CODE
A3
MTMOD EOV BACKUP SUBROUTINE
IJFSECVW
B3
SET BLOCK COUNT INSTRUCTION TO DECREASE BLOCK COUNT
C3
MAKE SIGN OF BLOCKSIZE-1 NEGATIVE IF REQUIRED
D3
SET INSTRUCTION TO INCREASE BLOCK COUNT IF REQUIRED
E3
UPDATE BLOCK COUNT AND IOAREA1 AND IOAREA2 POINTERS
F3
RETURN TO CALLING ROUTINE
A4
MTMOD TRANSLATE SUBROUTINE
IJFXTRA
B4
LOAD ADDRESS OF CURRENT I/O AREA AND BLOCKSIZE
C4
CALCULATE I/OAREA ADDRESS IF REQUIRED
IJFXFWD
D4
COMPUTE ADDRESSES OF TRANSLATE TABLES
E4
ANY BYTES TO TRANSLATE
NO
YES
F4
GET ADDRESS OF TRANSLATE TABLE IF REQUIRED
IJFXRST
G4
TRANSLATE ALL REQUIRED BYTES
H4
RETURN TO CALLING ROUTINE

Chart DL. MTMOD: Read/Write Subroutine, Fixed Length Records (Part 1 of 2)

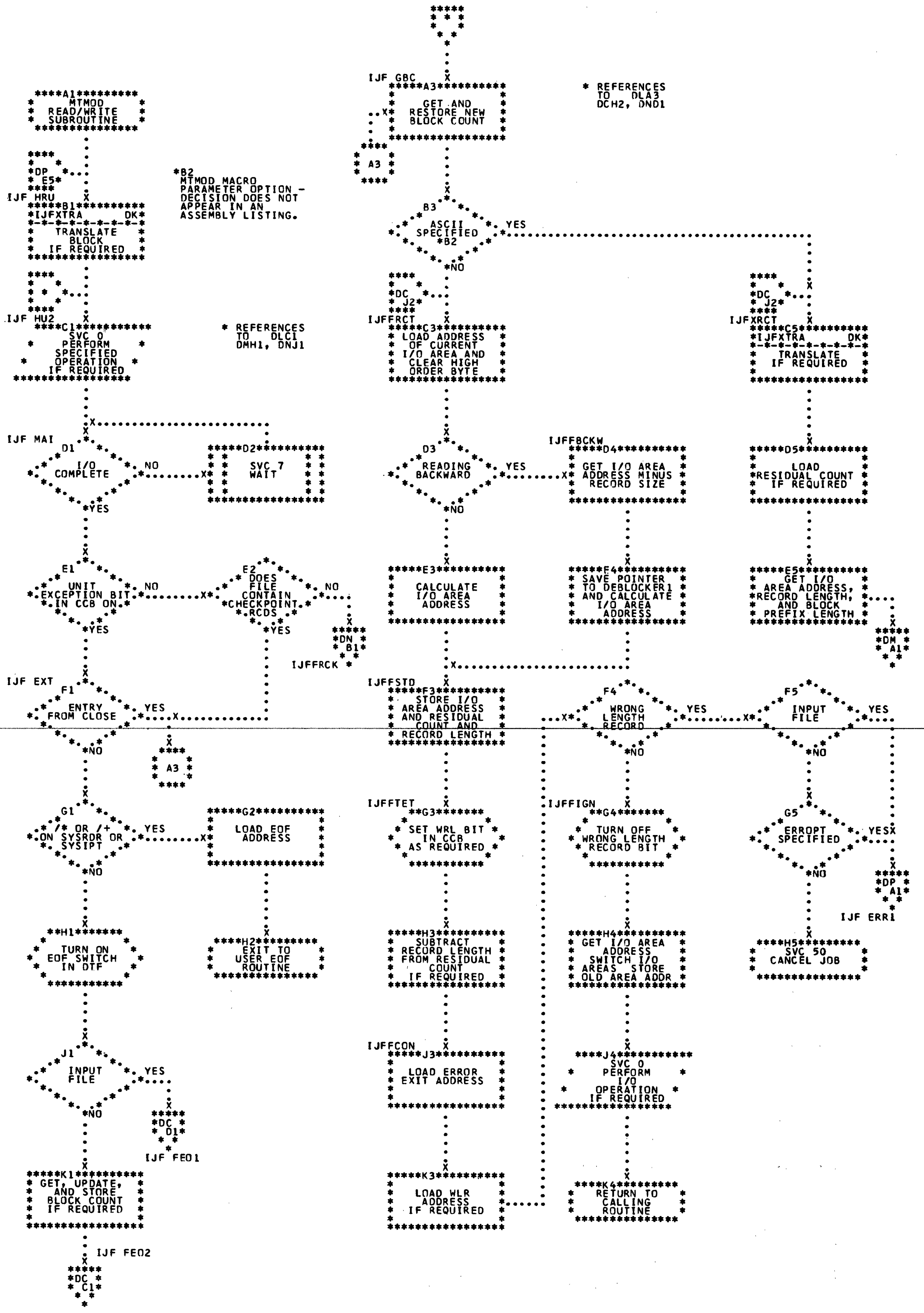

A1 MTMOD READ/WRITE SUBROUTINE
DP E5
IJF HRU
B1 IJFXTRA DK TRANSLATE BLOCK IF REQUIRED
*B2 MTMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
IJF HU2
C1 SVC 0 PERFORM SPECIFIED OPERATION IF REQUIRED
* REFERENCES TO DLC1 DMH1, DNJ1
IJF MAI
D1 I/O COMPLETE
NO
YES
D2 SVC 7 WAIT
E1 UNIT EXCEPTION BIT IN CCB ON
NO
YES
E2 DOES FILE CONTAIN CHECKPOINT RCDS
NO
YES
DN B1
IJFFRCK
IJF EXT
F1 ENTRY FROM CLOSE
YES
NO
A3
G1 /* OR /+ ON SYSRDR OR SYSIPT
YES
NO
G2 LOAD EOF ADDRESS
H1 TURN ON EOF SWITCH IN DTF
H2 EXIT TO USER EOF ROUTINE
J1 INPUT FILE
YES
NO
DC D1
IJF FE01
K1 GET, UPDATE, AND STORE BLOCK COUNT IF REQUIRED
IJF FE02
DC C1
IJF GBC
A3 GET AND RESTORE NEW BLOCK COUNT
A3
* REFERENCES TO DLA3 DCH2, DND1
B3 ASCII SPECIFIED *B2
YES
NO
DC J2
IJFFRCT
C3 LOAD ADDRESS OF CURRENT I/O AREA AND CLEAR HIGH ORDER BYTE
D3 READING BACKWARD
YES
NO
IJFFBCKW
D4 GET I/O AREA ADDRESS MINUS RECORD SIZE
E3 CALCULATE I/O AREA ADDRESS
E4 SAVE POINTER TO DEBLOCKER1 AND CALCULATE I/O AREA ADDRESS
IJFFSTD
F3 STORE I/O AREA ADDRESS AND RESIDUAL COUNT AND RECORD LENGTH
F4 WRONG LENGTH RECORD
YES
NO
IJFFTET
G3 SET WRL BIT IN CCB AS REQUIRED
IJFFIGN
G4 TURN OFF WRONG LENGTH RECORD BIT
H3 SUBTRACT RECORD LENGTH FROM RESIDUAL COUNT IF REQUIRED
H4 GET I/O AREA ADDRESS SWITCH I/O AREAS STORE OLD AREA ADDR
IJFFCON
J3 LOAD ERROR EXIT ADDRESS
J4 SVC 0 PERFORM I/O OPERATION IF REQUIRED
K3 LOAD WLR ADDRESS IF REQUIRED
K4 RETURN TO CALLING ROUTINE
DC J2
IJFXRCT
C5 IJFXTRA DK TRANSLATE IF REQUIRED
D5 LOAD RESIDUAL COUNT IF REQUIRED
E5 GET I/O AREA ADDRESS, RECORD LENGTH, AND BLOCK PREFIX LENGTH
DM A1
F5 INPUT FILE
YES
NO
G5 ERROPT SPECIFIED
YES
NO
DP A1
IJF ERR1
H5 SVC 50 CANCEL JOB

Chart DM. MTMOD: Read/Write Subroutine, Fixed Length Records (Part 2 of 2)

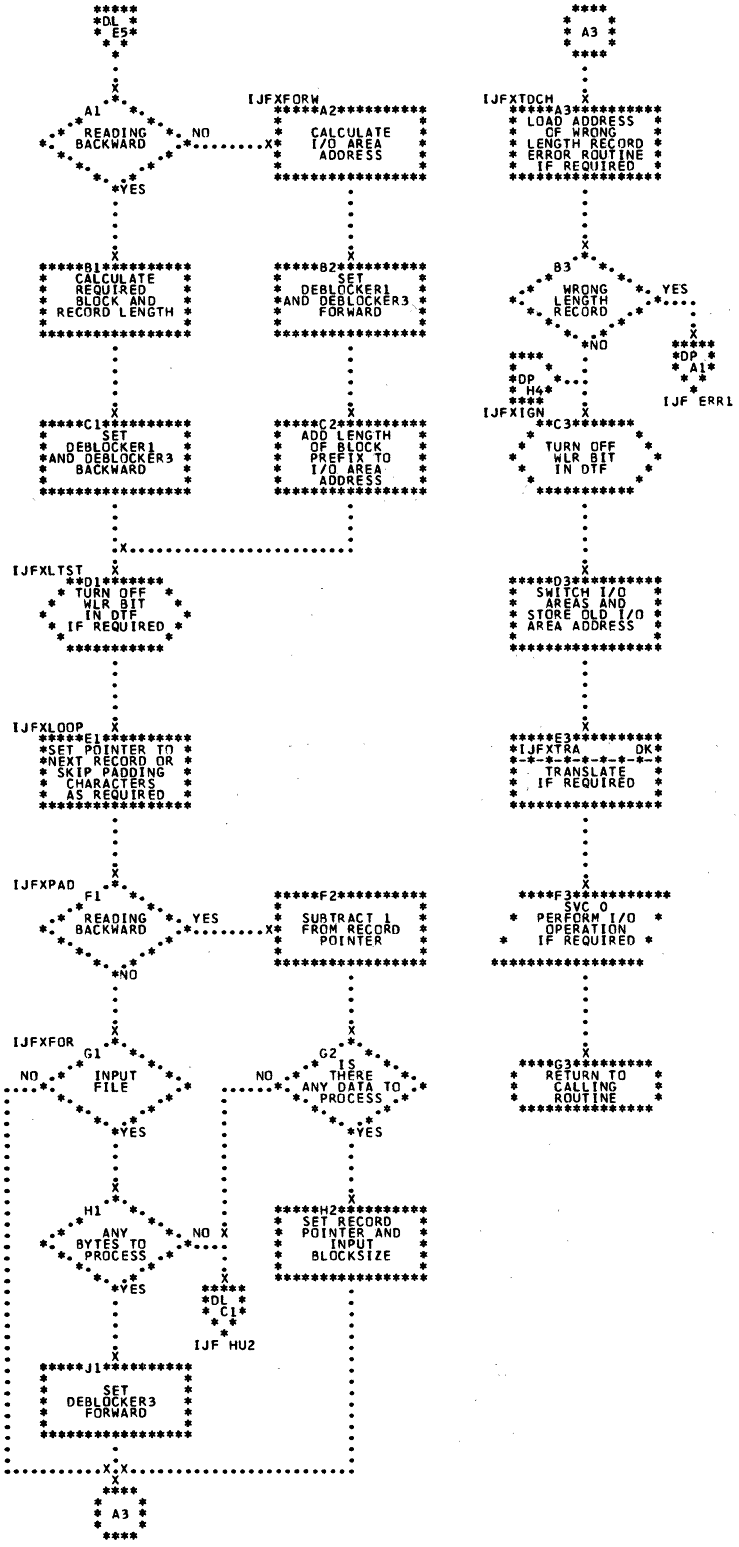

DL E5
A1 READING BACKWARD
NO
YES
IJFXFORW
A2 CALCULATE I/O AREA ADDRESS
B1 CALCULATE REQUIRED BLOCK AND RECORD LENGTH
B2 SET DEBLOCKER1 AND DEBLOCKER3 FORWARD
C1 SET DEBLOCKER1 AND DEBLOCKER3 BACKWARD
C2 ADD LENGTH OF BLOCK PREFIX TO I/O AREA ADDRESS
IJFXLTST
D1 TURN OFF WLR BIT IN DTF IF REQUIRED
IJFXLOOP
E1 SET POINTER TO NEXT RECORD OR SKIP PADDING CHARACTERS AS REQUIRED
IJFXPAD
F1 READING BACKWARD
YES
NO
F2 SUBTRACT 1 FROM RECORD POINTER
IJFXFOR
G1 INPUT FILE
NO
YES
G2 IS THERE ANY DATA TO PROCESS
NO
YES
H1 ANY BYTES TO PROCESS
NO
YES
DL C1
IJF HU2
H2 SET RECORD POINTER AND INPUT BLOCKSIZE
J1 SET DEBLOCKER3 FORWARD
A3
A3
IJFXTDCH
A3 LOAD ADDRESS OF WRONG LENGTH RECORD ERROR ROUTINE IF REQUIRED
B3 WRONG LENGTH RECORD
YES
NO
DP A1
IJF ERR1
DP H4
IJFXIGN
C3 TURN OFF WLR BIT IN DTF
D3 SWITCH I/O AREAS AND STORE OLD I/O AREA ADDRESS
E3 IJFXTRA DK
TRANSLATE IF REQUIRED
F3 SVC 0 PERFORM I/O OPERATION IF REQUIRED
G3 RETURN TO CALLING ROUTINE

Chart DN. MTMOD: Bypass Checkpoint Records Routine, Fixed Length Records

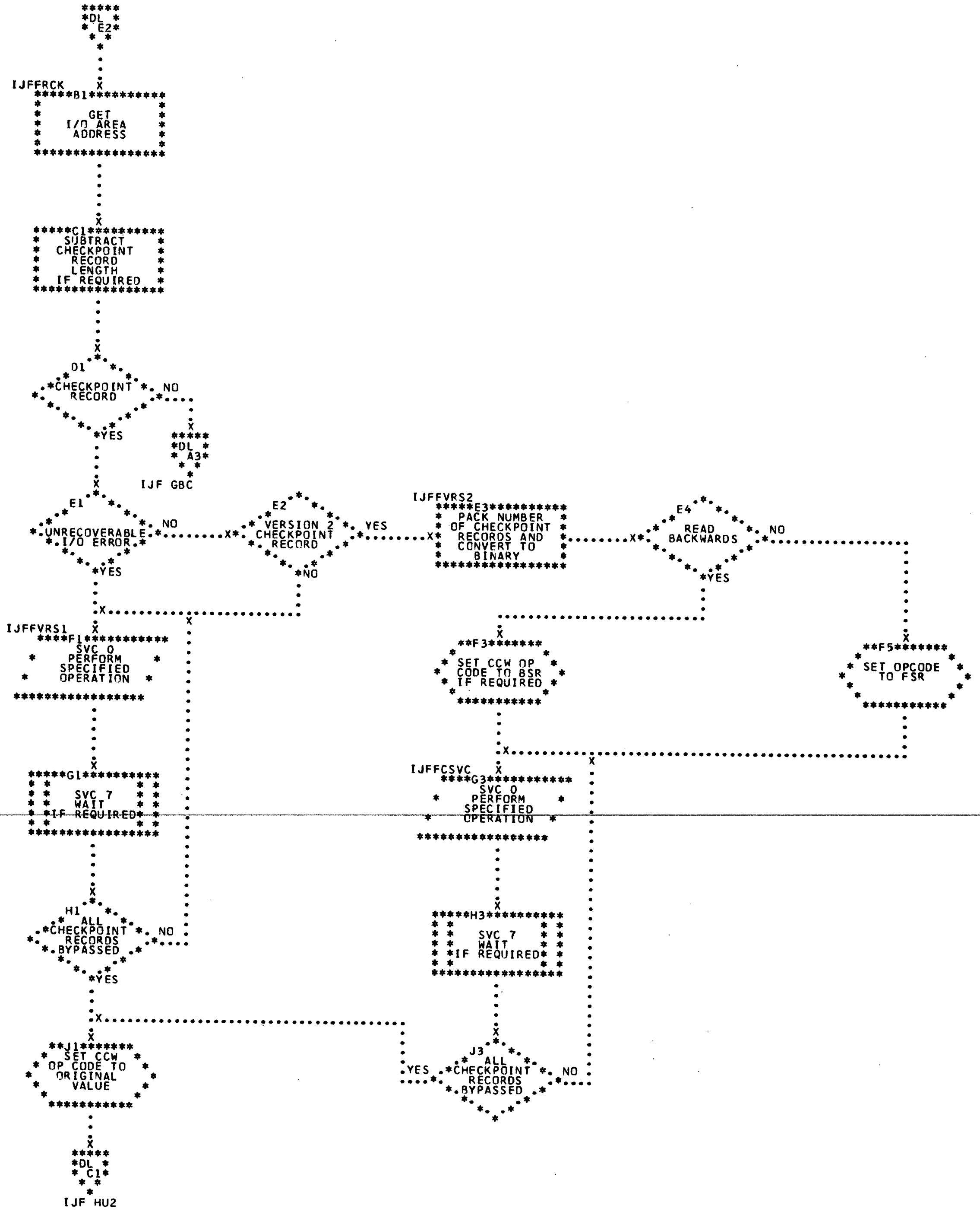

DL E2
IJFFRCK
B1
GET I/O AREA ADDRESS
C1
SUBTRACT CHECKPOINT RECORD LENGTH IF REQUIRED
D1
CHECKPOINT RECORD
NO
YES
DL A3
IJF GBC
E1
UNRECOVERABLE I/O ERROR
NO
YES
E2
VERSION 2 CHECKPOINT RECORD
YES
NO
IJFFVRS2
E3
PACK NUMBER OF CHECKPOINT RECORDS AND CONVERT TO BINARY
E4
READ BACKWARDS
NO
YES
IJFFVRS1
F1
SVC 0 PERFORM SPECIFIED OPERATION
F3
SET CCW OP CODE TO BSR IF REQUIRED
F5
SET OPCODE TO FSR
G1
SVC 7 WAIT IF REQUIRED
IJFFCSVC
G3
SVC 0 PERFORM SPECIFIED OPERATION
H1
ALL CHECKPOINT RECORDS BYPASSED
NO
YES
H3
SVC 7 WAIT IF REQUIRED
J1
SET CCW OP CODE TO ORIGINAL VALUE
J3
ALL CHECKPOINT RECORDS BYPASSED
YES
NO
DL C1
IJF HU2

Chart DP. MTMOD: Error Exit Routine, Fixed Length Records

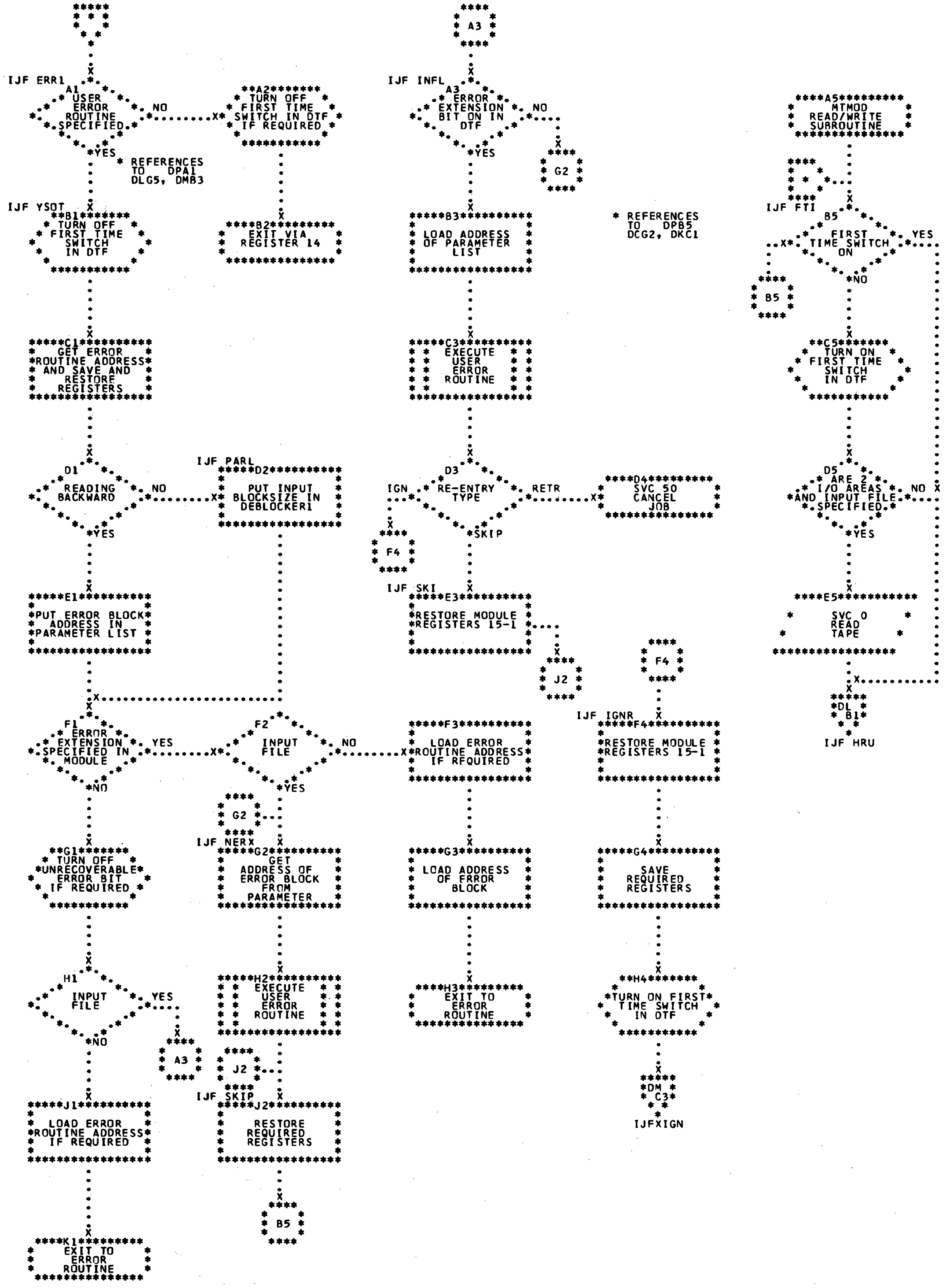
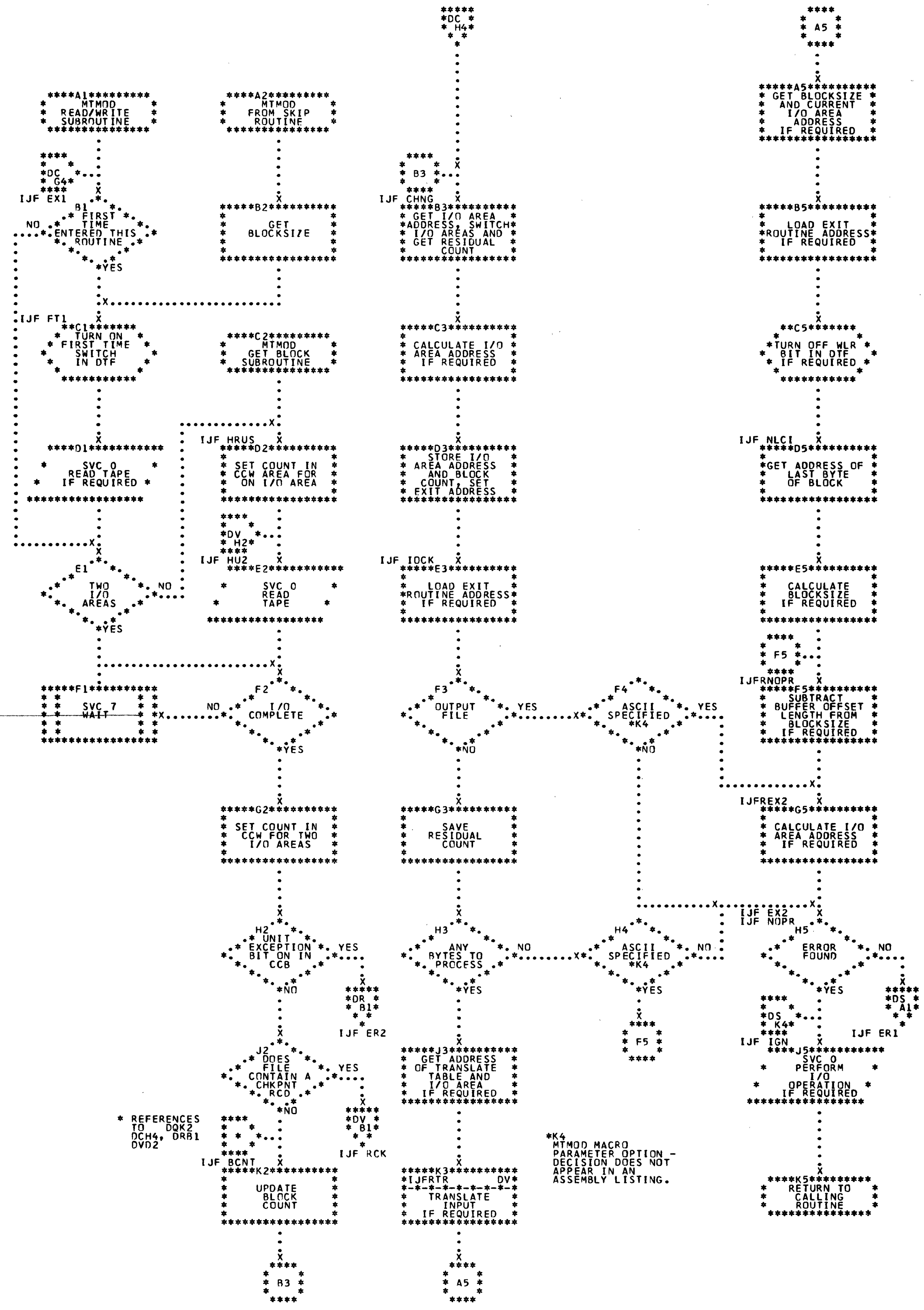

IJF ERR1
A1
USER ERROR ROUTINE SPECIFIED
NO
YES
REFERENCES TO DPA1 DLG5, DMB3
A2
TURN OFF FIRST TIME SWITCH IN DTF IF REQUIRED
IJF YSOT
B1
TURN OFF FIRST TIME SWITCH IN DTF
B2
EXIT VIA REGISTER 14
C1
GET ERROR ROUTINE ADDRESS AND SAVE AND RESTORE REGISTERS
D1
READING BACKWARD
NO
YES
IJF PARL
D2
PUT INPUT BLOCKSIZE IN DEBLOCKER1
E1
PUT ERROR BLOCK ADDRESS IN PARAMETER LIST
F1
ERROR EXTENSION SPECIFIED IN MODULE
YES
NO
F2
INPUT FILE
NO
YES
G2
IJF NERX
G1
TURN OFF UNRECOVERABLE ERROR BIT IF REQUIRED
G2
GET ADDRESS OF ERROR BLOCK FROM PARAMETER
H1
INPUT FILE
YES
NO
A3
H2
EXECUTE USER ERROR ROUTINE
J2
IJF SKIP
J1
LOAD ERROR ROUTINE ADDRESS IF REQUIRED
J2
RESTORE REQUIRED REGISTERS
B5
K1
EXIT TO ERROR ROUTINE
A3
IJF INFL
A3
ERROR EXTENSION BIT ON IN DTF
NO
G2
YES
B3
LOAD ADDRESS OF PARAMETER LIST
REFERENCES TO DPB5 DCG2, DKC1
C3
EXECUTE USER ERROR ROUTINE
D3
RE-ENTRY TYPE
IGN
RETR
SKIP
F4
D4
SVC 50 CANCEL JOB
IJF SKI
E3
RESTORE MODULE REGISTERS 15-1
J2
F4
IJF IGNR
F3
LOAD ERROR ROUTINE ADDRESS IF REQUIRED
F4
RESTORE MODULE REGISTERS 15-1
G3
LOAD ADDRESS OF ERROR BLOCK
G4
SAVE REQUIRED REGISTERS
H3
EXIT TO ERROR ROUTINE
H4
TURN ON FIRST TIME SWITCH IN DTF
DM C3
IJFXIGN
A5
MTMOD READ/WRITE SUBROUTINE
IJF FTI
B5
FIRST TIME SWITCH ON
YES
NO
B5
C5
TURN ON FIRST TIME SWITCH IN DTF
D5
ARE 2 I/O AREAS AND INPUT FILE SPECIFIED
NO
YES
E5
SVC 0 READ TAPE
DL B1
IJF HRU

Chart DQ. MTMOD: Read/Write Subroutine, Variable Length Records (Part 1 of 2)


DC H4
A5
A1 MTMOD READ/WRITE SUBROUTINE
A2 MTMOD FROM SKIP ROUTINE
A5 GET BLOCKSIZE AND CURRENT I/O AREA ADDRESS IF REQUIRED
DC G4
IJF EX1
B1 FIRST TIME ENTERED THIS ROUTINE
NO
YES
B2 GET BLOCKSIZE
B3
IJF CHNG
B3 GET I/O AREA ADDRESS, SWITCH I/O AREAS AND GET RESIDUAL COUNT
B5 LOAD EXIT ROUTINE ADDRESS IF REQUIRED
IJF FT1
C1 TURN ON FIRST TIME SWITCH IN DTF
C2 MTMOD GET BLOCK SUBROUTINE
C3 CALCULATE I/O AREA ADDRESS IF REQUIRED
C5 TURN OFF WLR BIT IN DTF IF REQUIRED
D1 SVC 0 READ TAPE IF REQUIRED
IJF HRUS
D2 SET COUNT IN CCW AREA FOR ON I/O AREA
D3 STORE I/O AREA ADDRESS AND BLOCK COUNT, SET EXIT ADDRESS
IJF NLCI
D5 GET ADDRESS OF LAST BYTE OF BLOCK
DV H2
E1 TWO I/O AREAS
NO
YES
IJF HU2
E2 SVC 0 READ TAPE
IJF IOCK
E3 LOAD EXIT ROUTINE ADDRESS IF REQUIRED
E5 CALCULATE BLOCKSIZE IF REQUIRED
F5
IJFRNOPR
F1 SVC 7 WAIT
F2 I/O COMPLETE
NO
YES
F3 OUTPUT FILE
YES
NO
F4 ASCII SPECIFIED *K4
YES
NO
F5 SUBTRACT BUFFER OFFSET LENGTH FROM BLOCKSIZE IF REQUIRED
IJFREX2
G2 SET COUNT IN CCW FOR TWO I/O AREAS
G3 SAVE RESIDUAL COUNT
G5 CALCULATE I/O AREA ADDRESS IF REQUIRED
IJF EX2
IJF NOPR
H2 UNIT EXCEPTION BIT ON IN CCB
YES
NO
DR B1
IJF ER2
H3 ANY BYTES TO PROCESS
NO
YES
H4 ASCII SPECIFIED *K4
NO
YES
F5
H5 ERROR FOUND
NO
YES
DS A1
DS K4
IJF IGN
IJF ER1
J2 DOES FILE CONTAIN A CHKPNT RCD
YES
NO
DV B1
IJF RCK
J3 GET ADDRESS OF TRANSLATE TABLE AND I/O AREA IF REQUIRED
J5 SVC 0 PERFORM I/O OPERATION IF REQUIRED
IJF BCNT
K2 UPDATE BLOCK COUNT
K3 IJFRTR DV TRANSLATE INPUT IF REQUIRED
K5 RETURN TO CALLING ROUTINE
B3
A5

* REFERENCES TO DQK2 DCH4, DRB1 DVD2

*K4 MTMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.

Chart DR. MTMOD: Read/Write Subroutine, Variable Length Records (Part 2 of 2)

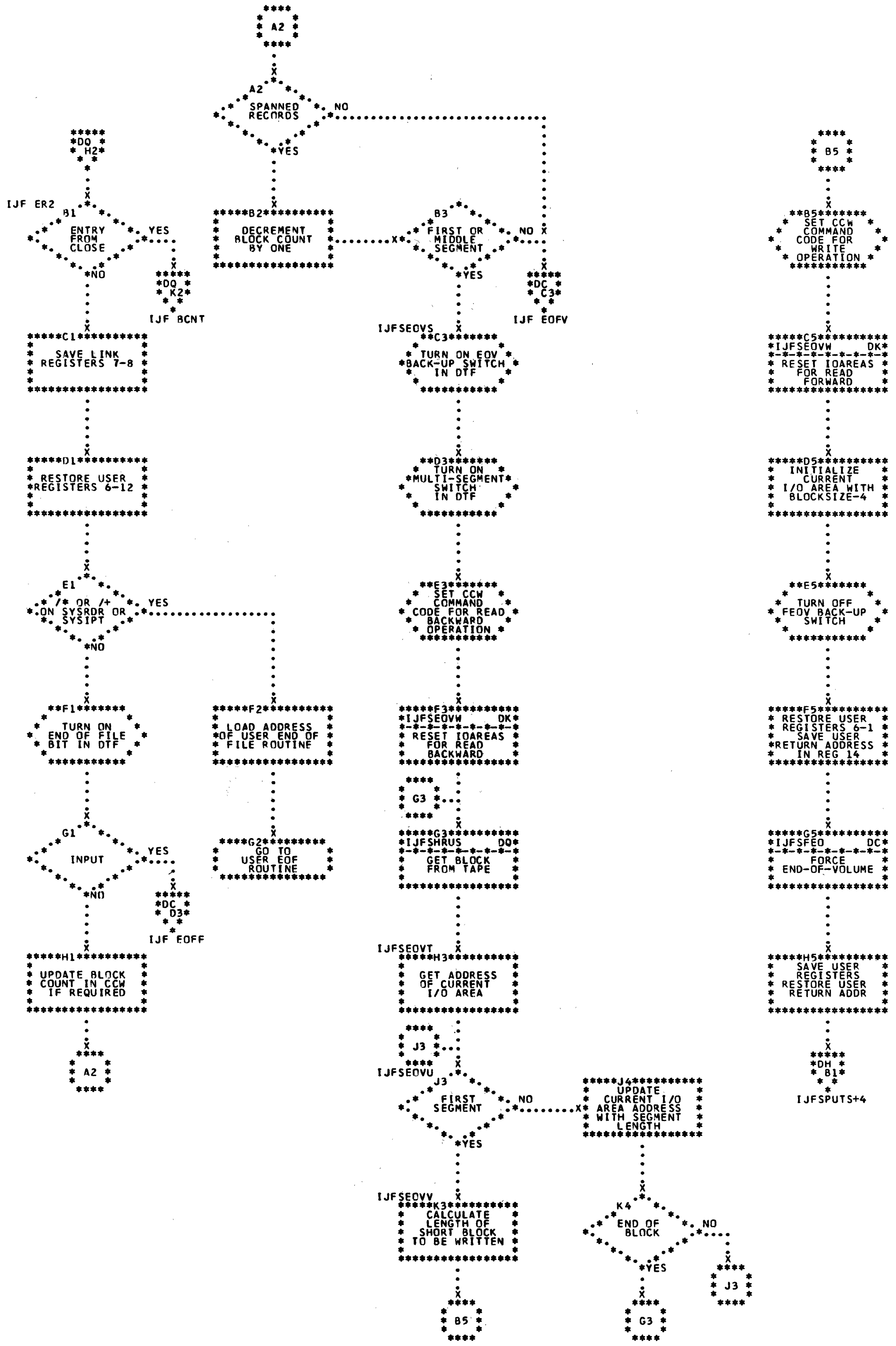

A2
A2 SPANNED RECORDS
NO
YES
B2 DECREMENT BLOCK COUNT BY ONE
B3 FIRST OR MIDDLE SEGMENT
NO
YES
DC C3
IJF EOFV
DQ H2
IJF ER2
B1 ENTRY FROM CLOSE
YES
NO
DQ K2
IJF BCNT
C1 SAVE LINK REGISTERS 7-8
D1 RESTORE USER REGISTERS 6-12
E1 /* OR /+ ON SYSRDR OR SYSIPT
YES
NO
F1 TURN ON END OF FILE BIT IN DTF
F2 LOAD ADDRESS OF USER END OF FILE ROUTINE
G2 GO TO USER EOF ROUTINE
G1 INPUT
YES
NO
DC D3
IJF EOFF
H1 UPDATE BLOCK COUNT IN CCW IF REQUIRED
A2
IJFSEOVS
C3 TURN ON EOV BACK-UP SWITCH IN DTF
D3 TURN ON MULTI-SEGMENT SWITCH IN DTF
E3 SET CCW COMMAND CODE FOR READ BACKWARD OPERATION
F3 IJFSEOVW DK
RESET IOAREAS FOR READ BACKWARD
G3
G3 IJFSHRUS DQ
GET BLOCK FROM TAPE
IJFSEOVT
H3 GET ADDRESS OF CURRENT I/O AREA
J3
IJFSEOVU
J3 FIRST SEGMENT
NO
YES
J4 UPDATE CURRENT I/O AREA ADDRESS WITH SEGMENT LENGTH
IJFSEOVV
K3 CALCULATE LENGTH OF SHORT BLOCK TO BE WRITTEN
B5
K4 END OF BLOCK
NO
YES
J3
G3
B5
B5 SET CCW COMMAND CODE FOR WRITE OPERATION
C5 IJFSEOVW DK
RESET IOAREAS FOR READ FORWARD
D5 INITIALIZE CURRENT I/O AREA WITH BLOCKSIZE-4
E5 TURN OFF FEOV BACK-UP SWITCH
F5 RESTORE USER REGISTERS 6-1 SAVE USER RETURN ADDRESS IN REG 14
G5 IJFSFEO DC
FORCE END-OF-VOLUME
H5 SAVE USER REGISTERS RESTORE USER RETURN ADDR
DH B1
IJFSPUTS+4

Chart DS. MTMOD: Error Exit Routine, Variable Length Records

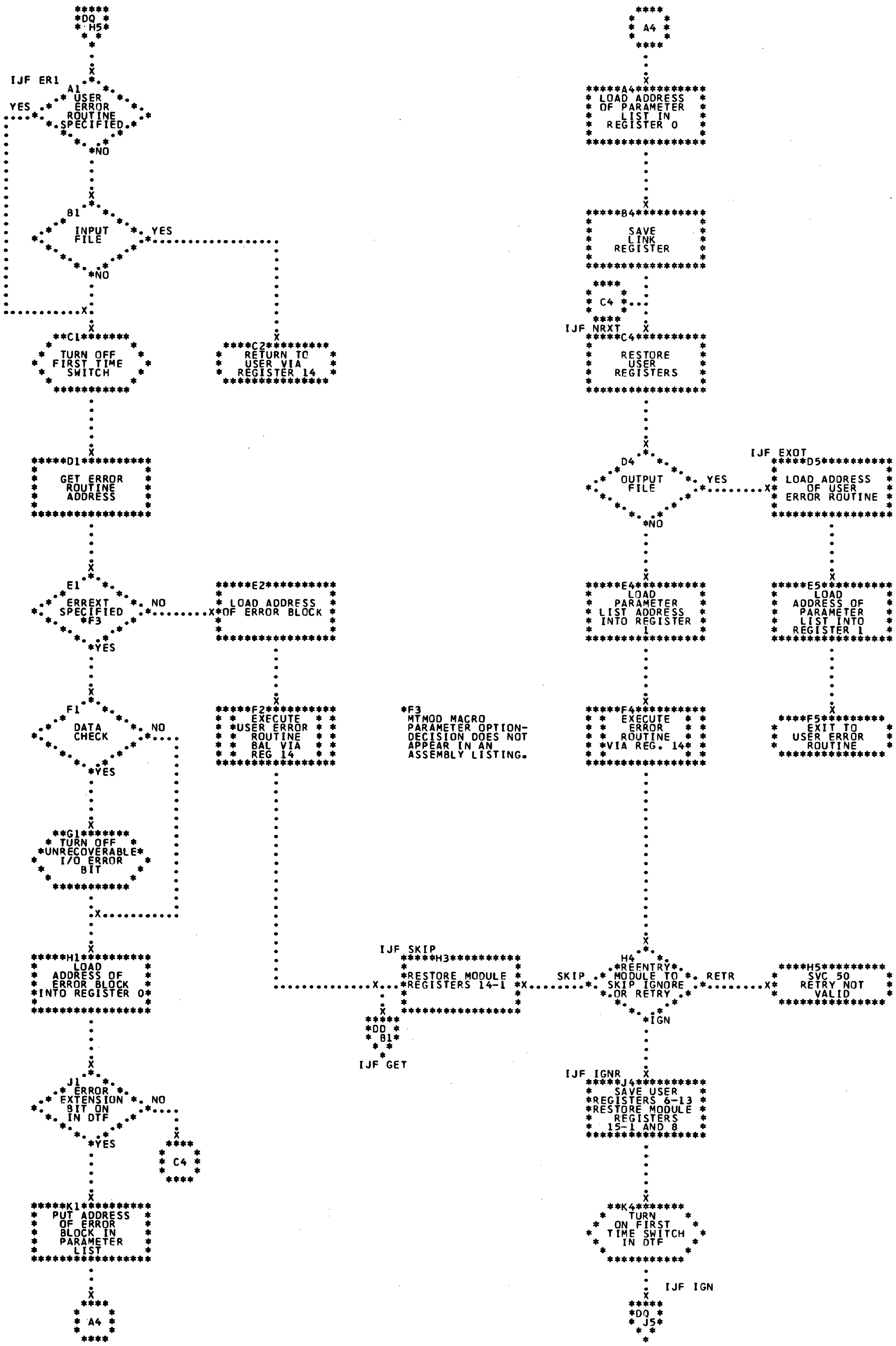

DQ H5
IJF ER1
A1 USER ERROR ROUTINE SPECIFIED
YES
NO
B1 INPUT FILE
YES
NO
C1 TURN OFF FIRST TIME SWITCH
C2 RETURN TO USER VIA REGISTER 14
D1 GET ERROR ROUTINE ADDRESS
E1 ERREXT SPECIFIED *F3
NO
YES
E2 LOAD ADDRESS OF ERROR BLOCK
F1 DATA CHECK
NO
YES
F2 EXECUTE USER ERROR ROUTINE BAL VIA REG 14
*F3 MTMOD MACRO PARAMETER OPTION-DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
G1 TURN OFF UNRECOVERABLE I/O ERROR BIT
H1 LOAD ADDRESS OF ERROR BLOCK INTO REGISTER 0
J1 ERROR EXTENSION BIT ON IN DTF
NO
YES
C4
K1 PUT ADDRESS OF ERROR BLOCK IN PARAMETER LIST
A4
A4
A4 LOAD ADDRESS OF PARAMETER LIST IN REGISTER 0
B4 SAVE LINK REGISTER
C4
IJF NRXT
C4 RESTORE USER REGISTERS
D4 OUTPUT FILE
YES
NO
IJF EXOT
D5 LOAD ADDRESS OF USER ERROR ROUTINE
E4 LOAD PARAMETER LIST ADDRESS INTO REGISTER 1
E5 LOAD ADDRESS OF PARAMETER LIST INTO REGISTER 1
F4 EXECUTE ERROR ROUTINE VIA REG. 14
F5 EXIT TO USER ERROR ROUTINE
IJF SKIP
H3 RESTORE MODULE REGISTERS 14-1
DD B1
IJF GET
SKIP
H4 REENTRY MODULE TO SKIP IGNORE OR RETRY
RETR
IGN
H5 SVC 50 RETRY NOT VALID
IJF IGNR
J4 SAVE USER REGISTERS 6-13 RESTORE MODULE REGISTERS 15-1 AND 8
K4 TURN ON FIRST TIME SWITCH IN DTF
IJF IGN
DQ J5

Chart DT. MTMOD: Read/Write Subroutine, Undefined Length Records (Part 1 of 2)

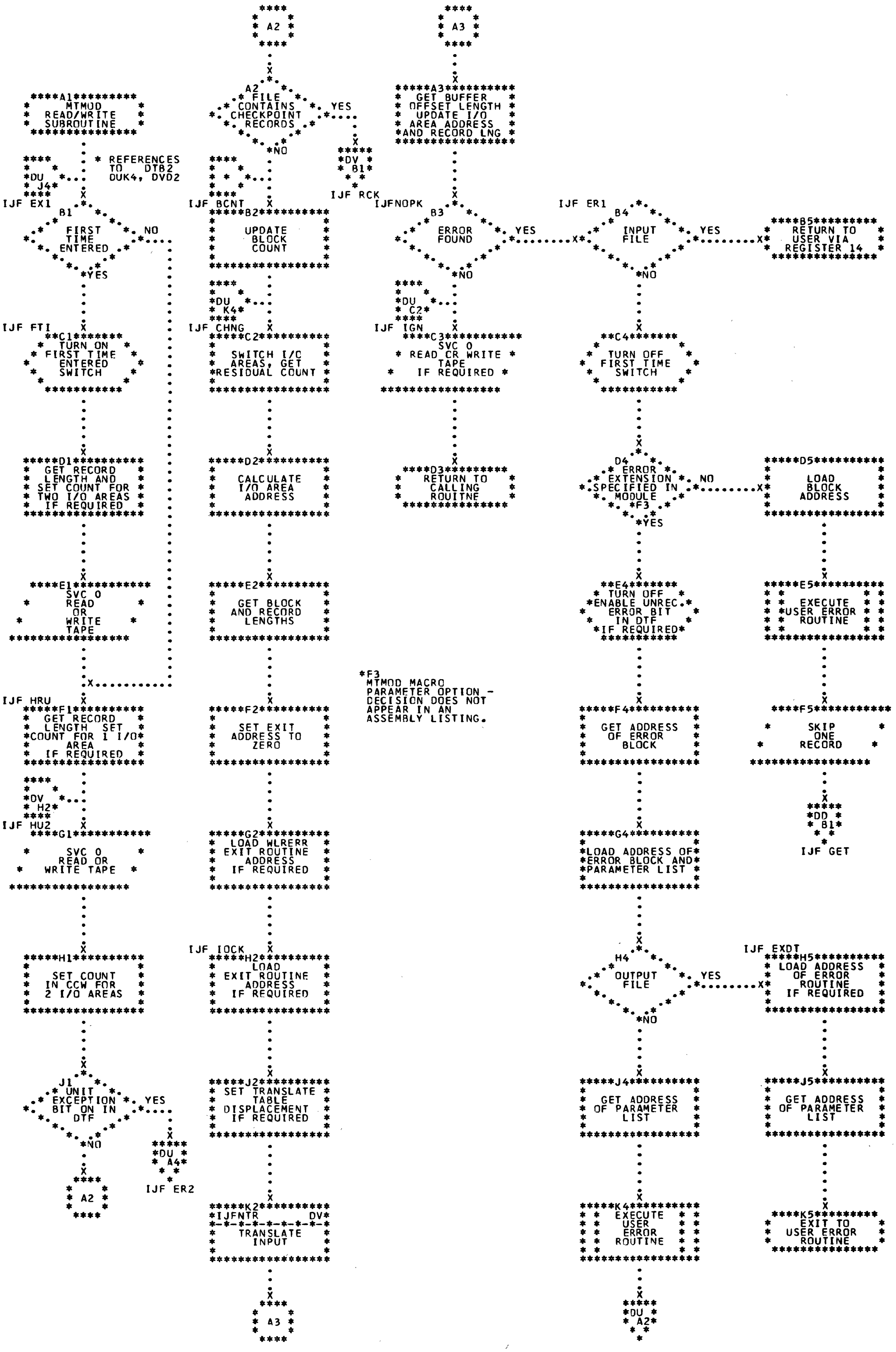


Chart DU. MTMOD: Read/Write Subroutine, Undefined Length Records (Part 2 of 2)

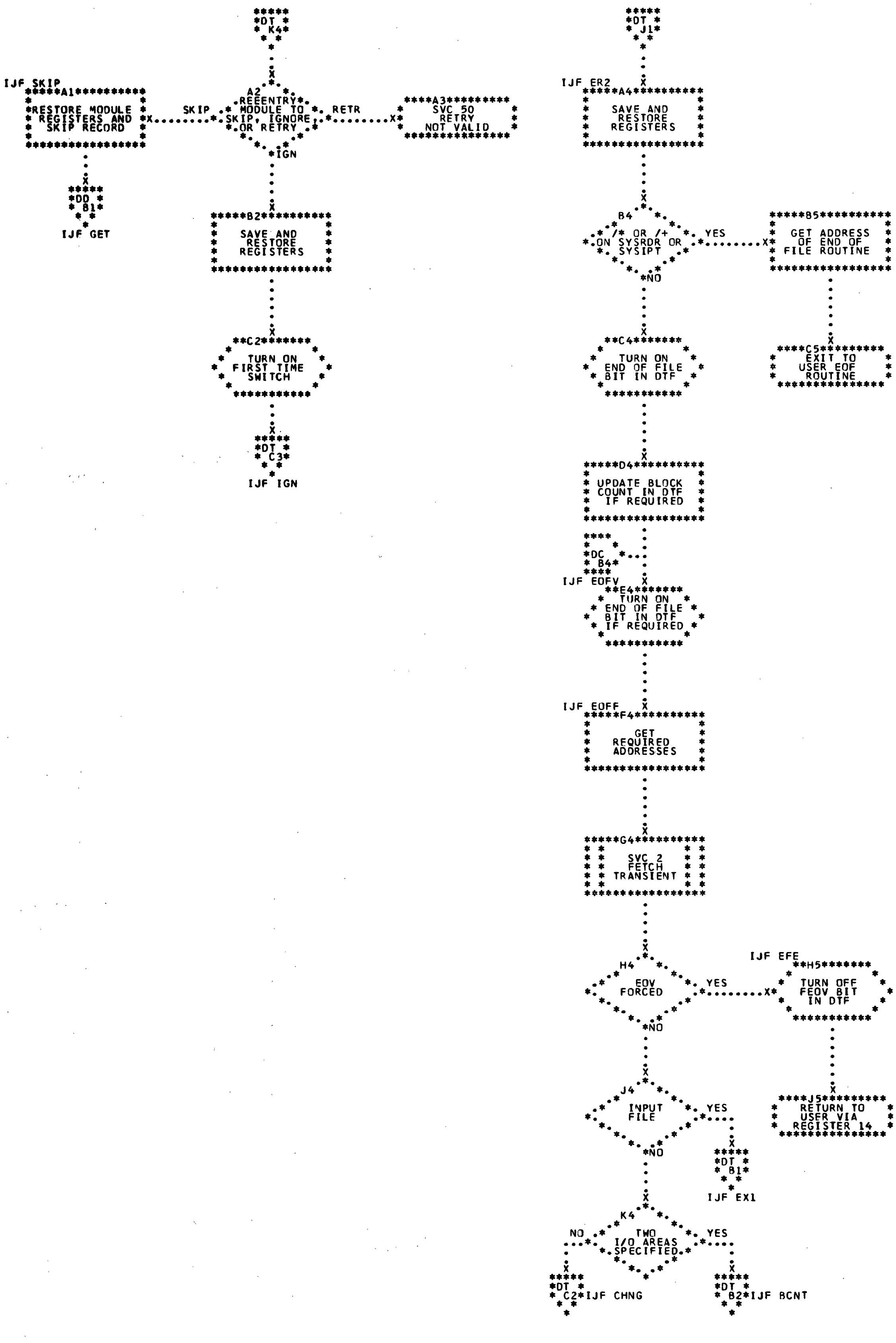

DT K4
A2
REENTRY MODULE TO SKIP, IGNORE OR RETRY
SKIP
RETR
IGN
IJF SKIP
A1
RESTORE MODULE REGISTERS AND SKIP RECORD
DD B1
IJF GET
A3
SVC 50 RETRY NOT VALID
B2
SAVE AND RESTORE REGISTERS
C2
TURN ON FIRST TIME SWITCH
DT C3
IJF IGN
DT J1
IJF ER2
A4
SAVE AND RESTORE REGISTERS
B4
/* OR /+ ON SYSRDR OR SYSIPT
YES
NO
B5
GET ADDRESS OF END OF FILE ROUTINE
C5
EXIT TO USER EOF ROUTINE
C4
TURN ON END OF FILE BIT IN DTF
D4
UPDATE BLOCK COUNT IN DTF IF REQUIRED
DC B4
IJF EOFV
E4
TURN ON END OF FILE BIT IN DTF IF REQUIRED
IJF EOFF
F4
GET REQUIRED ADDRESSES
G4
SVC 2 FETCH TRANSIENT
H4
EOV FORCED
YES
NO
IJF EFE
H5
TURN OFF FEOV BIT IN DTF
J5
RETURN TO USER VIA REGISTER 14
J4
INPUT FILE
YES
NO
DT B1
IJF EX1
K4
TWO I/O AREAS SPECIFIED
NO
YES
DT C2 IJF CHNG
DT B2 IJF BCNT

Chart DV. MTMOD: Bypass Checkpoint Records Routine and Translate Subroutine, Variable and Undefined Length Records

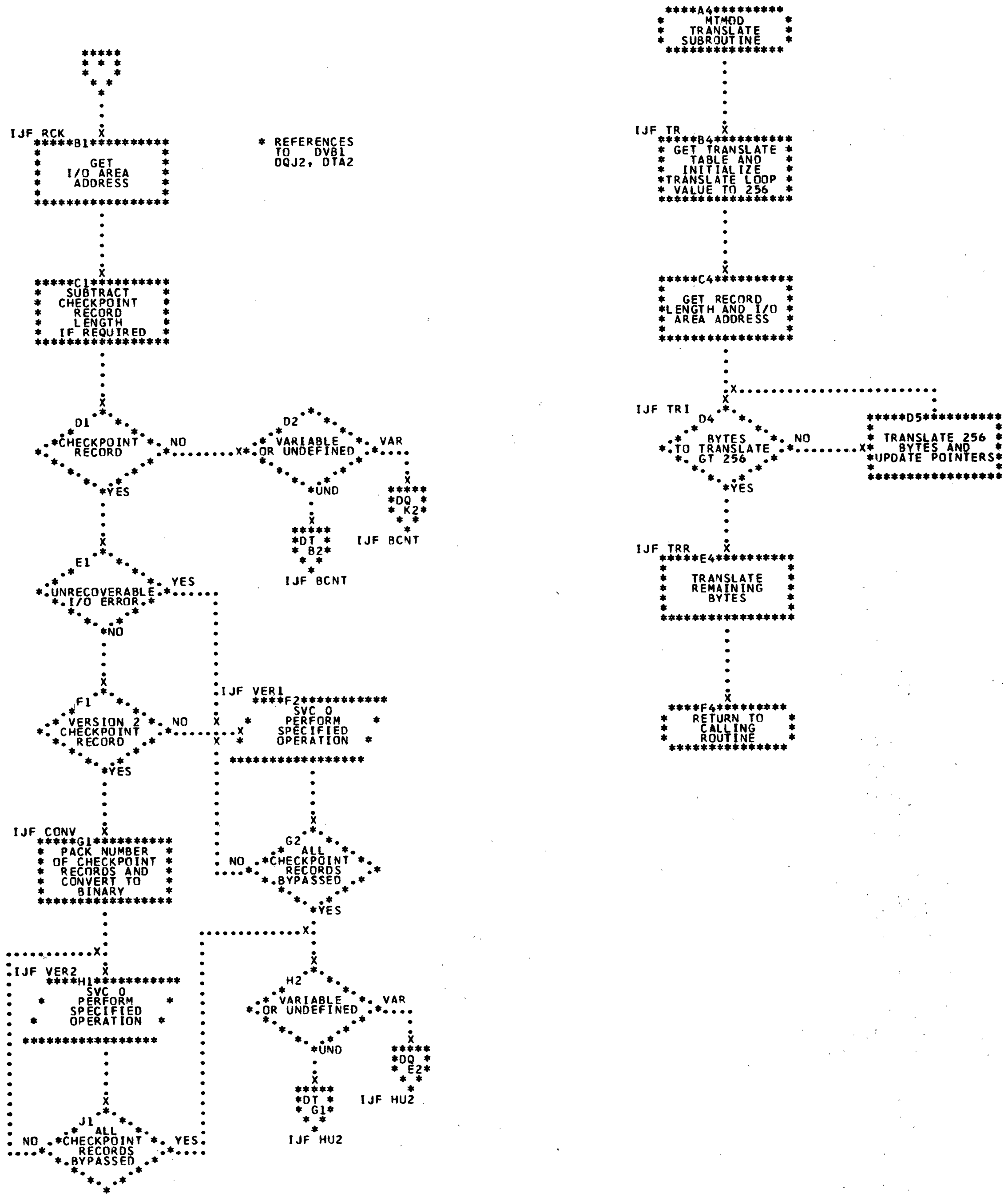

IJF RCK
B1
GET I/O AREA ADDRESS
* REFERENCES TO DVB1 DQJ2, DTA2
C1
SUBTRACT CHECKPOINT RECORD LENGTH IF REQUIRED
D1
CHECKPOINT RECORD
NO
YES
D2
VARIABLE OR UNDEFINED
VAR
UND
DQ K2
IJF BCNT
DT B2
IJF BCNT
E1
UNRECOVERABLE I/O ERROR
YES
NO
F1
VERSION 2 CHECKPOINT RECORD
NO
YES
IJF VER1
F2
SVC 0 PERFORM SPECIFIED OPERATION
G2
ALL CHECKPOINT RECORDS BYPASSED
NO
YES
IJF CONV
G1
PACK NUMBER OF CHECKPOINT RECORDS AND CONVERT TO BINARY
H2
VARIABLE OR UNDEFINED
VAR
UND
DQ E2
IJF HU2
DT G1
IJF HU2
IJF VER2
H1
SVC 0 PERFORM SPECIFIED OPERATION
J1
ALL CHECKPOINT RECORDS BYPASSED
NO
YES
A4
MTMOD TRANSLATE SUBROUTINE
IJF TR
B4
GET TRANSLATE TABLE AND INITIALIZE TRANSLATE LOOP VALUE TO 256
C4
GET RECORD LENGTH AND I/O AREA ADDRESS
IJF TRI
D4
BYTES TO TRANSLATE GT 256
NO
YES
D5
TRANSLATE 256 BYTES AND UPDATE POINTERS
IJF TRR
E4
TRANSLATE REMAINING BYTES
F4
RETURN TO CALLING ROUTINE

Chart DW. MTMOD: Work Area Subroutine

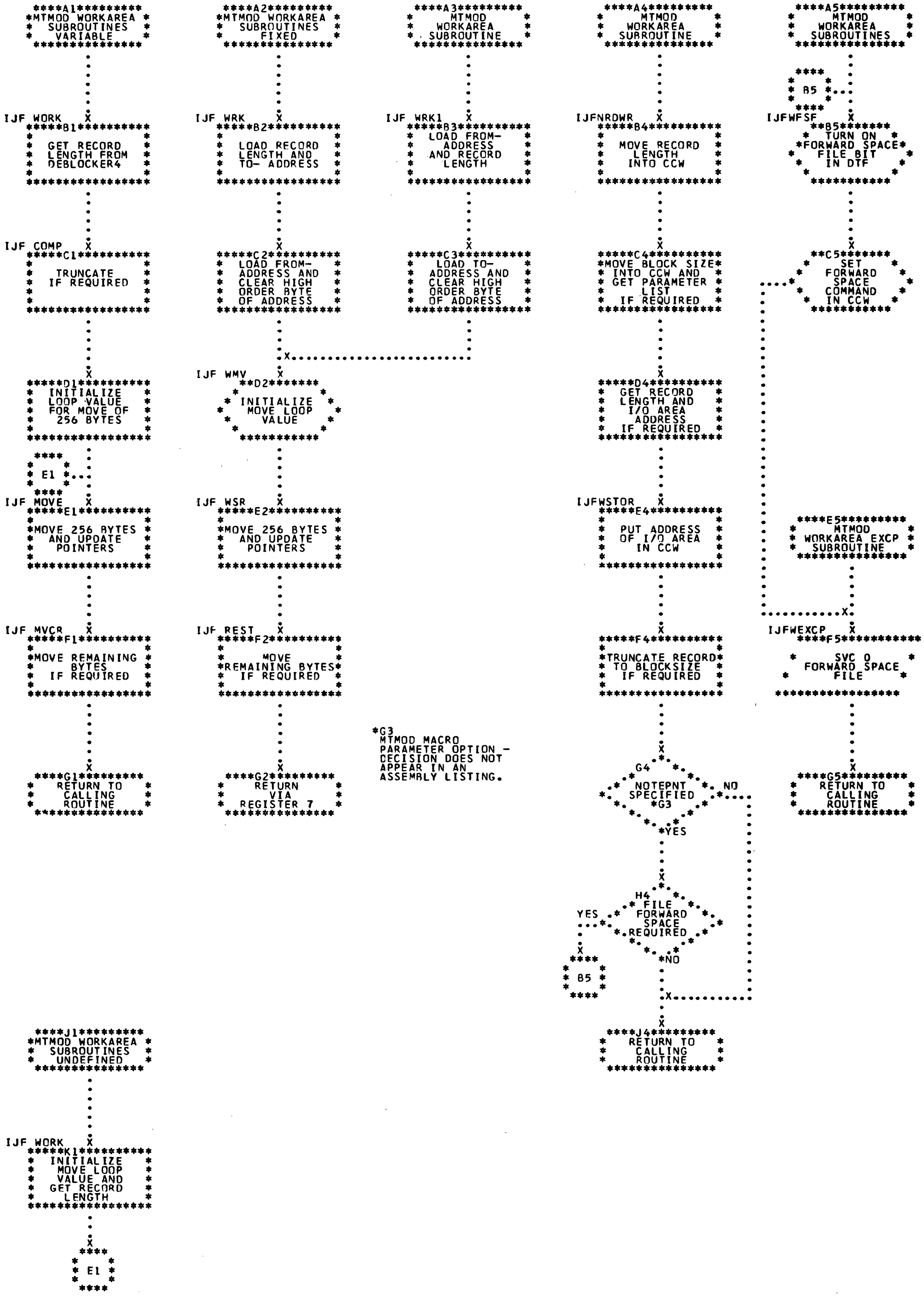

A1 MTMOD WORKAREA SUBROUTINES VARIABLE
IJF WORK
B1 GET RECORD LENGTH FROM DEBLOCKER4
IJF COMP
C1 TRUNCATE IF REQUIRED
D1 INITIALIZE LOOP VALUE FOR MOVE OF 256 BYTES
E1
IJF MOVE
E1 MOVE 256 BYTES AND UPDATE POINTERS
IJF MVCR
F1 MOVE REMAINING BYTES IF REQUIRED
G1 RETURN TO CALLING ROUTINE
J1 MTMOD WORKAREA SUBROUTINES UNDEFINED
IJF WORK
K1 INITIALIZE MOVE LOOP VALUE AND GET RECORD LENGTH
E1
A2 MTMOD WORKAREA SUBROUTINES FIXED
IJF WRK
B2 LOAD RECORD LENGTH AND TO- ADDRESS
C2 LOAD FROM- ADDRESS AND CLEAR HIGH ORDER BYTE OF ADDRESS
IJF WMV
D2 INITIALIZE MOVE LOOP VALUE
IJF WSR
E2 MOVE 256 BYTES AND UPDATE POINTERS
IJF REST
F2 MOVE REMAINING BYTES IF REQUIRED
G2 RETURN VIA REGISTER 7
A3 MTMOD WORKAREA SUBROUTINE
IJF WRK1
B3 LOAD FROM- ADDRESS AND RECORD LENGTH
C3 LOAD TO- ADDRESS AND CLEAR HIGH ORDER BYTE OF ADDRESS
*G3 MTMOD MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.
A4 MTMOD WORKAREA SUBROUTINE
IJFNRDWR
B4 MOVE RECORD LENGTH INTO CCW
C4 MOVE BLOCK SIZE INTO CCW AND GET PARAMETER LIST IF REQUIRED
D4 GET RECORD LENGTH AND I/O AREA ADDRESS IF REQUIRED
IJFWSTOR
E4 PUT ADDRESS OF I/O AREA IN CCW
F4 TRUNCATE RECORD TO BLOCKSIZE IF REQUIRED
G4 NOTEPNT SPECIFIED *G3
NO
YES
H4 FILE FORWARD SPACE REQUIRED
YES
NO
B5
J4 RETURN TO CALLING ROUTINE
A5 MTMOD WORKAREA SUBROUTINES
B5
IJFWFSF
B5 TURN ON FORWARD SPACE FILE BIT IN DTF
C5 SET FORWARD SPACE COMMAND IN CCW
E5 MTMOD WORKAREA EXCP SUBROUTINE
IJFWEXCP
F5 SVC 0 FORWARD SPACE FILE
G5 RETURN TO CALLING ROUTINE

Chart EA. $$BOMT01: Open Input Standard Labels, Forward (Part 1 of 2)

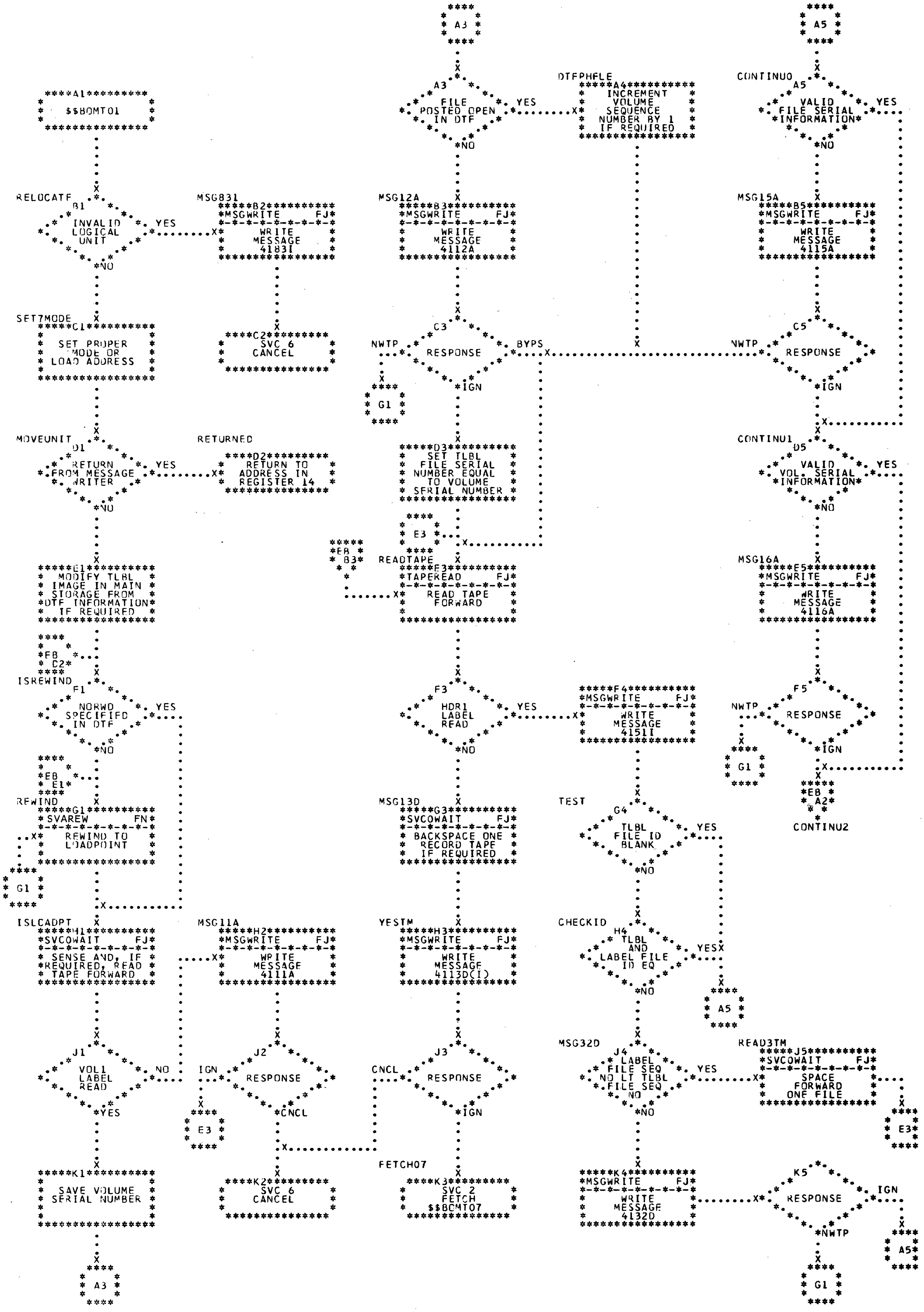


Chart EB. $$BOMT01: Open Input Standard Labels, Forward (Part 2 of 2)

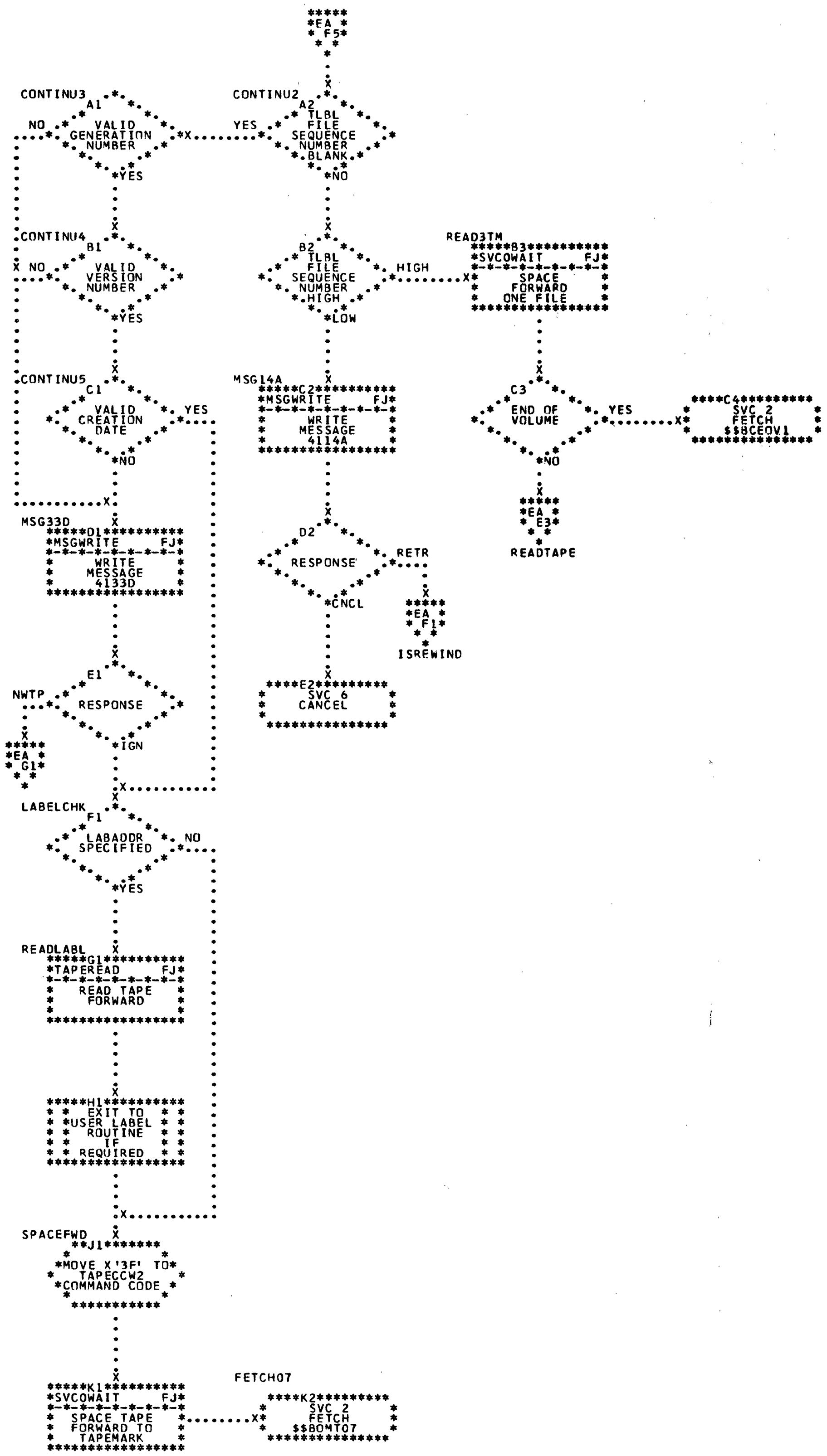


Chart EC. $$BOMT02: Open Input Standard Labels, Backward

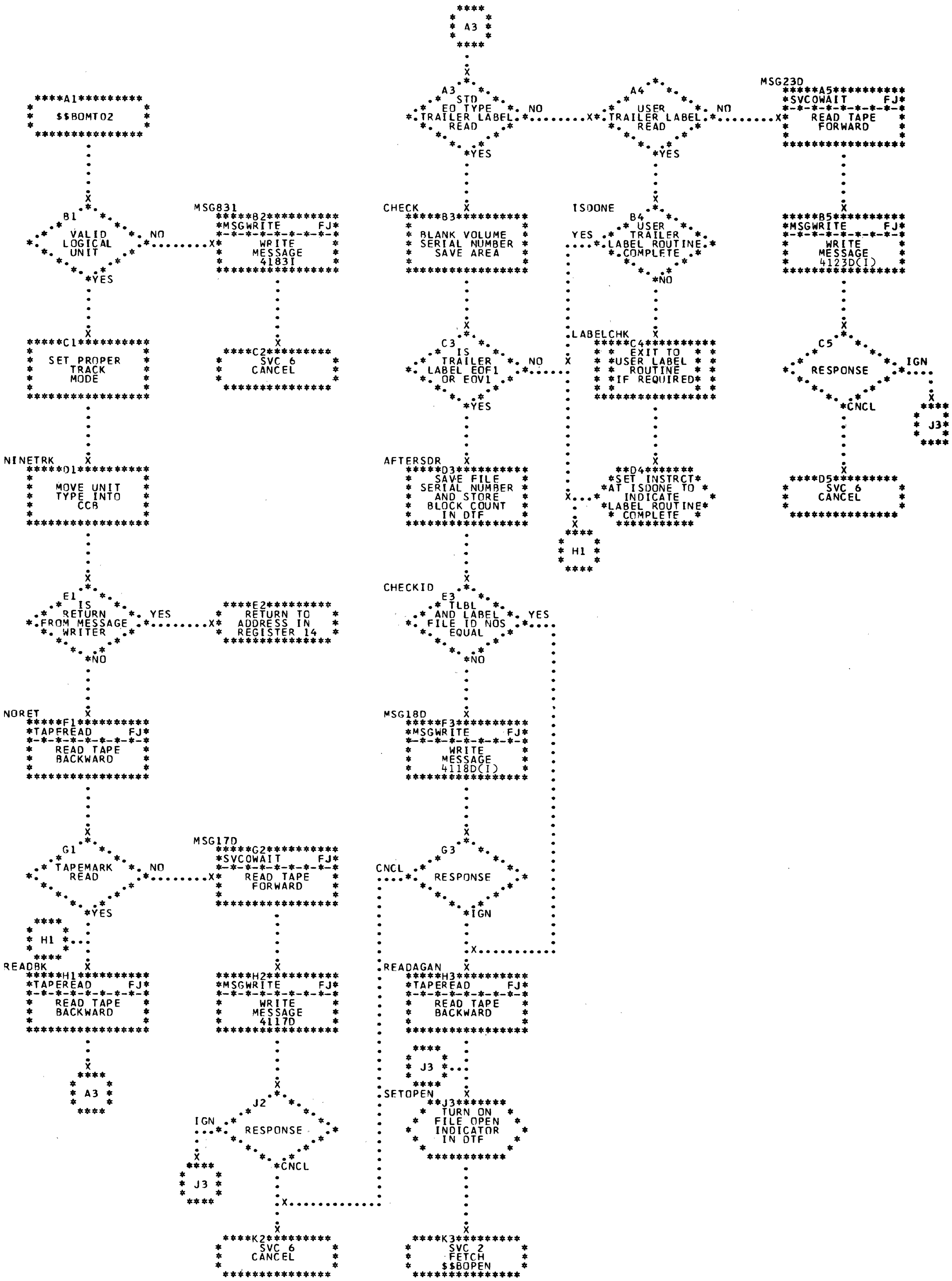


Chart ED. $$BOMT03: Open Output Standard Labels, Forward, Phase 1 (Part 1 of 2)

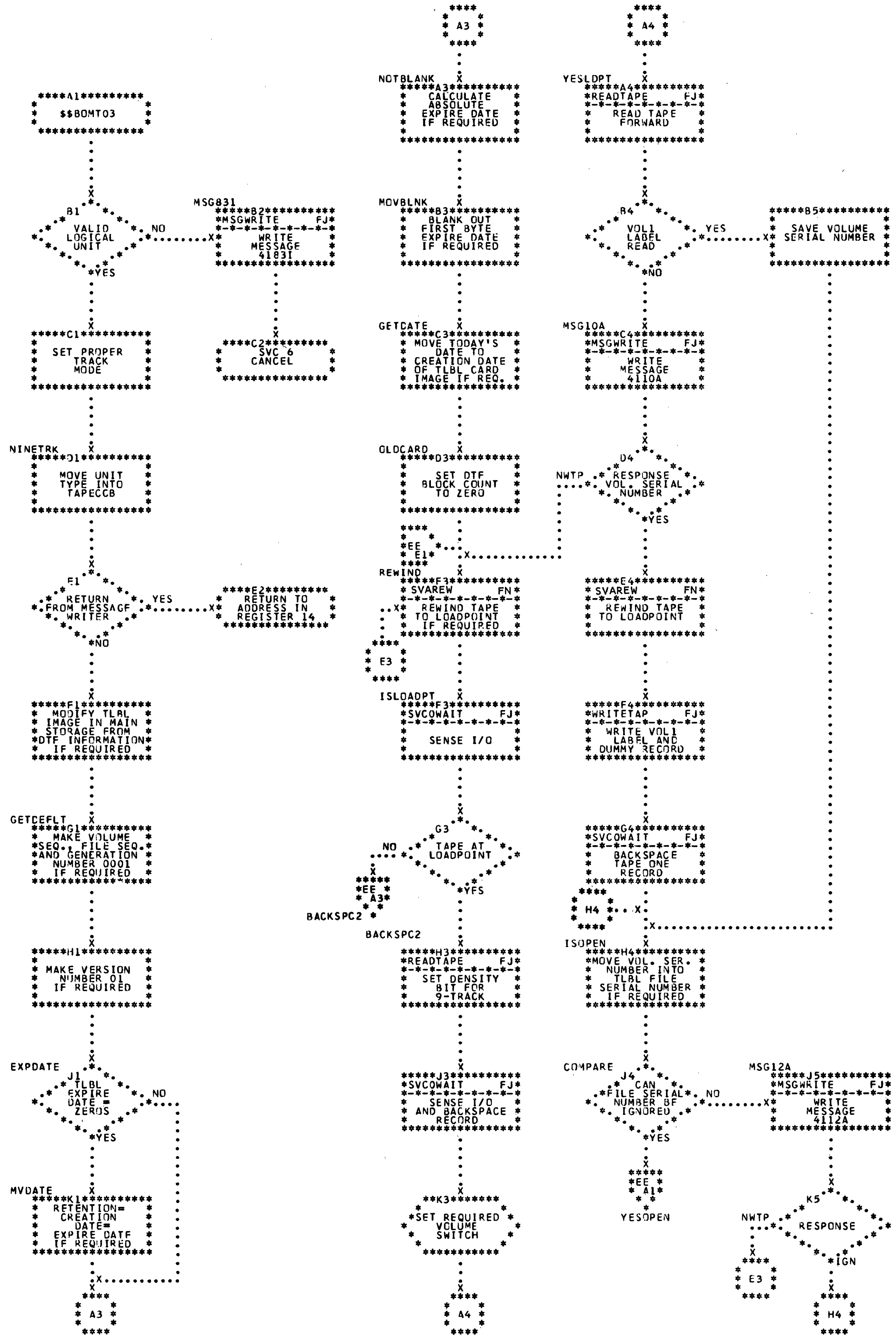

A3
A4
A1
$$BOMT03
NOTBLANK
A3
CALCULATE ABSOLUTE EXPIRE DATE IF REQUIRED
YESLDPT
A4
READTAPE FJ
READ TAPE FORWARD
B1
VALID LOGICAL UNIT
NO
YES
MSG831
B2
MSGWRITE FJ
WRITE MESSAGE 41831
MOVBLNK
B3
BLANK OUT FIRST BYTE EXPIRE DATE IF REQUIRED
B4
VOL1 LABEL READ
YES
NO
B5
SAVE VOLUME SERIAL NUMBER
C1
SET PROPER TRACK MODE
C2
SVC 6 CANCEL
GETDATE
C3
MOVE TODAY'S DATE TO CREATION DATE OF TLBL CARD IMAGE IF REQ.
MSG10A
C4
MSGWRITE FJ
WRITE MESSAGE 4110A
NINETRK
D1
MOVE UNIT TYPE INTO TAPECCB
OLDCARD
D3
SET DTF BLOCK COUNT TO ZERO
D4
NWTP
RESPONSE VOL. SERIAL NUMBER
YES
EE E1
E1
RETURN FROM MESSAGE WRITER
YES
NO
E2
RETURN TO ADDRESS IN REGISTER 14
REWIND
E3
SVAREW FN
REWIND TAPE TO LOADPOINT IF REQUIRED
E3
E4
SVAREW FN
REWIND TAPE TO LOADPOINT
F1
MODIFY TLBL IMAGE IN MAIN STORAGE FROM DTF INFORMATION IF REQUIRED
ISLOADPT
F3
SVCOWAIT FJ
SENSE I/O
F4
WRITETAP FJ
WRITE VOL1 LABEL AND DUMMY RECORD
GETDEFLT
G1
MAKE VOLUME SEQ., FILE SEQ. AND GENERATION NUMBER 0001 IF REQUIRED
G3
TAPE AT LOADPOINT
NO
YES
EE A3
BACKSPC2
G4
SVCOWAIT FJ
BACKSPACE TAPE ONE RECORD
H4
BACKSPC2
H1
MAKE VERSION NUMBER 01 IF REQUIRED
H3
READTAPE FJ
SET DENSITY BIT FOR 9-TRACK
ISOPEN
H4
MOVE VOL. SER. NUMBER INTO TLBL FILE SERIAL NUMBER IF REQUIRED
EXPDATE
J1
TLBL EXPIRE DATE = ZEROS
NO
YES
J3
SVCOWAIT FJ
SENSE I/O AND BACKSPACE RECORD
COMPARE
J4
CAN FILE SERIAL NUMBER BE IGNORED
NO
YES
MSG12A
J5
MSGWRITE FJ
WRITE MESSAGE 4112A
MVDATE
K1
RETENTION= CREATION DATE= EXPIRE DATE IF REQUIRED
K3
SET REQUIRED VOLUME SWITCH
EE A1
YESOPEN
K5
NWTP
RESPONSE
IGN
E3
A3
A4
H4

Chart EE. $$BOMT03: Open Output Standard Labels, Forward, Phase 1 (Part 2 of 2)

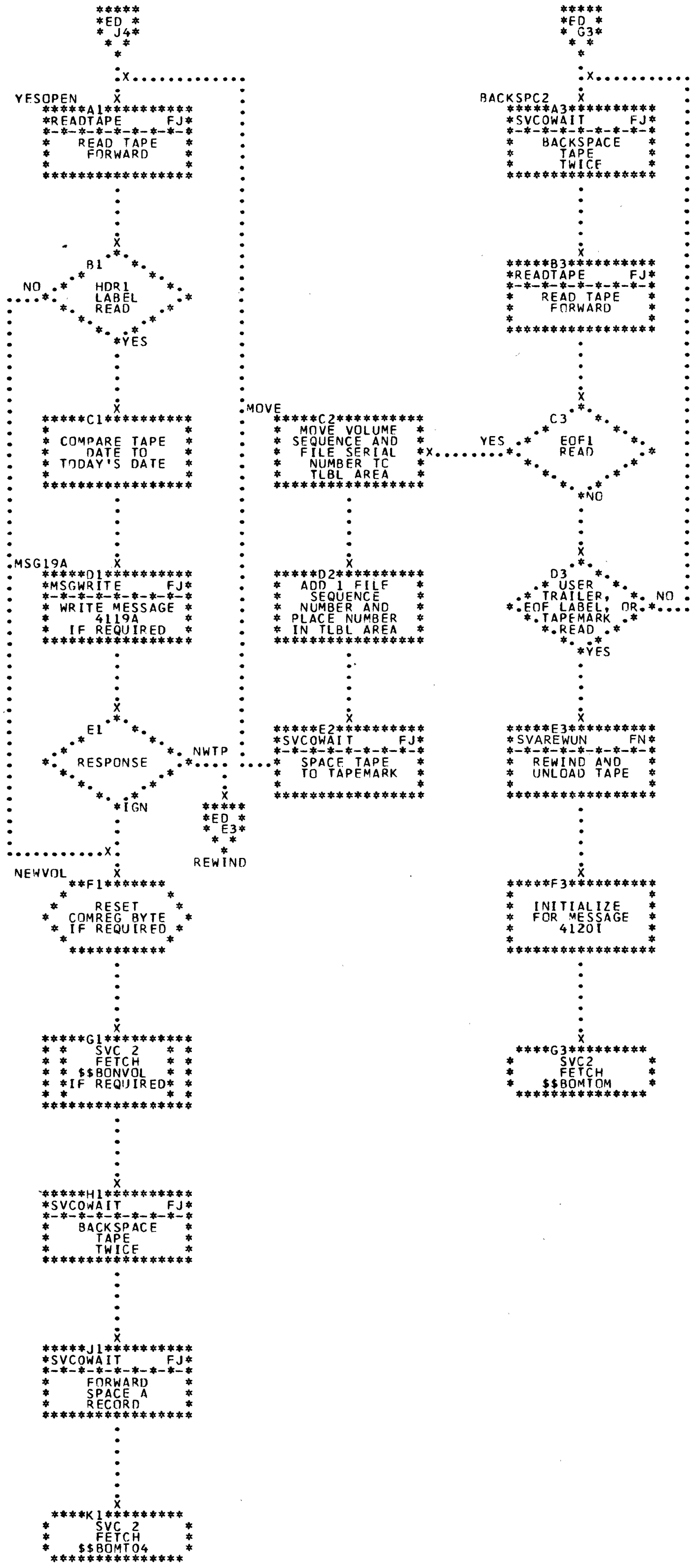


Chart EF. $$BOMT04: Open Output Standard Labels, Forward, Phase 2

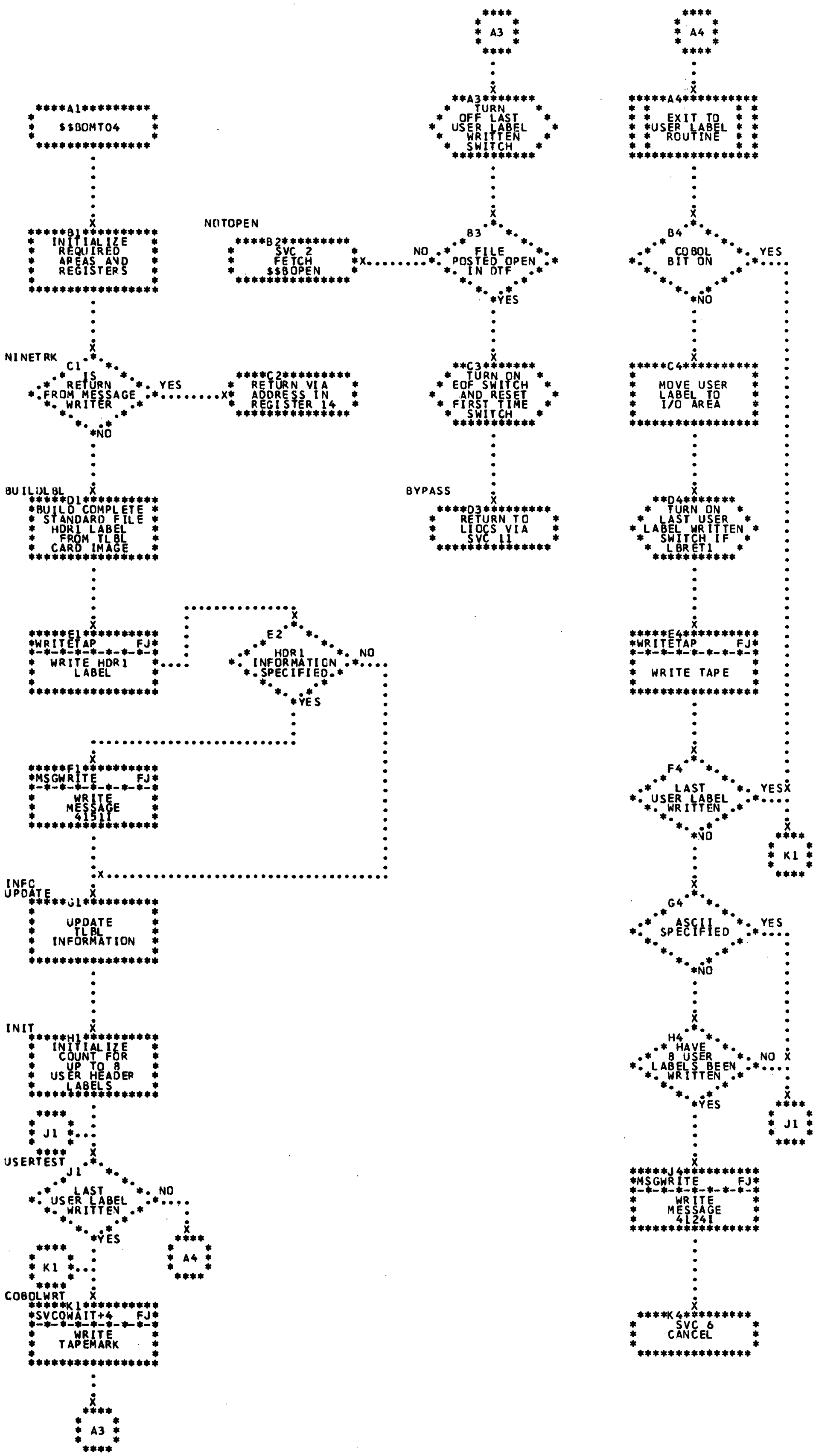

A3
A4
A1 $$BOMT04
B1 INITIALIZE REQUIRED AREAS AND REGISTERS
NOTOPEN
B2 SVC 2 FETCH $$BOPEN
NO
B3 FILE POSTED OPEN IN DTF
YES
A3 TURN OFF LAST USER LABEL WRITTEN SWITCH
A4 EXIT TO USER LABEL ROUTINE
B4 COBOL BIT ON
YES
NO
NINETRK
C1 IS RETURN FROM MESSAGE WRITER
YES
NO
C2 RETURN VIA ADDRESS IN REGISTER 14
C3 TURN ON EOF SWITCH AND RESET FIRST TIME SWITCH
C4 MOVE USER LABEL TO I/O AREA
BUILDLBL
D1 BUILD COMPLETE STANDARD FILE HDR1 LABEL FROM TLBL CARD IMAGE
BYPASS
D3 RETURN TO LIOCS VIA SVC 11
D4 TURN ON LAST USER LABEL WRITTEN SWITCH IF LBRET1
E1 WRITETAP FJ WRITE HDR1 LABEL
E2 HDR1 INFORMATION SPECIFIED
NO
YES
E4 WRITETAP FJ WRITE TAPE
F1 MSGWRITE FJ WRITE MESSAGE 4151I
F4 LAST USER LABEL WRITTEN
YES
NO
K1
INFO UPDATE
G1 UPDATE TLBL INFORMATION
G4 ASCII SPECIFIED
YES
NO
INIT
H1 INITIALIZE COUNT FOR UP TO 8 USER HEADER LABELS
H4 HAVE 8 USER LABELS BEEN WRITTEN
NO
YES
J1
USERTEST
J1 LAST USER LABEL WRITTEN
NO
YES
A4
K1
COBOLWRT
K1 SVCOWAIT+4 FJ WRITE TAPEMARK
A3
J4 MSGWRITE FJ WRITE MESSAGE 4124I
K4 SVC 6 CANCEL

Chart EG. $$BOMT05: Open Input or Output, Nonstandard Labeled or Unlabeled (Part 1 of 2)

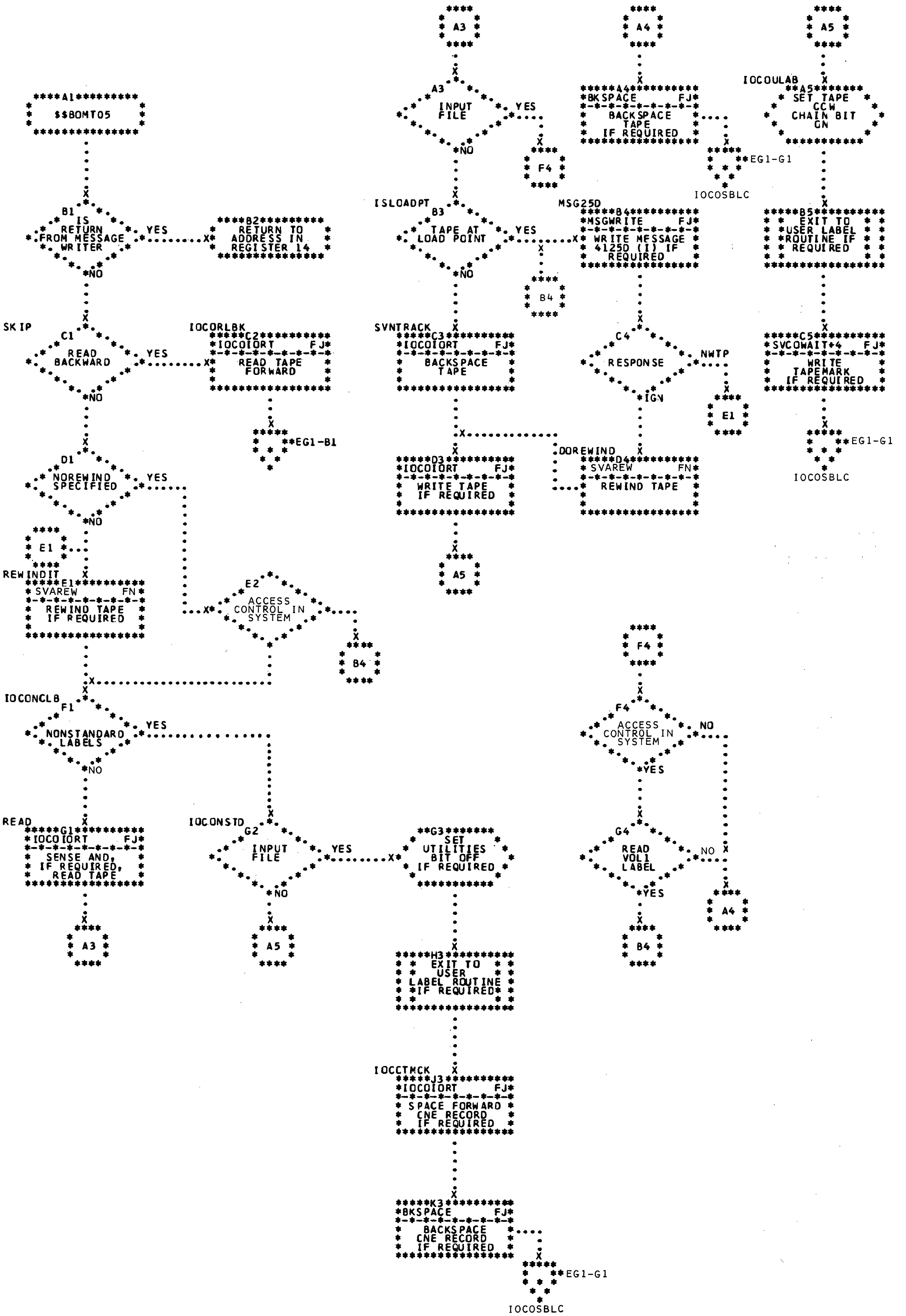


Chart EG1. $$BCMT05: Open Input or Output, Nonstandard Labeled or Unlabeled (Part 2 of 2)

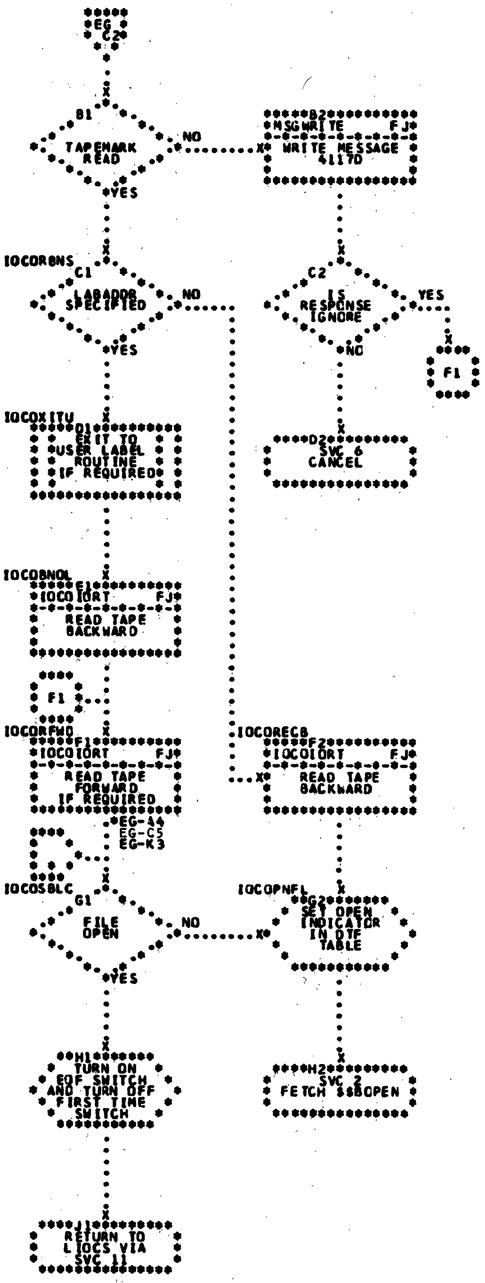


Chart EH. $$BOMT06: Open Work Files

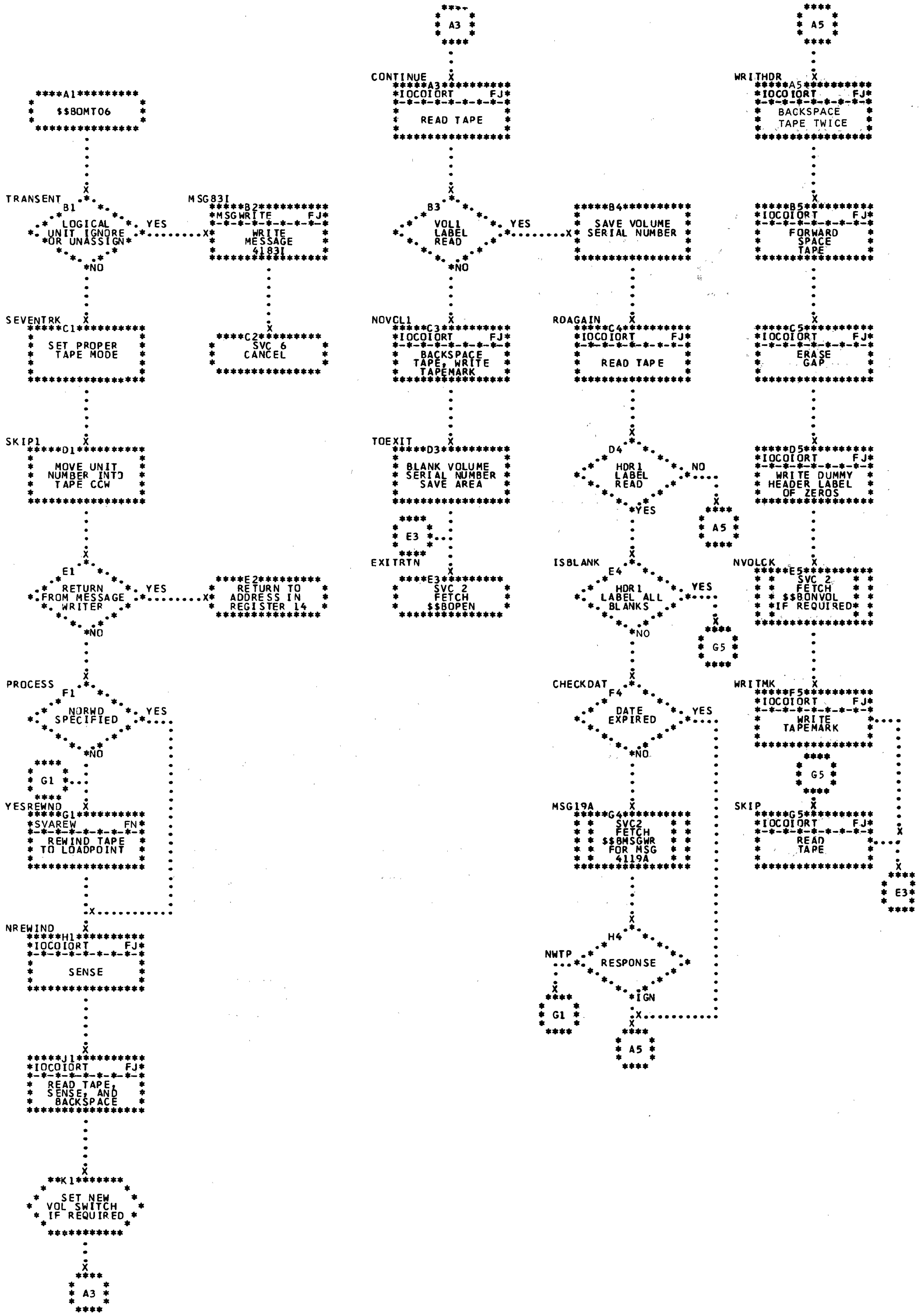


Chart EJ. $$BOMT07: Open Standard Labels, Input Forward

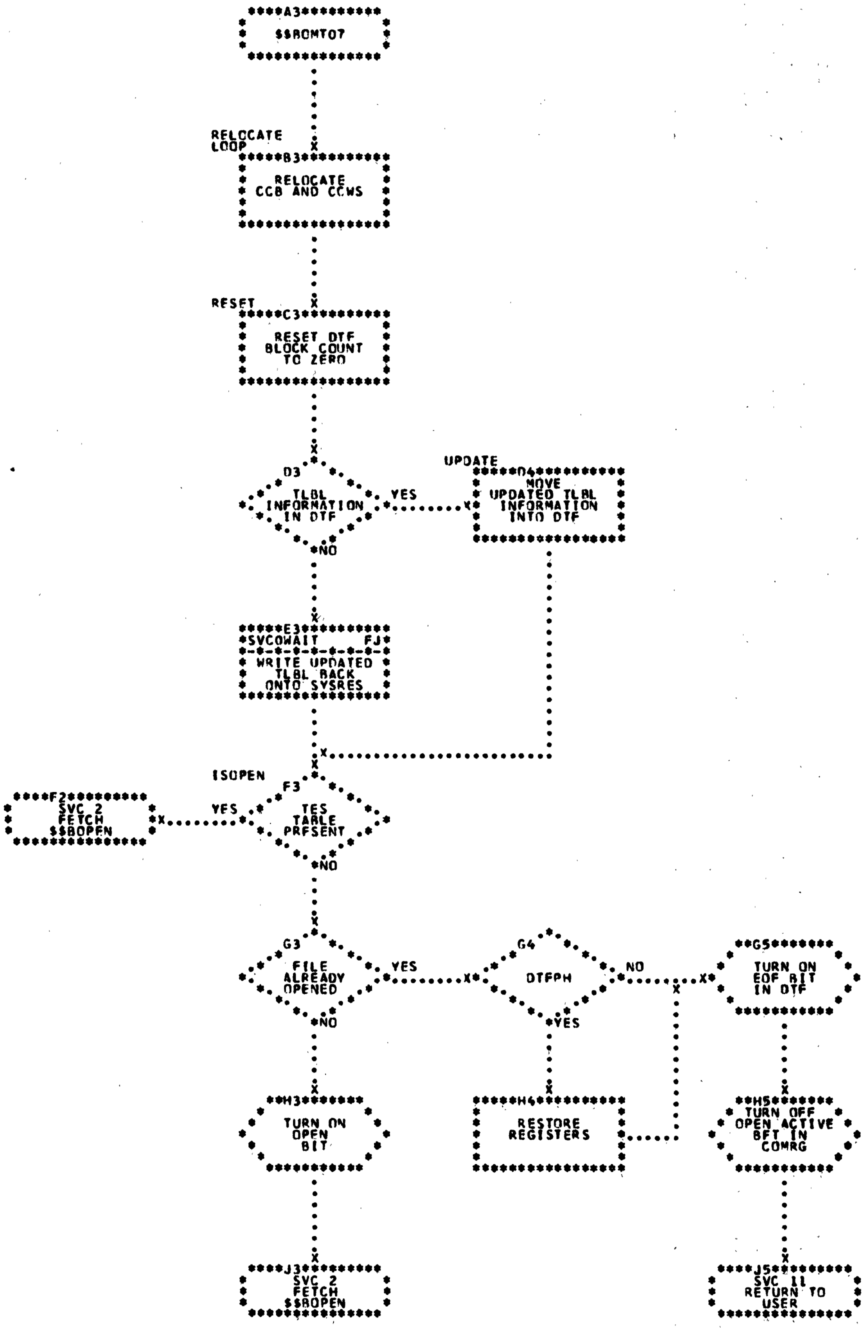


Chart EK. $$BJCOPT: Job Control Tape Open Routine, Phase 1

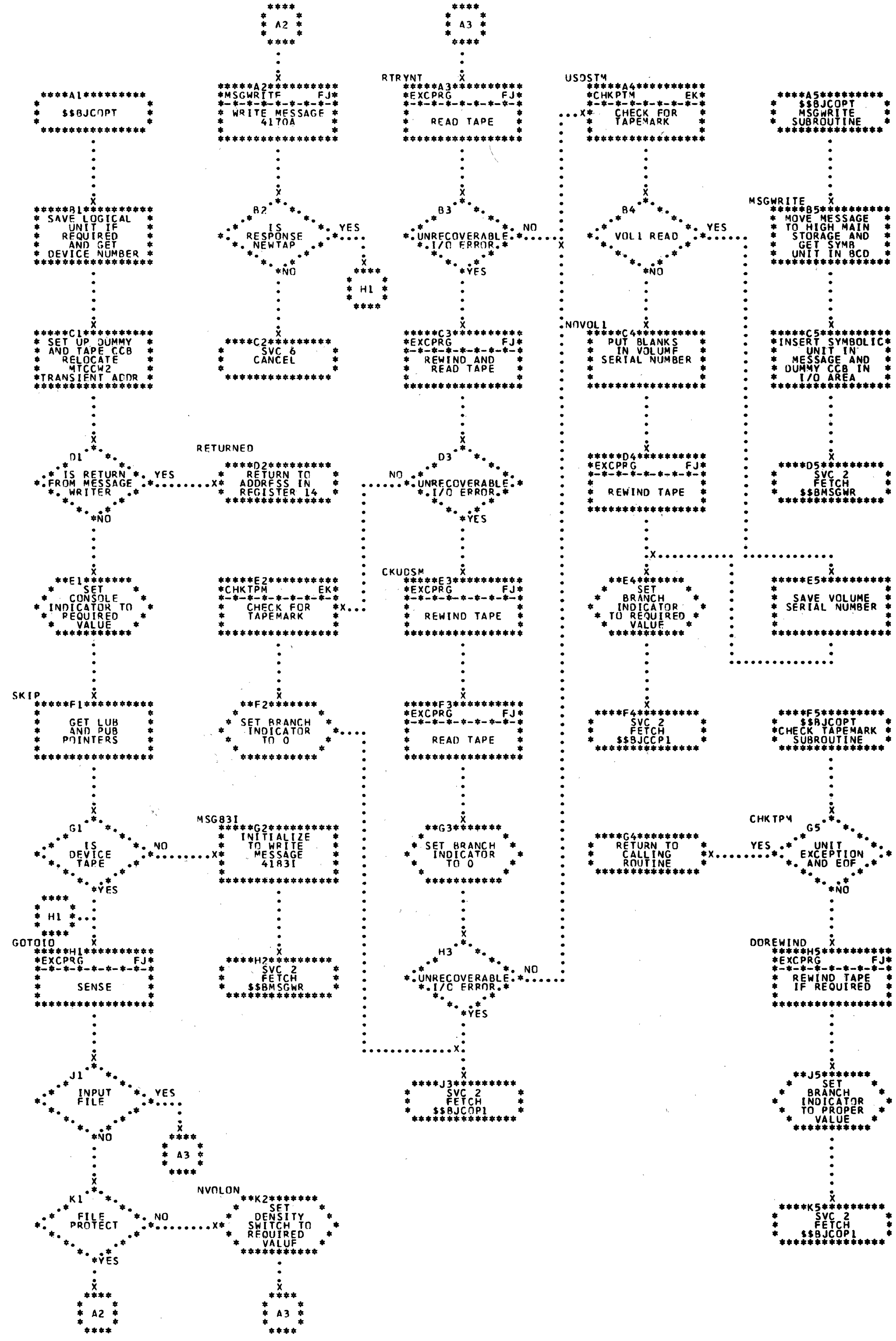


Chart EL. $$BJCOP1: Job Control Tape Open Routine, Phase 2

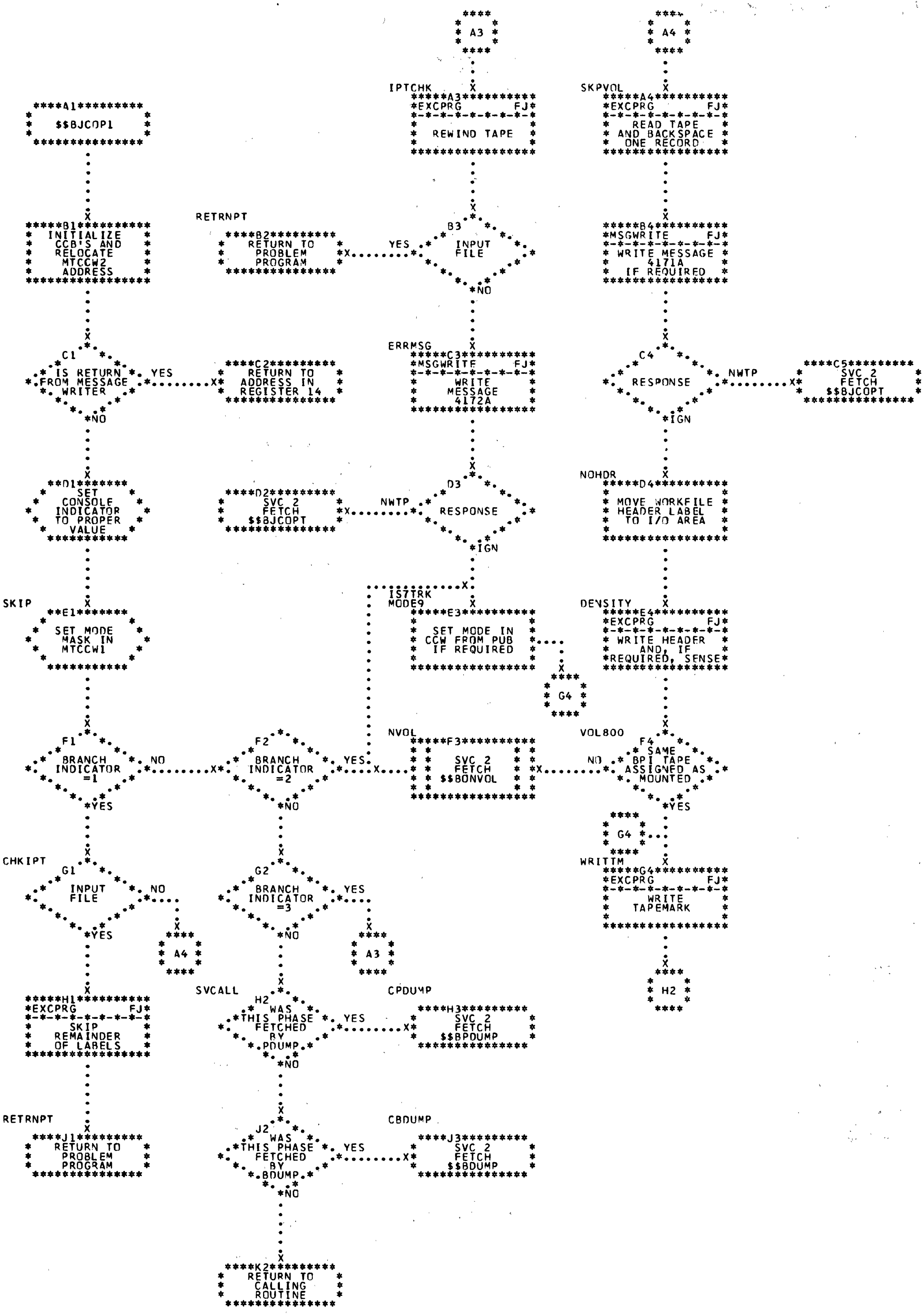


Chart EM. $$BCEOV1: EOF/EOV Monitor

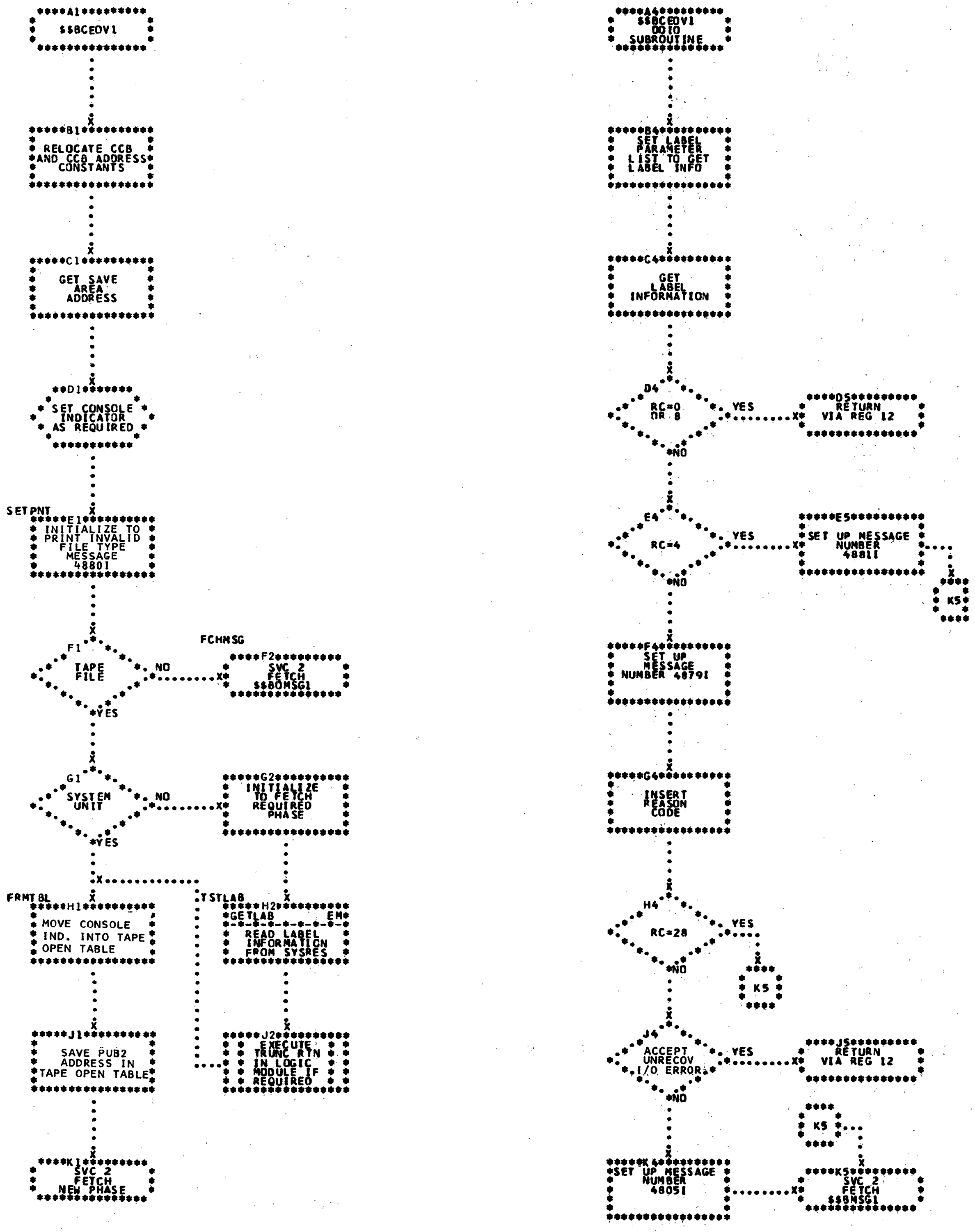

A1
$$BCEOV1
B1
RELOCATE CCB AND CCB ADDRESS CONSTANTS
C1
GET SAVE AREA ADDRESS
D1
SET CONSOLE INDICATOR AS REQUIRED
SETPNT
E1
INITIALIZE TO PRINT INVALID FILE TYPE MESSAGE 4880I
F1
TAPE FILE
NO
YES
FCHMSG
F2
SVC 2 FETCH $$BOMSG1
G1
SYSTEM UNIT
NO
YES
G2
INITIALIZE TO FETCH REQUIRED PHASE
FRMTBL
H1
MOVE CONSOLE IND. INTO TAPE OPEN TABLE
TSTLAB
H2
GETLAB
EM
READ LABEL INFORMATION FROM SYSRES
J1
SAVE PUB2 ADDRESS IN TAPE OPEN TABLE
J2
EXECUTE TRUNC RTN IN LOGIC MODULE IF REQUIRED
K1
SVC 2 FETCH NEW PHASE
A4
$$BCEOV1 DOIO SUBROUTINE
B4
SET LABEL PARAMETER LIST TO GET LABEL INFO
C4
GET LABEL INFORMATION
D4
RC=0 OR 8
YES
NO
D5
RETURN VIA REG 12
E4
RC=4
YES
NO
E5
SET UP MESSAGE NUMBER 4881I
K5
F4
SET UP MESSAGE NUMBER 4879I
G4
INSERT REASON CODE
H4
RC=28
YES
NO
K5
J4
ACCEPT UNRECOV I/O ERROR
YES
NO
J5
RETURN VIA REG 12
K5
K4
SET UP MESSAGE NUMBER 4805I
K5
SVC 2 FETCH $$BMSG1

Chart FA. $$BCMT01: Tape Close, EOF/EOV Input Forward, Phase 1

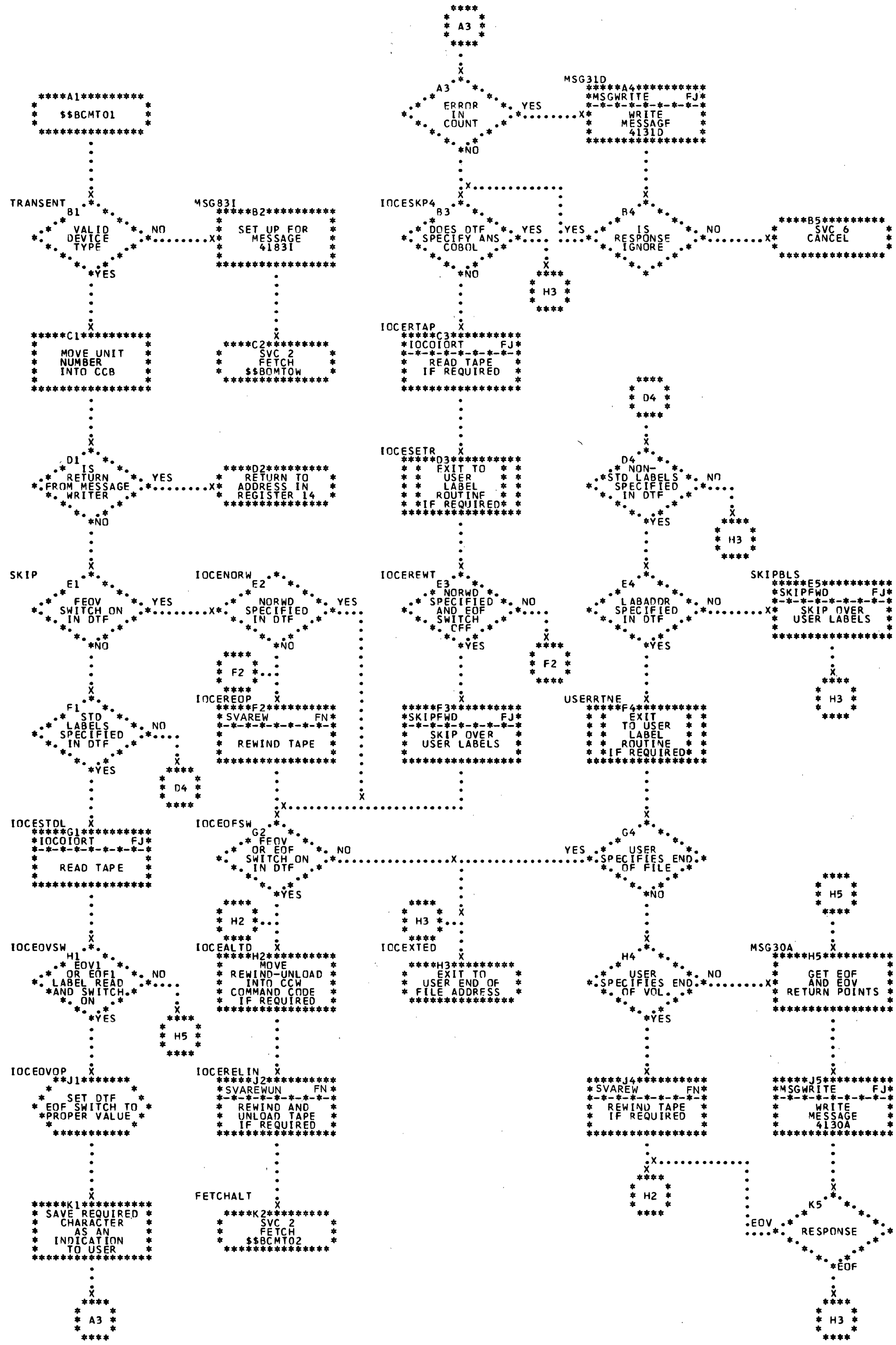


Chart FB. $$BCMT02: Tape Close, Alternate Switching for EOV

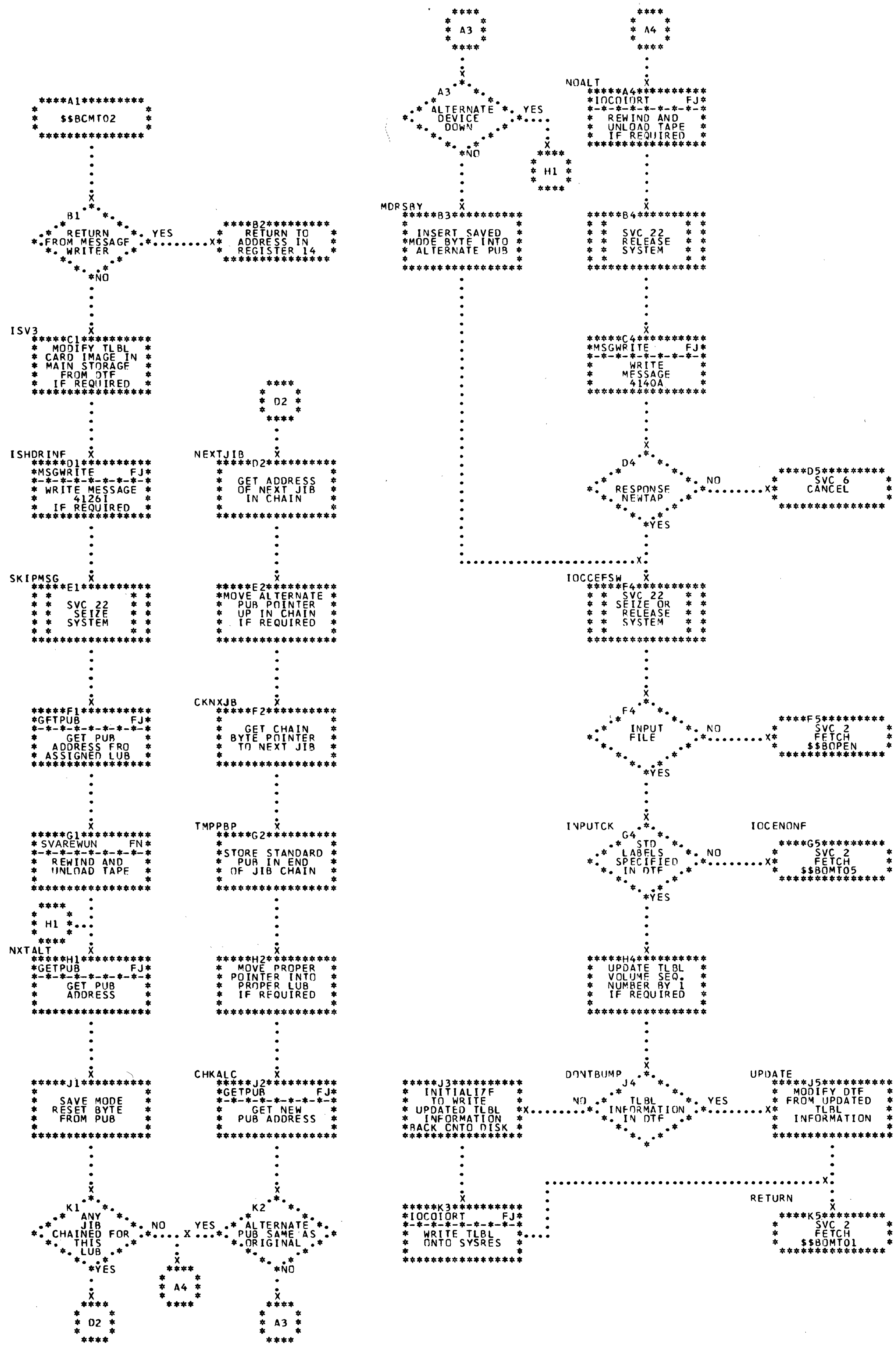

A1
$$BCMT02
B1
RETURN FROM MESSAGE WRITER
YES
NO
B2
RETURN TO ADDRESS IN REGISTER 14
ISV3
C1
MODIFY TLBL CARD IMAGE IN MAIN STORAGE FROM DTF IF REQUIRED
ISHDRINF
D1
MSGWRITE FJ
WRITE MESSAGE 4126I IF REQUIRED
SKIPMSG
E1
SVC 22 SEIZE SYSTEM
F1
GETPUB FJ
GET PUB ADDRESS FRO ASSIGNED LUB
G1
SVAREWUN FN
REWIND AND UNLOAD TAPE
H1
NXTALT
H1
GETPUB FJ
GET PUB ADDRESS
J1
SAVE MODE RESET BYTE FROM PUB
K1
ANY JIB CHAINED FOR THIS LUB
NO
YES
D2
A4
D2
NEXTJIB
D2
GET ADDRESS OF NEXT JIB IN CHAIN
E2
MOVE ALTERNATE PUB POINTER UP IN CHAIN IF REQUIRED
CKNXJB
F2
GET CHAIN BYTE POINTER TO NEXT JIB
TMPPBP
G2
STORE STANDARD PUB IN END OF JIB CHAIN
H2
MOVE PROPER POINTER INTO PROPER LUB IF REQUIRED
CHKALC
J2
GETPUB FJ
GET NEW PUB ADDRESS
K2
ALTERNATE PUB SAME AS ORIGINAL
YES
NO
A3
A3
A3
ALTERNATE DEVICE DOWN
YES
NO
H1
MDRSBY
B3
INSERT SAVED MODE BYTE INTO ALTERNATE PUB
A4
NOALT
A4
IOCOIORT FJ
REWIND AND UNLOAD TAPE IF REQUIRED
B4
SVC 22 RELEASE SYSTEM
C4
MSGWRITE FJ
WRITE MESSAGE 4140A
D4
RESPONSE NEWTAP
NO
YES
D5
SVC 6 CANCEL
IOCCEFSW
E4
SVC 22 SEIZE OR RELEASE SYSTEM
F4
INPUT FILE
NO
YES
F5
SVC 2 FETCH $$BOPEN
INPUTCK
G4
STD LABELS SPECIFIED IN DTF
NO
YES
IOCENONF
G5
SVC 2 FETCH $$BOMT05
H4
UPDATE TLBL VOLUME SEQ. NUMBER BY 1 IF REQUIRED
J3
INITIALIZE TO WRITE UPDATED TLBL INFORMATION BACK CNTO DISK
DONTBUMP
J4
TLBL INFORMATION IN DTF
NO
YES
UPDATE
J5
MODIFY DTF FROM UPDATED TLBL INFORMATION
K3
IOCOIORT FJ
WRITE TLBL ONTO SYSRES
RETURN
K5
SVC 2 FETCH $$BOMT01

Chart FC. $$BCMT03: Tape Close, EOF Input Backward

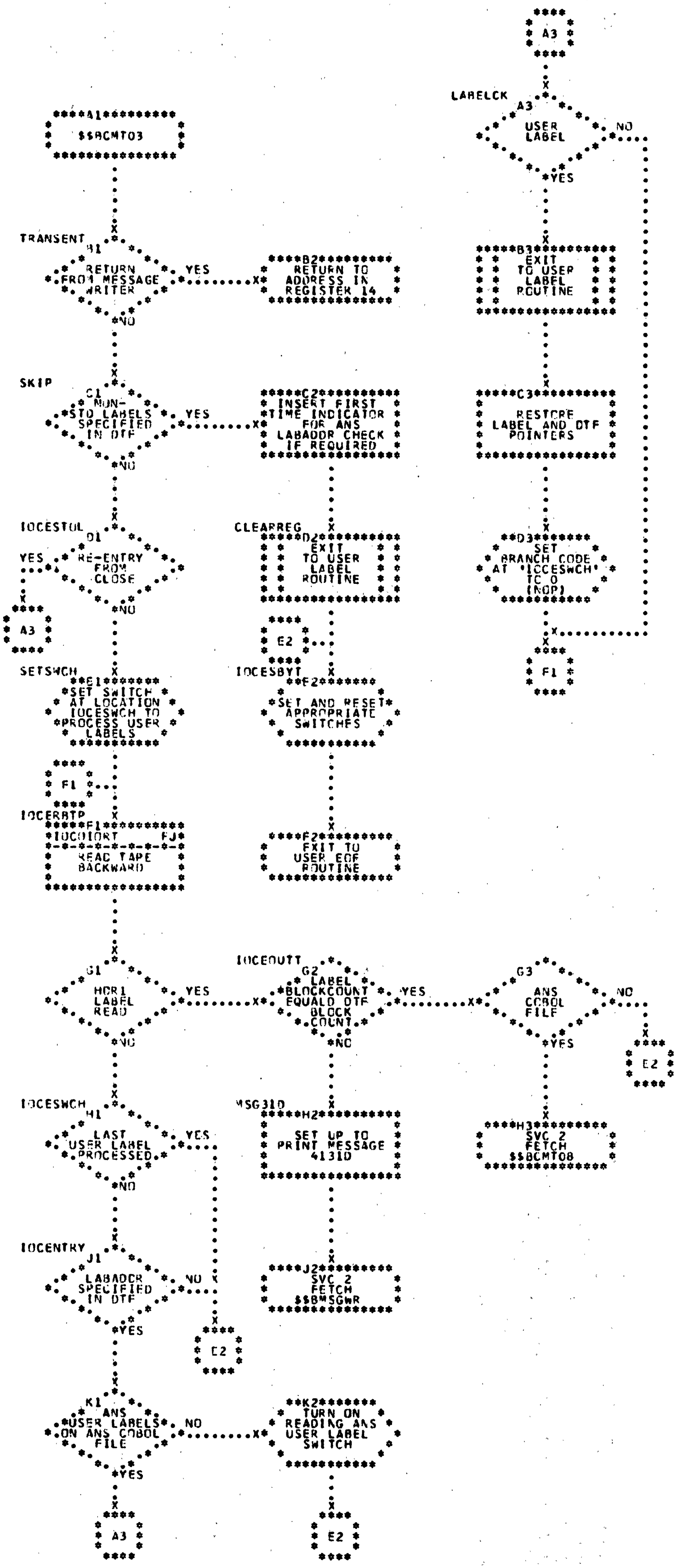

A1 $$BCMT03
TRANSENT B1 RETURN FROM MESSAGE WRITER
YES
NO
B2 RETURN TO ADDRESS IN REGISTER 14
SKIP C1 NON-STD LABELS SPECIFIED IN DTF
YES
NO
C2 INSERT FIRST TIME INDICATOR FOR ANS LABADDR CHECK IF REQUIRED
IOCESTUL D1 RE-ENTRY FROM CLOSE
YES
NO
A3
CLEARREG D2 EXIT TO USER LABEL ROUTINE
E2
SETSWCH E1 SET SWITCH AT LOCATION IOCESWCH TO PROCESS USER LABELS
IOCESBYT E2 SET AND RESET APPROPRIATE SWITCHES
F1
IOCERBTP F1 IOCOIORT FJ READ TAPE BACKWARD
F2 EXIT TO USER EOF ROUTINE
G1 HDR1 LABEL READ
YES
NO
IOCEOUTT G2 LABEL BLOCKCOUNT EQUALD DTF BLOCK COUNT
YES
NO
G3 ANS COBOL FILE
NO
YES
E2
IOCESWCH H1 LAST USER LABEL PROCESSED
YES
NO
MSG31D H2 SET UP TO PRINT MESSAGE 4131D
H3 SVC 2 FETCH $$BCMT08
IOCENTRY J1 LABADDR SPECIFIED IN DTF
NO
YES
E2
J2 SVC 2 FETCH $$BMSGWR
K1 ANS USER LABELS ON ANS COBOL FILE
NO
YES
K2 TURN ON READING ANS USER LABEL SWITCH
A3
E2
A3
LABELCK A3 USER LABEL
NO
YES
B3 EXIT TO USER LABEL ROUTINE
C3 RESTORE LABEL AND DTF POINTERS
D3 SET BRANCH CODE AT 'ICCESWCH' TC 0 (NOP)
F1

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

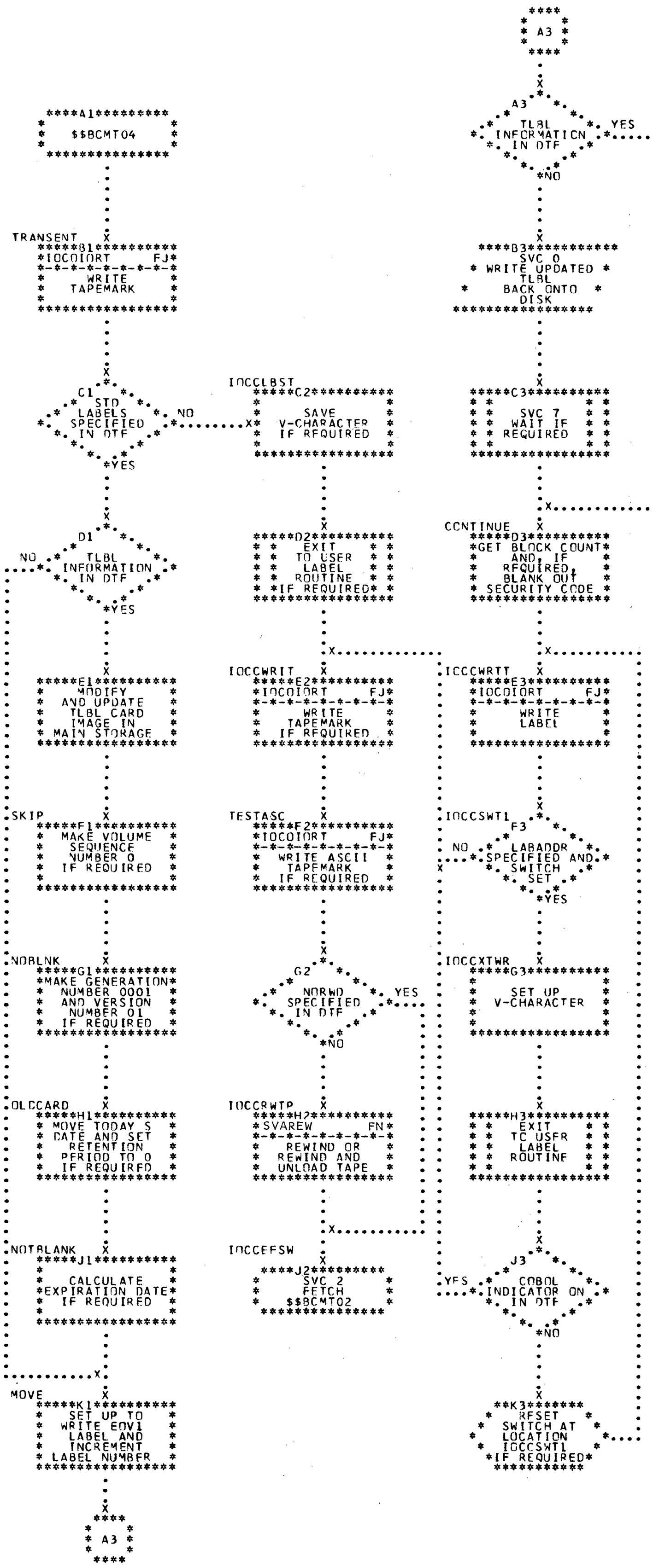

A3
A3 TLBL INFORMATICN IN DTF
YES
NO
UPDATE
A4 UPDATE DTF WITH NEW TLBL INFORMATION
A1 $$BCMT04
TRANSENT
B1 IOCOIORT FJ WRITE TAPEMARK
B3 SVC 0 WRITE UPDATED TLBL BACK ONTO DISK
C1 STD LABELS SPECIFIED IN DTF
NO
YES
IOCCLBST
C2 SAVE V-CHARACTER IF REQUIRED
C3 SVC 7 WAIT IF REQUIRED
D1 TLBL INFORMATION IN DTF
NO
YES
D2 EXIT TO USER LABEL ROUTINE IF REQUIRED
CONTINUE
D3 GET BLOCK COUNT AND, IF REQUIRED, BLANK OUT SECURITY CODE
E1 MODIFY AND UPDATE TLBL CARD IMAGE IN MAIN STORAGE
IOCCWRIT
E2 IOCOIORT FJ WRITE TAPEMARK IF REQUIRED
IOCCWRTT
E3 IOCOIORT FJ WRITE LABEL
SKIP
F1 MAKE VOLUME SEQUENCE NUMBER 0 IF REQUIRED
TESTASC
F2 IOCOIORT FJ WRITE ASCII TAPEMARK IF REQUIRED
IOCCSWT1
F3 LABADDR SPECIFIED AND SWITCH SET
NO
YES
NOBLNK
G1 MAKE GENERATION NUMBER 0001 AND VERSION NUMBER 01 IF REQUIRED
G2 NORWD SPECIFIED IN DTF
YES
NO
IOCCXTWR
G3 SET UP V-CHARACTER
OLCCARD
H1 MOVE TODAY S DATE AND SET RETENTION PERIOD TO 0 IF REQUIRED
IOCCRWTP
H2 SVAREW FN REWIND OR REWIND AND UNLOAD TAPE
H3 EXIT TO USER LABEL ROUTINE
NOTBLANK
J1 CALCULATE EXPIRATION DATE IF REQUIRED
IOCCEFSW
J2 SVC 2 FETCH $$BCMT02
J3 COBOL INDICATOR ON IN DTF
YES
NO
MOVE
K1 SET UP TO WRITE EOV1 LABEL AND INCREMENT LABEL NUMBER
K3 RESET SWITCH AT LOCATION IOCCSWT1 IF REQUIRED
A3

Chart FE. $$BCMT05: Output Close Standard, Nonstandard, and Unlabeled Files, all Types Except Work Files (Part 1 of 2)

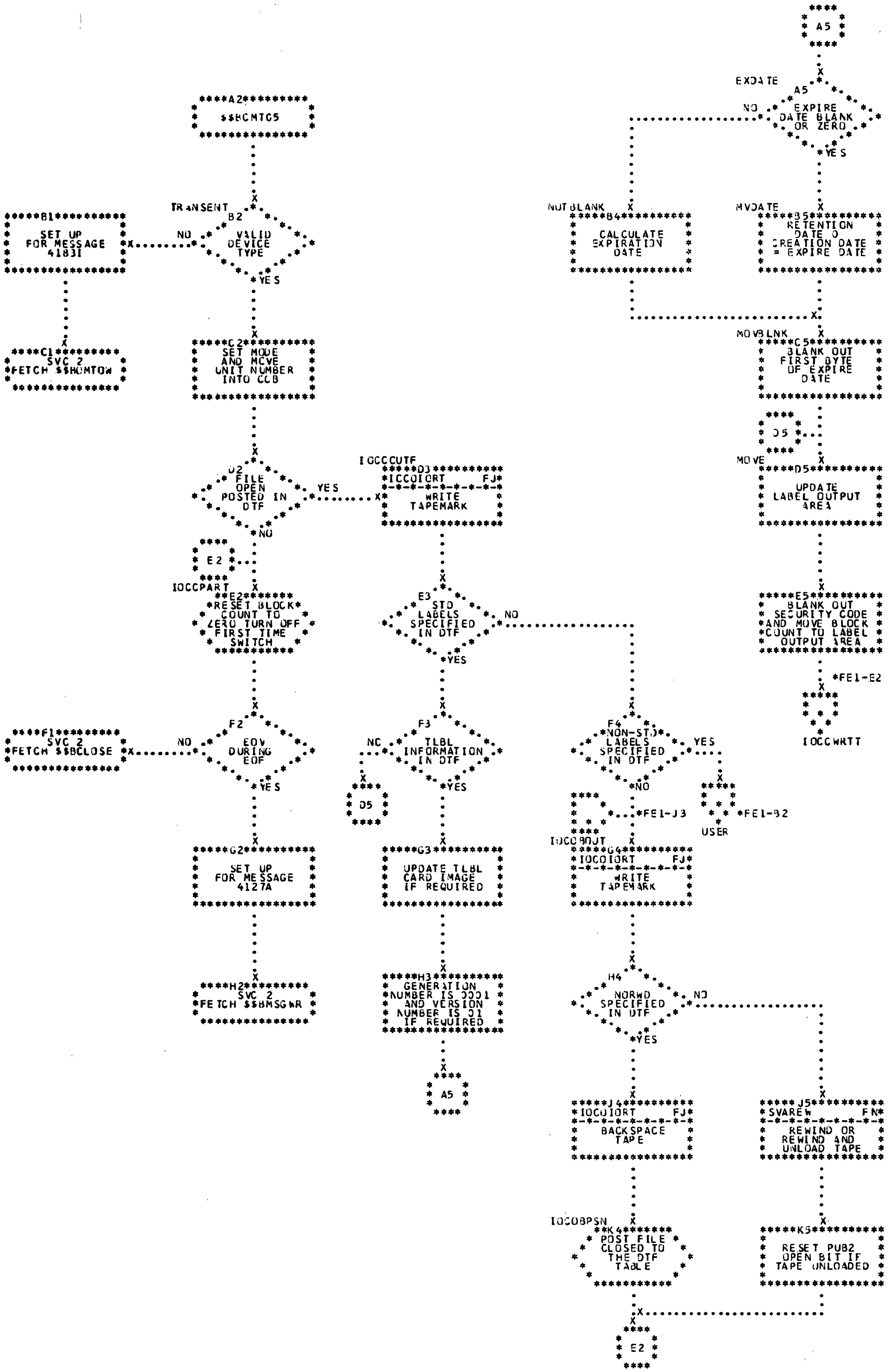

A2
$$BCMT05
TRANSENT
B2
VALID DEVICE TYPE
NO
YES
B1
SET UP FOR MESSAGE 4183I
C1
SVC 2
FETCH $$BCMTOW
C2
SET MODE AND MOVE UNIT NUMBER INTO CCB
D2
FILE OPEN POSTED IN DTF
YES
NO
IOCCCUTF
D3
IOCCOIORT FJ
WRITE TAPEMARK
E2
IOCCPART
E2
RESET BLOCK COUNT TO ZERO TURN OFF FIRST TIME SWITCH
E3
STD LABELS SPECIFIED IN DTF
NO
YES
F1
SVC 2
FETCH $$BCLOSE
F2
EOV DURING EOF
NO
YES
F3
TLBL INFORMATION IN DTF
NO
YES
D5
G2
SET UP FOR MESSAGE 4127A
G3
UPDATE TLBL CARD IMAGE IF REQUIRED
H2
SVC 2
FETCH $$BMSGWR
H3
GENERATION NUMBER IS 0001 AND VERSION NUMBER IS 01 IF REQUIRED
A5
F4
NON-STD LABELS SPECIFIED IN DTF
YES
NO
FE1-J3
FE1-B2
USER
IOCOBOUT
G4
IOCOIORT FJ
WRITE TAPEMARK
H4
NORWD SPECIFIED IN DTF
NO
YES
J4
IOCOIORT FJ
BACKSPACE TAPE
J5
SVAREW FN
REWIND OR REWIND AND UNLOAD TAPE
IOCOBPSN
K4
POST FILE CLOSED TO THE DTF TABLE
K5
RESET PUB2 OPEN BIT IF TAPE UNLOADED
E2
A5
EXDATE
A5
EXPIRE DATE BLANK OR ZERO
NO
YES
NOTBLANK
B4
CALCULATE EXPIRATION DATE
MVDATE
B5
RETENTION DATE 0 CREATION DATE = EXPIRE DATE
MOVBLNK
C5
BLANK OUT FIRST BYTE OF EXPIRE DATE
D5
MOVE
D5
UPDATE LABEL OUTPUT AREA
E5
BLANK OUT SECURITY CODE AND MOVE BLOCK COUNT TO LABEL OUTPUT AREA
FE1-E2
IOCCWRTT

Chart FE1. $$BCMT05; Output Close Standard, Nonstandard, and Unlabeled Files; all Types Except Work Files (Part 2 of 2)

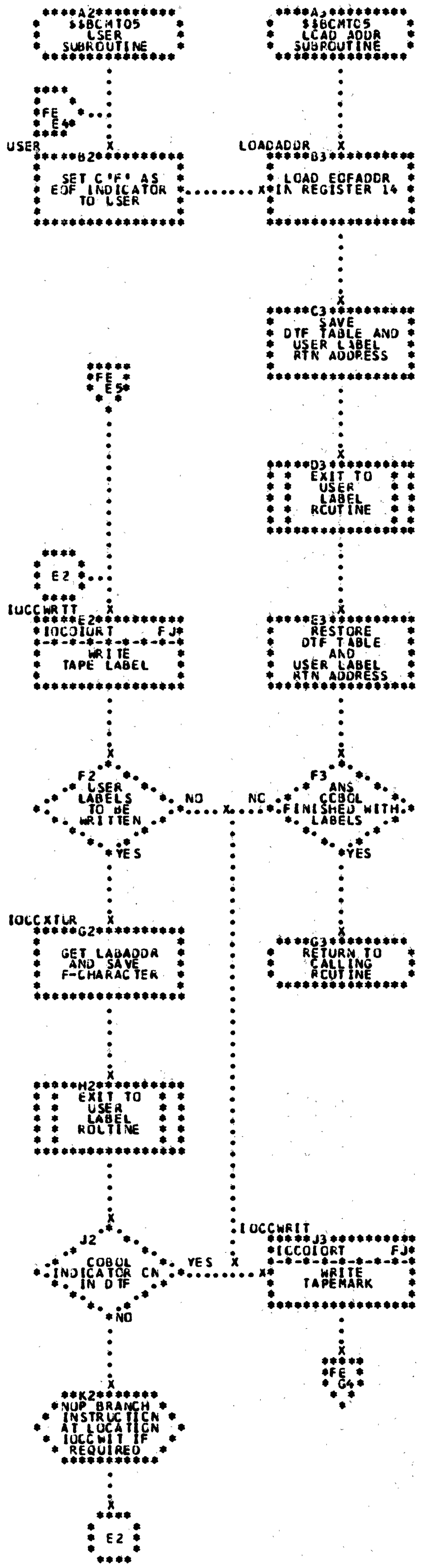


Chart FF. $$BCMT08: Input Close Standard, Nonstandard, and Unlabeled Files, all Types Except Work Files

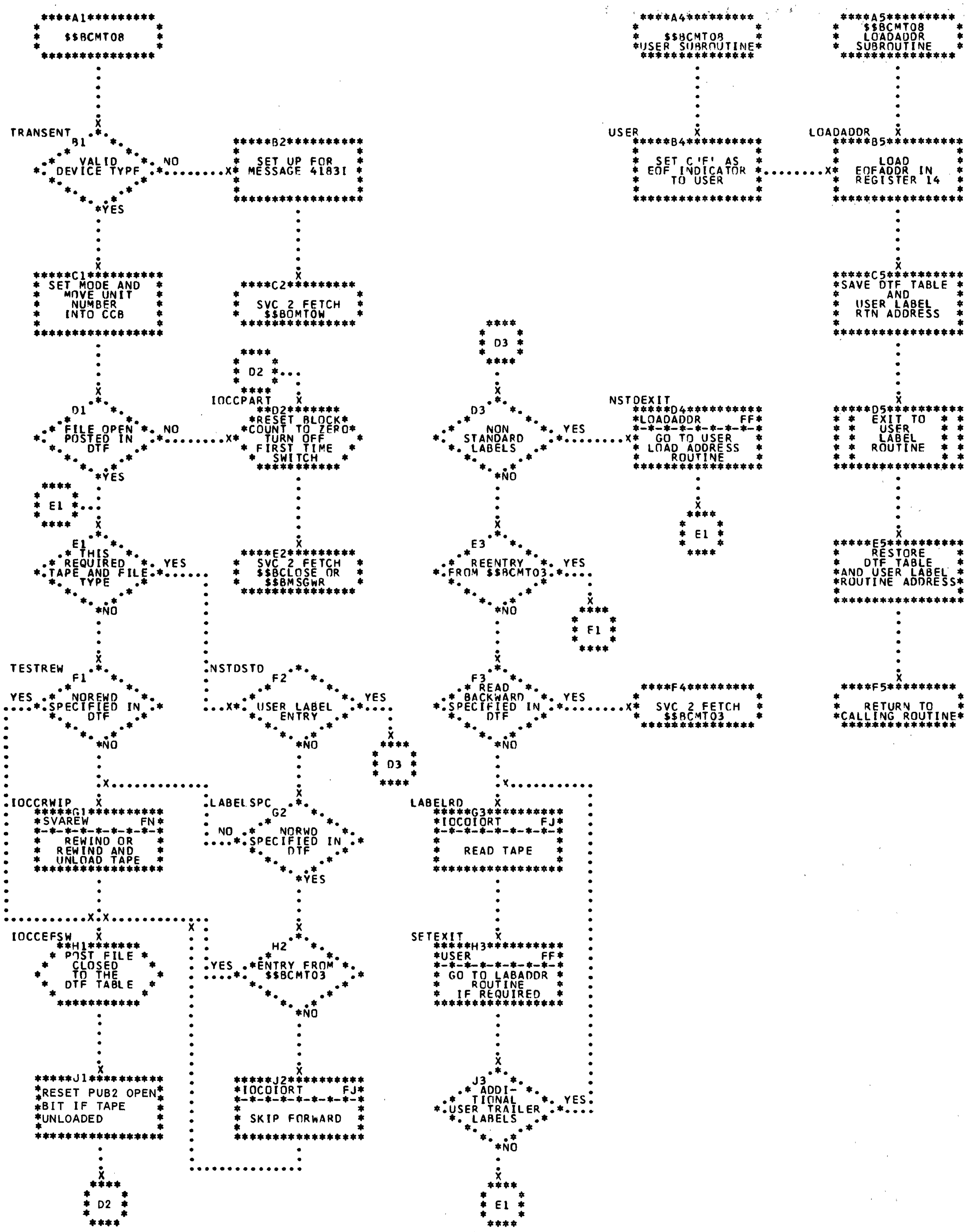

A1
$$BCMT08
TRANSENT
B1
VALID DEVICE TYPF
NO
B2
SET UP FOR MESSAGE 4183I
YES
C1
SET MODE AND MOVE UNIT NUMBER INTO CCB
C2
SVC 2 FETCH $$BOMTOW
D2
IOCCPART
D1
FILE OPEN POSTED IN DTF
NO
D2
RESET BLOCK COUNT TO ZERO TURN OFF FIRST TIME SWITCH
YES
E1
E1
THIS REQUIRED TAPE AND FILE TYPE
YES
E2
SVC 2 FETCH $$BCLOSE OR $$BMSGWR
NO
TESTREW
F1
NOREWD SPECIFIED IN DTF
YES
NO
NSTDSTD
F2
USER LABEL ENTRY
YES
NO
D3
IOCCRWIP
G1
SVAREW FN
REWIND OR REWIND AND UNLOAD TAPE
LABELSPC
G2
NORWD SPECIFIED IN DTF
NO
YES
IOCCEFSW
H1
POST FILE CLOSED TO THE DTF TABLE
H2
ENTRY FROM $$BCMT03
YES
NO
J1
RESET PUB2 OPEN BIT IF TAPE UNLOADED
J2
IOCOIORT FJ
SKIP FORWARD
D2
D3
D3
NON STANDARD LABELS
YES
NO
E3
REENTRY FROM $$BCMT03
YES
F1
NO
F3
READ BACKWARD SPECIFIED IN DTF
YES
NO
LABELRD
G3
IOCOIORT FJ
READ TAPE
SETEXIT
H3
USER FF
GO TO LABADDR ROUTINE IF REQUIRED
J3
ADDI- TIONAL USER TRAILER LABELS
YES
NO
E1
A4
$$BCMT08 USER SUBROUTINE
USER
B4
SET C'F' AS EOF INDICATOR TO USER
NSTDEXIT
D4
LOADADDR FF
GO TO USER LOAD ADDRESS ROUTINE
E1
F4
SVC 2 FETCH $$BCMT03
A5
$$BCMT08 LOADADDR SUBROUTINE
LOADADDR
B5
LOAD EOFADDR IN REGISTER 14
C5
SAVE DTF TABLE AND USER LABEL RTN ADDRESS
D5
EXIT TO USER LABEL ROUTINE
E5
RESTORE DTF TABLE AND USER LABEL ROUTINE ADDRESS
F5
RETURN TO CALLING ROUTINE

Chart FG. $$BCMT06: Close Tape Work Files

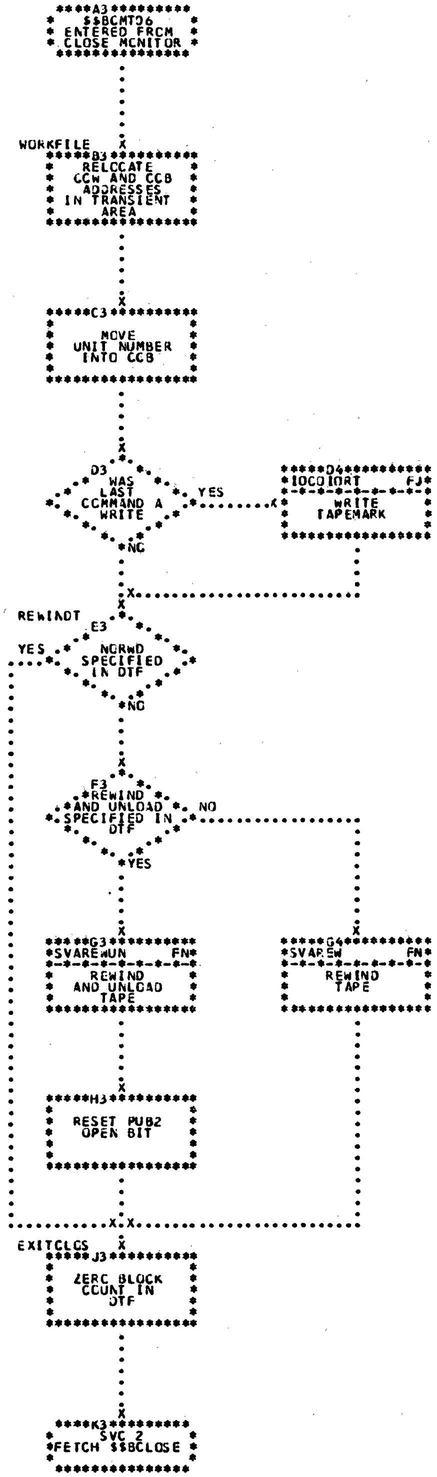


Chart FH. $$BCMT07: Tape Close, Alternate Switching for System Units

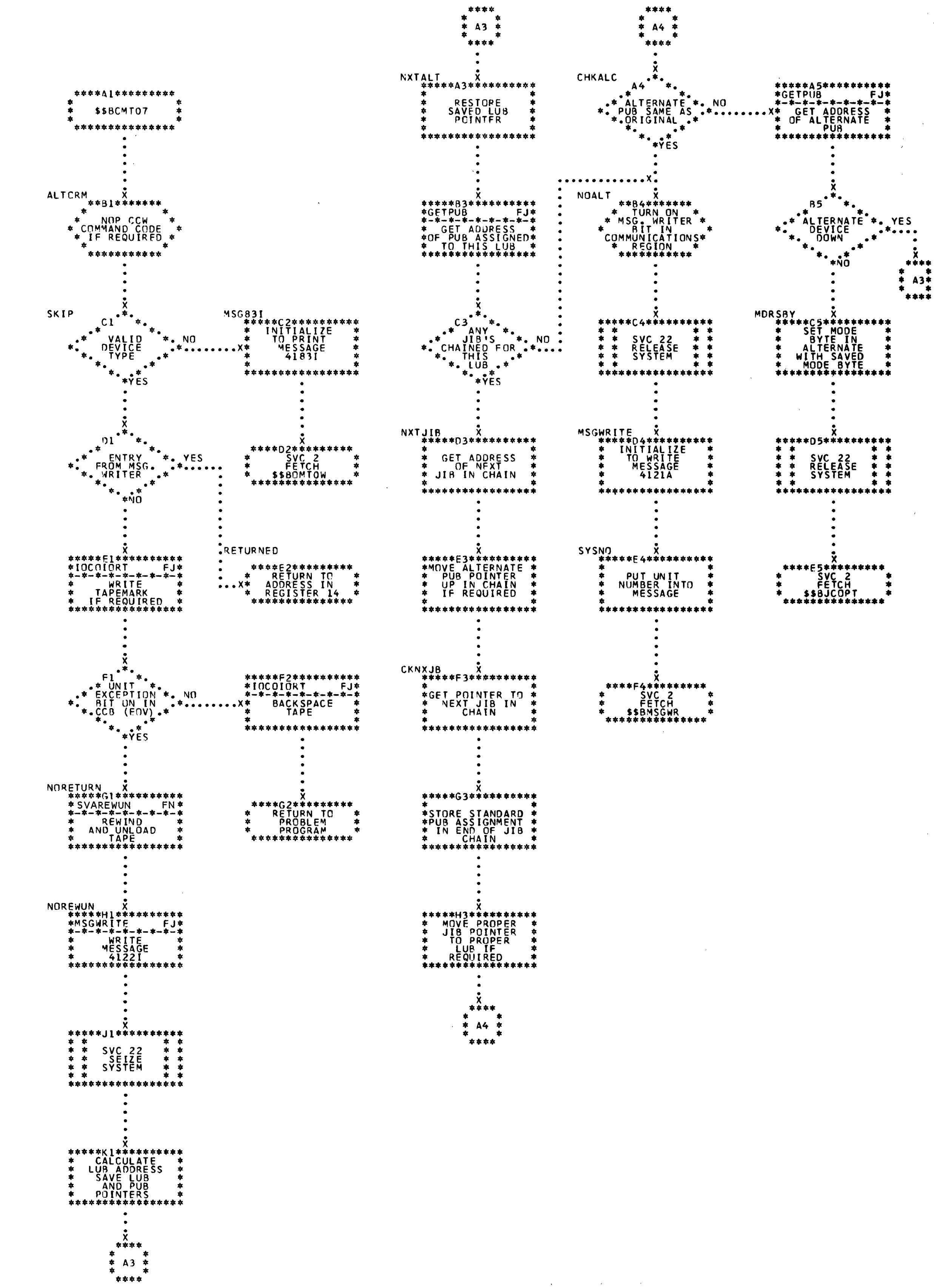

A1
$$BCMT07
ALTCRM
B1
NOP CCW COMMAND CODE IF REQUIRED
SKIP
C1
VALID DEVICE TYPE
NO
YES
MSG83I
C2
INITIALIZE TO PRINT MESSAGE 4183I
D1
ENTRY FROM MSG. WRITER
YES
NO
D2
SVC 2 FETCH $$BOMTOW
RETURNED
E1
IOCOIORT FJ
WRITE TAPEMARK IF REQUIRED
E2
RETURN TO ADDRESS IN REGISTER 14
F1
UNIT EXCEPTION BIT ON IN CCB (EOV)
NO
YES
F2
IOCOIORT FJ
BACKSPACE TAPE
NORETURN
G1
SVAREWUN FN
REWIND AND UNLOAD TAPE
G2
RETURN TO PROBLEM PROGRAM
NOREWUN
H1
MSGWRITE FJ
WRITE MESSAGE 4122I
J1
SVC 22 SEIZE SYSTEM
K1
CALCULATE LUB ADDRESS SAVE LUB AND PUB POINTERS
A3
A3
NXTALT
A3
RESTORE SAVED LUB POINTER
B3
GETPUB FJ
GET ADDRESS OF PUB ASSIGNED TO THIS LUB
C3
ANY JIB'S CHAINED FOR THIS LUB
NO
YES
NXTJIB
D3
GET ADDRESS OF NEXT JIB IN CHAIN
E3
MOVE ALTERNATE PUB POINTER UP IN CHAIN IF REQUIRED
CKNXJB
F3
GET POINTER TO NEXT JIB IN CHAIN
G3
STORE STANDARD PUB ASSIGNMENT IN END OF JIB CHAIN
H3
MOVE PROPER JIB POINTER TO PROPER LUB IF REQUIRED
A4
A4
CHKALC
A4
ALTERNATE PUB SAME AS ORIGINAL
NO
YES
NOALT
B4
TURN ON MSG. WRITER BIT IN COMMUNICATIONS REGION
C4
SVC 22 RELEASE SYSTEM
MSGWRITE
D4
INITIALIZE TO WRITE MESSAGE 4121A
SYSNO
E4
PUT UNIT NUMBER INTO MESSAGE
F4
SVC 2 FETCH $$BMSGWR
A5
GETPUB FJ
GET ADDRESS OF ALTERNATE PUB
B5
ALTERNATE DEVICE DOWN
YES
NO
A3
MDRSBY
C5
SET MODE BYTE IN ALTERNATE WITH SAVED MODE BYTE
D5
SVC 22 RELEASE SYSTEM
E5
SVC 2 FETCH $$BJCOPT

Chart FJ. Tape Open/Close Subroutines

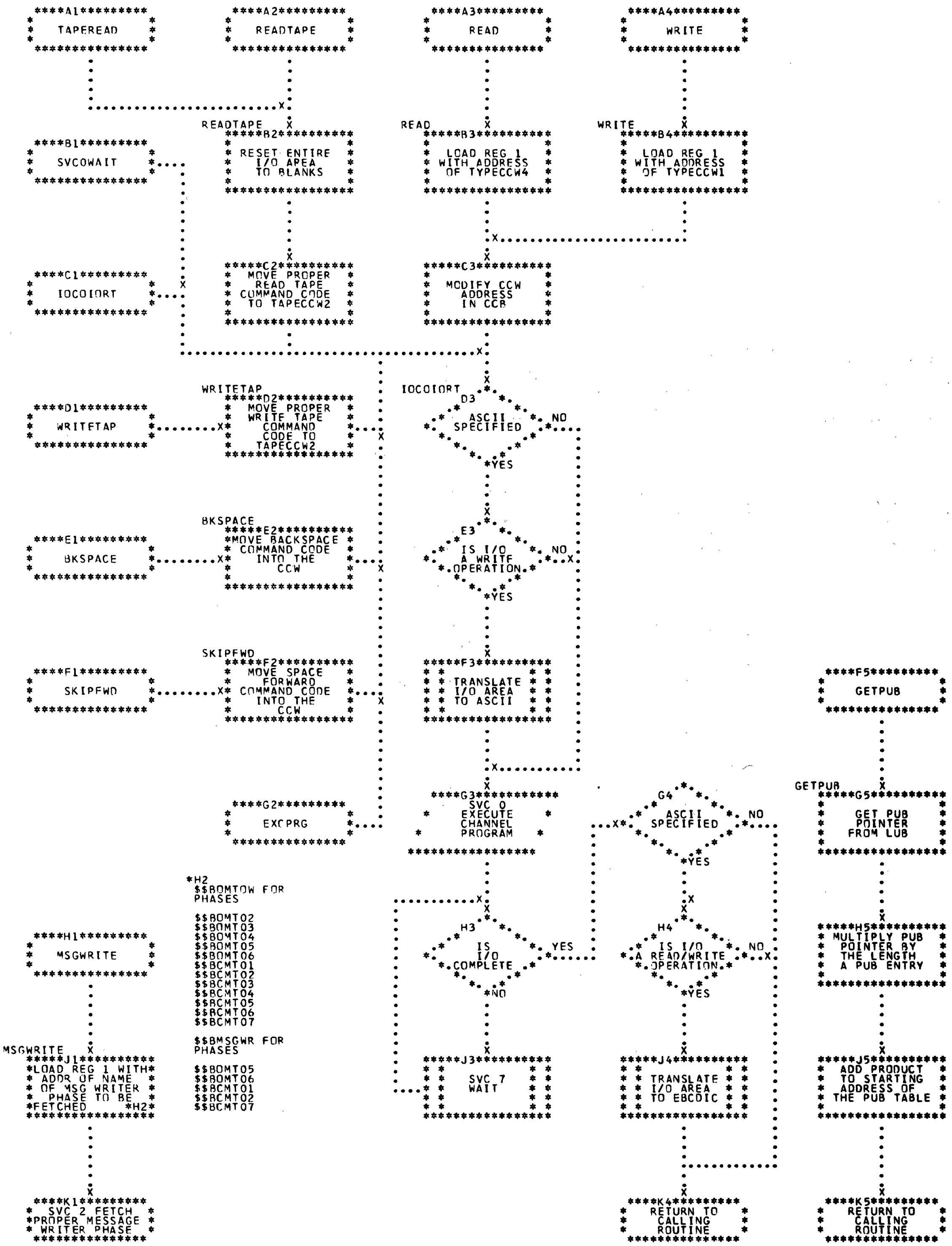

A1
TAPEREAD
A2
READTAPE
A3
READ
A4
WRITE
B1
SVCOWAIT
READTAPE
B2
RESET ENTIRE I/O AREA TO BLANKS
READ
B3
LOAD REG 1 WITH ADDRESS OF TYPECCW4
WRITE
B4
LOAD REG 1 WITH ADDRESS OF TYPECCW1
C1
IOCOIORT
C2
MOVE PROPER READ TAPE COMMAND CODE TO TAPECCW2
C3
MODIFY CCW ADDRESS IN CCB
D1
WRITETAP
WRITETAP
D2
MOVE PROPER WRITE TAPE COMMAND CODE TO TAPECCW2
IOCOIORT
D3
ASCII SPECIFIED
NO
YES
E1
BKSPACE
BKSPACE
E2
MOVE BACKSPACE COMMAND CODE INTO THE CCW
E3
IS I/O A WRITE OPERATION
NO
YES
F1
SKIPFWD
SKIPFWD
F2
MOVE SPACE FORWARD COMMAND CODE INTO THE CCW
F3
TRANSLATE I/O AREA TO ASCII
F5
GETPUB
G2
EXCPRG
G3
SVC 0 EXECUTE CHANNEL PROGRAM
G4
ASCII SPECIFIED
NO
YES
GETPUB
G5
GET PUB POINTER FROM LUB
H2
$$BOMTOW FOR PHASES
$$BOMT02
$$BOMT03
$$BOMT04
$$BOMT05
$$BOMT06
$$BCMT01
$$BCMT02
$$BCMT03
$$BCMT04
$$BCMT05
$$BCMT06
$$BCMT07
$$BMSGWR FOR PHASES
$$BOMT05
$$BOMT06
$$BCMT01
$$BCMT02
$$BCMT07
H1
MSGWRITE
H3
IS I/O COMPLETE
YES
NO
H4
IS I/O A READ/WRITE OPERATION
NO
YES
H5
MULTIPLY PUB POINTER BY THE LENGTH A PUB ENTRY
MSGWRITE
J1
LOAD REG 1 WITH ADDR OF NAME OF MSG WRITER PHASE TO BE FETCHED
H2
J3
SVC 7 WAIT
J4
TRANSLATE I/O AREA TO EBCDIC
J5
ADD PRODUCT TO STARTING ADDRESS OF THE PUB TABLE
K1
SVC 2 FETCH PROPER MESSAGE WRITER PHASE
K4
RETURN TO CALLING ROUTINE
K5
RETURN TO CALLING ROUTINE

Chart FK. $$BOMTOM and $$BOMTOW: Tape Open Message Writers

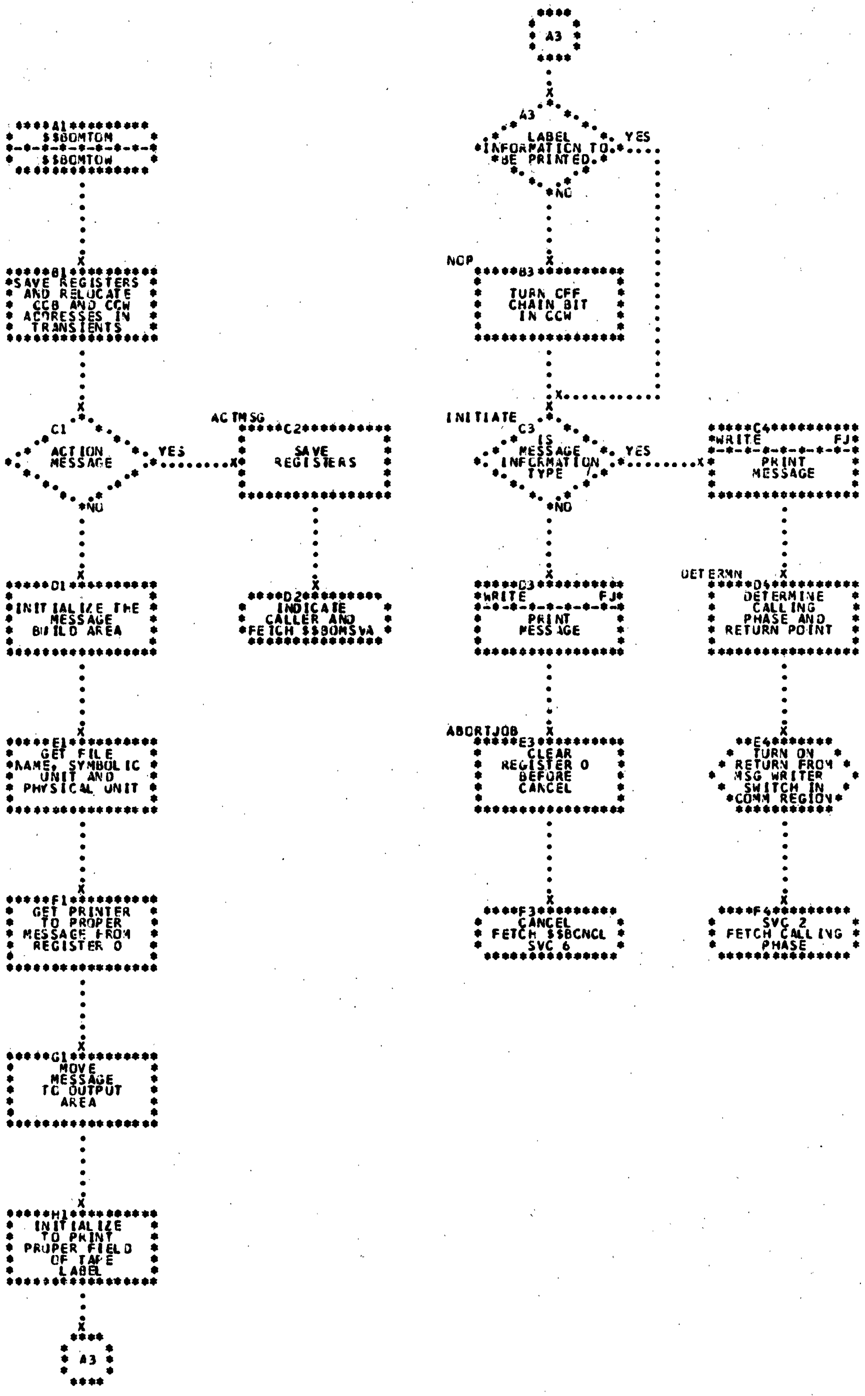

A1
$$BOMTOM
$$BOMTOW
B1
SAVE REGISTERS AND RELOCATE CCB AND CCW ADDRESSES IN TRANSIENTS
C1
ACTION MESSAGE
YES
NO
ACTMSG
C2
SAVE REGISTERS
D1
INITIALIZE THE MESSAGE BUILD AREA
D2
INDICATE CALLER AND FETCH $$BOMSVA
E1
GET FILE NAME, SYMBOLIC UNIT AND PHYSICAL UNIT
F1
GET PRINTER TO PROPER MESSAGE FROM REGISTER 0
G1
MOVE MESSAGE TC OUTPUT AREA
H1
INITIALIZE TO PRINT PROPER FIELD OF TAPE LABEL
A3
A3
A3
LABEL INFORMATION TO BE PRINTED
YES
NO
NOP
B3
TURN OFF CHAIN BIT IN CCW
INITIATE
C3
IS MESSAGE INFORMATION TYPE
YES
NO
C4
WRITE FJ
PRINT MESSAGE
D3
WRITE FJ
PRINT MESSAGE
DETERMN
D4
DETERMINE CALLING PHASE AND RETURN POINT
ABORTJOB
E3
CLEAR REGISTER 0 BEFORE CANCEL
E4
TURN ON RETURN FROM MSG WRITER SWITCH IN COMM REGION
F3
CANCEL FETCH $$BCNCL SVC 6
F4
SVC 2 FETCH CALLING PHASE

Chart FL. $$BMSGWR: Tape Open/Close Message Writer (Part 1 of 2)

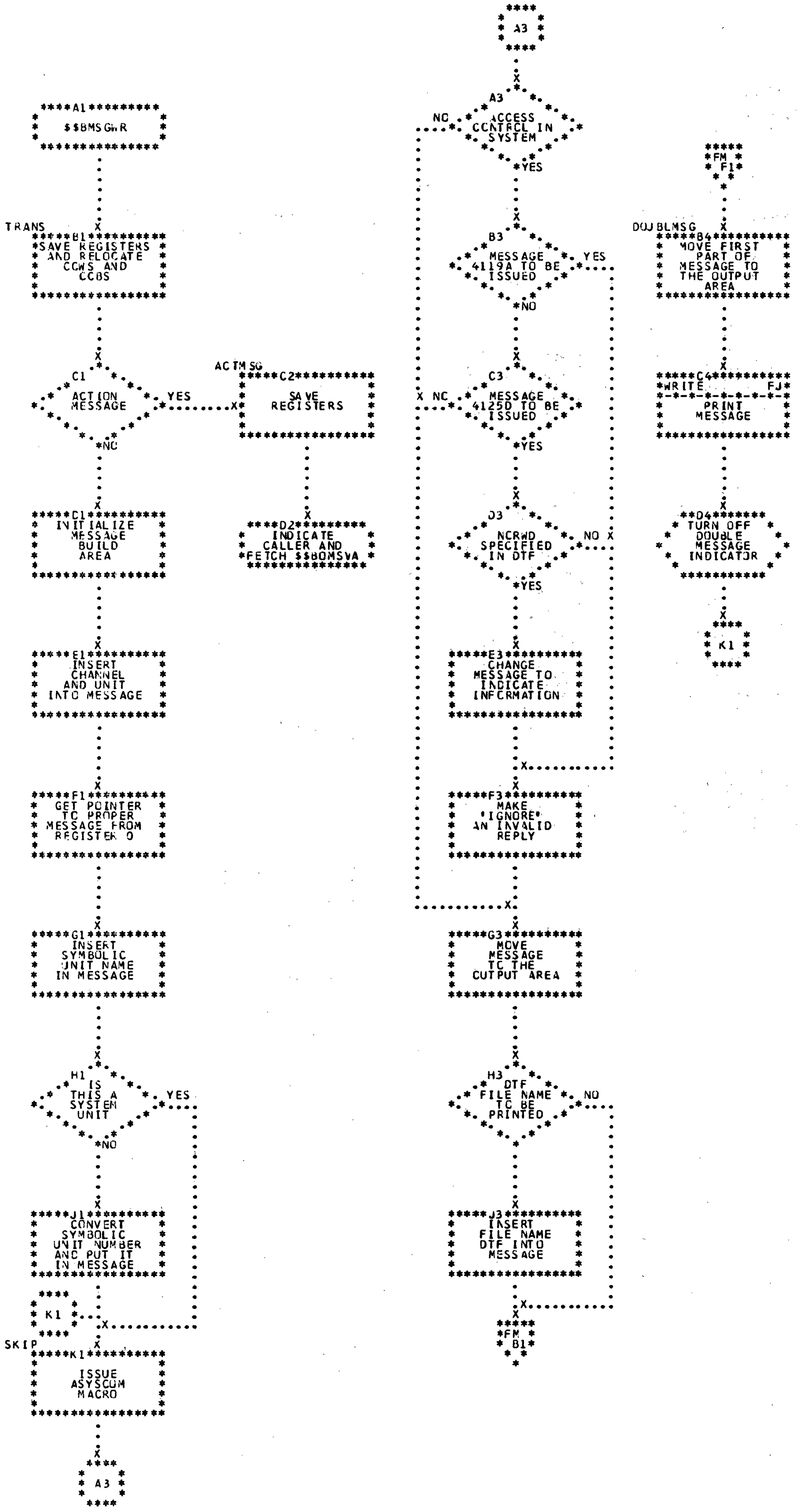

A1
$$BMSGWR
TRANS
B1
SAVE REGISTERS AND RELOCATE CCWS AND CCBS
C1
ACTION MESSAGE
YES
NO
ACTMSG
C2
SAVE REGISTERS
D2
INDICATE CALLER AND FETCH $$BOMSVA
D1
INITIALIZE MESSAGE BUILD AREA
E1
INSERT CHANNEL AND UNIT INTO MESSAGE
F1
GET POINTER TO PROPER MESSAGE FROM REGISTER 0
G1
INSERT SYMBOLIC UNIT NAME IN MESSAGE
H1
IS THIS A SYSTEM UNIT
YES
NO
J1
CONVERT SYMBOLIC UNIT NUMBER AND PUT IT IN MESSAGE
K1
SKIP
K1
ISSUE ASYSCOM MACRO
A3
A3
A3
ACCESS CONTROL IN SYSTEM
NO
YES
B3
MESSAGE 4119A TO BE ISSUED
YES
NO
C3
MESSAGE 4125D TO BE ISSUED
NO
YES
D3
NORWD SPECIFIED IN DTF
NO
YES
E3
CHANGE MESSAGE TO INDICATE INFORMATION
F3
MAKE 'IGNORE' AN INVALID REPLY
G3
MOVE MESSAGE TO THE OUTPUT AREA
H3
DTF FILE NAME TO BE PRINTED
NO
J3
INSERT FILE NAME DTF INTO MESSAGE
FM
B1
FM
F1
DOUBLMSG
B4
MOVE FIRST PART OF MESSAGE TO THE OUTPUT AREA
C4
WRITE FJ
PRINT MESSAGE
D4
TURN OFF DOUBLE MESSAGE INDICATOR
K1

Chart FM. $$BMSGWR: Tape Open/Close Message Writer (Part 2 of 2)

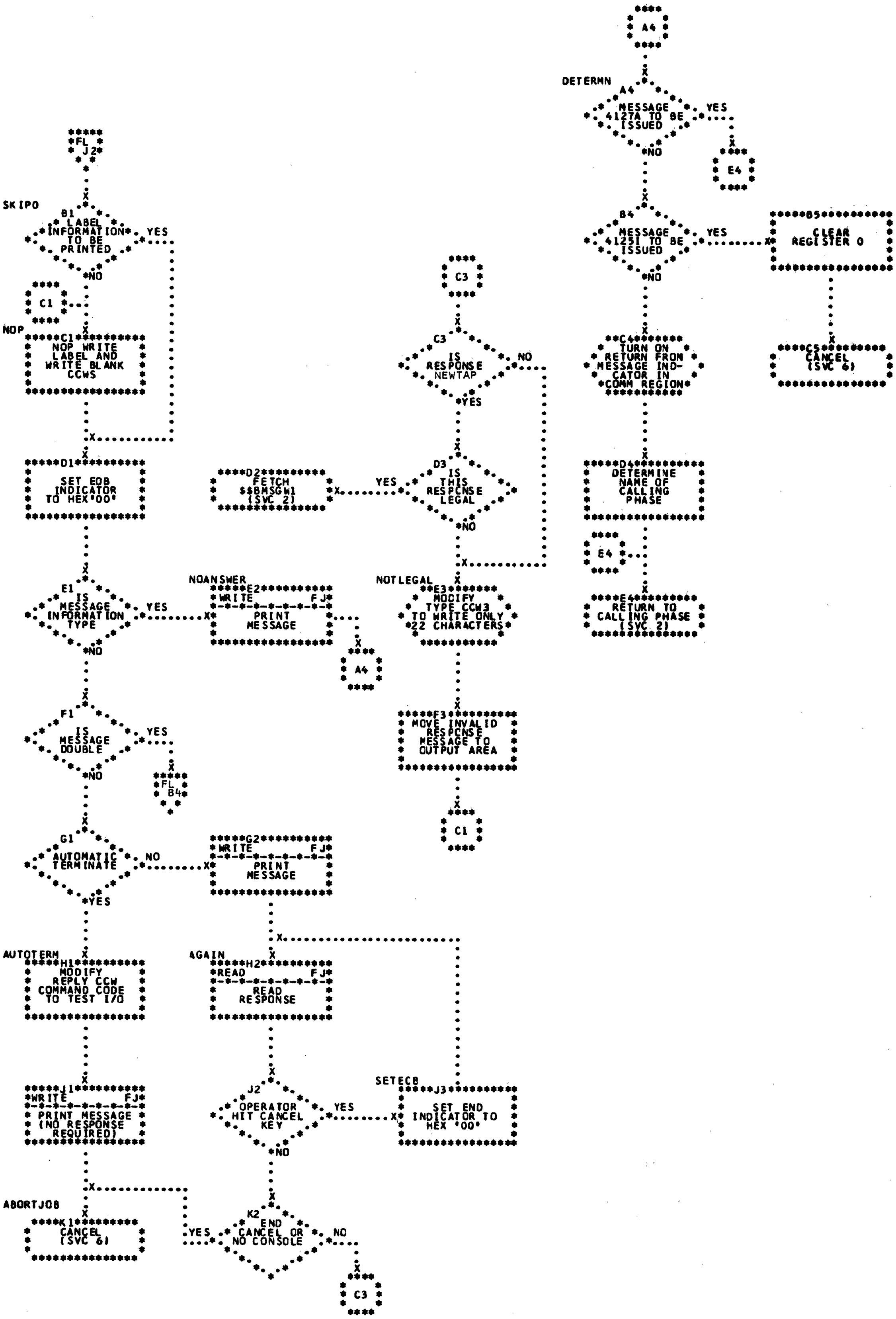

FL J2
SKIPO
B1 LABEL INFORMATION TO BE PRINTED
YES
NO
C1
NOP
C1 NOP WRITE LABEL AND WRITE BLANK CCWS
D1 SET EOB INDICATOR TO HEX'00'
E1 IS MESSAGE INFORMATION TYPE
YES
NO
NOANSWER
E2 WRITE FJ PRINT MESSAGE
A4
F1 IS MESSAGE DOUBLE
YES
NO
FL B4
G1 AUTOMATIC TERMINATE
NO
YES
G2 WRITE FJ PRINT MESSAGE
AUTOTERM
H1 MODIFY REPLY CCW COMMAND CODE TO TEST I/O
AGAIN
H2 READ FJ READ RESPONSE
J1 WRITE FJ PRINT MESSAGE (NO RESPONSE REQUIRED)
J2 OPERATOR HIT CANCEL KEY
YES
NO
SETECB
J3 SET END INDICATOR TO HEX '00'
ABORTJOB
K1 CANCEL (SVC 6)
K2 END CANCEL OR NO CONSOLE
YES
NO
C3
C3
C3 IS RESPONSE NEWTAP
NO
YES
D2 FETCH $$BMSGW1 (SVC 2)
D3 IS THIS RESPONSE LEGAL
YES
NO
NOTLEGAL
E3 MODIFY TYPE CCW3 TO WRITE ONLY 22 CHARACTERS
F3 MOVE INVALID RESPONSE MESSAGE TO OUTPUT AREA
C1
A4
DETERMN
A4 MESSAGE 4127A TO BE ISSUED
YES
NO
E4
B4 MESSAGE 4125I TO BE ISSUED
YES
NO
B5 CLEAR REGISTER 0
C4 TURN ON RETURN FROM MESSAGE INDICATOR IN COMM REGION
C5 CANCEL (SVC 6)
D4 DETERMINE NAME OF CALLING PHASE
E4
E4 RETURN TO CALLING PHASE (SVC 2)

Chart FN. SAVEREW and SAVEREWUN: SVA Rewind and Rewind/Unload

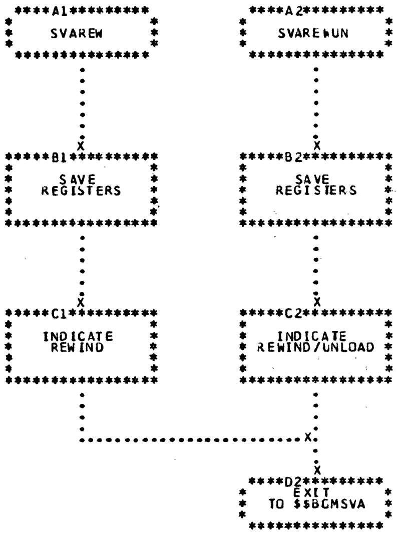


Chart FP. $$BCNVOL: Standard Volume Label Rewriter

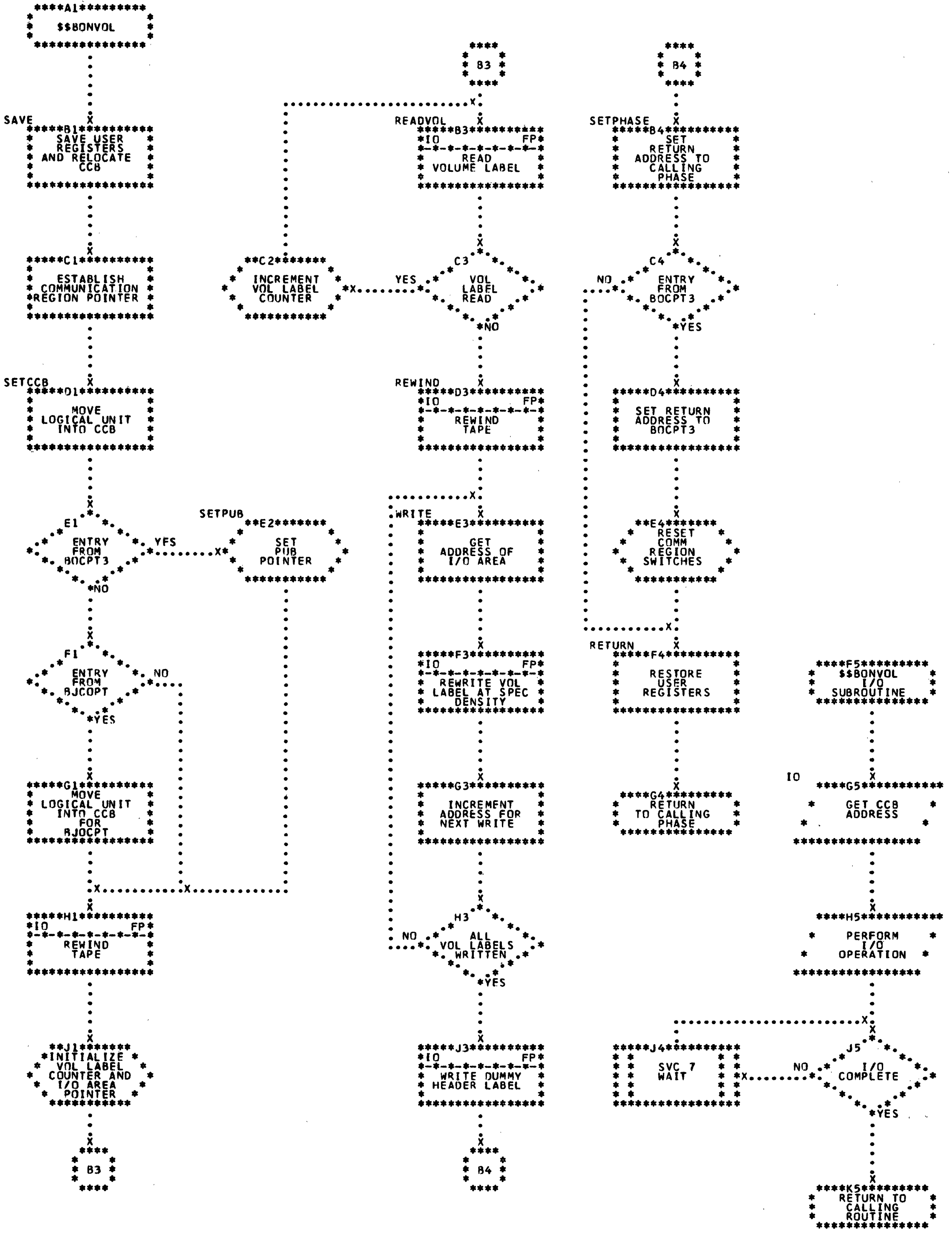

A1 $$BONVOL
SAVE
B1 SAVE USER REGISTERS AND RELOCATE CCB
C1 ESTABLISH COMMUNICATION REGION POINTER
SETCCB
D1 MOVE LOGICAL UNIT INTO CCB
E1 ENTRY FROM BOCPT3
YES
NO
SETPUB
E2 SET PUB POINTER
F1 ENTRY FROM BJCOPT
NO
YES
G1 MOVE LOGICAL UNIT INTO CCB FOR BJOCPT
H1 IO FP REWIND TAPE
J1 INITIALIZE VOL LABEL COUNTER AND I/O AREA POINTER
B3
B3
C2 INCREMENT VOL LABEL COUNTER
READVOL
B3 IO FP READ VOLUME LABEL
C3 VOL LABEL READ
YES
NO
REWIND
D3 IO FP REWIND TAPE
WRITE
E3 GET ADDRESS OF I/O AREA
F3 IO FP REWRITE VOL LABEL AT SPEC DENSITY
G3 INCREMENT ADDRESS FOR NEXT WRITE
H3 ALL VOL LABELS WRITTEN
NO
YES
J3 IO FP WRITE DUMMY HEADER LABEL
B4
B4
SETPHASE
B4 SET RETURN ADDRESS TO CALLING PHASE
C4 ENTRY FROM BOCPT3
NO
YES
D4 SET RETURN ADDRESS TO BOCPT3
E4 RESET COMM REGION SWITCHES
RETURN
F4 RESTORE USER REGISTERS
G4 RETURN TO CALLING PHASE
F5 $$BONVOL I/O SUBROUTINE
IO
G5 GET CCB ADDRESS
H5 PERFORM I/O OPERATION
J4 SVC 7 WAIT
J5 I/O COMPLETE
NO
YES
K5 RETURN TO CALLING ROUTINE

Chart FQ. $$BOMSVA and $$BOMSV2: Transfer Control to SVA Message Writer

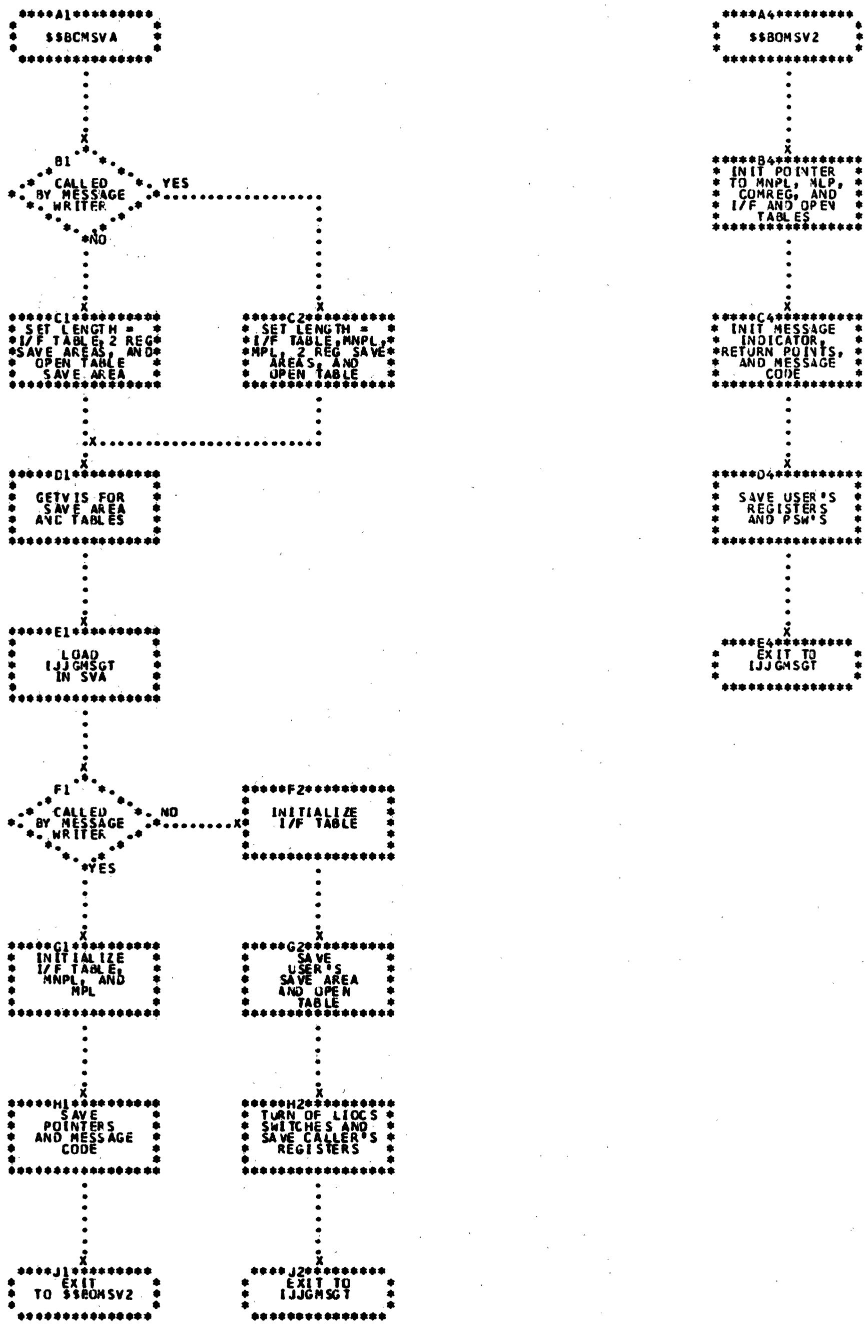

A1
$$BCMSVA
B1
CALLED BY MESSAGE WRITER
YES
NO
C1
SET LENGTH = I/F TABLE, 2 REG SAVE AREAS, AND OPEN TABLE SAVE AREA
C2
SET LENGTH = I/F TABLE, MNPL, MPL, 2 REG SAVE AREAS, AND OPEN TABLE
D1
GETVIS FOR SAVE AREA AND TABLES
E1
LOAD IJJGMSGT IN SVA
F1
CALLED BY MESSAGE WRITER
NO
YES
F2
INITIALIZE I/F TABLE
G1
INITIALIZE I/F TABLE, MNPL, AND MPL
G2
SAVE USER'S SAVE AREA AND OPEN TABLE
H1
SAVE POINTERS AND MESSAGE CODE
H2
TURN OF LIOCS SWITCHES AND SAVE CALLER'S REGISTERS
J1
EXIT TO $$BOMSV2
J2
EXIT TO IJJGMSGT
A4
$$BOMSV2
B4
INIT POINTER TO MNPL, MLP, COMREG, AND I/F AND OPEN TABLES
C4
INIT MESSAGE INDICATOR, RETURN POINTS, AND MESSAGE CODE
D4
SAVE USER'S REGISTERS AND PSW'S
E4
EXIT TO IJJGMSGT

Chart FR. IJJGMSGT: Issue Error Message

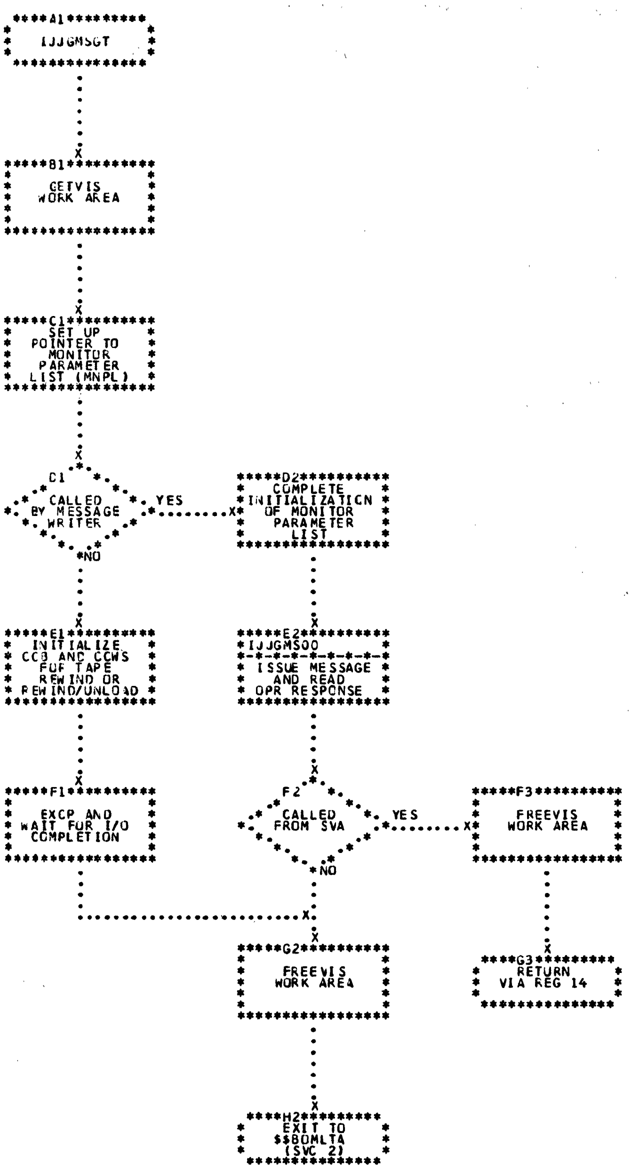

A1
IJJGMSGT
B1
GETVIS WORK AREA
C1
SET UP POINTER TO MONITOR PARAMETER LIST (MNPL)
D1
CALLED BY MESSAGE WRITER
YES
NO
D2
COMPLETE INITIALIZATION OF MONITOR PARAMETER LIST
E1
INITIALIZE CCB AND CCWS FOR TAPE REWIND OR REWIND/UNLOAD
E2
IJJGMS00
ISSUE MESSAGE AND READ OPR RESPONSE
F1
EXCP AND WAIT FOR I/O COMPLETION
F2
CALLED FROM SVA
YES
NO
F3
FREEVIS WORK AREA
G2
FREEVIS WORK AREA
G3
RETURN VIA REG 14
H2
EXIT TO $$BOMLTA (SVC 2)

Chart FS. $$BCMLTA: Return to Logical Transient Area

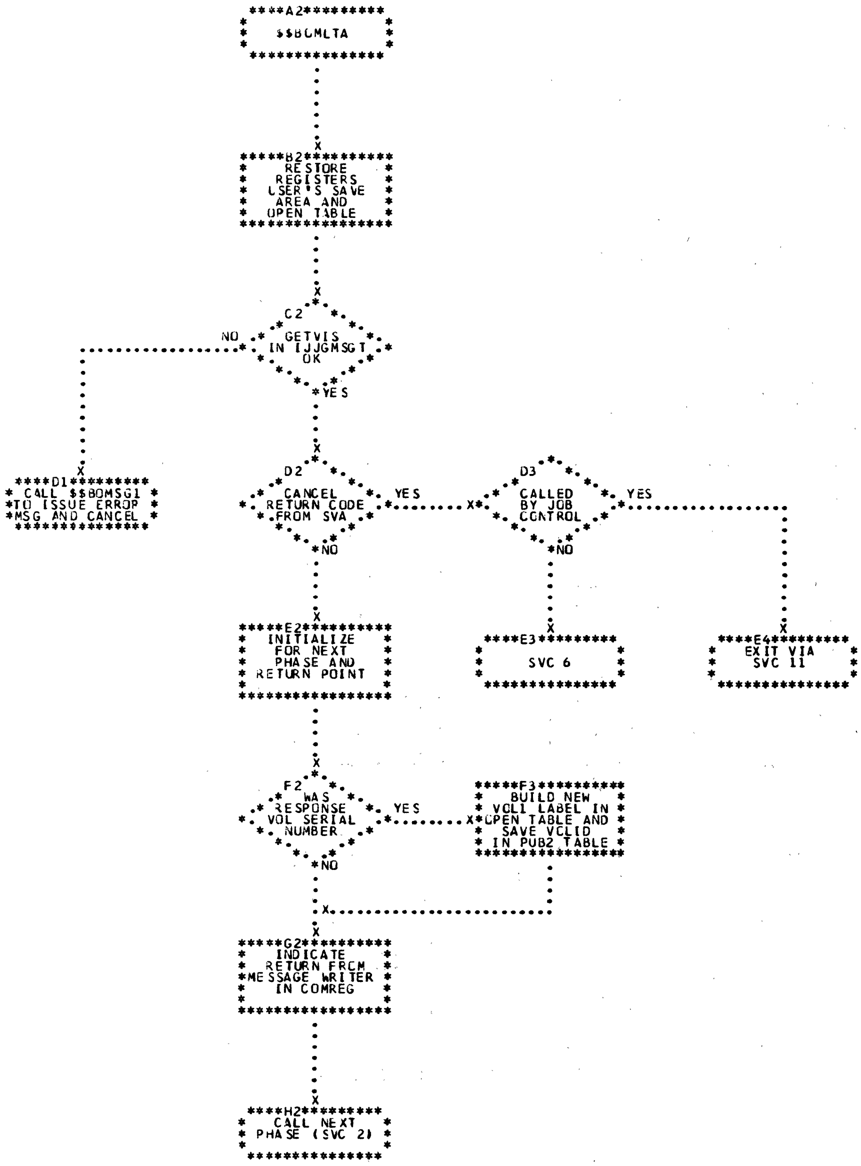

A2
$$BCMLTA
B2
RESTORE REGISTERS LSER'S SAVE AREA AND OPEN TABLE
C2
GETVIS IN IJJGMSGT OK
NO
YES
D1
CALL $$BOMSG1 TO ISSUE ERROP MSG AND CANCEL
D2
CANCEL RETURN CODE FROM SVA
YES
NO
D3
CALLED BY JOB CONTROL
YES
NO
E2
INITIALIZE FOR NEXT PHASE AND RETURN POINT
E3
SVC 6
E4
EXIT VIA SVC 11
F2
WAS RESPONSE VOL SERIAL NUMBER
YES
NO
F3
BUILD NEW VCL1 LABEL IN CPEN TABLE AND SAVE VCLID IN PUB2 TABLE
G2
INDICATE RETURN FRCM MESSAGE WRITER IN COMREG
H2
CALL NEXT PHASE (SVC 2)

Chart NA. CPMOD: GET Macro, Two I/O Areas

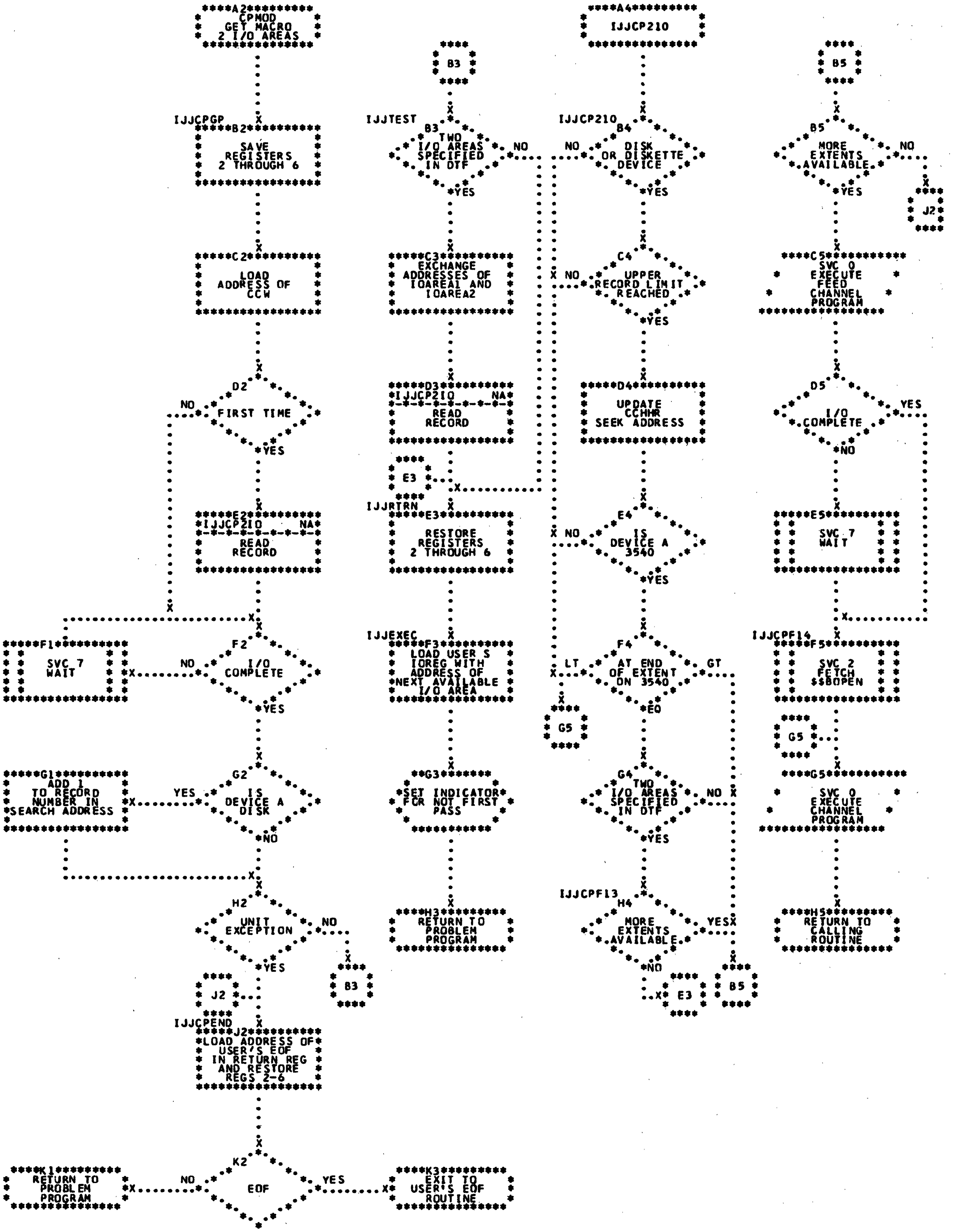

A2
CPMOD GET MACRO 2 I/O AREAS
IJJCPGP
B2
SAVE REGISTERS 2 THROUGH 6
C2
LOAD ADDRESS OF CCW
D2
FIRST TIME
NO
YES
E2
IJJCP210 NA
READ RECORD
F1
SVC 7 WAIT
F2
I/O COMPLETE
NO
YES
G1
ADD 1 TO RECORD NUMBER IN SEARCH ADDRESS
G2
IS DEVICE A DISK
YES
NO
H2
UNIT EXCEPTION
NO
YES
J2
B3
IJJCPEND
J2
LOAD ADDRESS OF USER'S EOF IN RETURN REG AND RESTORE REGS 2-6
K1
RETURN TO PROBLEM PROGRAM
K2
EOF
NO
YES
K3
EXIT TO USER'S EOF ROUTINE
B3
IJJTEST
B3
TWO I/O AREAS SPECIFIED IN DTF
NO
YES
C3
EXCHANGE ADDRESSES OF IOAREA1 AND IOAREA2
D3
IJJCP210 NA
READ RECORD
E3
IJJRTRN
E3
RESTORE REGISTERS 2 THROUGH 6
IJJEXEC
F3
LOAD USER S IOREG WITH ADDRESS OF NEXT AVAILABLE I/O AREA
G3
SET INDICATOR FOR NOT FIRST PASS
H3
RETURN TO PROBLEM PROGRAM
A4
IJJCP210
IJJCP210
B4
DISK OR DISKETTE DEVICE
NO
YES
C4
UPPER RECORD LIMIT REACHED
NO
YES
D4
UPDATE CCHHR SEEK ADDRESS
E4
IS DEVICE A 3540
NO
YES
F4
AT END OF EXTENT ON 3540
LT
GT
EQ
G5
G4
TWO I/O AREAS SPECIFIED IN DTF
NO
YES
IJJCPF13
H4
MORE EXTENTS AVAILABLE
YES
NO
E3
B5
B5
B5
MORE EXTENTS AVAILABLE
NO
YES
J2
C5
SVC 0 EXECUTE FEED CHANNEL PROGRAM
D5
I/O COMPLETE
YES
NO
E5
SVC 7 WAIT
IJJCPF14
F5
SVC 2 FETCH $$BOPEN
G5
G5
SVC 0 EXECUTE CHANNEL PROGRAM
H5
RETURN TO CALLING ROUTINE

Chart NB. CPMOD: GET Macro, One I/O Area or IOPTR=YES

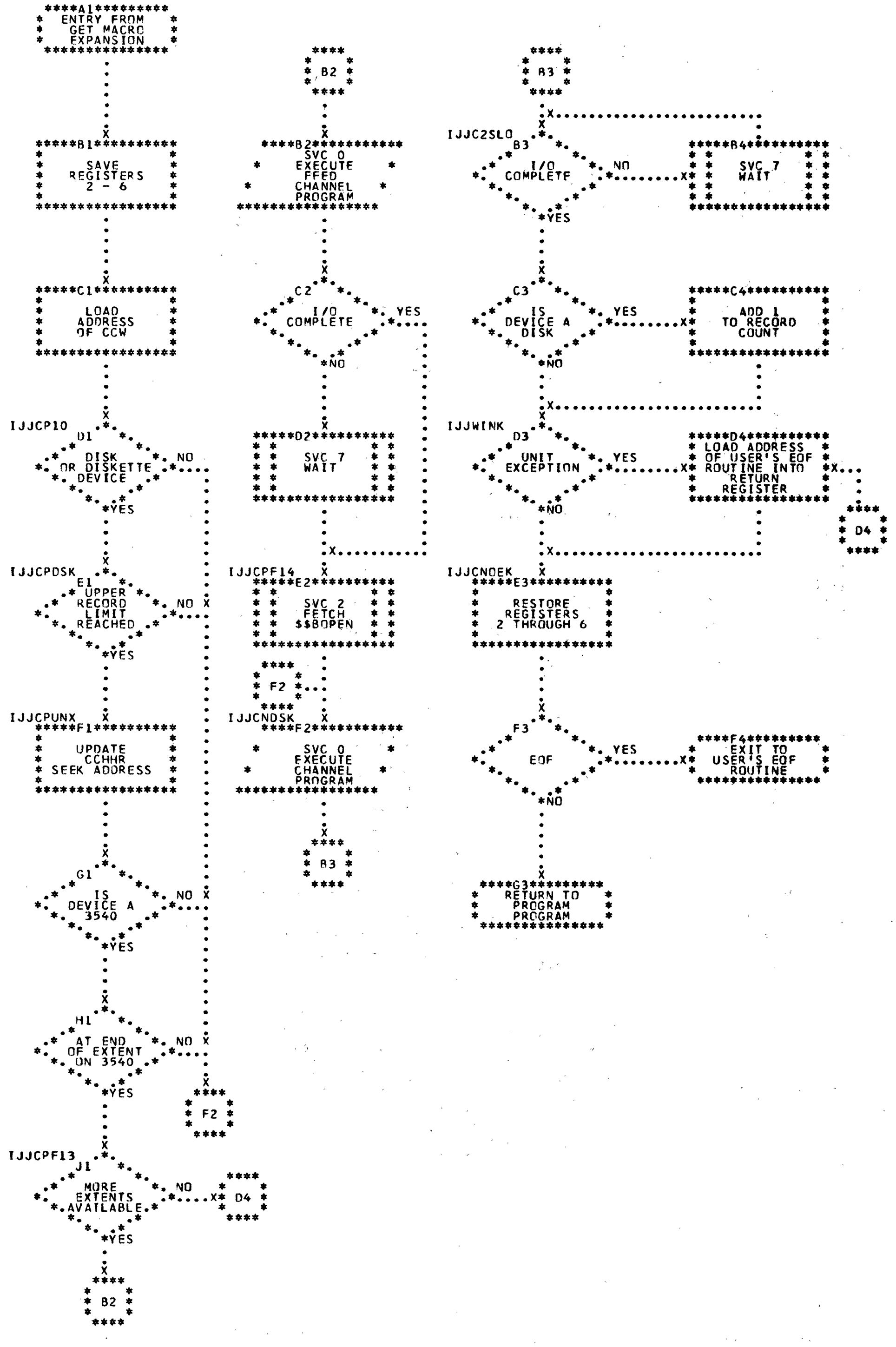


Chart NC. CPMOD: PUT Macro, Two I/O Areas (Part 1 of 2)

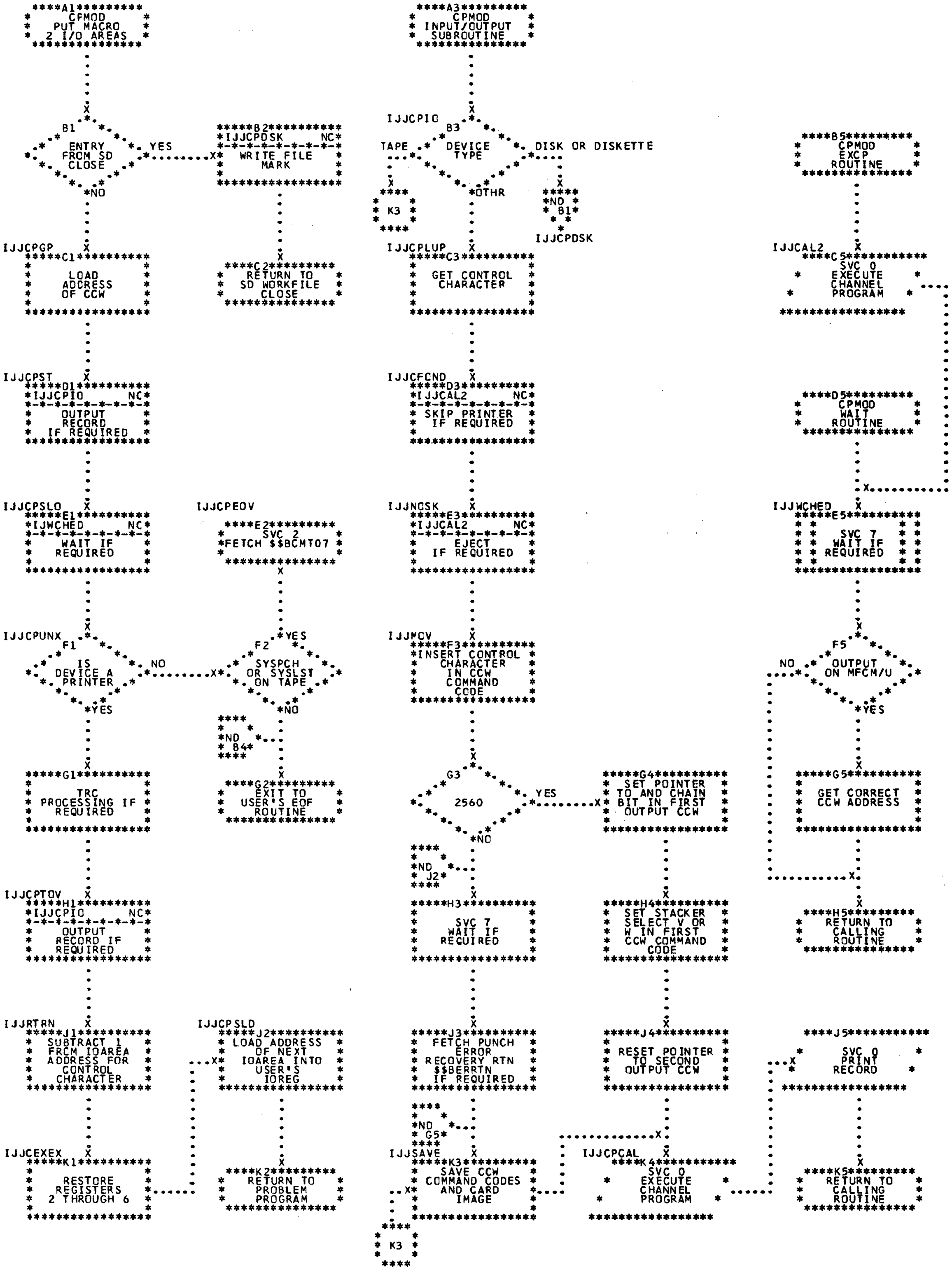


Chart ND. CPMOD: PUT Macro, Two I/O Areas (Part 2 of 2)

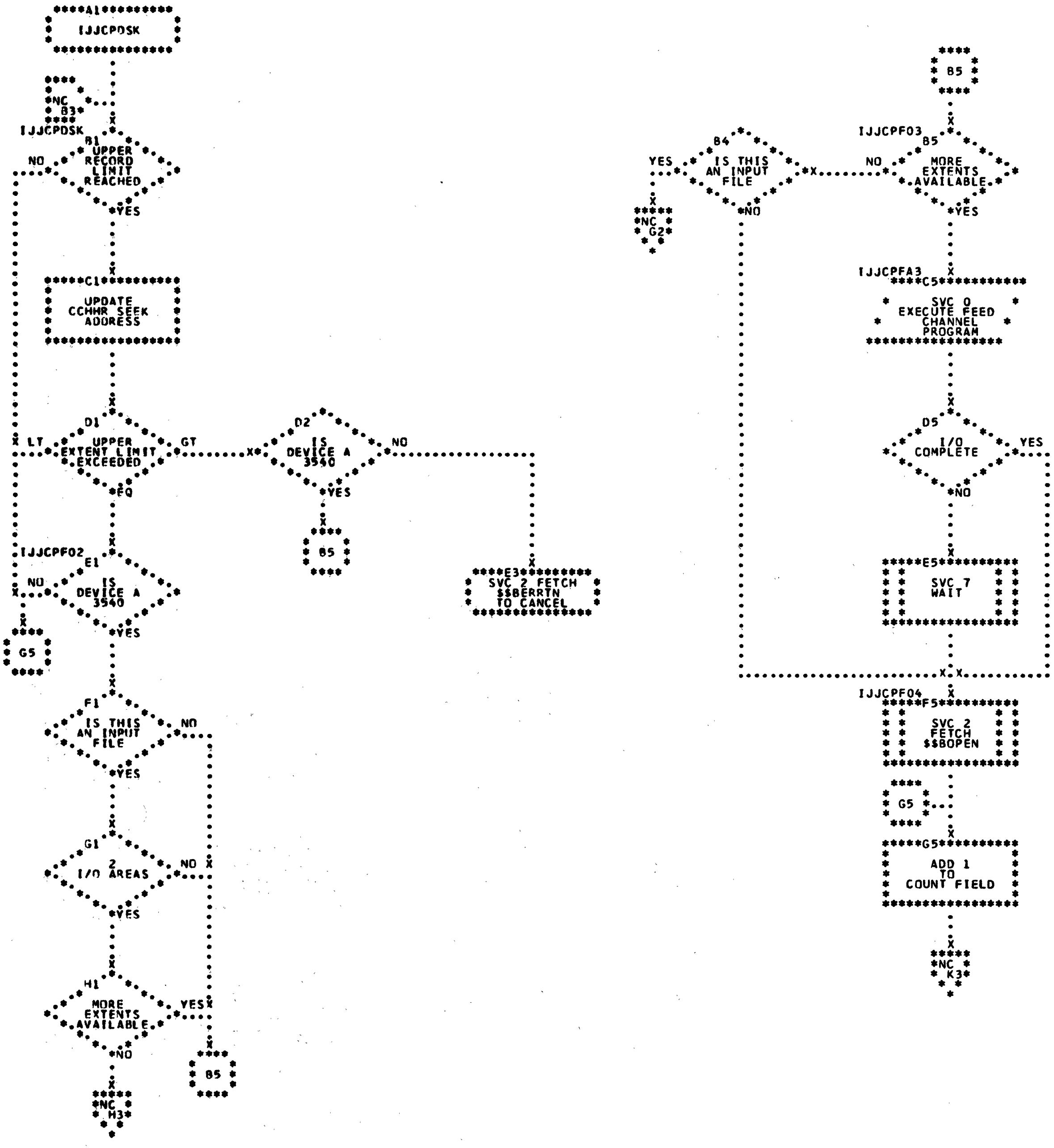


Chart NE. CPMOD: PUT Macro, One I/O Area (Part 1 of 2)

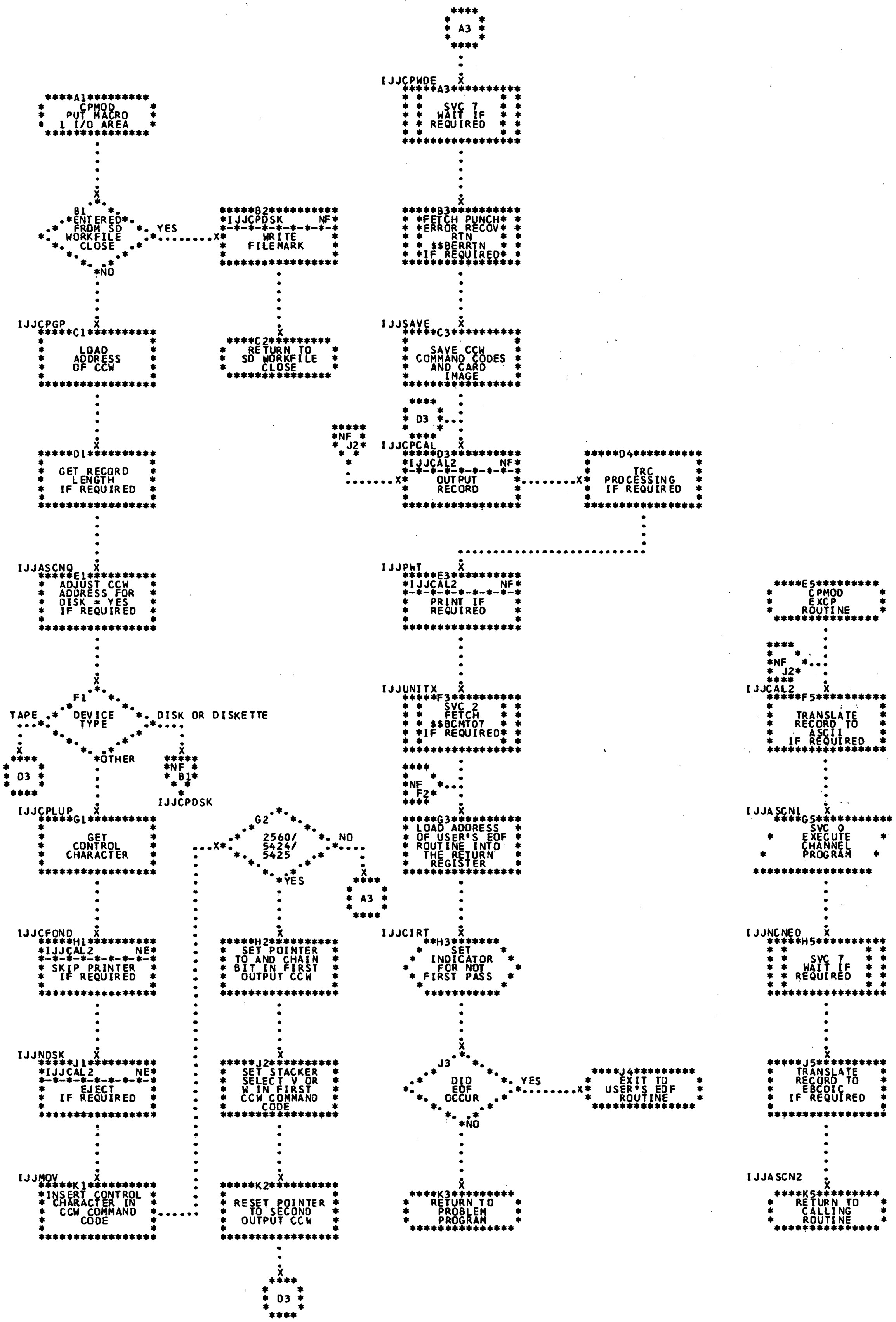

A3
A1 CPMOD PUT MACRO 1 I/O AREA
B1 ENTERED FROM SD WORKFILE CLOSE
YES
NO
B2 IJJCPDSK NF WRITE FILEMARK
C2 RETURN TO SD WORKFILE CLOSE
IJJCPGP C1 LOAD ADDRESS OF CCW
D1 GET RECORD LENGTH IF REQUIRED
IJJASCNQ E1 ADJUST CCW ADDRESS FOR DISK = YES IF REQUIRED
F1 DEVICE TYPE
TAPE
DISK OR DISKETTE
OTHER
D3
NF B1
IJJCPDSK
IJJCPLUP G1 GET CONTROL CHARACTER
IJJCFOND H1 IJJCAL2 NE SKIP PRINTER IF REQUIRED
IJJNDSK J1 IJJCAL2 NE EJECT IF REQUIRED
IJJMOV K1 INSERT CONTROL CHARACTER IN CCW COMMAND CODE
G2 2560/ 5424/ 5425
NO
YES
A3
H2 SET POINTER TO AND CHAIN BIT IN FIRST OUTPUT CCW
J2 SET STACKER SELECT V OR W IN FIRST CCW COMMAND CODE
K2 RESET POINTER TO SECOND OUTPUT CCW
D3
IJJCPWDE A3 SVC 7 WAIT IF REQUIRED
B3 FETCH PUNCH ERROR RECOV RTN $$BERRTN IF REQUIRED
IJJSAVE C3 SAVE CCW COMMAND CODES AND CARD IMAGE
D3
NF J2
IJJCPCAL D3 IJJCAL2 NF OUTPUT RECORD
D4 TRC PROCESSING IF REQUIRED
IJJPWT E3 IJJCAL2 NF PRINT IF REQUIRED
IJJUNITX F3 SVC 2 FETCH $$BCMT07 IF REQUIRED
NF F2
G3 LOAD ADDRESS OF USER'S EOF ROUTINE INTO THE RETURN REGISTER
IJJCIRT H3 SET INDICATOR FOR NOT FIRST PASS
J3 DID EOF OCCUR
YES
NO
J4 EXIT TO USER'S EOF ROUTINE
K3 RETURN TO PROBLEM PROGRAM
E5 CPMOD EXCP ROUTINE
NF J2
IJJCAL2 F5 TRANSLATE RECORD TO ASCII IF REQUIRED
IJJASCN1 G5 SVC 0 EXECUTE CHANNEL PROGRAM
IJJNCNED H5 SVC 7 WAIT IF REQUIRED
J5 TRANSLATE RECORD TO EBCDIC IF REQUIRED
IJJASCN2 K5 RETURN TO CALLING ROUTINE

Chart NF. CFMCI: PUT Macro, One I/O Area (Part 2 of 2)

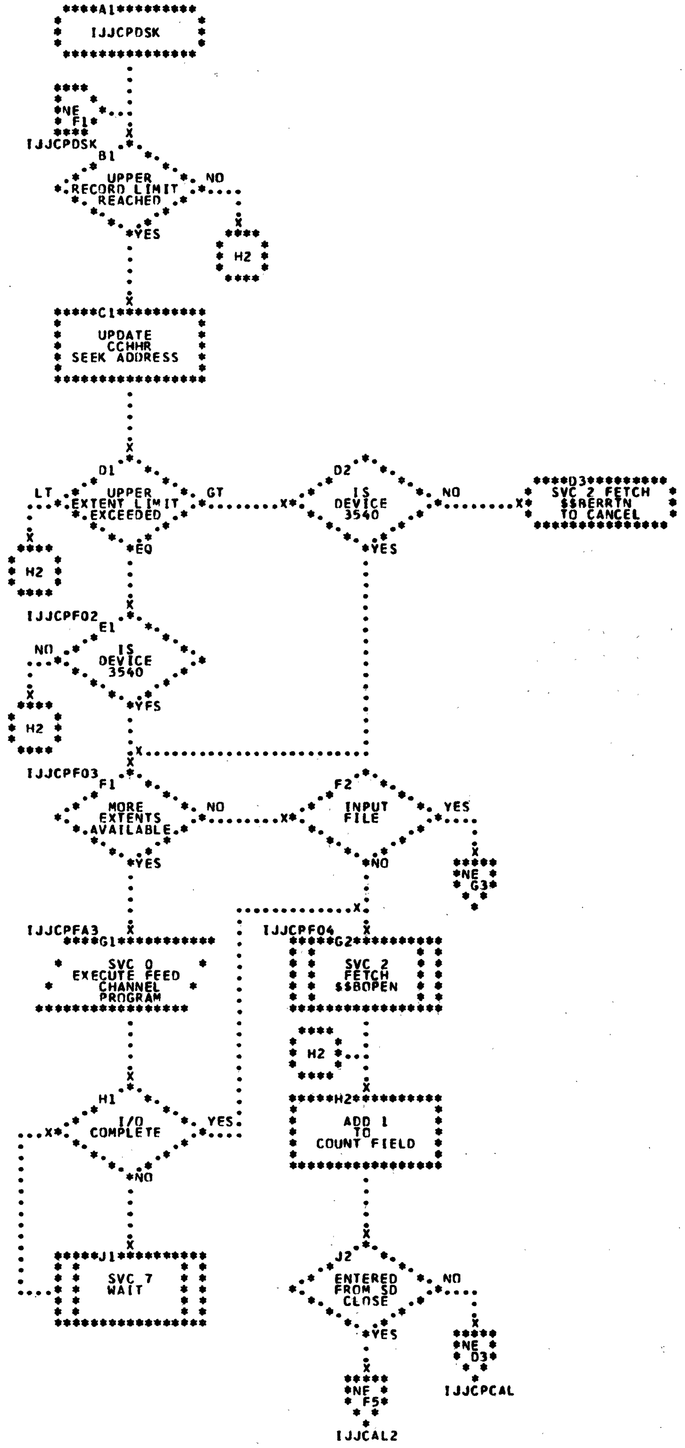


Chart NG. CPMOD: PUT Macro, IOPTR=YES (Part 1 of 2)

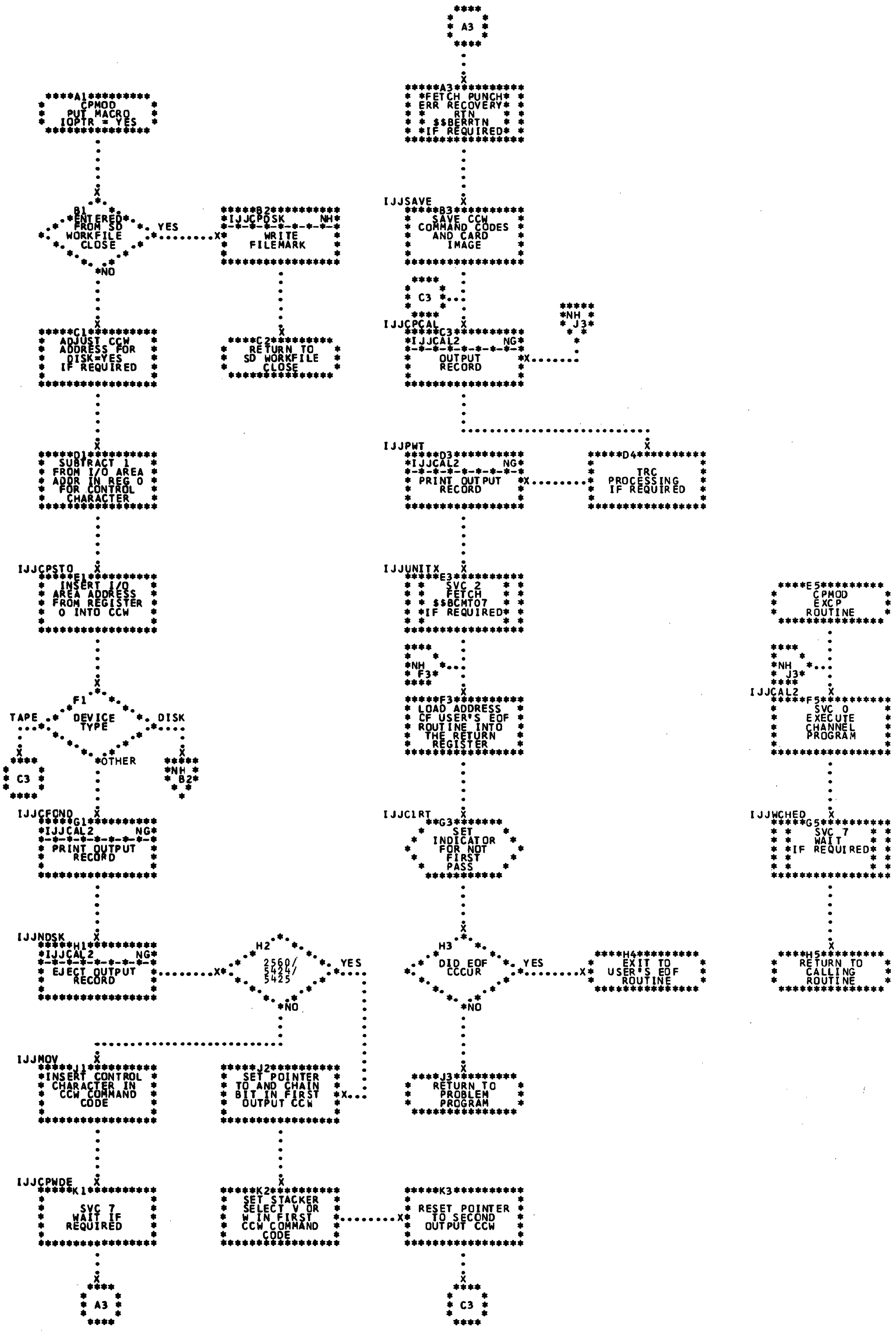

A1
CPMOD PUT MACRO IOPTR = YES
B1
ENTERED FROM SD WORKFILE CLOSE
YES
NO
B2
IJJCPDSK NH
WRITE FILEMARK
C1
ADJUST CCW ADDRESS FOR DISK=YES IF REQUIRED
C2
RETURN TO SD WORKFILE CLOSE
D1
SUBTRACT 1 FROM I/O AREA ADDR IN REG 0 FOR CONTROL CHARACTER
IJJCPSTO
E1
INSERT I/O AREA ADDRESS FROM REGISTER 0 INTO CCW
F1
DEVICE TYPE
TAPE
DISK
OTHER
C3
NH B2
IJJCFOND
G1
IJJCAL2 NG
PRINT OUTPUT RECORD
IJJNDSK
H1
IJJCAL2 NG
EJECT OUTPUT RECORD
H2
2560/ 5424/ 5425
YES
NO
IJJMOV
J1
INSERT CONTROL CHARACTER IN CCW COMMAND CODE
J2
SET POINTER TO AND CHAIN BIT IN FIRST OUTPUT CCW
IJJCPWDE
K1
SVC 7 WAIT IF REQUIRED
K2
SET STACKER SELECT V OR W IN FIRST CCW COMMAND CODE
A3
A3
FETCH PUNCH ERR RECOVERY RTN $$BERRTN IF REQUIRED
IJJSAVE
B3
SAVE CCW COMMAND CODES AND CARD IMAGE
C3
IJJCPCAL
C3
IJJCAL2 NG
OUTPUT RECORD
NH J3
IJJPWT
D3
IJJCAL2 NG
PRINT OUTPUT RECORD
D4
TRC PROCESSING IF REQUIRED
IJJUNITX
E3
SVC 2 FETCH $$BCMT07 IF REQUIRED
NH F3
F3
LOAD ADDRESS CF USER'S EOF ROUTINE INTO THE RETURN REGISTER
IJJC1RT
G3
SET INDICATOR FOR NOT FIRST PASS
H3
DID EOF CCCUR
YES
NO
H4
EXIT TO USER'S EOF ROUTINE
J3
RETURN TO PROBLEM PROGRAM
K3
RESET POINTER TO SECOND OUTPUT CCW
C3
E5
CPMOD EXCP ROUTINE
NH J3
IJJCAL2
F5
SVC 0 EXECUTE CHANNEL PROGRAM
IJJWCHED
G5
SVC 7 WAIT IF REQUIRED
H5
RETURN TO CALLING ROUTINE

Chart NH. CPMOD: PUT Macro, IOPTR=YES (Part 2 of 2)

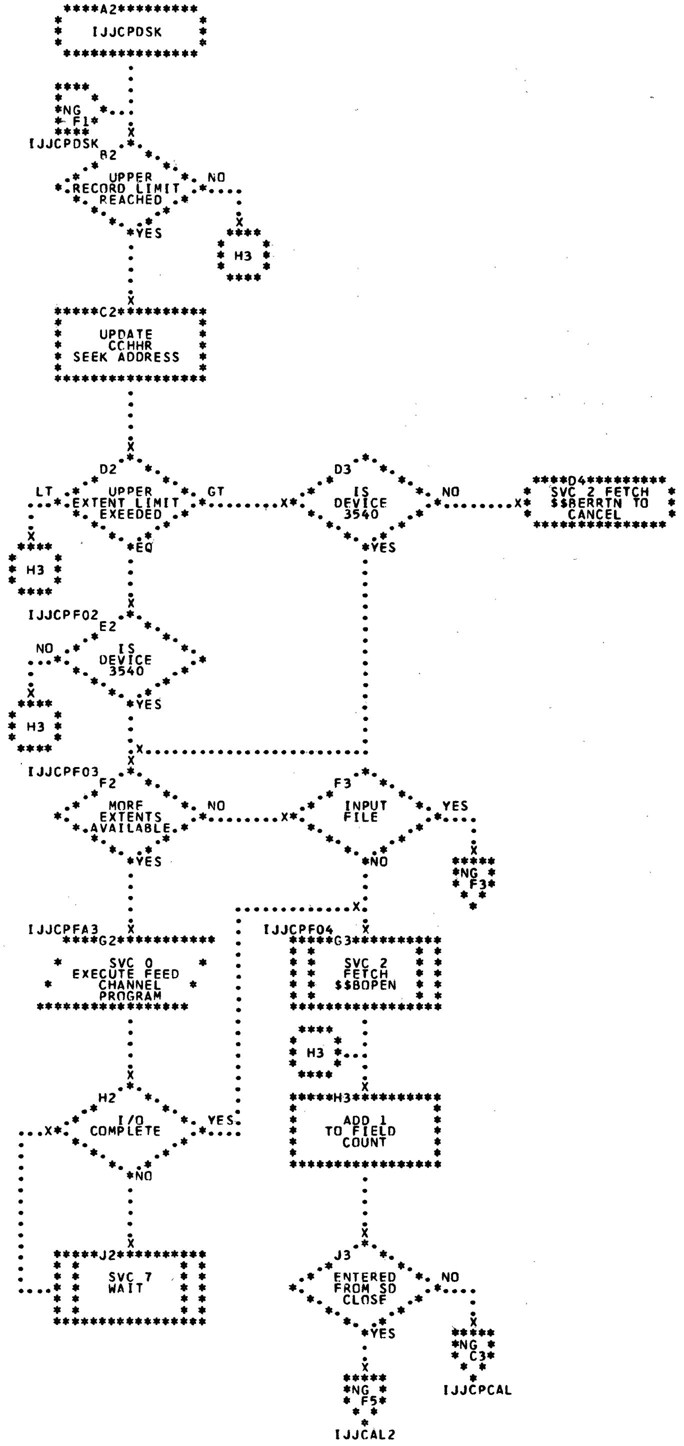
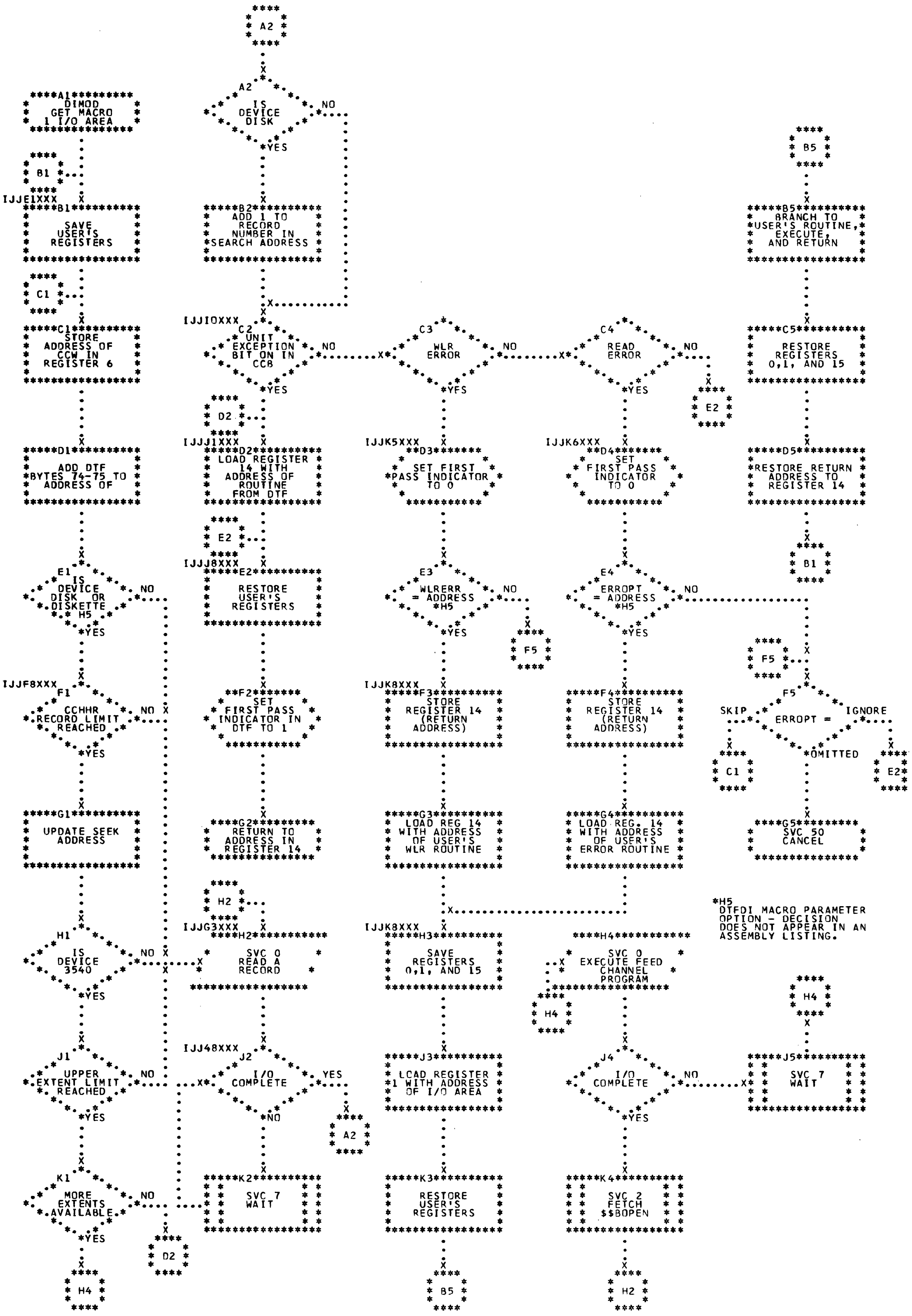


Chart PA. DIMOD: GET Macro, One I/O Area


A1 DIMOD GET MACRO 1 I/O AREA
B1
IJJE1XXX
B1 SAVE USER'S REGISTERS
C1
C1 STORE ADDRESS OF CCW IN REGISTER 6
D1 ADD DTF BYTES 74-75 TO ADDRESS OF
E1 IS DEVICE DISK OR DISKETTE H5
NO
YES
IJJF8XXX
F1 CCHHR RECORD LIMIT REACHED
NO
YES
G1 UPDATE SEEK ADDRESS
H1 IS DEVICE 3540
NO
YES
J1 UPPER EXTENT LIMIT REACHED
NO
YES
K1 MORE EXTENTS AVAILABLE
NO
YES
H4
D2
A2
A2 IS DEVICE DISK
NO
YES
B2 ADD 1 TO RECORD NUMBER IN SEARCH ADDRESS
IJJI0XXX
C2 UNIT EXCEPTION BIT ON IN CCB
NO
YES
D2
IJJJ1XXX
D2 LOAD REGISTER 14 WITH ADDRESS OF ROUTINE FROM DTF
E2
IJJJ8XXX
E2 RESTORE USER'S REGISTERS
F2 SET FIRST PASS INDICATOR IN DTF TO 1
G2 RETURN TO ADDRESS IN REGISTER 14
H2
IJJG3XXX
H2 SVC 0 READ A RECORD
IJJ48XXX
J2 I/O COMPLETE
YES
NO
A2
K2 SVC 7 WAIT
C3 WLR ERROR
NO
YES
IJJK5XXX
D3 SET FIRST PASS INDICATOR TO 0
E3 WLRERR = ADDRESS *H5
NO
YES
F5
IJJK8XXX
F3 STORE REGISTER 14 (RETURN ADDRESS)
G3 LOAD REG 14 WITH ADDRESS OF USER'S WLR ROUTINE
IJJK8XXX
H3 SAVE REGISTERS 0,1, AND 15
J3 LOAD REGISTER 1 WITH ADDRESS OF I/O AREA
K3 RESTORE USER'S REGISTERS
B5
C4 READ ERROR
NO
YES
E2
IJJK6XXX
D4 SET FIRST PASS INDICATOR TO 0
E4 ERROPT = ADDRESS *H5
NO
YES
F4 STORE REGISTER 14 (RETURN ADDRESS)
G4 LOAD REG. 14 WITH ADDRESS OF USER'S ERROR ROUTINE
H4 SVC 0 EXECUTE FEED CHANNEL PROGRAM
H4
J4 I/O COMPLETE
NO
YES
K4 SVC 2 FETCH $$BOPEN
H2
B5
B5 BRANCH TO USER'S ROUTINE, EXECUTE, AND RETURN
C5 RESTORE REGISTERS 0,1, AND 15
D5 RESTORE RETURN ADDRESS TO REGISTER 14
B1
F5
F5 SKIP ERROPT = IGNORE
OMITTED
C1
E2
G5 SVC 50 CANCEL
H4
J5 SVC 7 WAIT

*H5 DTFDI MACRO PARAMETER OPTION - DECISION DOES NOT APPEAR IN AN ASSEMBLY LISTING.

Chart PB. DIMOD: GET Macro, Two I/O Areas

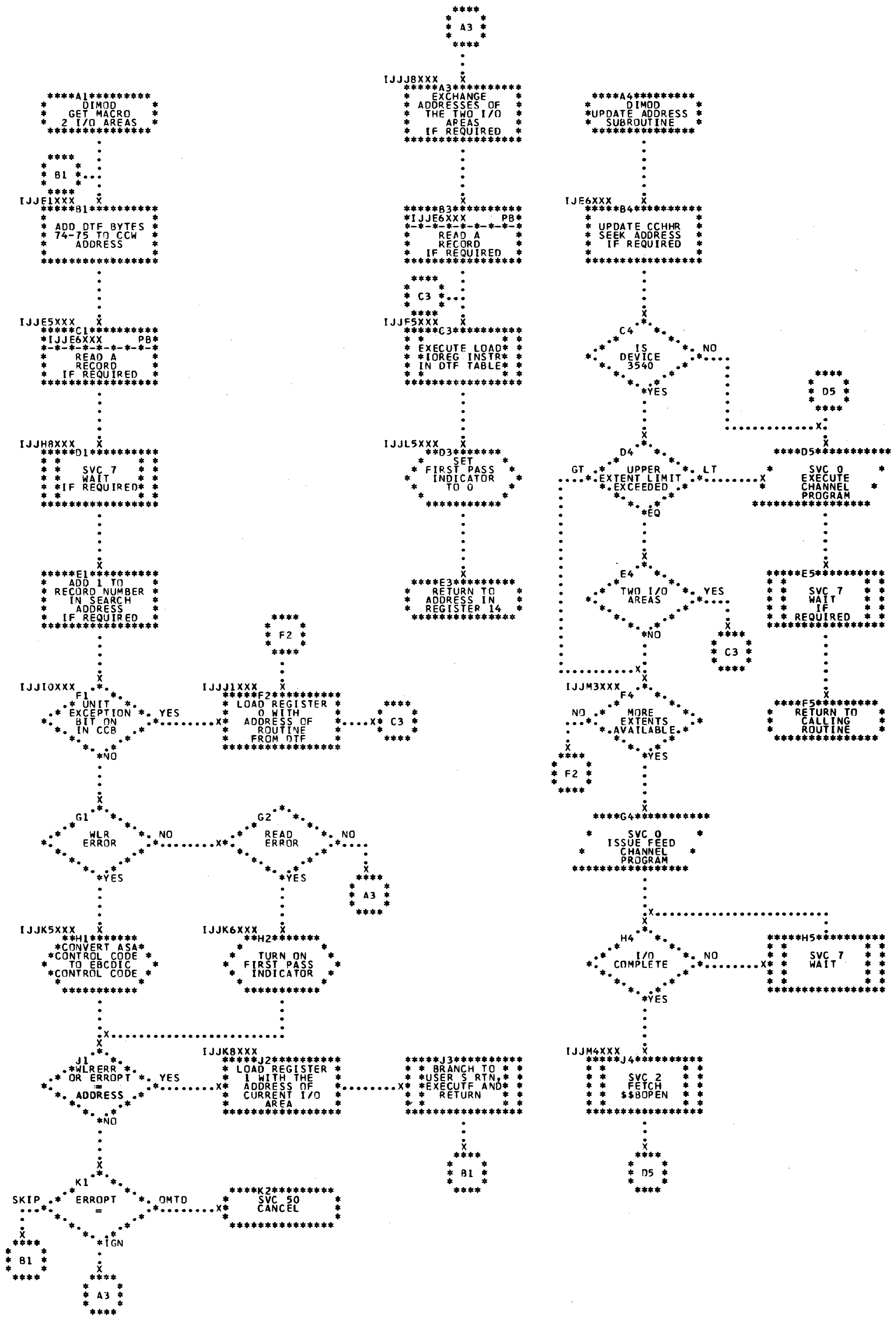

A1 DIMOD GET MACRO 2 I/O AREAS
B1
IJJE1XXX B1 ADD DTF BYTES 74-75 TO CCW ADDRESS
IJJE5XXX C1 IJJE6XXX PB READ A RECORD IF REQUIRED
IJJH8XXX D1 SVC 7 WAIT IF REQUIRED
E1 ADD 1 TO RECORD NUMBER IN SEARCH ADDRESS IF REQUIRED
IJJI0XXX F1 UNIT EXCEPTION BIT ON IN CCB
YES
NO
G1 WLR ERROR
NO
YES
IJJK5XXX H1 CONVERT ASA CONTROL CODE TO EBCDIC CONTROL CODE
J1 WLRERR OR ERROPT = ADDRESS
YES
NO
K1 ERROPT =
SKIP
OMTD
IGN
B1
A3
F2
IJJJ1XXX F2 LOAD REGISTER 0 WITH ADDRESS OF ROUTINE FROM DTF
C3
G2 READ ERROR
NO
YES
A3
IJJK6XXX H2 TURN ON FIRST PASS INDICATOR
IJJK8XXX J2 LOAD REGISTER 1 WITH THE ADDRESS OF CURRENT I/O AREA
K2 SVC 50 CANCEL
A3
IJJJ8XXX A3 EXCHANGE ADDRESSES OF THE TWO I/O AREAS IF REQUIRED
B3 IJJE6XXX PB READ A RECORD IF REQUIRED
C3
IJJF5XXX C3 EXECUTE LOAD IOREG INSTR IN DTF TABLE
IJJL5XXX D3 SET FIRST PASS INDICATOR TO 0
E3 RETURN TO ADDRESS IN REGISTER 14
J3 BRANCH TO USER S RTN, EXECUTE AND RETURN
B1
A4 DIMOD UPDATE ADDRESS SUBROUTINE
IJE6XXX B4 UPDATE CCHHR SEEK ADDRESS IF REQUIRED
C4 IS DEVICE 3540
NO
YES
D4 UPPER EXTENT LIMIT EXCEEDED
GT
LT
EQ
E4 TWO I/O AREAS
YES
NO
C3
IJJM3XXX F4 MORE EXTENTS AVAILABLE
NO
YES
F2
G4 SVC 0 ISSUE FEED CHANNEL PROGRAM
H4 I/O COMPLETE
NO
YES
IJJM4XXX J4 SVC 2 FETCH $$BOPEN
D5
D5
D5 SVC 0 EXECUTE CHANNEL PROGRAM
E5 SVC 7 WAIT IF REQUIRED
F5 RETURN TO CALLING ROUTINE
H5 SVC 7 WAIT

Chart PC. DIMOD: PUT Macro, One I/O Area (Part 1 of 2)

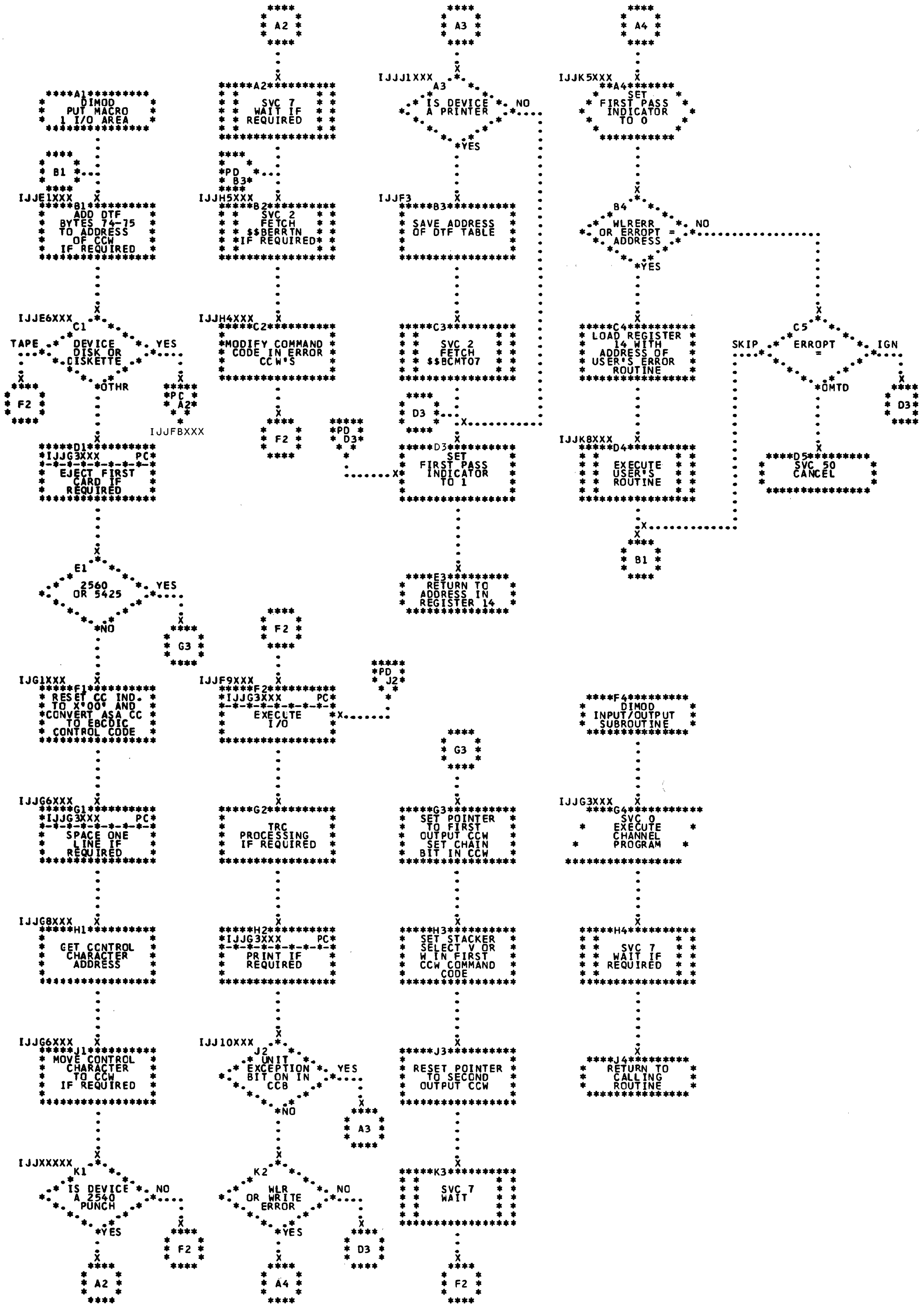

A2
A3
A4
A1
DIMOD PUT MACRO 1 I/O AREA
B1
IJJE1XXX
B1
ADD DTF BYTES 74-75 TO ADDRESS OF CCW IF REQUIRED
IJJE6XXX
C1
DEVICE DISK OR DISKETTE
TAPE
YES
OTHR
F2
PC A2
IJJFBXXX
D1
IJJG3XXX PC
EJECT FIRST CARD IF REQUIRED
E1
2560 OR 5425
YES
NO
G3
IJG1XXX
F1
RESET CC IND. TO X'00' AND CONVERT ASA CC TO EBCDIC CONTROL CODE
IJJG6XXX
G1
IJJG3XXX PC
SPACE ONE LINE IF REQUIRED
IJJG8XXX
H1
GET CONTROL CHARACTER ADDRESS
IJJG6XXX
J1
MOVE CONTROL CHARACTER TO CCW IF REQUIRED
IJJXXXXX
K1
IS DEVICE A 2540 PUNCH
NO
YES
F2
A2
A2
SVC 7 WAIT IF REQUIRED
PD B3
IJJH5XXX
B2
SVC 2 FETCH $$BERRTN IF REQUIRED
IJJH4XXX
C2
MODIFY COMMAND CODE IN ERROR CCW'S
F2
PD D3
F2
IJJF9XXX
F2
IJJG3XXX PC
EXECUTE I/O
PD J2
G2
TRC PROCESSING IF REQUIRED
H2
IJJG3XXX PC
PRINT IF REQUIRED
IJJ10XXX
J2
UNIT EXCEPTION BIT ON IN CCB
YES
NO
A3
K2
WLR OR WRITE ERROR
NO
YES
D3
A4
IJJJ1XXX
A3
IS DEVICE A PRINTER
NO
YES
IJJF3
B3
SAVE ADDRESS OF DTF TABLE
C3
SVC 2 FETCH $$BCMT07
D3
D3
SET FIRST PASS INDICATOR TO 1
E3
RETURN TO ADDRESS IN REGISTER 14
G3
G3
SET POINTER TO FIRST OUTPUT CCW SET CHAIN BIT IN CCW
H3
SET STACKER SELECT V OR W IN FIRST CCW COMMAND CODE
J3
RESET POINTER TO SECOND OUTPUT CCW
K3
SVC 7 WAIT
F2
IJJK5XXX
A4
SET FIRST PASS INDICATOR TO 0
B4
WLRERR OR ERROPT = ADDRESS
NO
YES
C4
LOAD REGISTER 14 WITH ADDRESS OF USER'S ERROR ROUTINE
IJJK8XXX
D4
EXECUTE USER'S ROUTINE
B1
C5
ERROPT =
SKIP
IGN
OMTD
D3
D5
SVC 50 CANCEL
F4
DIMOD INPUT/OUTPUT SUBROUTINE
IJJG3XXX
G4
SVC 0 EXECUTE CHANNEL PROGRAM
H4
SVC 7 WAIT IF REQUIRED
J4
RETURN TO CALLING ROUTINE

Chart PC. DIMOD: PUT Macro, One I/O Area (Part 2 of 2)

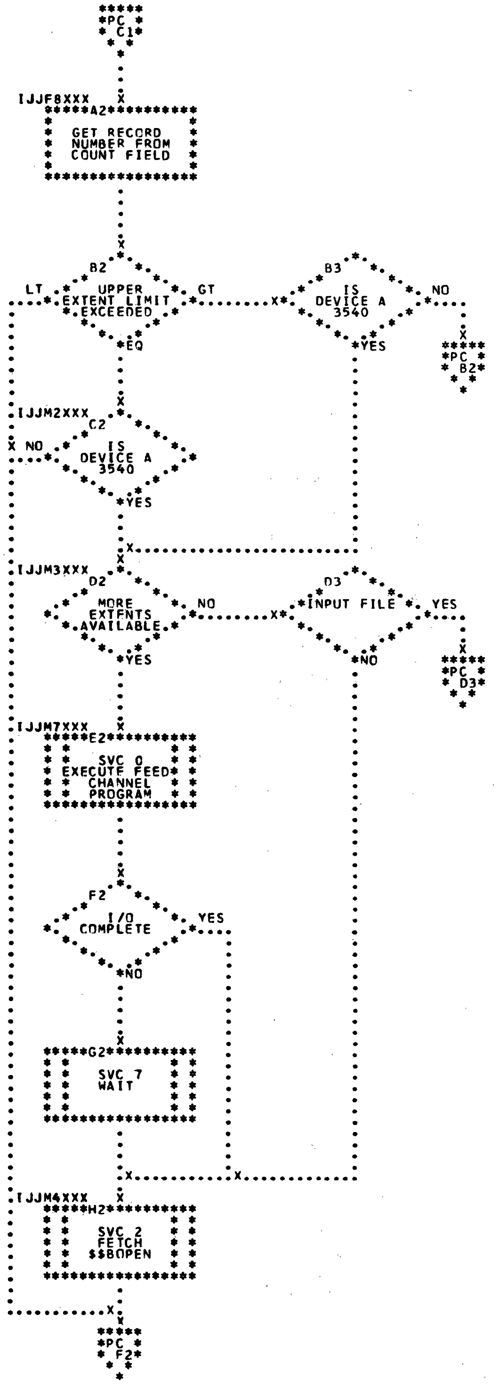


Chart PE. DIMOD: PUT Macro, Two I/O Areas (Part 1 of 2)

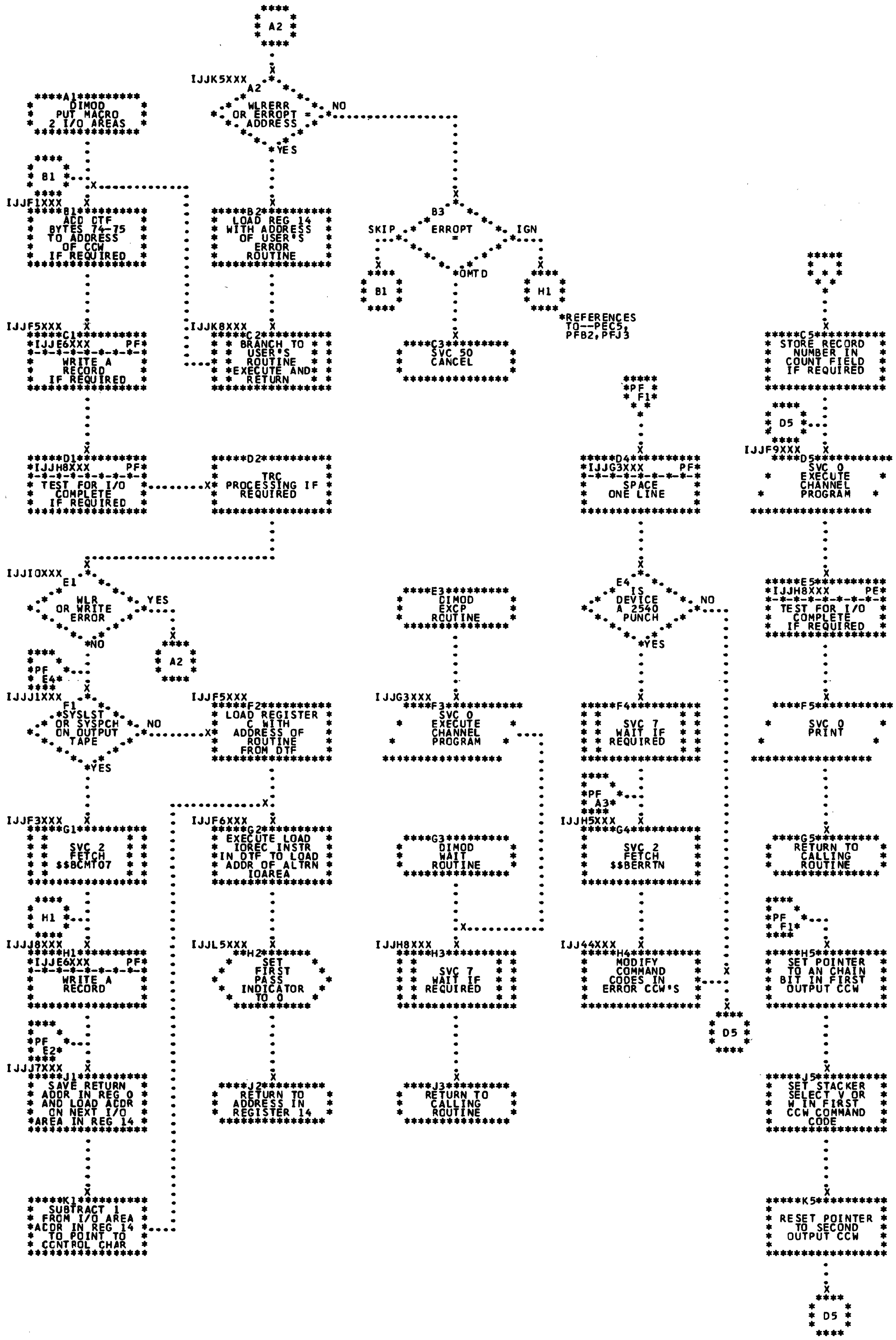

A2
IJJK5XXX
A2
WLRERR OR ERROPT = ADDRESS
NO
YES
A1
DIMOD PUT MACRO 2 I/O AREAS
B1
IJJF1XXX
B1
ADD CTF BYTES 74-75 TO ADDRESS OF CCW IF REQUIRED
B2
LOAD REG 14 WITH ADDRESS OF USER'S ERROR ROUTINE
B3
SKIP
ERROPT =
IGN
OMTD
B1
H1
REFERENCES TO--PEC5, PFB2,PFJ3
IJJF5XXX
C1
IJJE6XXX PF
WRITE A RECORD IF REQUIRED
IJJK8XXX
C2
BRANCH TO USER'S ROUTINE EXECUTE AND RETURN
C3
SVC 50 CANCEL
C5
STORE RECORD NUMBER IN COUNT FIELD IF REQUIRED
PF F1
D5
IJJF9XXX
D1
IJJH8XXX PF
TEST FOR I/O COMPLETE IF REQUIRED
D2
TRC PROCESSING IF REQUIRED
D4
IJJG3XXX PF
SPACE ONE LINE
D5
SVC 0 EXECUTE CHANNEL PROGRAM
IJJI0XXX
E1
WLR OR WRITE ERROR
YES
NO
A2
E3
DIMOD EXCP ROUTINE
E4
IS DEVICE A 2540 PUNCH
NO
YES
E5
IJJH8XXX PE
TEST FOR I/O COMPLETE IF REQUIRED
PF E4
IJJJ1XXX
F1
SYSLST OR SYSPCH ON OUTPUT TAPE
NO
YES
IJJF5XXX
F2
LOAD REGISTER C WITH ADDRESS OF ROUTINE FROM DTF
IJJG3XXX
F3
SVC 0 EXECUTE CHANNEL PROGRAM
F4
SVC 7 WAIT IF REQUIRED
F5
SVC 0 PRINT
PF A3
IJJF3XXX
G1
SVC 2 FETCH $$BCMT07
IJJF6XXX
G2
EXECUTE LOAD IOREC INSTR IN DTF TO LOAD ADDR OF ALTRN IOAREA
G3
DIMOD WAIT ROUTINE
IJJH5XXX
G4
SVC 2 FETCH $$BERRTN
G5
RETURN TO CALLING ROUTINE
H1
PF F1
IJJJ8XXX
H1
IJJE6XXX PF
WRITE A RECORD
IJJL5XXX
H2
SET FIRST PASS INDICATOR TO 0
IJJH8XXX
H3
SVC 7 WAIT IF REQUIRED
IJJ44XXX
H4
MODIFY COMMAND CODES IN ERROR CCW'S
H5
SET POINTER TO AN CHAIN BIT IN FIRST OUTPUT CCW
D5
PF E2
IJJJ7XXX
J1
SAVE RETURN ADDR IN REG 0 AND LOAD ACDR ON NEXT I/O AREA IN REG 14
J2
RETURN TO ADDRESS IN REGISTER 14
J3
RETURN TO CALLING ROUTINE
J5
SET STACKER SELECT V OR W IN FIRST CCW COMMAND CODE
K1
SUBTRACT 1 FROM I/O AREA ACDR IN REG 14 TO POINT TO CCNTROL CHAR
K5
RESET POINTER TO SECOND OUTPUT CCW
D5

Chart PF. DIMOD: PUT Macro, Two I/O Areas (Part 2 of 2)

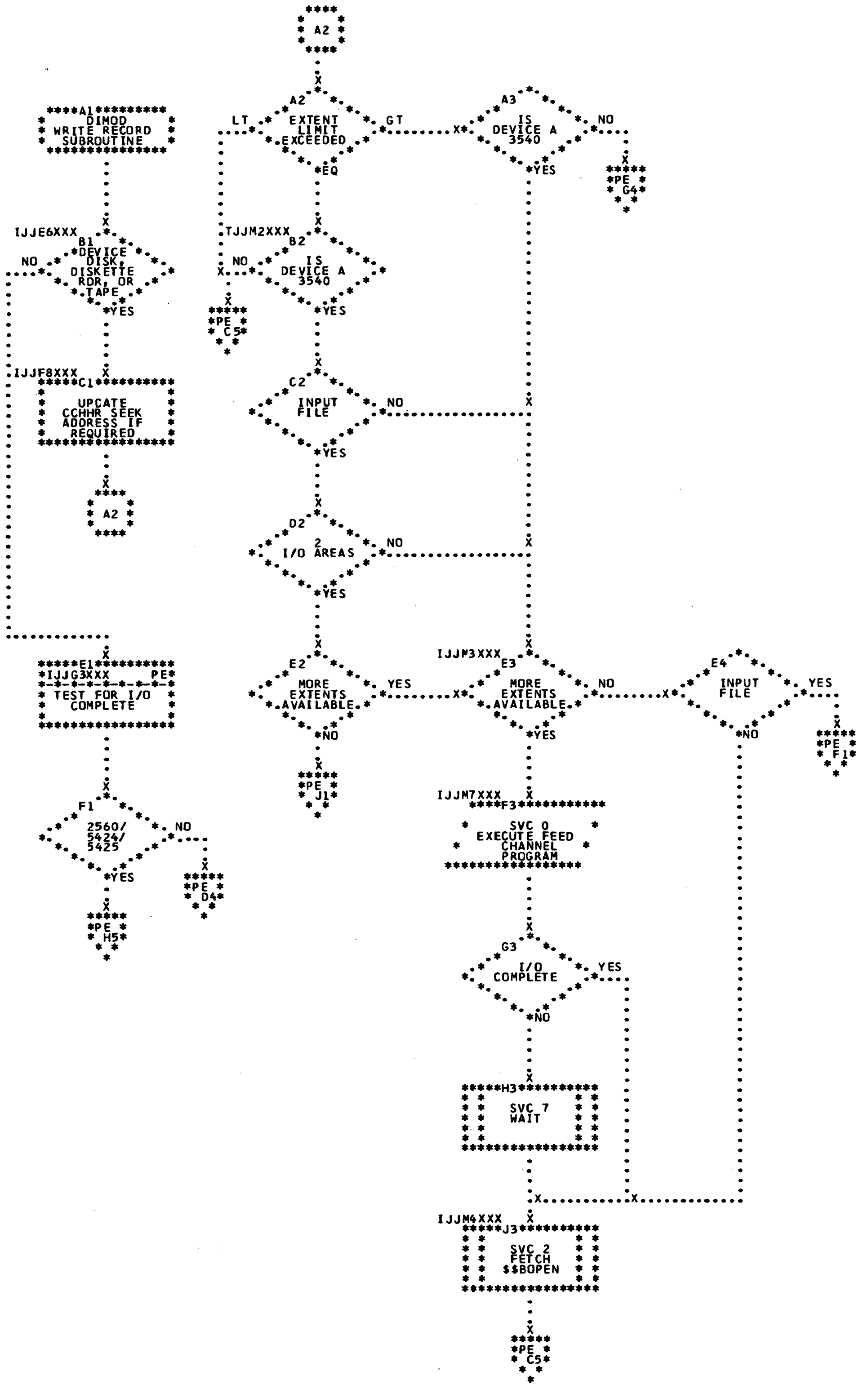


Chart PG. $$BERRTN: Punch Error Recovery Routine

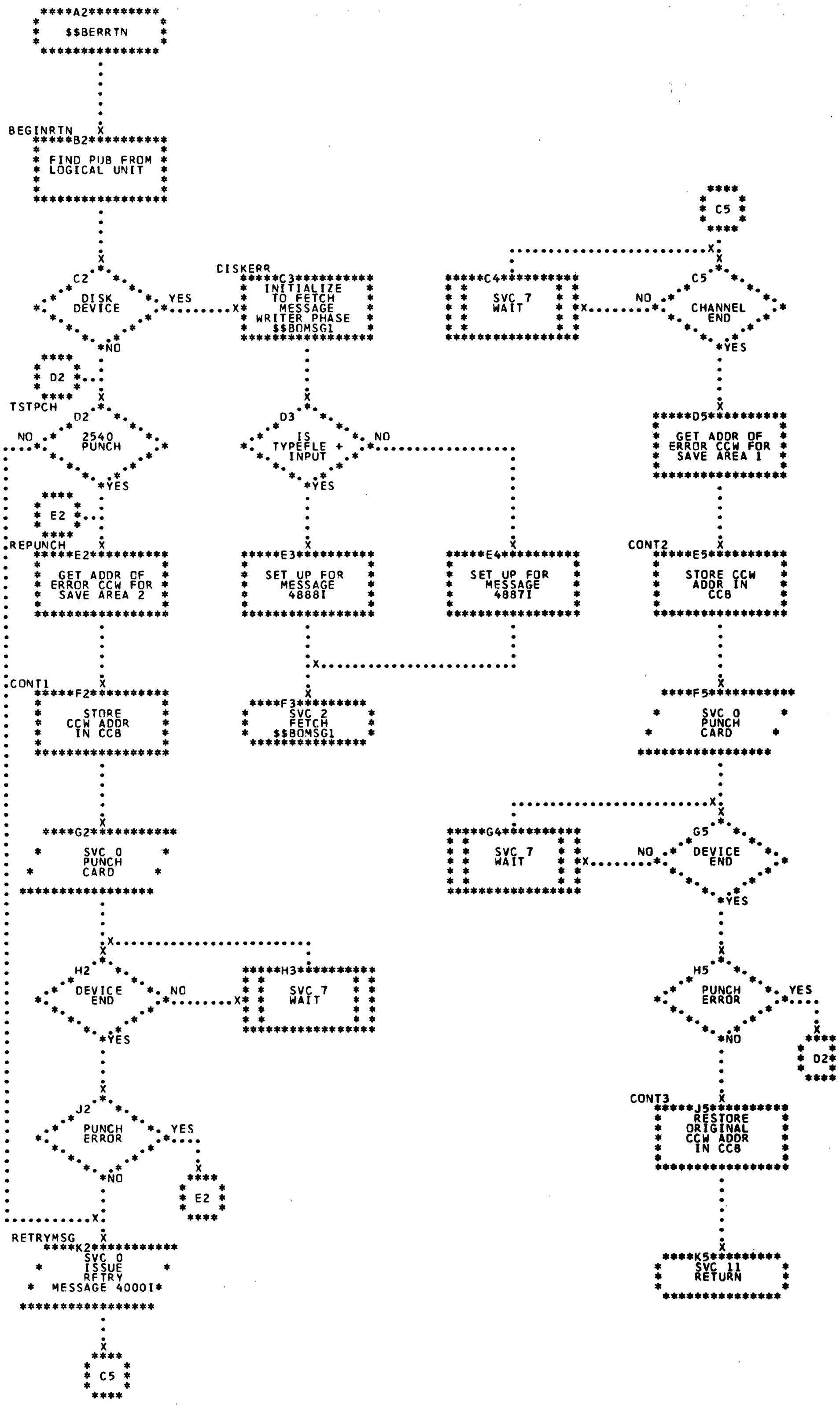


Chart PH. $$BERPTP: 1018 Punch Tape Punch Error Recovery Routine (Part 1 of 2)

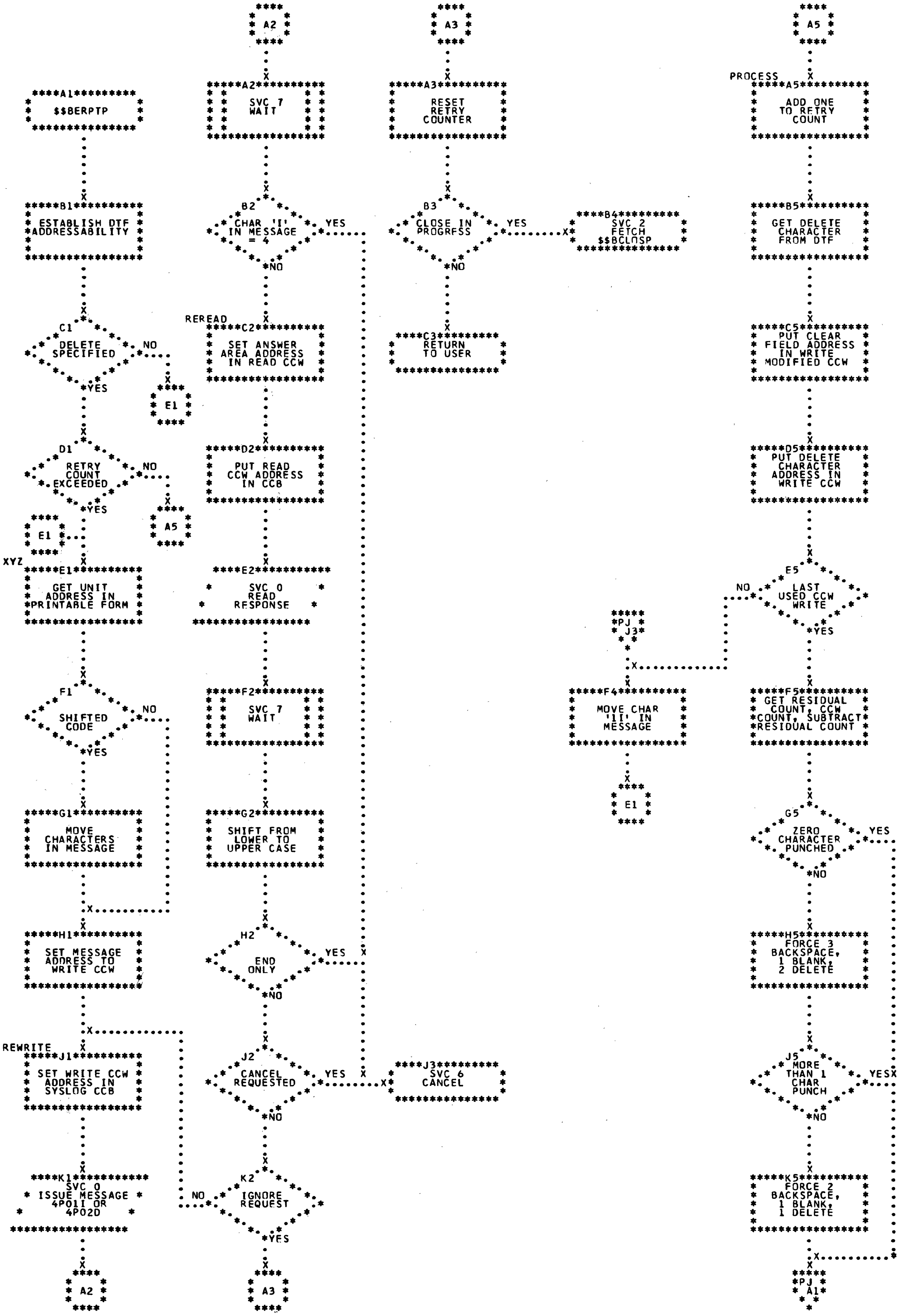

A1
$$BERPTP
B1
ESTABLISH DTF ADDRESSABILITY
C1
DELETE SPECIFIED
NO
YES
E1
D1
RETRY COUNT EXCEEDED
NO
YES
A5
E1
XYZ
E1
GET UNIT ADDRESS IN PRINTABLE FORM
F1
SHIFTED CODE
NO
YES
G1
MOVE CHARACTERS IN MESSAGE
H1
SET MESSAGE ADDRESS TO WRITE CCW
REWRITE
J1
SET WRITE CCW ADDRESS IN SYSLOG CCB
K1
SVC 0 ISSUE MESSAGE 4P01I OR 4P02D
A2
A2
A2
SVC 7 WAIT
B2
CHAR 'I' IN MESSAGE = 4
YES
NO
REREAD
C2
SET ANSWER AREA ADDRESS IN READ CCW
D2
PUT READ CCW ADDRESS IN CCB
E2
SVC 0 READ RESPONSE
F2
SVC 7 WAIT
G2
SHIFT FROM LOWER TO UPPER CASE
H2
END ONLY
YES
NO
J2
CANCEL REQUESTED
YES
NO
K2
IGNORE REQUEST
NO
YES
A3
A3
A3
RESET RETRY COUNTER
B3
CLOSE IN PROGRESS
YES
NO
B4
SVC 2 FETCH $$BCLOSP
C3
RETURN TO USER
J3
SVC 6 CANCEL
PJ J3
F4
MOVE CHAR 'II' IN MESSAGE
E1
A5
PROCESS
A5
ADD ONE TO RETRY COUNT
B5
GET DELETE CHARACTER FROM DTF
C5
PUT CLEAR FIELD ADDRESS IN WRITE MODIFIED CCW
D5
PUT DELETE CHARACTER ADDRESS IN WRITE CCW
E5
LAST USED CCW WRITE
NO
YES
F5
GET RESIDUAL COUNT, CCW COUNT, SUBTRACT RESIDUAL COUNT
G5
ZERO CHARACTER PUNCHED
YES
NO
H5
FORCE 3 BACKSPACE, 1 BLANK, 2 DELETE
J5
MORE THAN 1 CHAR PUNCH
YES
NO
K5
FORCE 2 BACKSPACE, 1 BLANK, 1 DELETE
PJ A1

Chart PJ. $$BEFPTP: 1018 Punch Tape Punch Error Recovery Routine (Part 2 of 2)

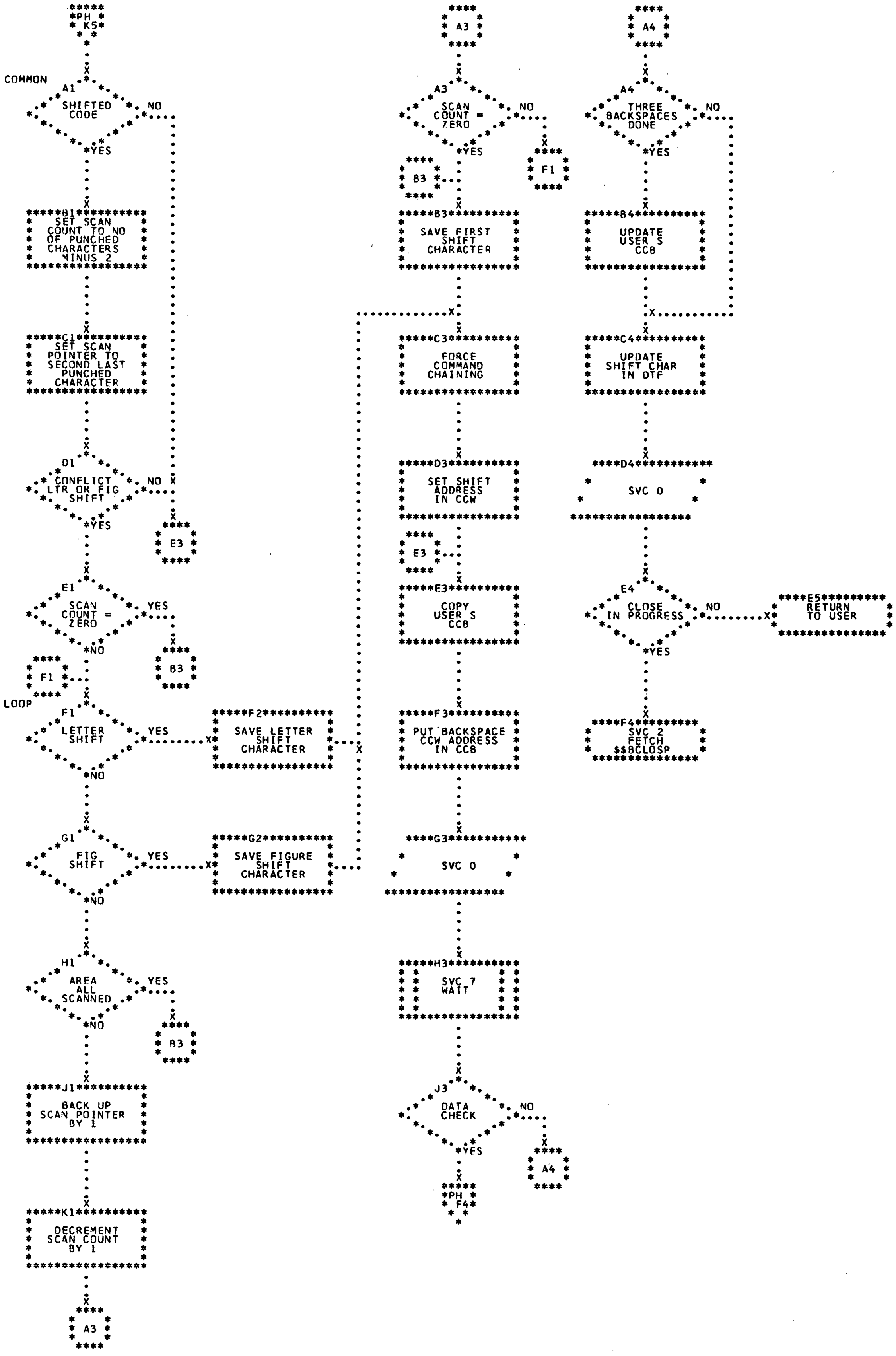


Chart QA. $$BOCP01: Open Device Independent Files, Phase 1

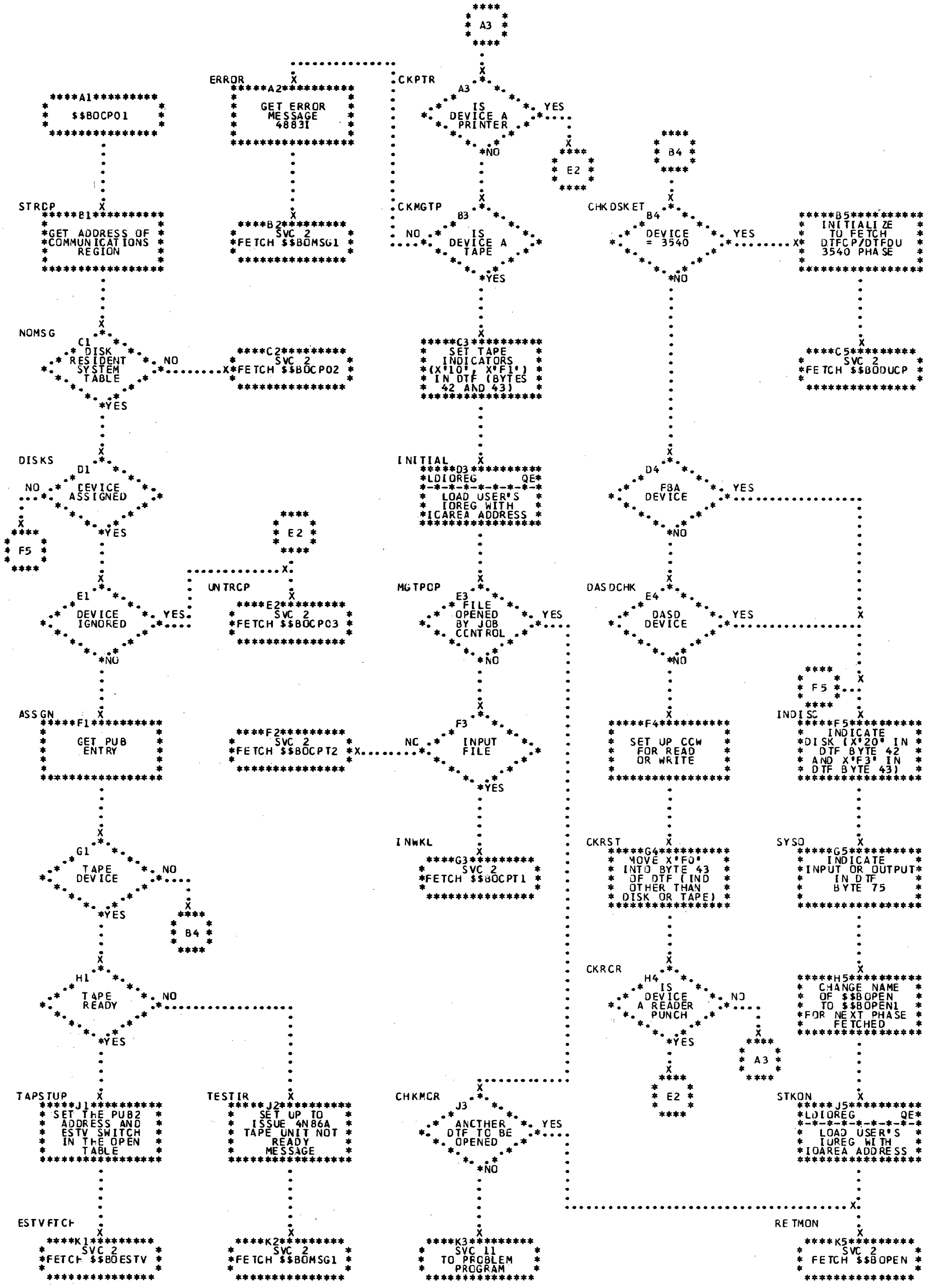

A3
A1 $$BOCP01
ERROR A2 GET ERROR MESSAGE 4883I
CKPTR A3 IS DEVICE A PRINTER YES NO
E2
B4
STRCP B1 GET ADDRESS OF COMMUNICATIONS REGION
B2 SVC 2 FETCH $$BOMSG1
CKMGTP B3 IS DEVICE A TAPE NO YES
CHKDSKET B4 DEVICE = 3540 YES NO
B5 INITIALIZE TO FETCH DTFCP/DTFDU 3540 PHASE
NOMSG C1 DISK RESIDENT SYSTEM TABLE NO YES
C2 SVC 2 FETCH $$BOCP02
C3 SET TAPE INDICATORS (X'10', X'F1') IN DTF (BYTES 42 AND 43)
C5 SVC 2 FETCH $$BODUCP
DISKS D1 DEVICE ASSIGNED NO YES
F5
INITIAL D3 LDIOREG QE LOAD USER'S IOREG WITH ICAREA ADDRESS
D4 FBA DEVICE YES NO
E2
UNTRCP E2 SVC 2 FETCH $$BOCP03
E1 DEVICE IGNORED YES NO
MGTPOP E3 FILE OPENED BY JOB CONTROL YES NO
DASDCHK E4 DASD DEVICE YES NO
F5
ASSGN F1 GET PUB ENTRY
F2 SVC 2 FETCH $$BOCPT2
F3 INPUT FILE NO YES
F4 SET UP CCW FOR READ OR WRITE
INDISC F5 INDICATE DISK (X'20' IN DTF BYTE 42 AND X'F3' IN DTF BYTE 43)
G1 TAPE DEVICE NO YES
B4
INWKL G3 SVC 2 FETCH $$BOCPT1
CKRST G4 MOVE X'F0' INTO BYTE 43 OF DTF (IND OTHER THAN DISK OR TAPE)
SYSO G5 INDICATE INPUT OR OUTPUT IN DTF BYTE 75
H1 TAPE READY NO YES
CKRCR H4 IS DEVICE A READER PUNCH NO YES
A3
E2
H5 CHANGE NAME OF $$BOPEN TO $$BOPEN1 FOR NEXT PHASE FETCHED
TAPSTUP J1 SET THE PUB2 ADDRESS AND ESTV SWITCH IN THE OPEN TABLE
TESTIR J2 SET UP TO ISSUE 4N86A TAPE UNIT NOT READY MESSAGE
CHKMCR J3 ANOTHER DTF TO BE OPENED YES NO
STKON J5 LDIOREG QE LOAD USER'S IOREG WITH IOAREA ADDRESS
ESTVFTCH K1 SVC 2 FETCH $$BOESTV
K2 SVC 2 FETCH $$BOMSG1
K3 SVC 11 TO PROBLEM PROGRAM
RETMON K5 SVC 2 FETCH $$BOPEN

Chart QE. $$BCCP02: Open Device Independent Files, Phase 2

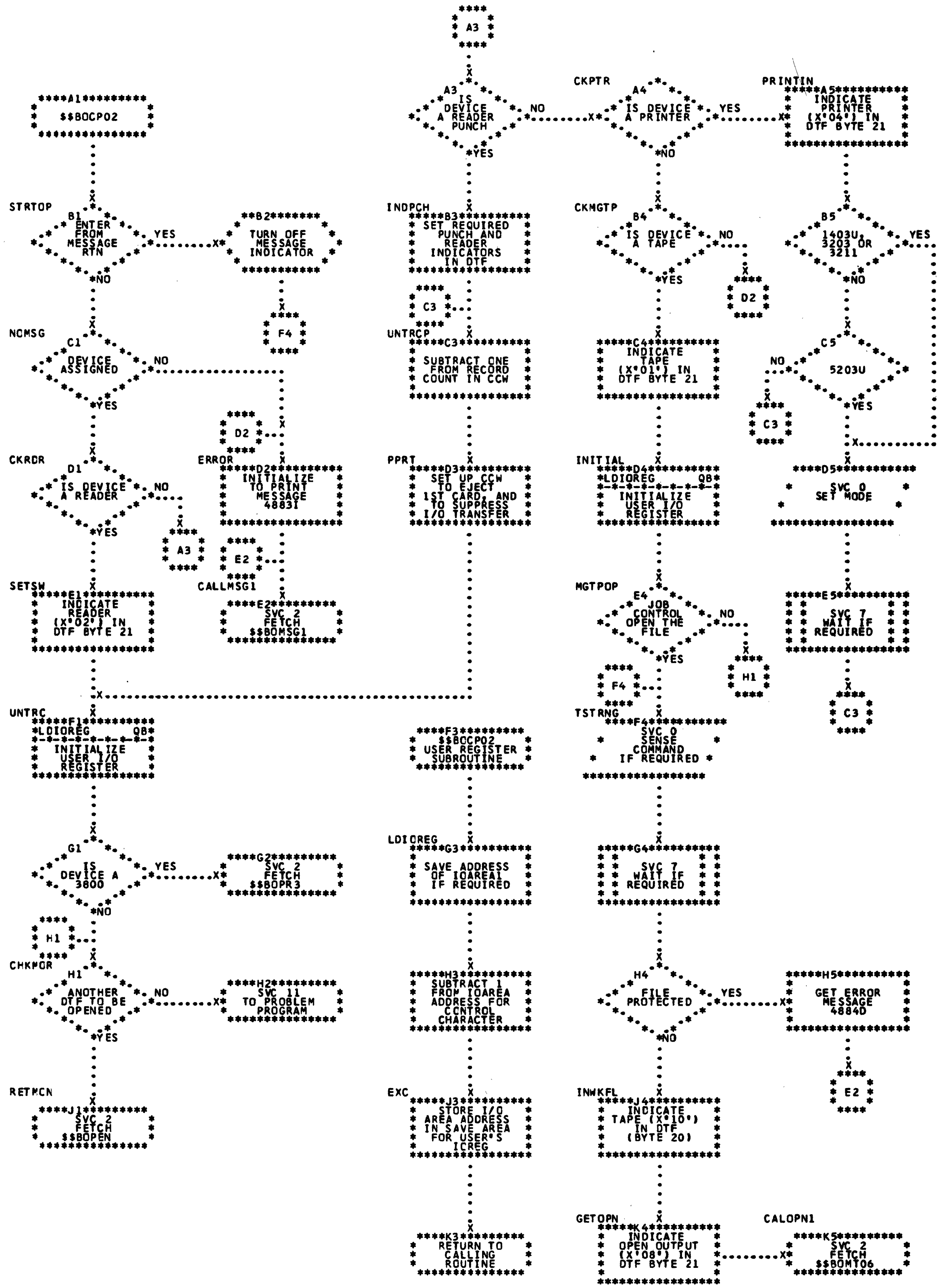

A3
A1
$$BOCP02
A3
IS DEVICE A READER PUNCH
NO
YES
CKPTR
A4
IS DEVICE A PRINTER
YES
NO
PRINTIN
A5
INDICATE PRINTER (X'04') IN DTF BYTE 21
STRTOP
B1
ENTER FROM MESSAGE RTN
YES
NO
B2
TURN OFF MESSAGE INDICATOR
F4
INDPCH
B3
SET REQUIRED PUNCH AND READER INDICATORS IN DTF
CKMGTP
B4
IS DEVICE A TAPE
NO
YES
D2
B5
1403U, 3203 OR 3211
YES
NO
C3
NCMSG
C1
DEVICE ASSIGNED
NO
YES
UNTRCP
C3
SUBTRACT ONE FROM RECORD COUNT IN CCW
C4
INDICATE TAPE (X'01') IN DTF BYTE 21
C5
5203U
NO
YES
C3
D2
CKRDR
D1
IS DEVICE A READER
NO
YES
A3
ERROR
D2
INITIALIZE TO PRINT MESSAGE 4883I
PPRT
D3
SET UP CCW TO EJECT 1ST CARD, AND TO SUPPRESS I/O TRANSFER
INITIAL
D4
LDIOREG QB
INITIALIZE USER I/O REGISTER
D5
SVC 0 SET MODE
E2
SETSW
E1
INDICATE READER (X'02') IN DTF BYTE 21
CALLMSG1
E2
SVC 2 FETCH $$BOMSG1
MGTPOP
E4
JOB CONTROL OPEN THE FILE
NO
YES
H1
F4
E5
SVC 7 WAIT IF REQUIRED
C3
UNTRC
F1
LDIOREG QB
INITIALIZE USER I/O REGISTER
F3
$$BOCP02 USER REGISTER SUBROUTINE
TSTRNG
F4
SVC 0 SENSE COMMAND IF REQUIRED
G1
IS DEVICE A 3800
YES
NO
G2
SVC 2 FETCH $$BOPR3
LDIOREG
G3
SAVE ADDRESS OF IOAREA1 IF REQUIRED
G4
SVC 7 WAIT IF REQUIRED
H1
CHKMOR
H1
ANOTHER DTF TO BE OPENED
NO
YES
H2
SVC 11 TO PROBLEM PROGRAM
H3
SUBTRACT 1 FROM IOAREA ADDRESS FOR CONTROL CHARACTER
H4
FILE PROTECTED
YES
NO
H5
GET ERROR MESSAGE 4884D
E2
RETMCN
J1
SVC 2 FETCH $$BOPEN
EXC
J3
STORE I/O AREA ADDRESS IN SAVE AREA FOR USER'S IOREG
INWKFL
J4
INDICATE TAPE (X'10') IN DTF (BYTE 20)
GETOPN
K4
INDICATE OPEN OUTPUT (X'08') IN DTF BYTE 21
CALOPN1
K5
SVC 2 FETCH $$BOMT06
K3
RETURN TO CALLING ROUTINE

Chart QC. $$BOCP03: Open Device Independent Files, Phase 3

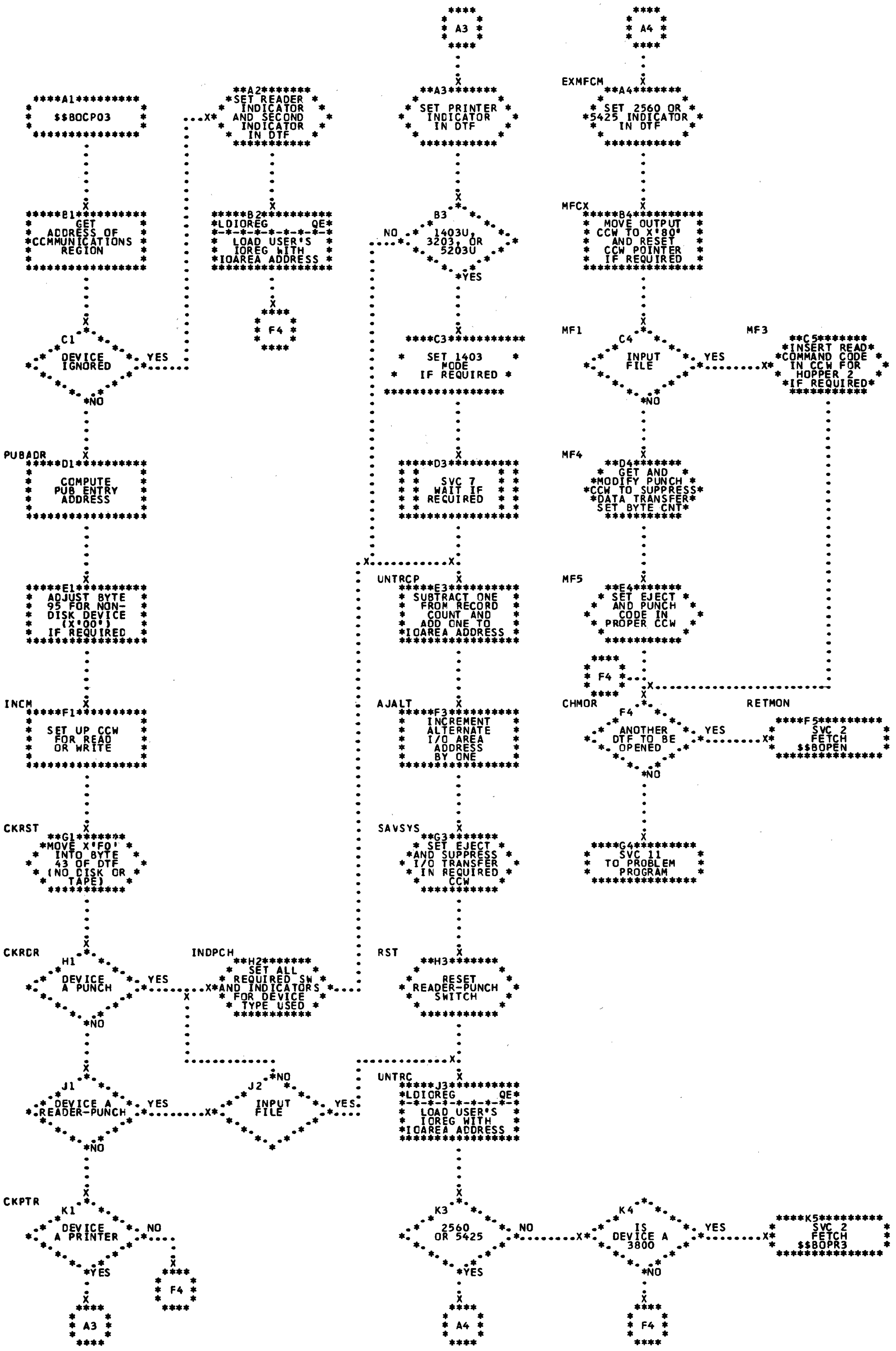

A1
$$BOCP03
B1
GET ADDRESS OF COMMUNICATIONS REGION
C1
DEVICE IGNORED
YES
NO
PUBADR
D1
COMPUTE PUB ENTRY ADDRESS
E1
ADJUST BYTE 95 FOR NON-DISK DEVICE (X'00') IF REQUIRED
INCM
F1
SET UP CCW FOR READ OR WRITE
CKRST
G1
MOVE X'F0' INTO BYTE 43 OF DTF (NO DISK OR TAPE)
CKRDR
H1
DEVICE A PUNCH
YES
NO
J1
DEVICE A READER-PUNCH
YES
NO
CKPTR
K1
DEVICE A PRINTER
NO
YES
F4
A3
A2
SET READER INDICATOR AND SECOND INDICATOR IN DTF
B2
LDIOREG QE
LOAD USER'S IOREG WITH IOAREA ADDRESS
F4
INDPCH
H2
SET ALL REQUIRED SW AND INDICATORS FOR DEVICE TYPE USED
J2
INPUT FILE
YES
NO
A3
A3
SET PRINTER INDICATOR IN DTF
B3
1403U, 3203, OR 5203U
NO
YES
C3
SET 1403 MODE IF REQUIRED
D3
SVC 7 WAIT IF REQUIRED
UNTRCP
E3
SUBTRACT ONE FROM RECORD COUNT AND ADD ONE TO IOAREA ADDRESS
AJALT
F3
INCREMENT ALTERNATE I/O AREA ADDRESS BY ONE
SAVSYS
G3
SET EJECT AND SUPPRESS I/O TRANSFER IN REQUIRED CCW
RST
H3
RESET READER-PUNCH SWITCH
UNTRC
J3
LDIOREG QE
LOAD USER'S IOREG WITH IOAREA ADDRESS
K3
2560 OR 5425
NO
YES
A4
A4
EXMFCM
A4
SET 2560 OR 5425 INDICATOR IN DTF
MFCX
B4
MOVE OUTPUT CCW TO X'80' AND RESET CCW POINTER IF REQUIRED
MF1
C4
INPUT FILE
YES
NO
MF4
D4
GET AND MODIFY PUNCH CCW TO SUPPRESS DATA TRANSFER SET BYTE CNT
MF5
E4
SET EJECT AND PUNCH CODE IN PROPER CCW
F4
CHMOR
F4
ANOTHER DTF TO BE OPENED
YES
NO
G4
SVC 11 TO PROBLEM PROGRAM
K4
IS DEVICE A 3800
YES
NO
F4
MF3
C5
INSERT READ COMMAND CODE IN CCW FOR HOPPER 2 IF REQUIRED
RETMON
F5
SVC 2 FETCH $$BOPEN
K5
SVC 2 FETCH $$BOPR3

Chart QD. $$BOCP11: Open DTFCP (Version 1 Only), Phase 1 (Part 1 of 2)

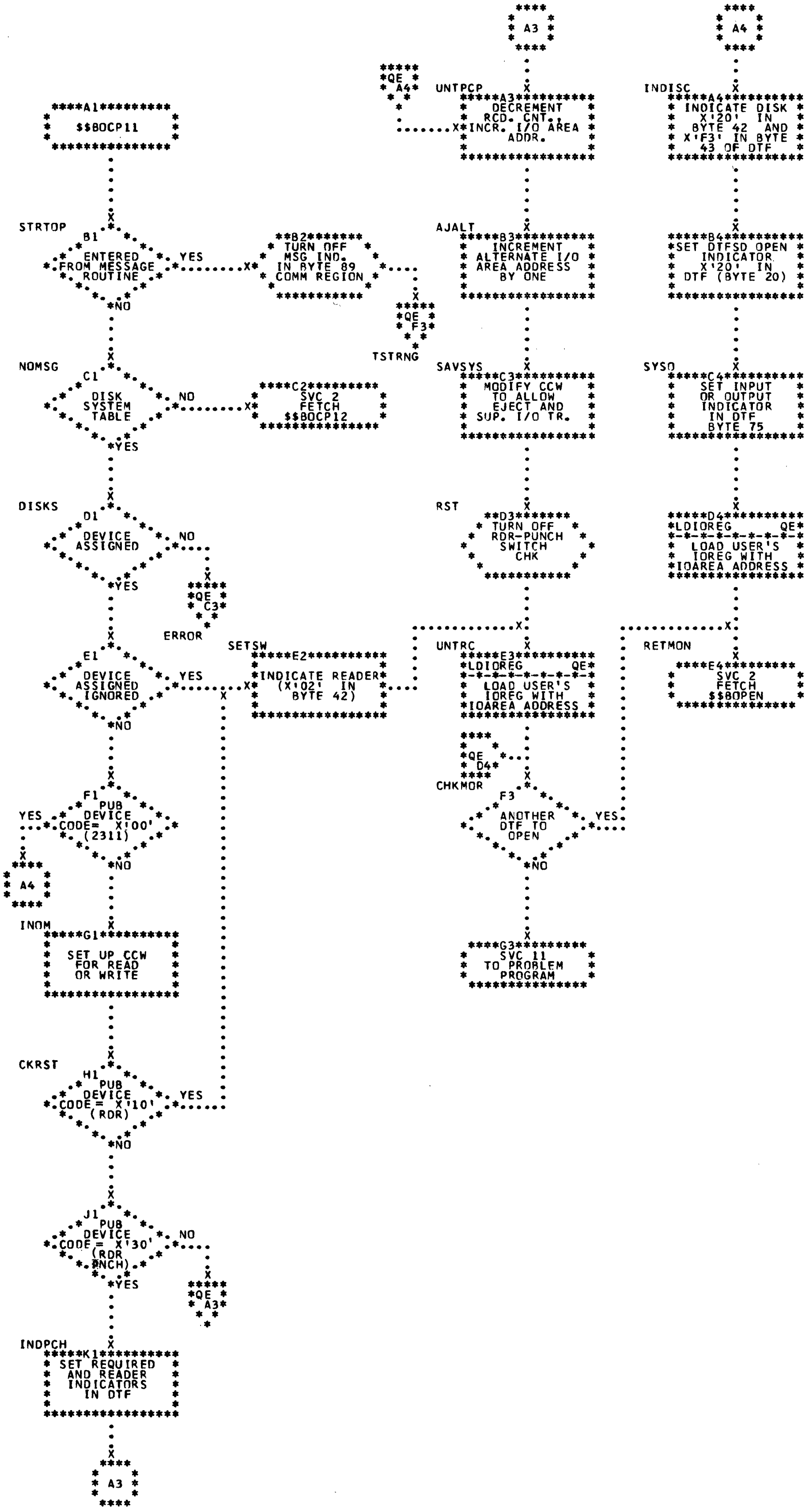

A1
$$BOCP11
STRTOP
B1
ENTERED FROM MESSAGE ROUTINE
YES
NO
B2
TURN OFF MSG IND. IN BYTE 89 COMM REGION
QE F3
TSTRNG
NOMSG
C1
DISK SYSTEM TABLE
NO
YES
C2
SVC 2 FETCH $$BOCP12
DISKS
D1
DEVICE ASSIGNED
NO
YES
QE C3
ERROR
E1
DEVICE ASSIGNED IGNORED
YES
NO
SETSW
E2
INDICATE READER (X'02' IN BYTE 42)
F1
PUB DEVICE CODE= X'00' (2311)
YES
NO
A4
INDM
G1
SET UP CCW FOR READ OR WRITE
CKRST
H1
PUB DEVICE CODE= X'10' (RDR)
YES
NO
J1
PUB DEVICE CODE= X'30' (RDR PNCH)
NO
YES
QE A3
INDPCH
K1
SET REQUIRED AND READER INDICATORS IN DTF
A3
A3
QE A4
UNTPCP
A3
DECREMENT RCD. CNT., INCR. I/O AREA ADDR.
AJALT
B3
INCREMENT ALTERNATE I/O AREA ADDRESS BY ONE
SAVSYS
C3
MODIFY CCW TO ALLOW EJECT AND SUP. I/O TR.
RST
D3
TURN OFF RDR-PUNCH SWITCH CHK
UNTRC
E3
LDIOREG QE
LOAD USER'S IOREG WITH IOAREA ADDRESS
QE D4
CHKMOR
F3
ANOTHER DTF TO OPEN
YES
NO
G3
SVC 11 TO PROBLEM PROGRAM
A4
INDISC
A4
INDICATE DISK X'20' IN BYTE 42 AND X'F3' IN BYTE 43 OF DTF
B4
SET DTFSD OPEN INDICATOR X'20' IN DTF (BYTE 20)
SYSO
C4
SET INPUT OR OUTPUT INDICATOR IN DTF BYTE 75
D4
LDIOREG QE
LOAD USER'S IOREG WITH IOAREA ADDRESS
RETMON
E4
SVC 2 FETCH $$BOPEN

Chart QE. $$BOCP11: Open DTFCP (Version 1 Only), Phase 1 (Part 2 of 2)

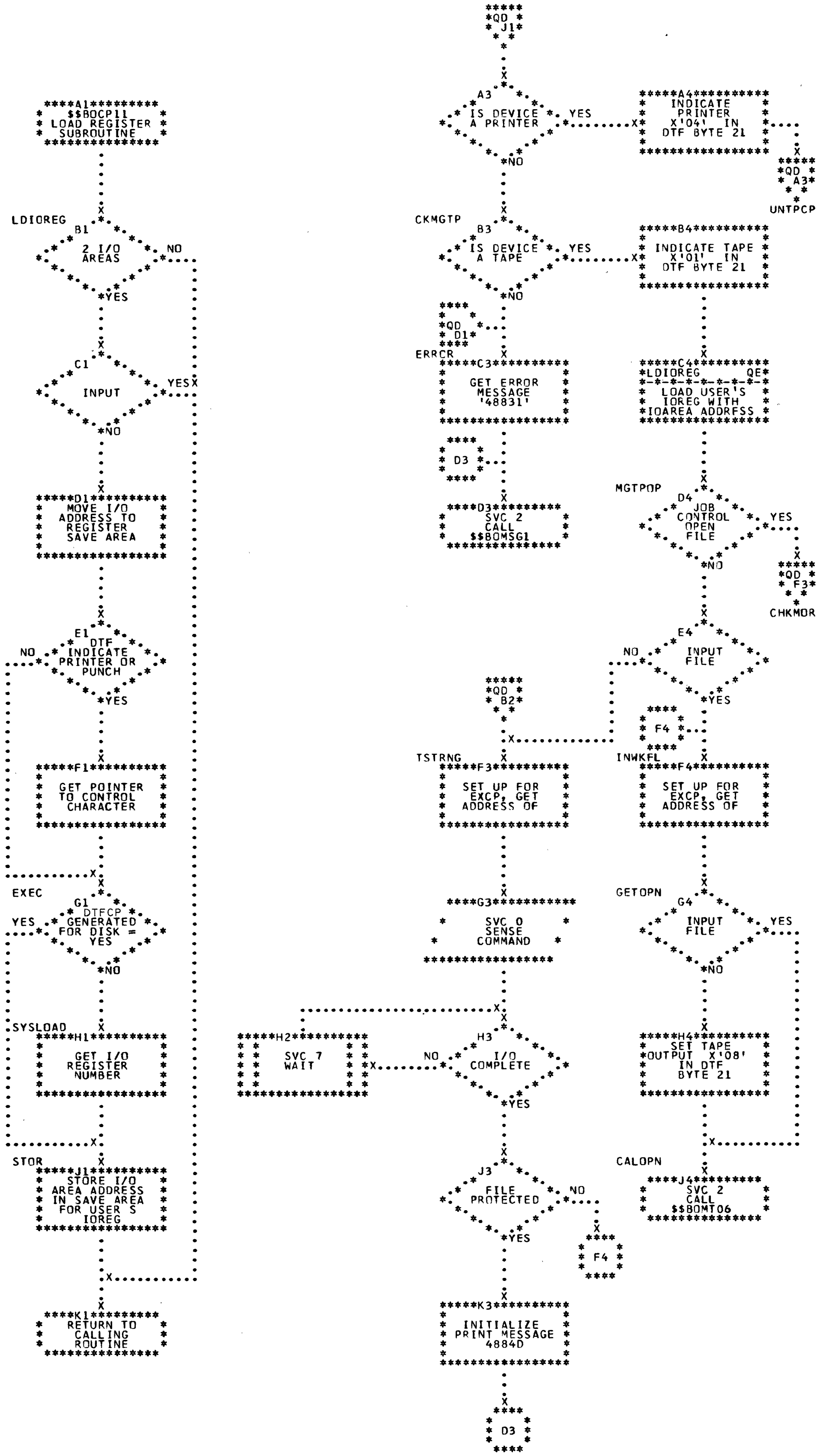

QD J1
A1
$$BOCP11 LOAD REGISTER SUBROUTINE
LDIOREG
B1
2 I/O AREAS
NO
YES
C1
INPUT
YES
NO
D1
MOVE I/O ADDRESS TO REGISTER SAVE AREA
E1
DTF INDICATE PRINTER OR PUNCH
NO
YES
F1
GET POINTER TO CONTROL CHARACTER
EXEC
G1
DTFCP GENERATED FOR DISK = YES
YES
NO
SYSLOAD
H1
GET I/O REGISTER NUMBER
STOR
J1
STORE I/O AREA ADDRESS IN SAVE AREA FOR USER S IOREG
K1
RETURN TO CALLING ROUTINE
A3
IS DEVICE A PRINTER
YES
NO
A4
INDICATE PRINTER X'04' IN DTF BYTE 21
QD A3
UNTPCP
CKMGTP
B3
IS DEVICE A TAPE
YES
NO
B4
INDICATE TAPE X'01' IN DTF BYTE 21
QD D1
ERRCR
C3
GET ERROR MESSAGE '48831'
D3
D3
SVC 2 CALL $$BOMSG1
C4
LDIOREG QE
LOAD USER'S IOREG WITH IOAREA ADDRESS
MGTPOP
D4
JOB CONTROL OPEN FILE
YES
NO
QD F3
CHKMOR
E4
INPUT FILE
NO
YES
QD B2
F4
TSTRNG
F3
SET UP FOR EXCP, GET ADDRESS OF
INWKFL
F4
SET UP FOR EXCP, GET ADDRESS OF
G3
SVC 0 SENSE COMMAND
GETOPN
G4
INPUT FILE
YES
NO
H2
SVC 7 WAIT
H3
I/O COMPLETE
NO
YES
H4
SET TAPE OUTPUT X'08' IN DTF BYTE 21
J3
FILE PROTECTED
NO
YES
F4
CALOPN
J4
SVC 2 CALL $$BOMT06
K3
INITIALIZE PRINT MESSAGE 4884D
D3

Chart QF. $$BOCP12: Open DTFCP (Version 1 Only), Phase 2

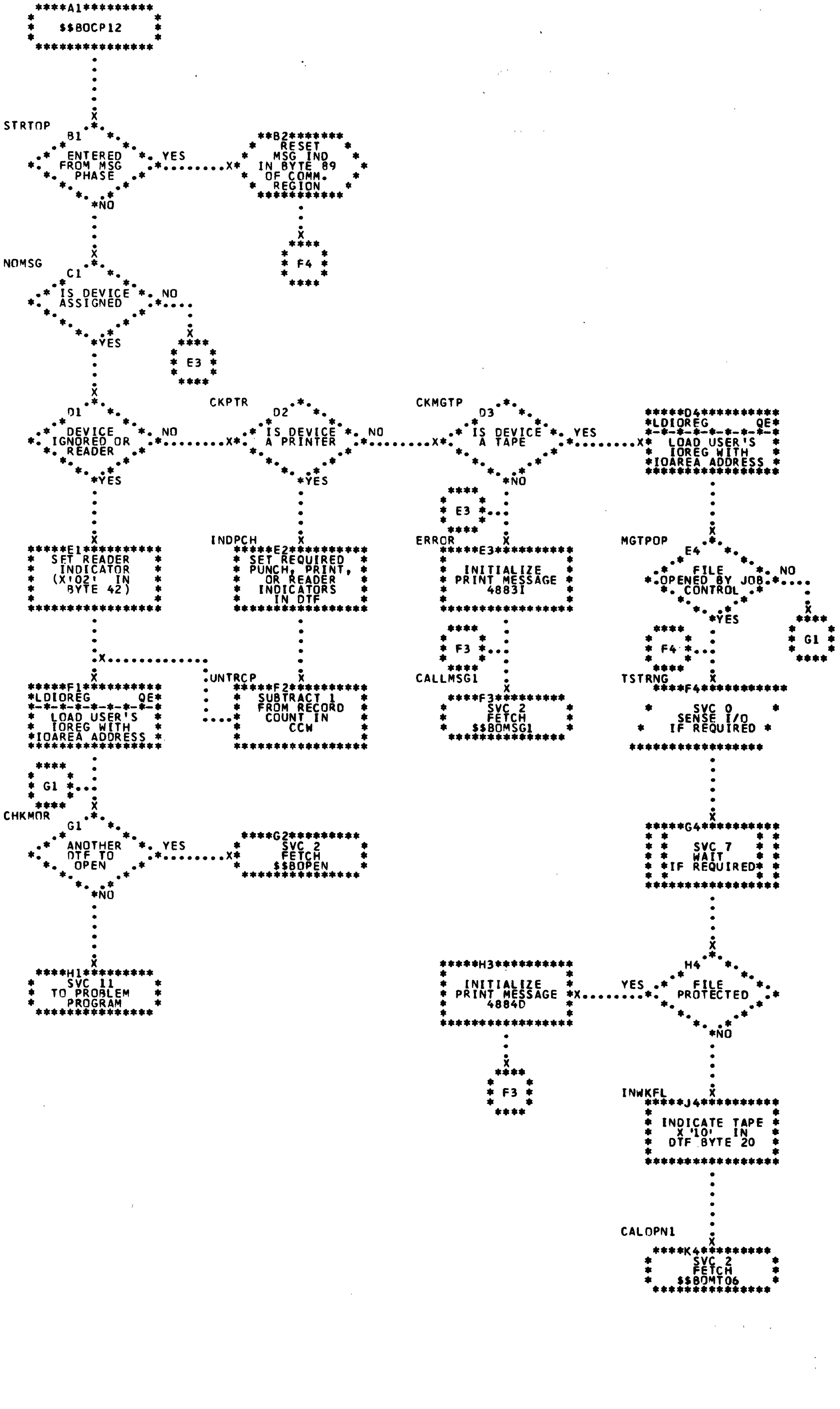


Chart QG. $$BCFR3: OPEN Processing for 3800 Printer file (Part 1 of 2)

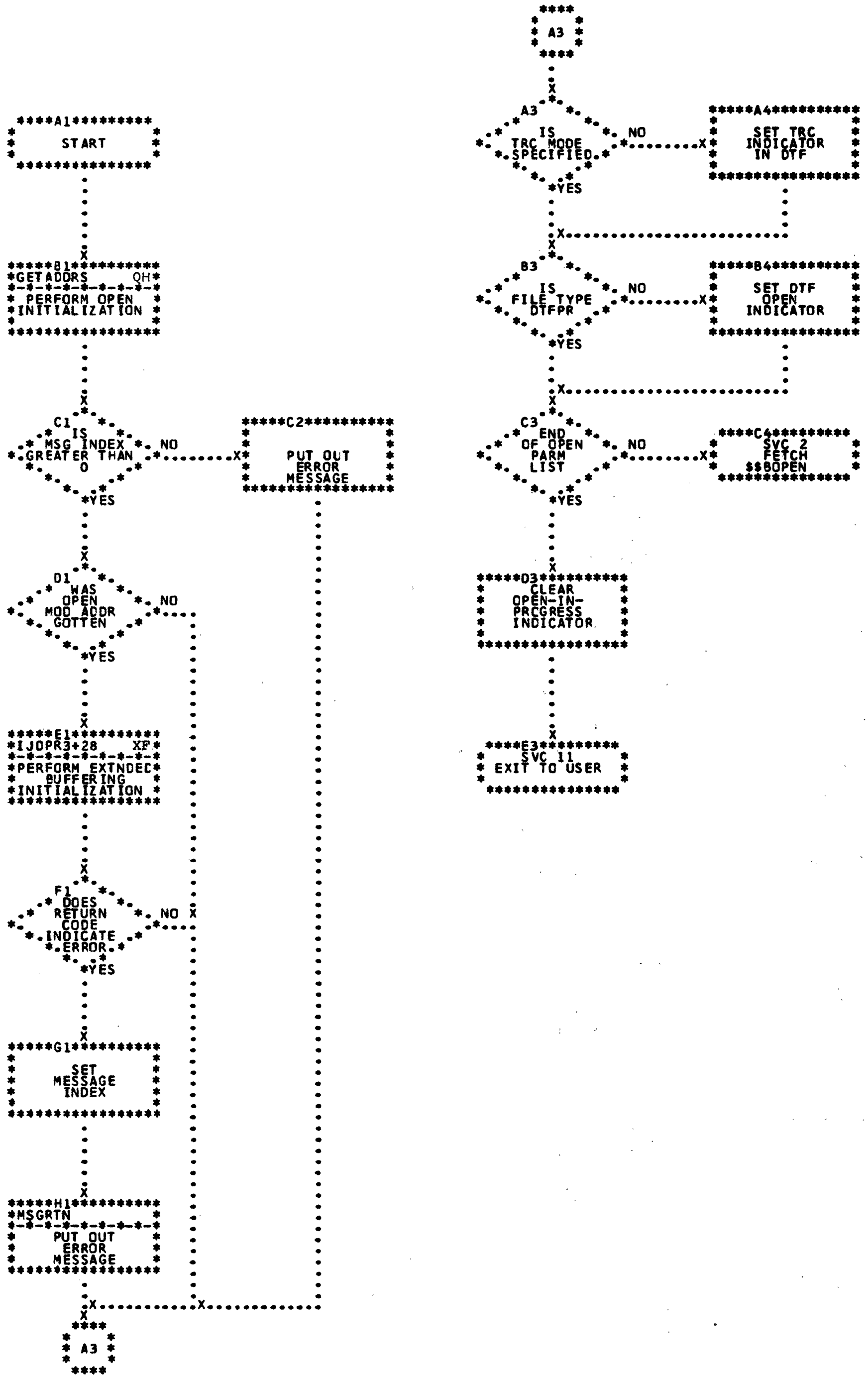


Chart QH. $$BOPR3: OPEN Processing for 3800 Printer file (Part 2 of 2)

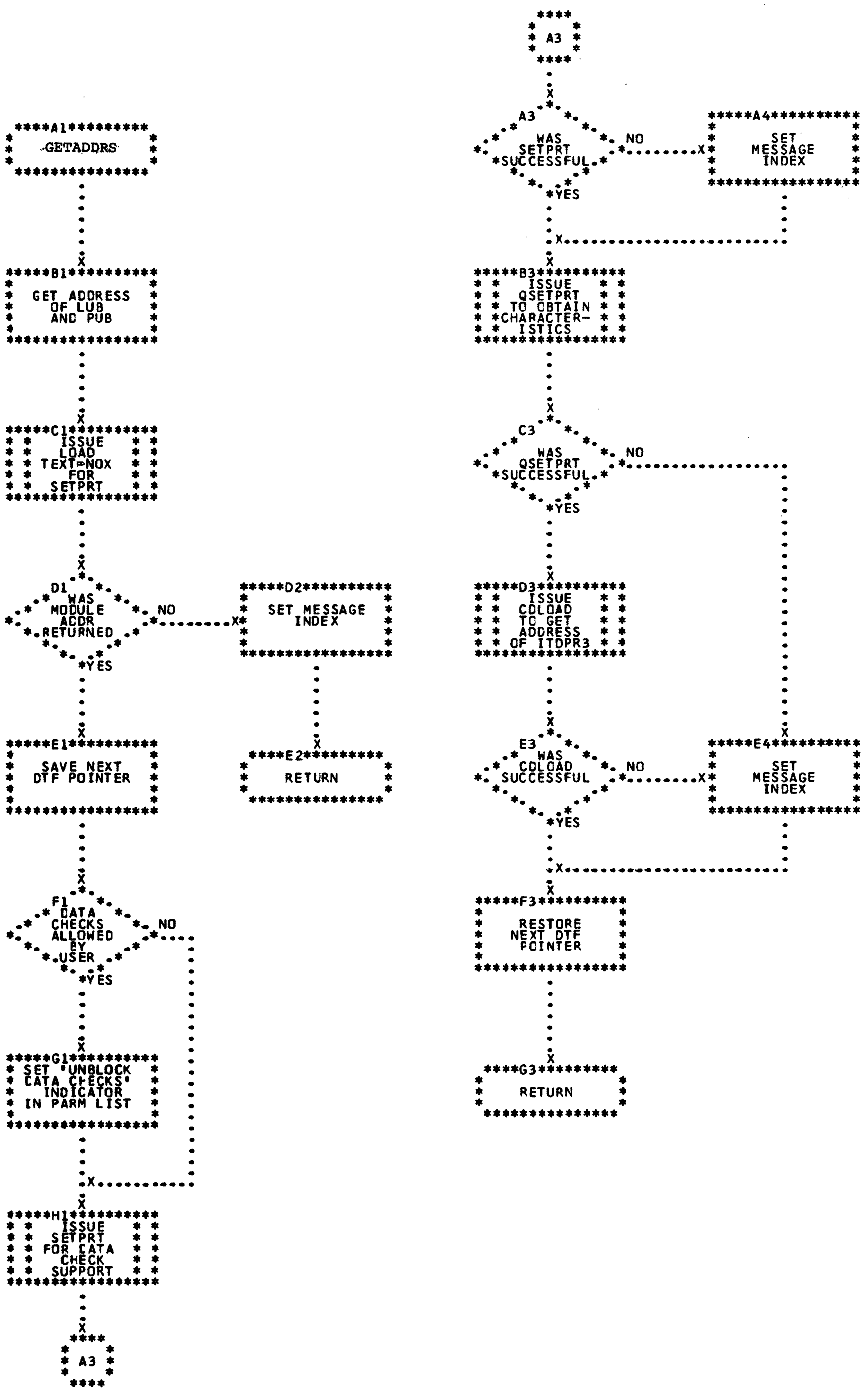


Chart RA. $$BOCPT1: Open DTFCP and DTFDI Input Tape (Part 1 of 3)

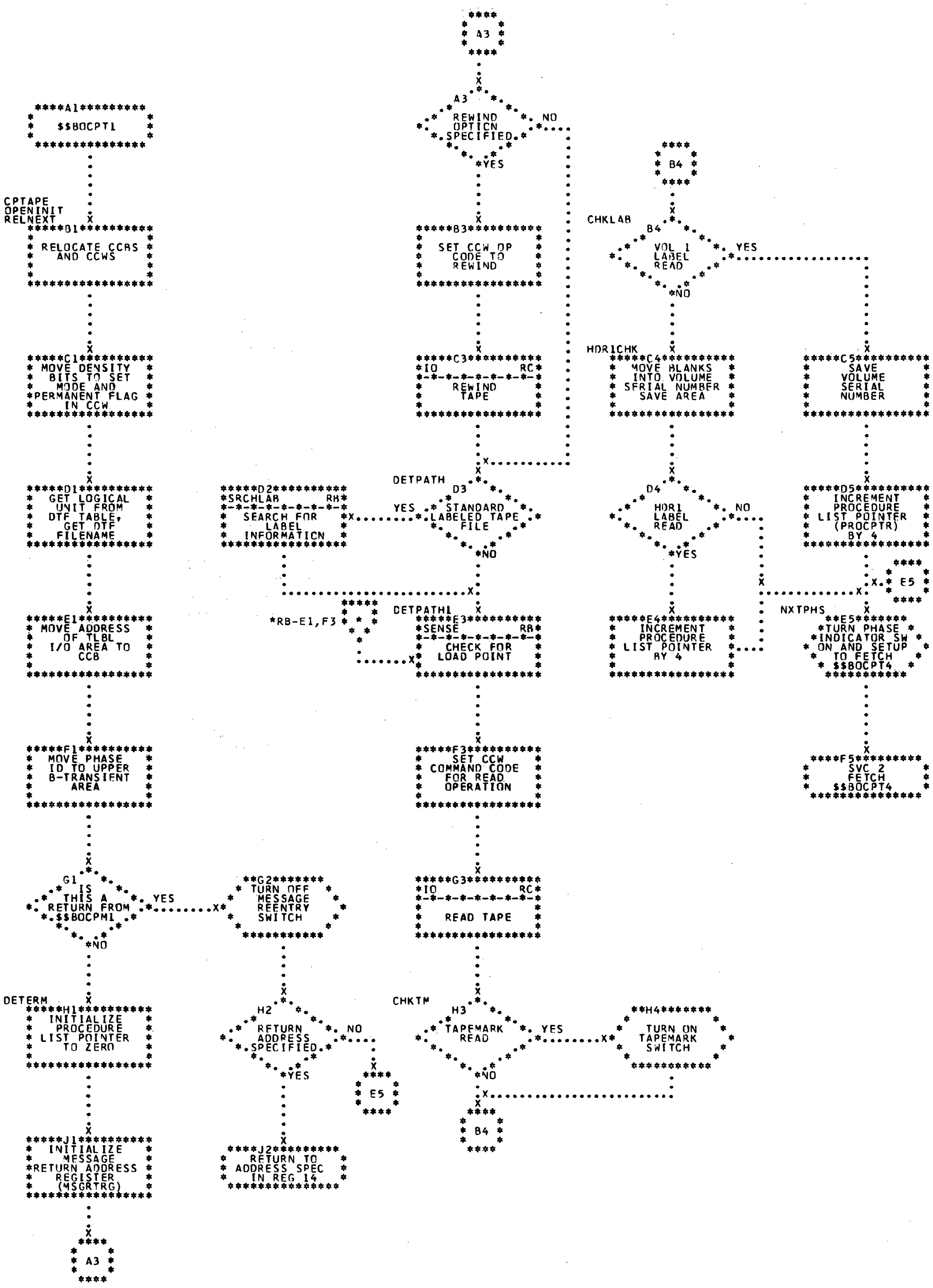

A1
$$BOCPT1
CPTAPE
OPENINIT
RELNEXT
B1
RELOCATE CCBS AND CCWS
C1
MOVE DENSITY BITS TO SET MODE AND PERMANENT FLAG IN CCW
D1
GET LOGICAL UNIT FROM DTF TABLE, GET DTF FILENAME
E1
MOVE ADDRESS OF TLBL I/O AREA TO CCB
F1
MOVE PHASE ID TO UPPER B-TRANSIENT AREA
G1
IS THIS A RETURN FROM $$BOCPM1
YES
NO
DETERM
H1
INITIALIZE PROCEDURE LIST POINTER TO ZERO
J1
INITIALIZE MESSAGE RETURN ADDRESS REGISTER (MSGRTRG)
A3
D2
SRCHLAB RB
SEARCH FOR LABEL INFORMATICN
*RB-E1,F3
G2
TURN OFF MESSAGE REENTRY SWITCH
H2
RETURN ADDRESS SPECIFIED
NO
YES
E5
J2
RETURN TO ADDRESS SPEC IN REG 14
A3
A3
REWIND OPTICN SPECIFIED
NO
YES
B3
SET CCW OP CODE TO REWIND
C3
IO RC
REWIND TAPE
DETPATH
D3
STANDARD LABELED TAPE FILE
YES
NO
DETPATH1
E3
SENSE RB
CHECK FOR LOAD POINT
F3
SET CCW COMMAND CODE FOR READ OPERATION
G3
IO RC
READ TAPE
CHKTM
H3
TAPEMARK READ
YES
NO
H4
TURN ON TAPEMARK SWITCH
B4
B4
CHKLAB
B4
VOL 1 LABEL READ
YES
NO
HDR1CHK
C4
MOVE BLANKS INTO VOLUME SERIAL NUMBER SAVE AREA
D4
HDR1 LABEL READ
NO
YES
E4
INCREMENT PROCEDURE LIST POINTER BY 4
C5
SAVE VOLUME SERIAL NUMBER
D5
INCREMENT PROCEDURE LIST POINTER (PROCPTR) BY 4
E5
NXTPHS
E5
TURN PHASE INDICATOR SW ON AND SETUP TO FETCH $$BOCPT4
F5
SVC 2 FETCH $$BOCPT4

Chart RB. $$BOCPT1: Subroutines for Open DTFCP and DTFDI Input Tape (Part 2 of 3)

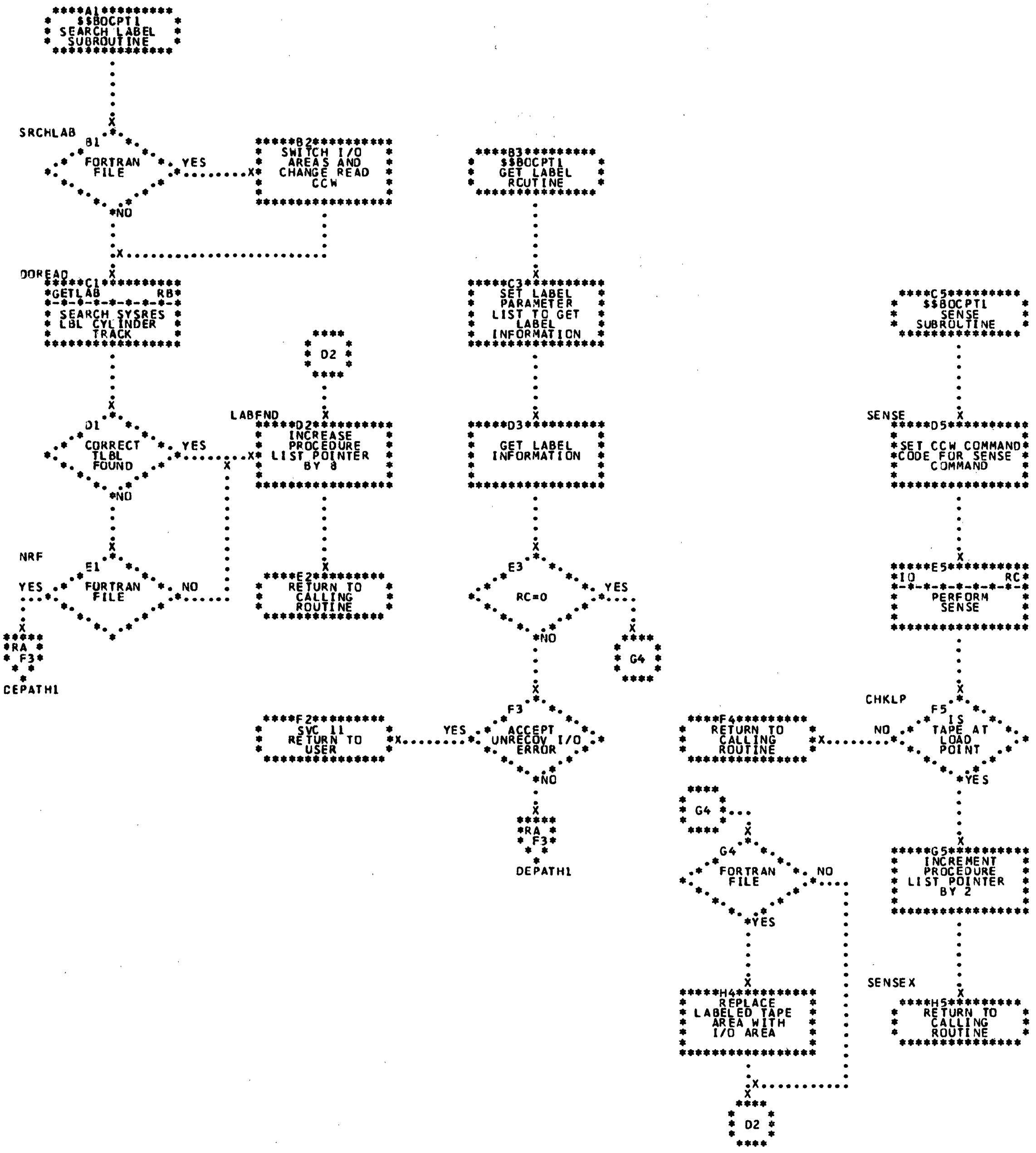

A1
$$BOCPT1 SEARCH LABEL SUBROUTINE
SRCHLAB
B1
FORTRAN FILE
YES
NO
B2
SWITCH I/O AREAS AND CHANGE READ CCW
B3
$$BOCPT1 GET LABEL ROUTINE
DOREAD
C1
GETLAB RB
SEARCH SYSRES LBL CYLINDER TRACK
C3
SET LABEL PARAMETER LIST TO GET LABEL INFORMATION
C5
$$BOCPT1 SENSE SUBROUTINE
D2
LABEND
D1
CORRECT TLBL FOUND
YES
NO
D2
INCREASE PROCEDURE LIST POINTER BY 8
D3
GET LABEL INFORMATION
SENSE
D5
SET CCW COMMAND CODE FOR SENSE COMMAND
NRF
E1
FORTRAN FILE
YES
NO
RA F3
CEPATH1
E2
RETURN TO CALLING ROUTINE
E3
RC=0
YES
NO
G4
E5
IO RC
PERFORM SENSE
CHKLP
F2
SVC 11 RETURN TO USER
YES
F3
ACCEPT UNRECOV I/O ERROR
NO
RA F3
DEPATH1
F4
RETURN TO CALLING ROUTINE
NO
F5
IS TAPE AT LOAD POINT
YES
G4
G4
FORTRAN FILE
NO
YES
G5
INCREMENT PROCEDURE LIST POINTER BY 2
H4
REPLACE LABELED TAPE AREA WITH I/O AREA
SENSEX
H5
RETURN TO CALLING ROUTINE
D2

Chart RC. $$BOCPT1: Subroutines for Open DTFCP and DTFDI Input Tape (Part 3 of 3)

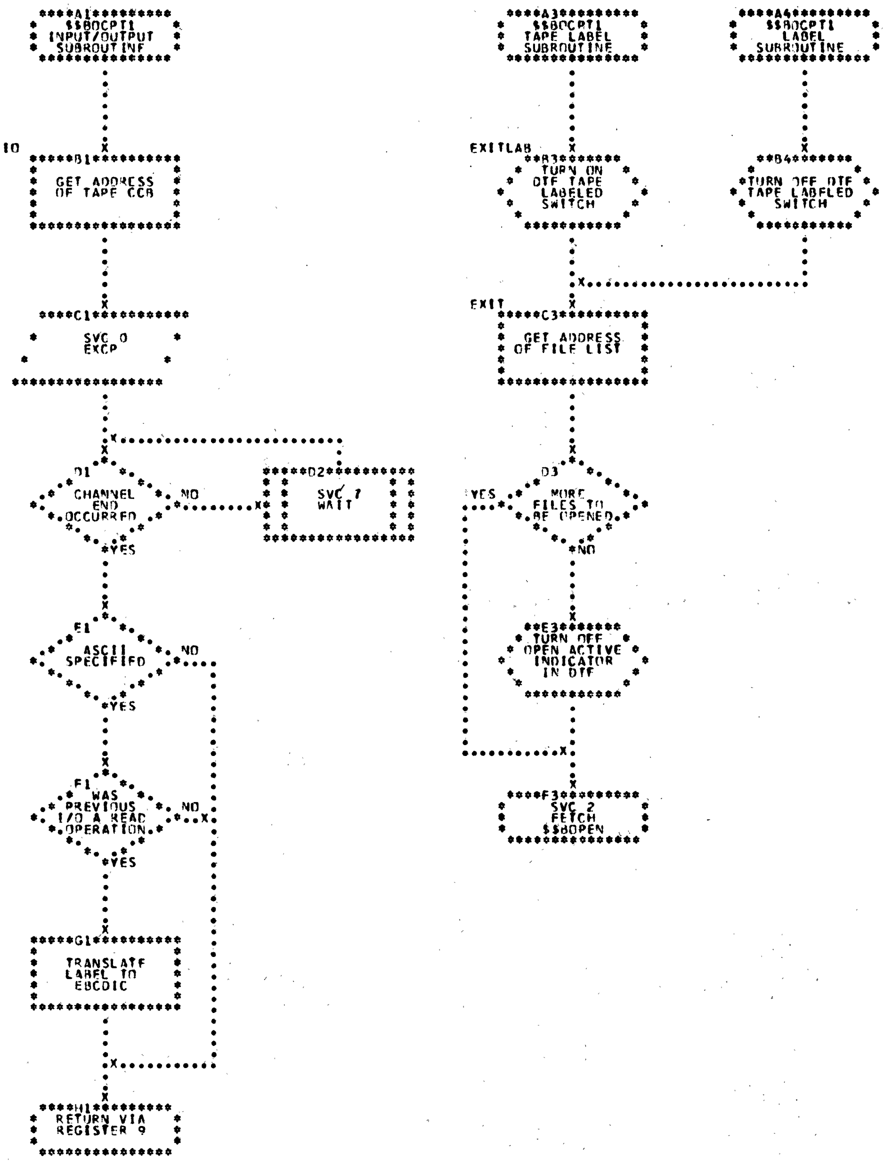


Chart RD. $$BOCPT2: Open DTFCP and DTFDI Output Tape (Part 1 of 3)

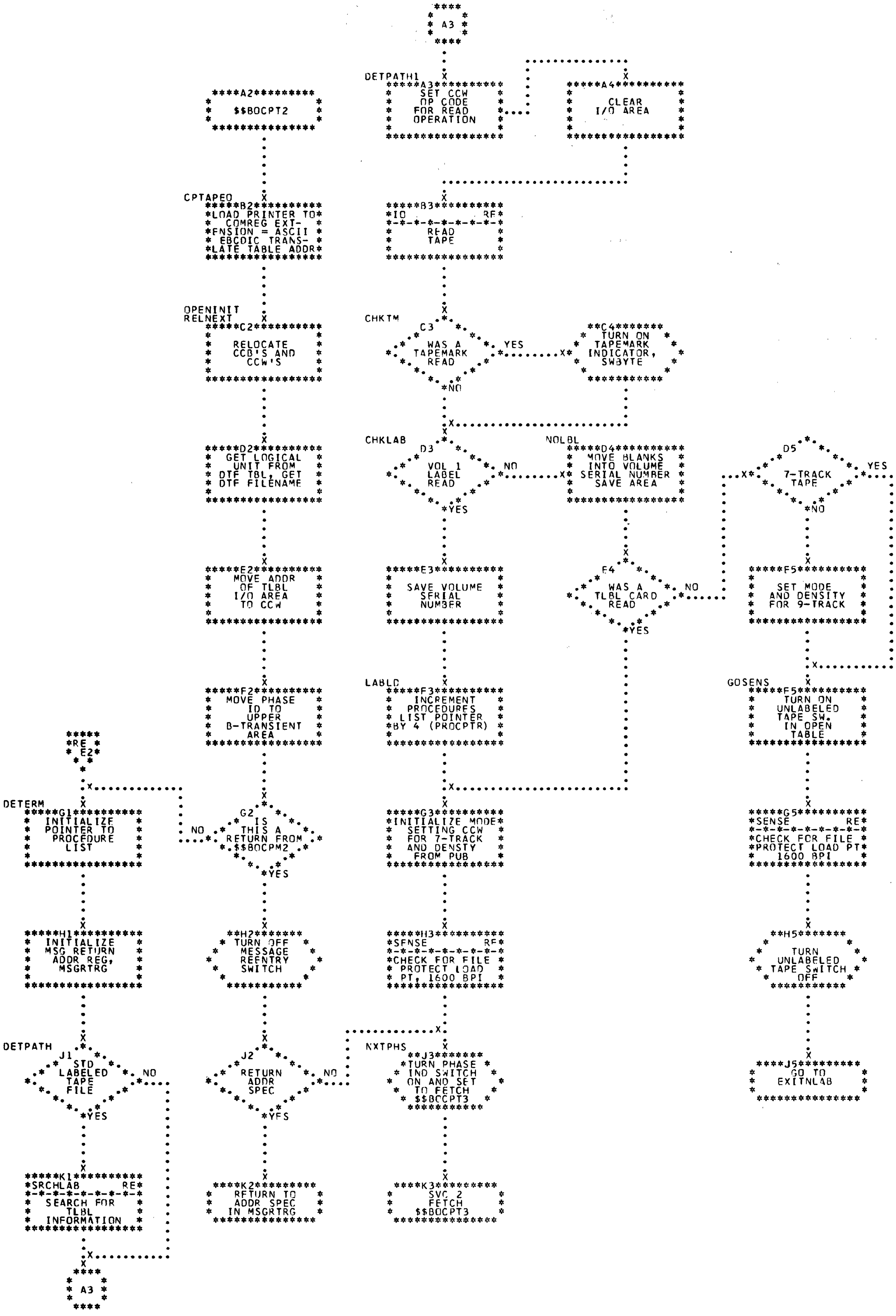

A3
DETPATH1
A3
SET CCW OP CODE FOR READ OPERATION
A4
CLEAR I/O AREA
A2
$$BOCPT2
CPTAPEO
B2
LOAD PRINTER TO COMREG EXTENSION = ASCII EBCDIC TRANSLATE TABLE ADDR
B3
IO RE
READ TAPE
OPENINIT
RELNEXT
C2
RELOCATE CCB'S AND CCW'S
CHKTM
C3
WAS A TAPEMARK READ
YES
NO
C4
TURN ON TAPEMARK INDICATOR, SWBYTE
D2
GET LOGICAL UNIT FROM DTF TBL, GET DTF FILENAME
CHKLAB
D3
VOL 1 LABEL READ
NO
YES
NOLBL
D4
MOVE BLANKS INTO VOLUME SERIAL NUMBER SAVE AREA
D5
7-TRACK TAPE
YES
NO
E2
MOVE ADDR OF TLBL I/O AREA TO CCW
E3
SAVE VOLUME SERIAL NUMBER
E4
WAS A TLBL CARD READ
NO
YES
F5
SET MODE AND DENSITY FOR 9-TRACK
F2
MOVE PHASE ID TO UPPER B-TRANSIENT AREA
LABLD
F3
INCREMENT PROCEDURES LIST POINTER BY 4 (PROCPTR)
GOSENS
F5
TURN ON UNLABELED TAPE SW. IN OPEN TABLE
RE
E2
DETERM
G1
INITIALIZE POINTER TO PROCEDURE LIST
G2
IS THIS A RETURN FROM $$BOCPM2
NO
YES
G3
INITIALIZE MODE SETTING CCW FOR 7-TRACK AND DENSTY FROM PUB
G5
SENSE RE
CHECK FOR FILE PROTECT LOAD PT 1600 BPI
H1
INITIALIZE MSG RETURN ADDR REG, MSGRTRG
H2
TURN OFF MESSAGE REENTRY SWITCH
H3
SENSE RE
CHECK FOR FILE PROTECT LOAD PT, 1600 BPI
H5
TURN UNLABELED TAPE SWITCH OFF
DETPATH
J1
STD LABELED TAPE FILE
NO
YES
J2
RETURN ADDR SPEC
NO
YES
NXTPHS
J3
TURN PHASE IND SWITCH ON AND SET TO FETCH $$BOCPT3
J5
GO TO EXITNLAB
K1
SRCHLAB RE
SEARCH FOR TLBL INFORMATION
K2
RETURN TO ADDR SPEC IN MSGRTRG
K3
SVC 2 FETCH $$BOCPT3
A3

Chart RE. $$BOCPT2: Subroutines for Open DTFCP and DTFDI Output Tape (Part 2 of 3)

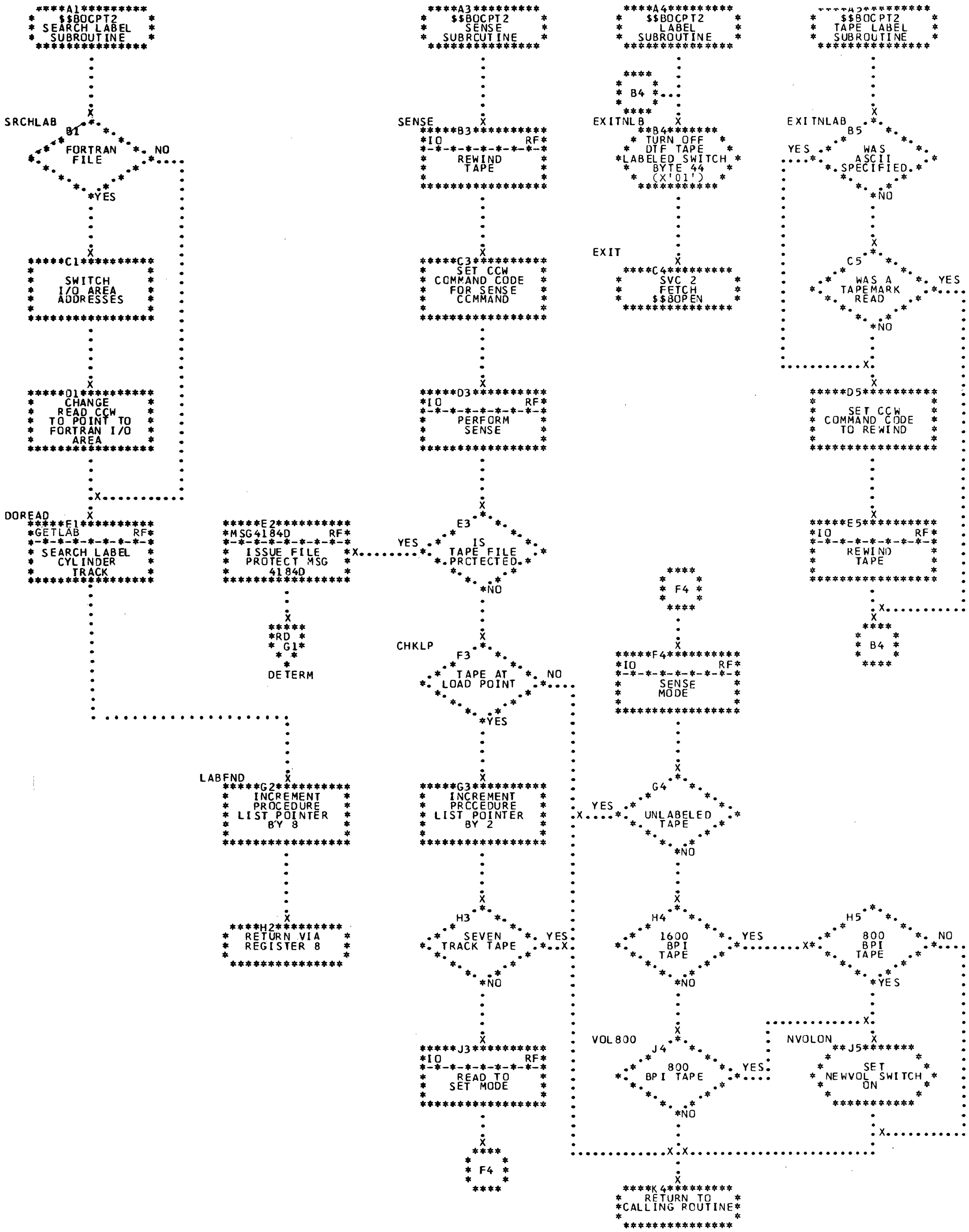


Chart RF. $$BOCPT2: Subroutines for Open DTFCP and DTFDI Output Tape (Part 3 of 3)

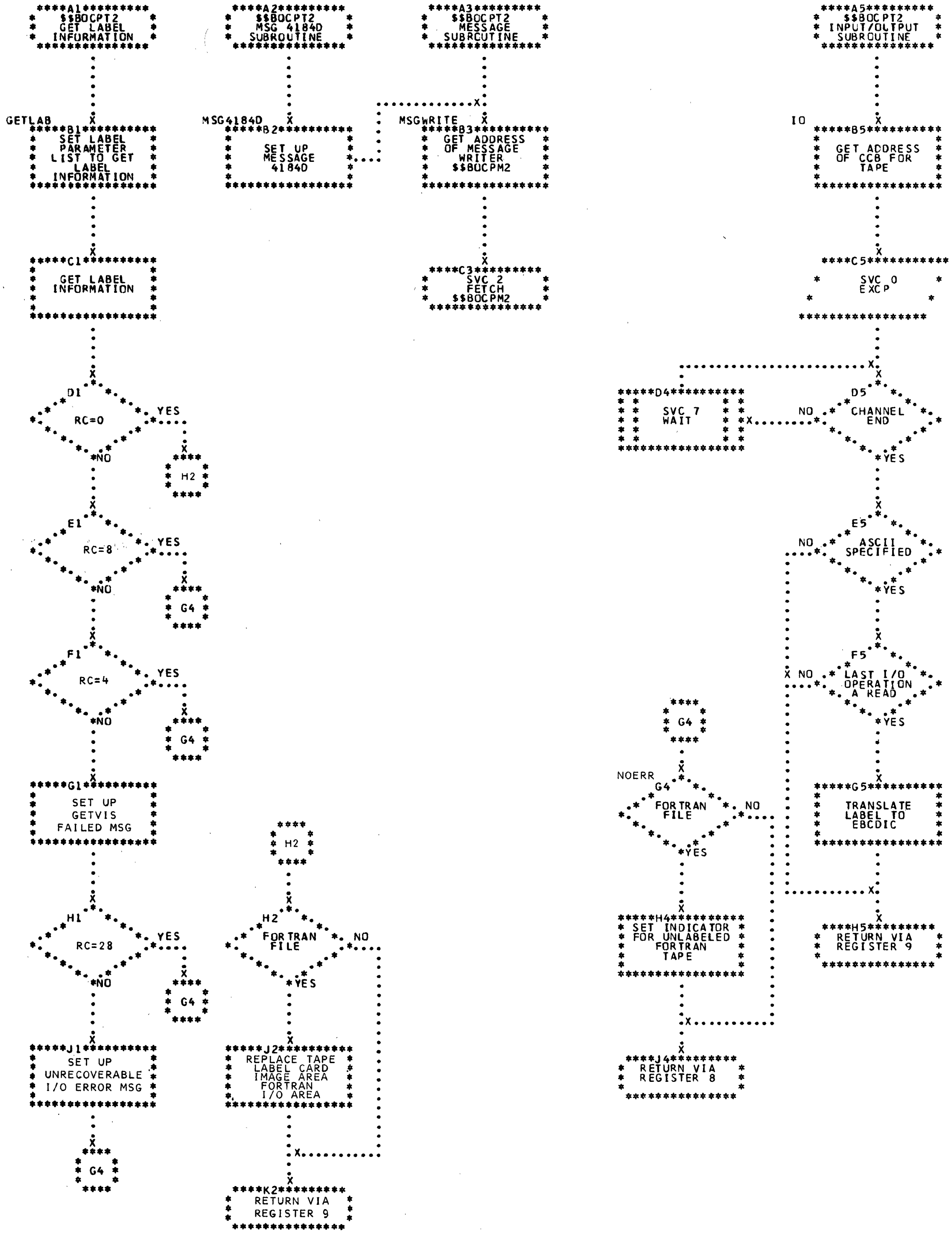


Chart BG. $$BOCPT3: Open DTFCP and DTFDI Labeled Output Tape (Part 1 of 3)

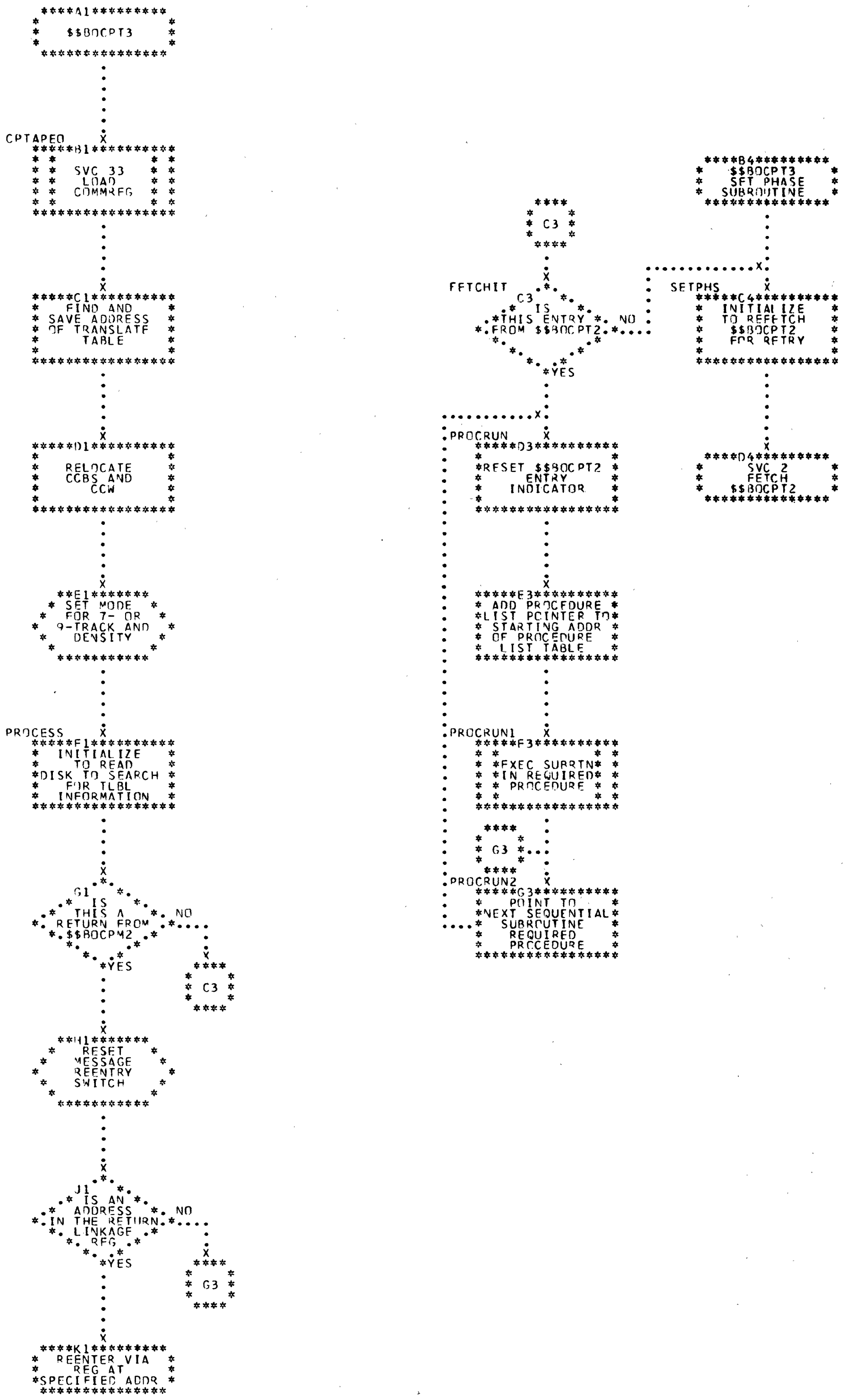


Chart RH. $$BOCPT3: Subroutines for Open DTFCP and DTFDI Labeled Output Tape (Part 2 of 3)

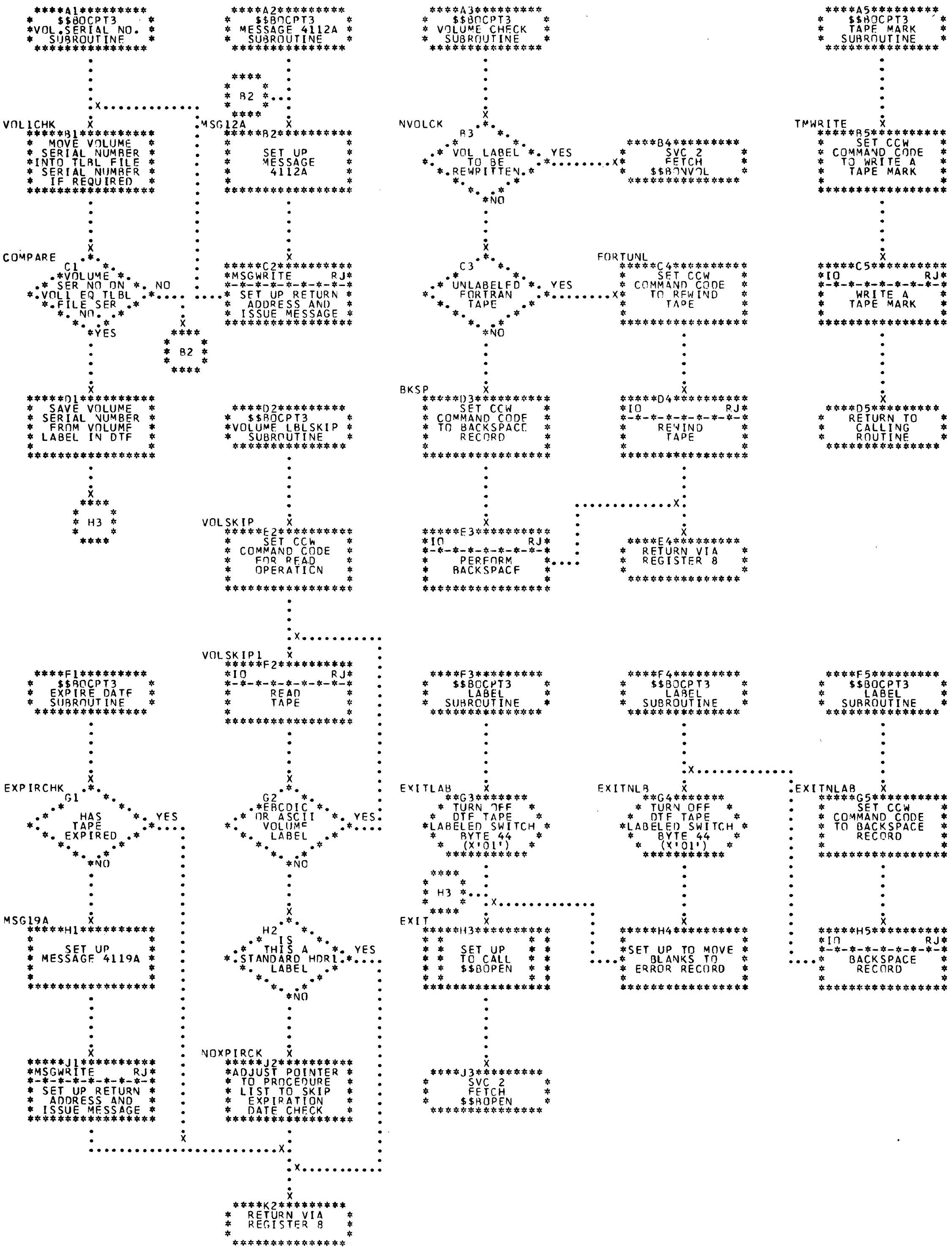

A1 $$BOCPT3 VOL.SERIAL NO. SUBROUTINE
VOL1CHK
B1 MOVE VOLUME SERIAL NUMBER INTO TLBL FILE SERIAL NUMBER IF REQUIRED
COMPARE
C1 VOLUME SER NO ON VOL1 EQ TLBL FILE SER NO.
NO
YES
B2
D1 SAVE VOLUME SERIAL NUMBER FROM VOLUME LABEL IN DTF
H3
A2 $$BOCPT3 MESSAGE 4112A SUBROUTINE
B2
MSG12A
B2 SET UP MESSAGE 4112A
C2 MSGWRITE RJ SET UP RETURN ADDRESS AND ISSUE MESSAGE
D2 $$BOCPT3 VOLUME LBLSKIP SUBROUTINE
VOLSKIP
E2 SET CCW COMMAND CODE FOR READ OPERATICN
VOLSKIP1
F2 IO RJ READ TAPE
G2 EBCDIC OR ASCII VOLUME LABEL
YES
NO
H2 IS THIS A STANDARD HDR1 LABEL
YES
NO
NOXPIRCK
J2 ADJUST POINTER TO PROCEDURE LIST TO SKIP EXPIRATION DATE CHECK
K2 RETURN VIA REGISTER 8
F1 $$BOCPT3 EXPIRE DATE SUBROUTINE
EXPIRCHK
G1 HAS TAPE EXPIRED
YES
NO
MSG19A
H1 SET UP MESSAGE 4119A
J1 MSGWRITE RJ SET UP RETURN ADDRESS AND ISSUE MESSAGE
A3 $$BOCPT3 VOLUME CHECK SUBROUTINE
NVOLCK
B3 VOL LABEL TO BE REWRITTEN
YES
NO
B4 SVC 2 FETCH $$BONVOL
C3 UNLABELED FORTRAN TAPE
YES
NO
FORTUNL
C4 SET CCW COMMAND CODE TO REWIND TAPE
D4 IO RJ REWIND TAPE
BKSP
D3 SET CCW COMMAND CODE TO BACKSPACE RECORD
E3 IO RJ PERFORM BACKSPACE
E4 RETURN VIA REGISTER 8
F3 $$BOCPT3 LABEL SUBROUTINE
EXITLAB
G3 TURN OFF DTF TAPE LABELED SWITCH BYTE 44 (X'01')
H3
EXIT
H3 SET UP TO CALL $$BOPEN
J3 SVC 2 FETCH $$BOPEN
F4 $$BOCPT3 LABEL SUBROUTINE
EXITNLB
G4 TURN OFF DTF TAPE LABELED SWITCH BYTE 44 (X'01')
H4 SET UP TO MOVE BLANKS TO ERROR RECORD
A5 $$BOCPT3 TAPE MARK SUBROUTINE
TMWRITE
B5 SET CCW COMMAND CODE TO WRITE A TAPE MARK
C5 IO RJ WRITE A TAPE MARK
D5 RETURN TO CALLING ROUTINE
F5 $$BOCPT3 LABEL SUBROUTINE
EXITNLAB
G5 SET CCW COMMAND CODE TO BACKSPACE RECORD
H5 IO RJ BACKSPACE RECORD

Chart RJ. $$BOCPT3: Subroutines for Open DTFCP and DTFDI Labeled Output Tape (Part 3 of 3)

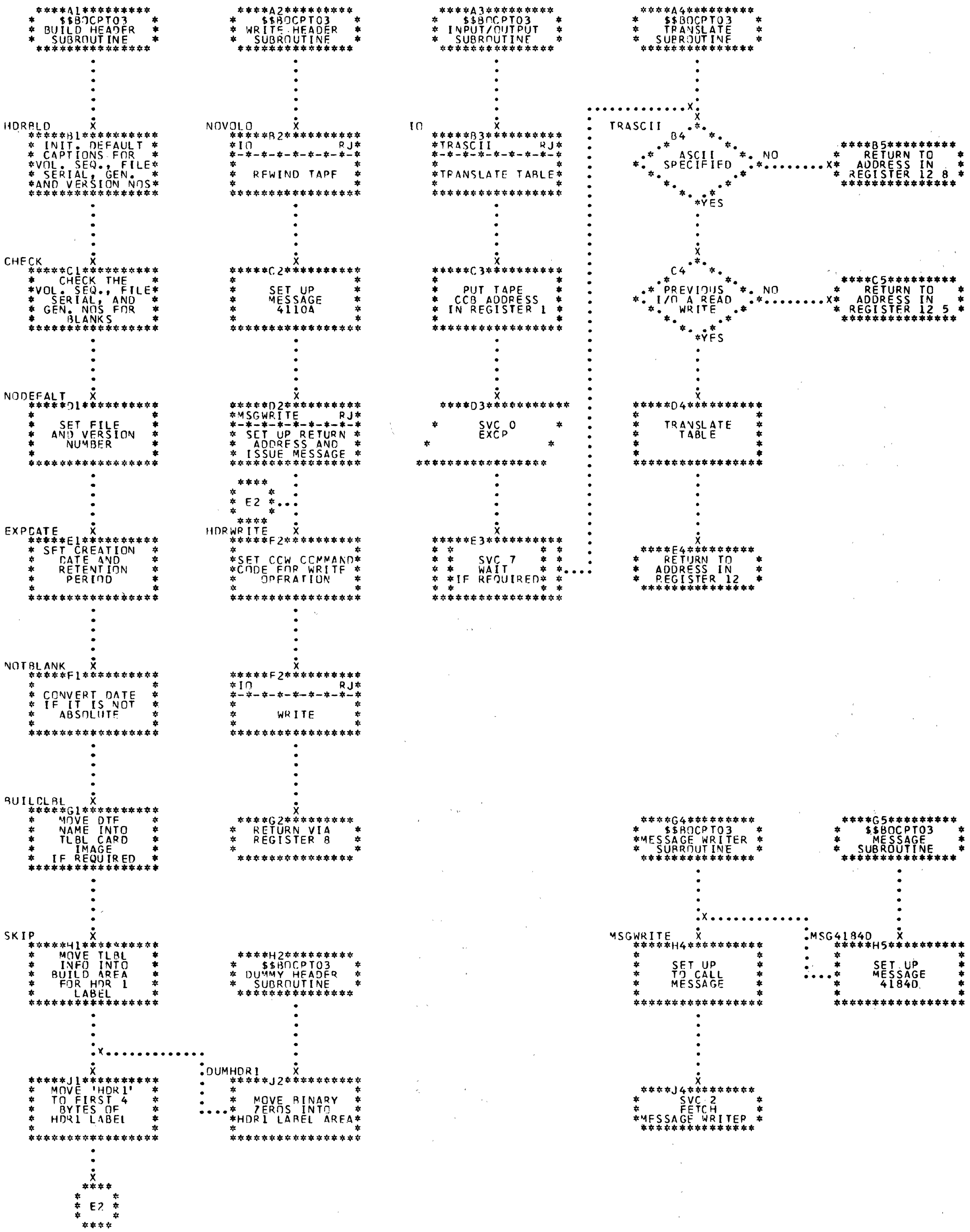

A1 $$BOCPT03 BUILD HEADER SUBROUTINE
HDRBLD B1 INIT. DEFAULT CAPTIONS FOR VOL. SEQ., FILE SERIAL, GEN. AND VERSION NOS
CHECK C1 CHECK THE VOL. SEQ., FILE SERIAL, AND GEN. NOS FOR BLANKS
NODEFALT D1 SET FILE AND VERSION NUMBER
EXPDATE E1 SET CREATION DATE AND RETENTION PERIOD
NOTBLANK F1 CONVERT DATE IF IT IS NOT ABSOLUTE
BUILDLBL G1 MOVE DTF NAME INTO TLBL CARD IMAGE IF REQUIRED
SKIP H1 MOVE TLBL INFO INTO BUILD AREA FOR HDR 1 LABEL
J1 MOVE 'HDR1' TO FIRST 4 BYTES OF HDR1 LABEL
E2
A2 $$BOCPT03 WRITE HEADER SUBROUTINE
NOVOLO B2 IO RJ REWIND TAPE
C2 SET UP MESSAGE 4110A
D2 MSGWRITE RJ SET UP RETURN ADDRESS AND ISSUE MESSAGE
E2
HDRWRITE F2 SET CCW COMMAND CODE FOR WRITE OPERATION
F2 IO RJ WRITE
G2 RETURN VIA REGISTER 8
H2 $$BOCPT03 DUMMY HEADER SUBROUTINE
DUMHDR1 J2 MOVE BINARY ZEROS INTO HDR1 LABEL AREA
A3 $$BOCPT03 INPUT/OUTPUT SUBROUTINE
IO B3 TRASCII RJ TRANSLATE TABLE
C3 PUT TAPE CCB ADDRESS IN REGISTER 1
D3 SVC 0 EXCP
E3 SVC 7 WAIT IF REQUIRED
A4 $$BOCPT03 TRANSLATE SUBROUTINE
TRASCII B4 ASCII SPECIFIED
NO
YES
C4 PREVIOUS I/O A READ WRITE
NO
YES
D4 TRANSLATE TABLE
E4 RETURN TO ADDRESS IN REGISTER 12
B5 RETURN TO ADDRESS IN REGISTER 12 8
C5 RETURN TO ADDRESS IN REGISTER 12 5
G4 $$BOCPT03 MESSAGE WRITER SUBROUTINE
MSGWRITE H4 SET UP TO CALL MESSAGE
J4 SVC 2 FETCH MESSAGE WRITER
G5 $$BOCPT03 MESSAGE SUBROUTINE
MSG4184D H5 SET UP MESSAGE 4184D

Chart RK. $$BOCPT4: Oper DTFCP and DTFDI Labeled Input Tape

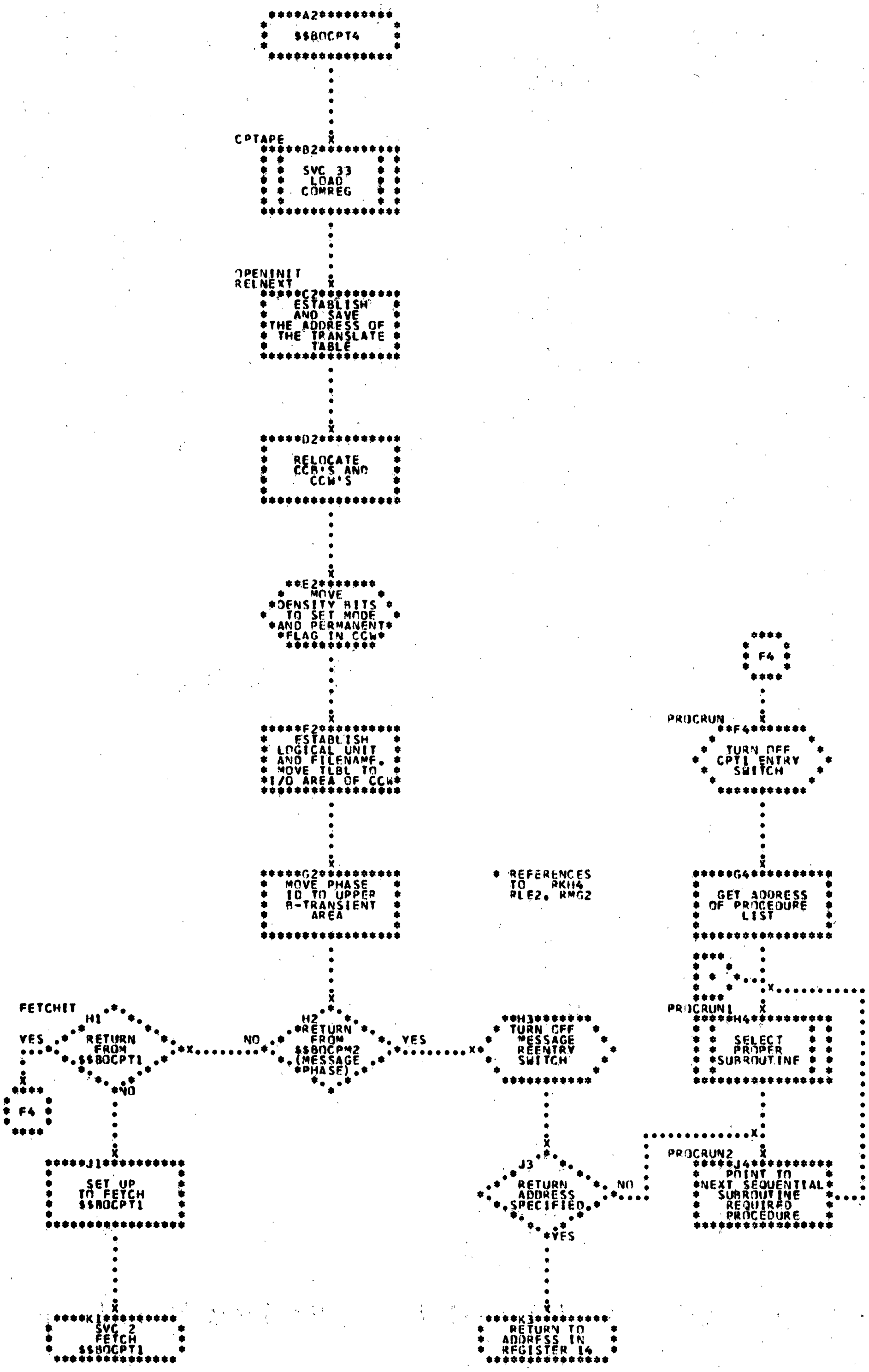


Chart RI. $$BOCPT4: Subroutines for Open DTFCP and DTFDI Labeled Input Tape (Part 1 of 2)

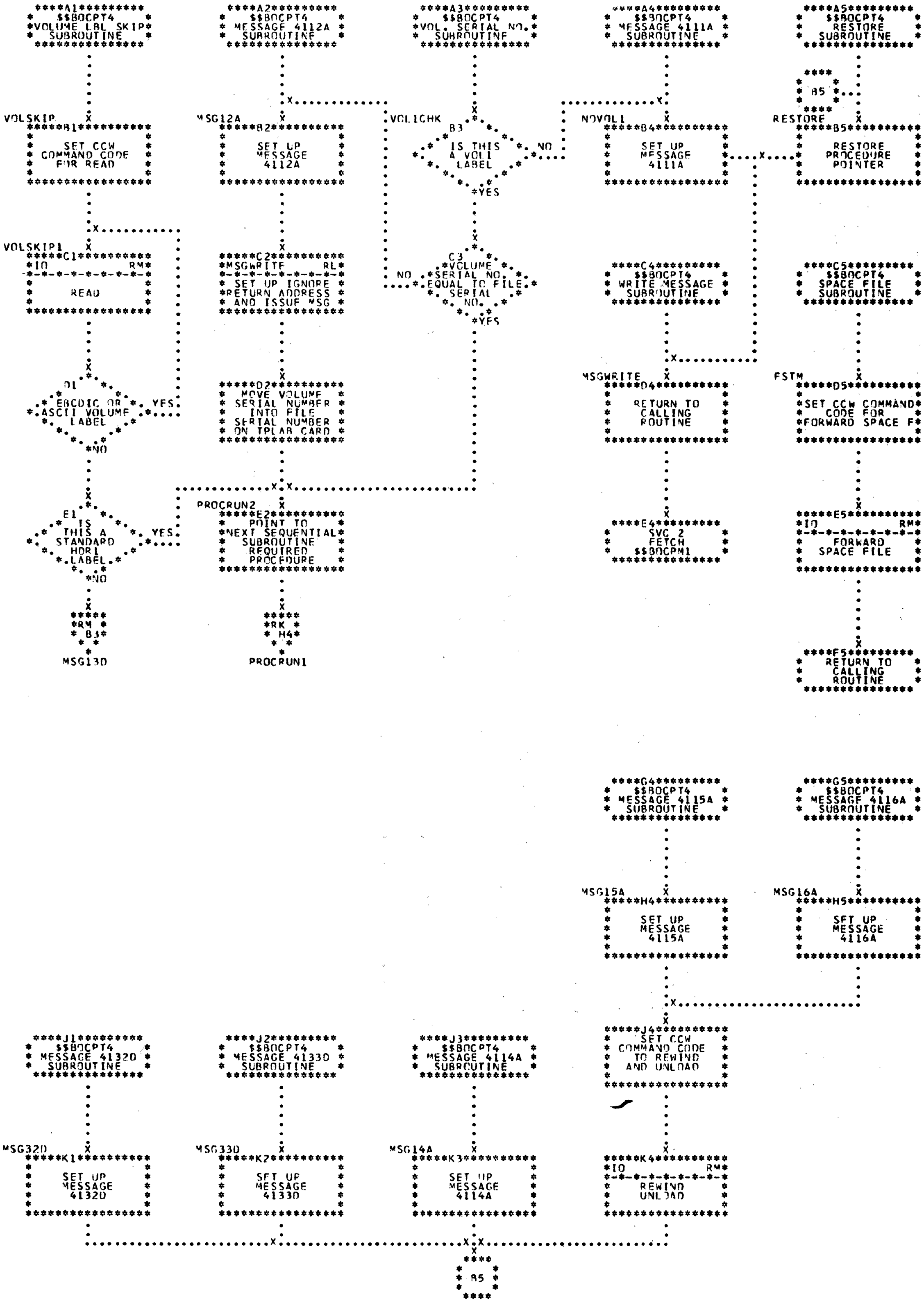

A1 $$BOCPT4 VOLUME LBL SKIP SUBROUTINE
A2 $$BOCPT4 MESSAGE 4112A SUBROUTINE
A3 $$BOCPT4 VOL. SERIAL NO. SUBROUTINE
A4 $$BOCPT4 MESSAGE 4111A SUBROUTINE
A5 $$BOCPT4 RESTORE SUBROUTINE
B5
VOLSKIP
B1 SET CCW COMMAND CODE FOR READ
MSG12A
B2 SET UP MESSAGE 4112A
VOLICHK
B3 IS THIS A VOL1 LABEL
NO
YES
NOVOL1
B4 SET UP MESSAGE 4111A
RESTORE
B5 RESTORE PROCEDURE POINTER
VOLSKIP1
C1 IO RM
READ
C2 MSGWRITE RL
SET UP IGNORE RETURN ADDRESS AND ISSUE MSG
C3 VOLUME SERIAL NO. EQUAL TO FILE SERIAL NO.
NO
YES
C4 $$BOCPT4 WRITE MESSAGE SUBROUTINE
C5 $$BOCPT4 SPACE FILE SUBROUTINE
D1 EBCDIC OR ASCII VOLUME LABEL
YES
NO
D2 MOVE VOLUME SERIAL NUMBER INTO FILE SERIAL NUMBER ON TPLAB CARD
MSGWRITE
D4 RETURN TO CALLING ROUTINE
FSTM
D5 SET CCW COMMAND CODE FOR FORWARD SPACE F
E1 IS THIS A STANDARD HDR1 LABEL
YES
NO
PROCRUN2
E2 POINT TO NEXT SEQUENTIAL SUBROUTINE REQUIRED PROCEDURE
E4 SVC 2 FETCH $$BOCPM1
E5 IO RM
FORWARD SPACE FILE
RM B3
MSG13D
RK H4
PROCRUN1
F5 RETURN TO CALLING ROUTINE
G4 $$BOCPT4 MESSAGE 4115A SUBROUTINE
G5 $$BOCPT4 MESSAGE 4116A SUBROUTINE
MSG15A
H4 SET UP MESSAGE 4115A
MSG16A
H5 SET UP MESSAGE 4116A
J1 $$BOCPT4 MESSAGE 4132D SUBROUTINE
J2 $$BOCPT4 MESSAGE 4133D SUBROUTINE
J3 $$BOCPT4 MESSAGE 4114A SUBROUTINE
J4 SET CCW COMMAND CODE TO REWIND AND UNLOAD
MSG32D
K1 SET UP MESSAGE 4132D
MSG33D
K2 SET UP MESSAGE 4133D
MSG14A
K3 SET UP MESSAGE 4114A
K4 IO RM
REWIND UNLOAD
B5

Chart RM. $$BOCPT4: Subroutines for Open DTFCP and DTFDI Tape Labeled Input (Part 2 of 2)

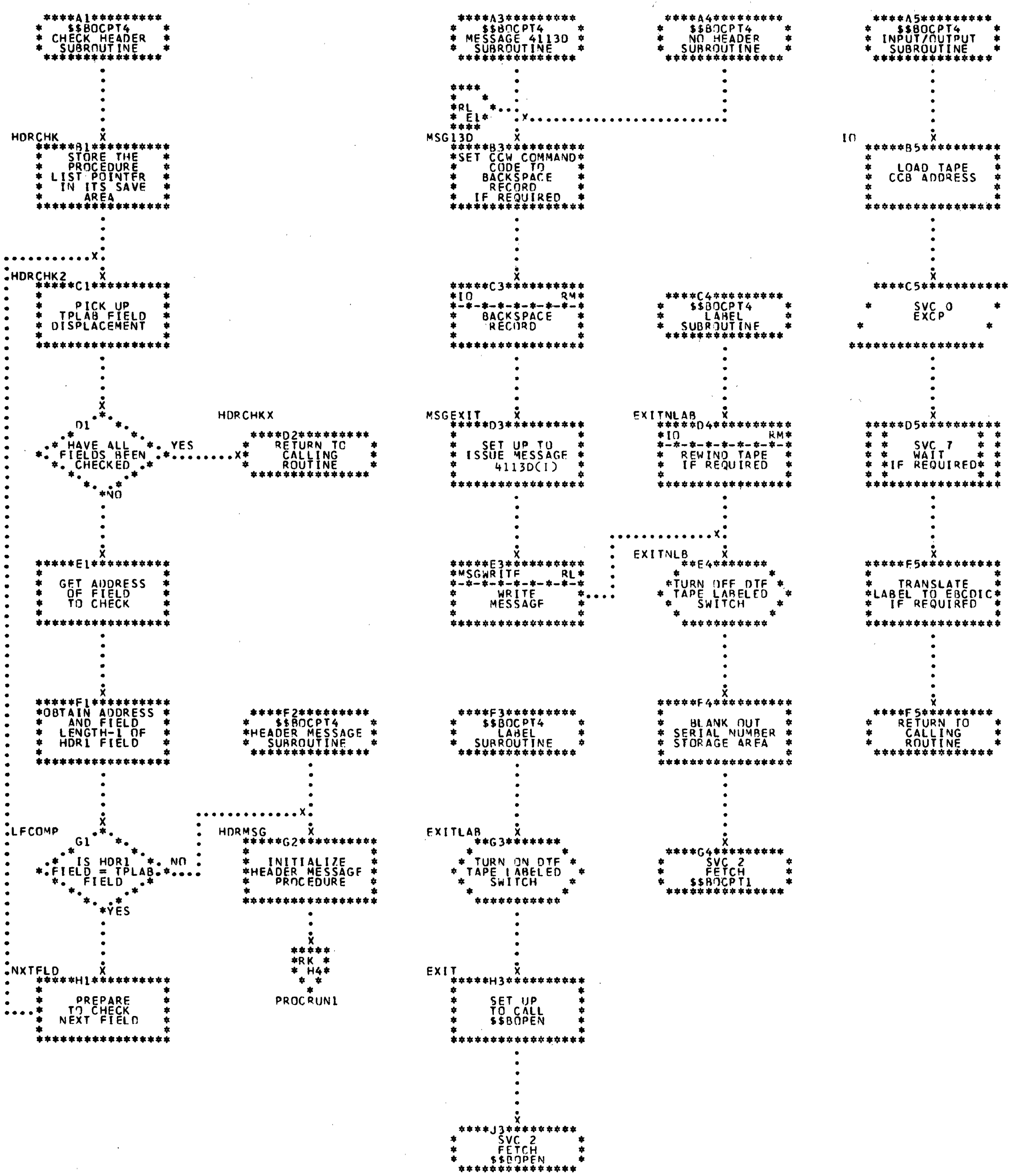

A1 $$BOCPT4 CHECK HEADER SUBROUTINE
HDRCHK
B1 STORE THE PROCEDURE LIST POINTER IN ITS SAVE AREA
HDRCHK2
C1 PICK UP TPLAB FIELD DISPLACEMENT
D1 HAVE ALL FIELDS BEEN CHECKED
YES
NO
HDRCHKX
D2 RETURN TO CALLING ROUTINE
E1 GET ADDRESS OF FIELD TO CHECK
F1 OBTAIN ADDRESS AND FIELD LENGTH-1 OF HDR1 FIELD
LFCOMP
G1 IS HDR1 FIELD = TPLAB FIELD
NO
YES
NXTFLD
H1 PREPARE TO CHECK NEXT FIELD
F2 $$BOCPT4 HEADER MESSAGE SUBROUTINE
HDRMSG
G2 INITIALIZE HEADER MESSAGE PROCEDURE
RK H4
PROCRUN1
A3 $$BOCPT4 MESSAGE 4113D SUBROUTINE
RL E1
MSG13D
B3 SET CCW COMMAND CODE TO BACKSPACE RECORD IF REQUIRED
C3 IO RM
BACKSPACE RECORD
MSGEXIT
D3 SET UP TO ISSUE MESSAGE 4113D(I)
E3 MSGWRITE RL
WRITE MESSAGE
F3 $$BOCPT4 LABEL SUBROUTINE
EXITLAB
G3 TURN ON DTF TAPE LABELED SWITCH
EXIT
H3 SET UP TO CALL $$BOPEN
J3 SVC 2 FETCH $$BOPEN
A4 $$BOCPT4 NO HEADER SUBROUTINE
C4 $$BOCPT4 LABEL SUBROUTINE
EXITNLAB
D4 IO RM
REWIND TAPE IF REQUIRED
EXITNLB
E4 TURN OFF DTF TAPE LABELED SWITCH
F4 BLANK OUT SERIAL NUMBER STORAGE AREA
G4 SVC 2 FETCH $$BOCPT1
A5 $$BOCPT4 INPUT/OUTPUT SUBROUTINE
IO
B5 LOAD TAPE CCB ADDRESS
C5 SVC 0 EXCP
D5 SVC 7 WAIT IF REQUIRED
E5 TRANSLATE LABEL TO EBCDIC IF REQUIRED
F5 RETURN TO CALLING ROUTINE

Chart SA. $$BCCPT1: Close DTFCP and DTFDI Tape Files (Part 1 of 2)

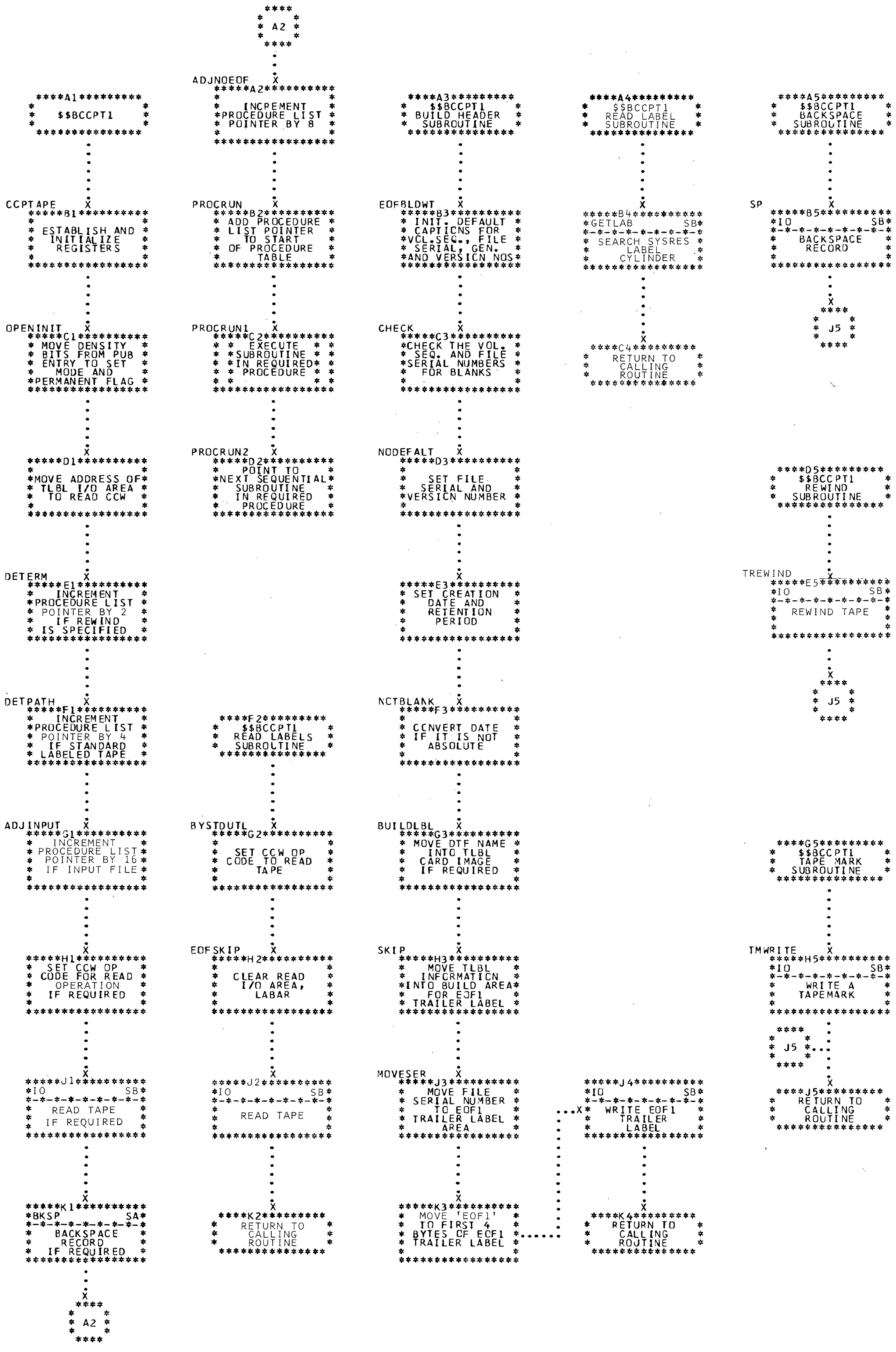

A2
ADJNOEOF
A1 $$BCCPT1
A2 INCREMENT PROCEDURE LIST POINTER BY 8
A3 $$BCCPT1 BUILD HEADER SUBROUTINE
A4 $$BCCPT1 READ LABEL SUBROUTINE
A5 $$BCCPT1 BACKSPACE SUBROUTINE
CCPTAPE
B1 ESTABLISH AND INITIALIZE REGISTERS
PROCRUN
B2 ADD PROCEDURE LIST POINTER TO START OF PROCEDURE TABLE
EOFBLDWT
B3 INIT. DEFAULT CAPTIONS FOR VOL.SEQ., FILE SERIAL, GEN. AND VERSION NOS
B4 GETLAB SB SEARCH SYSRES LABEL CYLINDER
SP
B5 IO SB BACKSPACE RECORD
J5
OPENINIT
C1 MOVE DENSITY BITS FROM PUB ENTRY TO SET MODE AND PERMANENT FLAG
PROCRUN1
C2 EXECUTE SUBROUTINE IN REQUIRED PROCEDURE
CHECK
C3 CHECK THE VOL. SEQ. AND FILE SERIAL NUMBERS FOR BLANKS
C4 RETURN TO CALLING ROUTINE
D1 MOVE ADDRESS OF TLBL I/O AREA TO READ CCW
PROCRUN2
D2 POINT TO NEXT SEQUENTIAL SUBROUTINE IN REQUIRED PROCEDURE
NODEFALT
D3 SET FILE SERIAL AND VERSION NUMBER
D5 $$BCCPT1 REWIND SUBROUTINE
DETERM
E1 INCREMENT PROCEDURE LIST POINTER BY 2 IF REWIND IS SPECIFIED
E3 SET CREATION DATE AND RETENTION PERIOD
TREWIND
E5 IO SB REWIND TAPE
J5
DETPATH
F1 INCREMENT PROCEDURE LIST POINTER BY 4 IF STANDARD LABELED TAPE
F2 $$BCCPT1 READ LABELS SUBROUTINE
NCTBLANK
F3 CONVERT DATE IF IT IS NOT ABSOLUTE
ADJINPUT
G1 INCREMENT PROCEDURE LIST POINTER BY 16 IF INPUT FILE
BYSTDUTL
G2 SET CCW OP CODE TO READ TAPE
BUILDLBL
G3 MOVE DTF NAME INTO TLBL CARD IMAGE IF REQUIRED
G5 $$BCCPT1 TAPE MARK SUBROUTINE
H1 SET CCW OP CODE FOR READ OPERATION IF REQUIRED
EOFSKIP
H2 CLEAR READ I/O AREA, LABAR
SKIP
H3 MOVE TLBL INFORMATION INTO BUILD AREA FOR EOF1 TRAILER LABEL
TMWRITE
H5 IO SB WRITE A TAPEMARK
J5
J1 IO SB READ TAPE IF REQUIRED
J2 IO SB READ TAPE
MOVESER
J3 MOVE FILE SERIAL NUMBER TO EOF1 TRAILER LABEL AREA
J4 IO SB WRITE EOF1 TRAILER LABEL
J5 RETURN TO CALLING ROUTINE
K1 BKSP SA BACKSPACE RECORD IF REQUIRED
K2 RETURN TO CALLING ROUTINE
K3 MOVE 'EOF1' TO FIRST 4 BYTES OF EOF1 TRAILER LABEL
K4 RETURN TO CALLING ROUTINE
A2

Chart SE. $$BCCPT1: Close DTFCP and DTFDI Tape Files (Part 2 of 2)

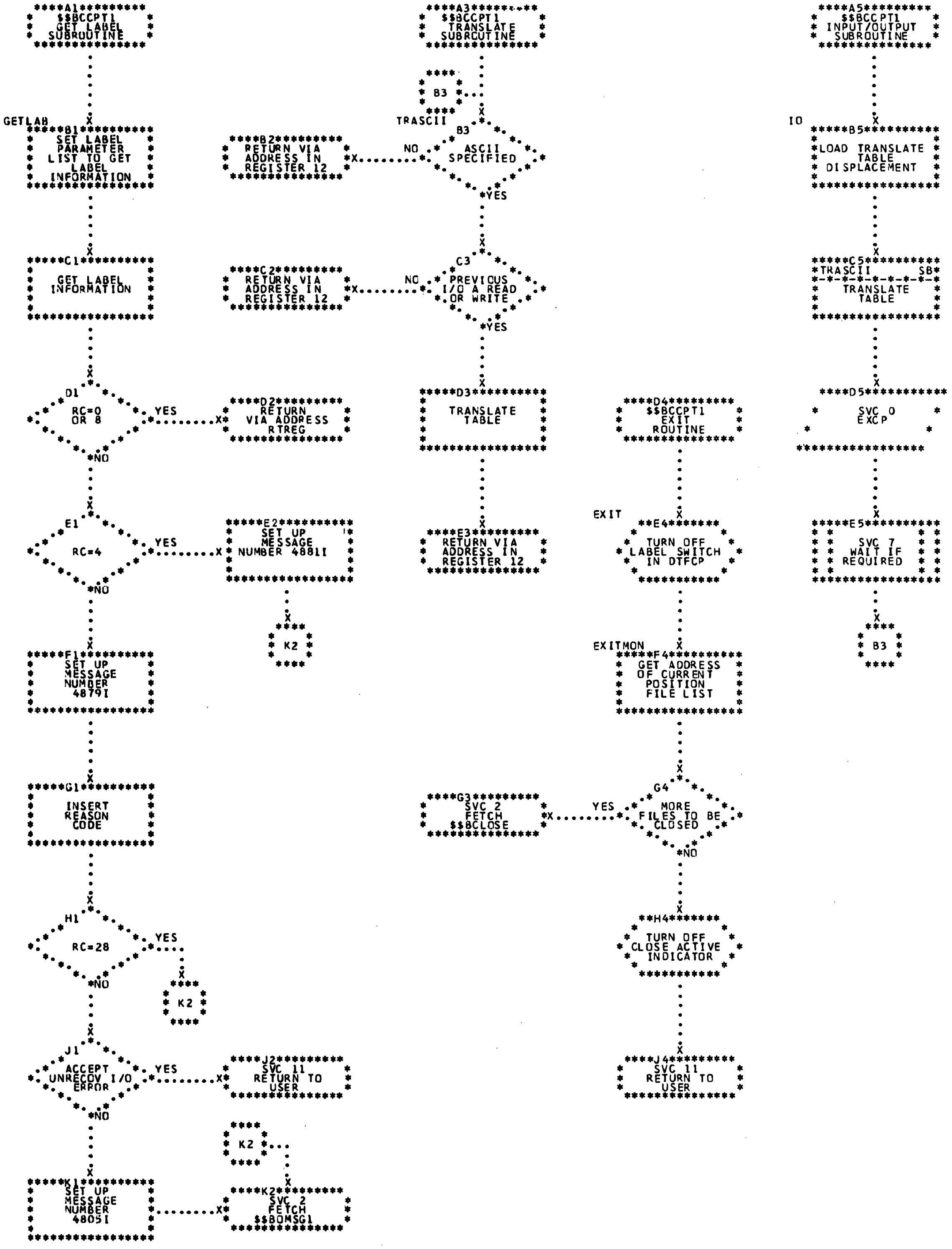

A1 $$BCCPT1 GET LABEL SUBROUTINE
GETLAB
B1 SET LABEL PARAMETER LIST TO GET LABEL INFORMATION
C1 GET LABEL INFORMATION
D1 RC=0 OR 8
YES
NO
D2 RETURN VIA ADDRESS RTREG
E1 RC=4
YES
NO
E2 SET UP MESSAGE NUMBER 4881I
K2
F1 SET UP MESSAGE NUMBER 4879I
G1 INSERT REASON CODE
H1 RC=28
YES
NO
K2
J1 ACCEPT UNRECOV I/O ERROR
YES
NO
J2 SVC 11 RETURN TO USER
K2
K1 SET UP MESSAGE NUMBER 4805I
K2 SVC 2 FETCH $$BOMSG1
A3 $$BCCPT1 TRANSLATE SUBRCUTINE
B3
TRASCII
B3 ASCII SPECIFIED
NO
YES
B2 RETURN VIA ADDRESS IN REGISTER 12
C3 PREVIOUS I/O A READ OR WRITE
NO
YES
C2 RETURN VIA ADDRESS IN REGISTER 12
D3 TRANSLATE TABLE
E3 RETURN VIA ADDRESS IN REGISTER 12
G3 SVC 2 FETCH $$BCLOSE
D4 $$BCCPT1 EXIT ROUTINE
EXIT
E4 TURN OFF LABEL SWITCH IN DTFCP
EXITMON
F4 GET ADDRESS OF CURRENT POSITION FILE LIST
G4 MORE FILES TO BE CLOSED
YES
NO
H4 TURN OFF CLOSE ACTIVE INDICATOR
J4 SVC 11 RETURN TO USER
A5 $$BCCPT1 INPUT/OUTPUT SUBROUTINE
IO
B5 LOAD TRANSLATE TABLE DISPLACEMENT
C5 TRASCII SB
TRANSLATE TABLE
D5 SVC 0 EXCP
E5 SVC 7 WAIT IF REQUIRED
B3

Chart SC. $$BCICSF: Punch File Close (Part 1 of 2)

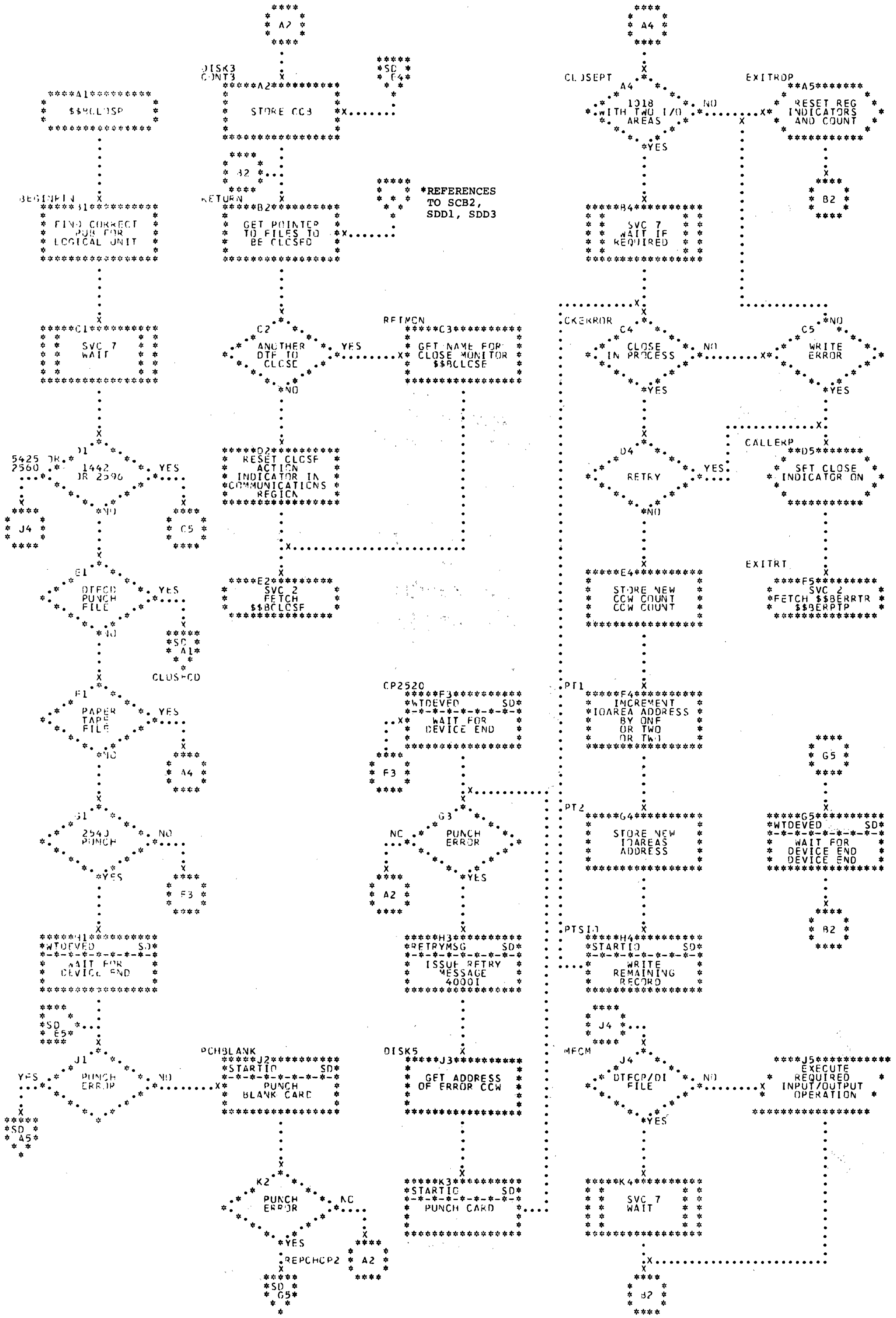


Chart SD. $$BCLOSP: Punch File Close (Part 2 of 2)

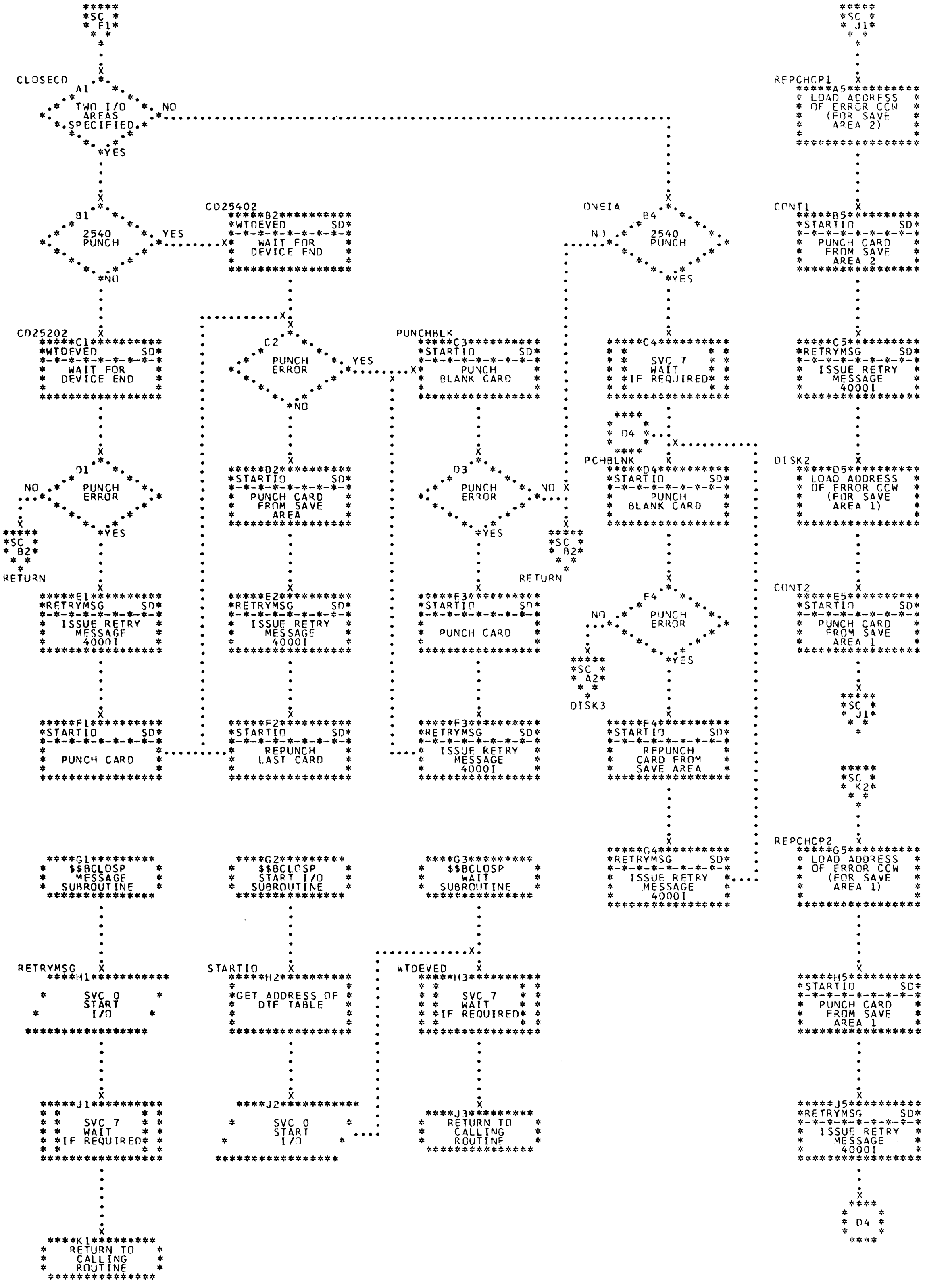


Chart SE. $$BOCPM1 and $$BOCPM2: DTFCP/DTFDI Message Writers

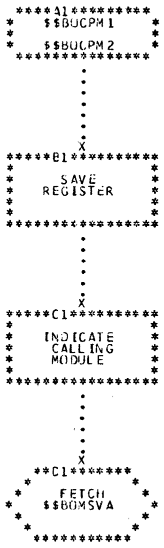
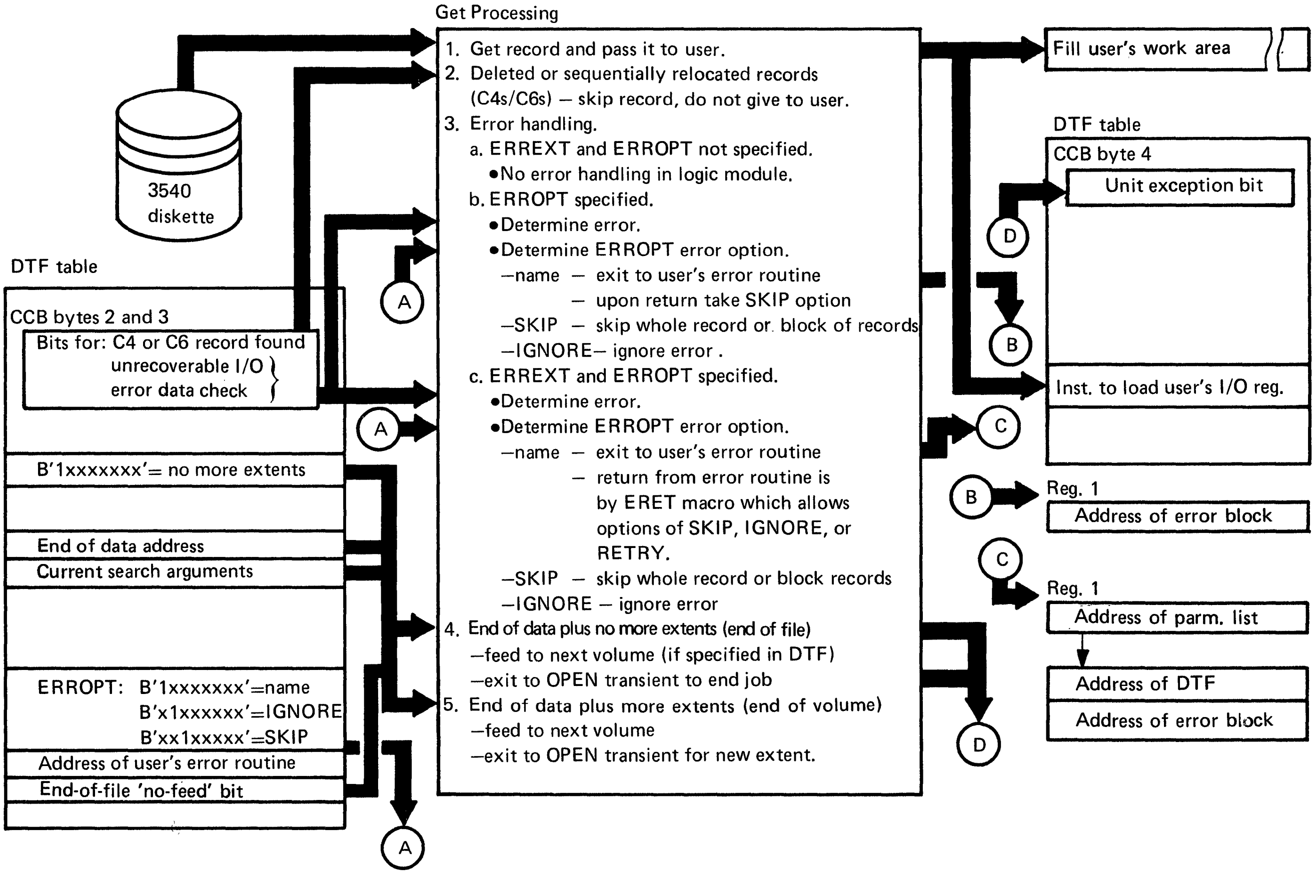


CHART SE

Chart UA. DUMCDFI: Get Routine



1. Get record and pass it to user.
2. Deleted or sequentially relocated records (C4s/C6s) – skip record, do not give to user.
3. Error handling.
   a. ERREXT and ERROPT not specified.
      - •No error handling in logic module.
   b. ERROPT specified.
      - •Determine error.
      - •Determine ERROPT error option.
        - –name – exit to user's error routine
          – upon return take SKIP option
        - –SKIP – skip whole record or block of records
        - –IGNORE– ignore error .
   c. ERREXT and ERROPT specified.
      - •Determine error.
      - •Determine ERROPT error option.
        - –name – exit to user's error routine
          – return from error routine is by ERET macro which allows options of SKIP, IGNORE, or RETRY.
        - –SKIP – skip whole record or block records
        - –IGNORE – ignore error
4. End of data plus no more extents (end of file)
   - –feed to next volume (if specified in DTF)
   - –exit to OPEN transient to end job
5. End of data plus more extents (end of volume)
   - –feed to next volume
   - –exit to OPEN transient for new extent.

Chart UB. DUMODFO: Put and Close Routines

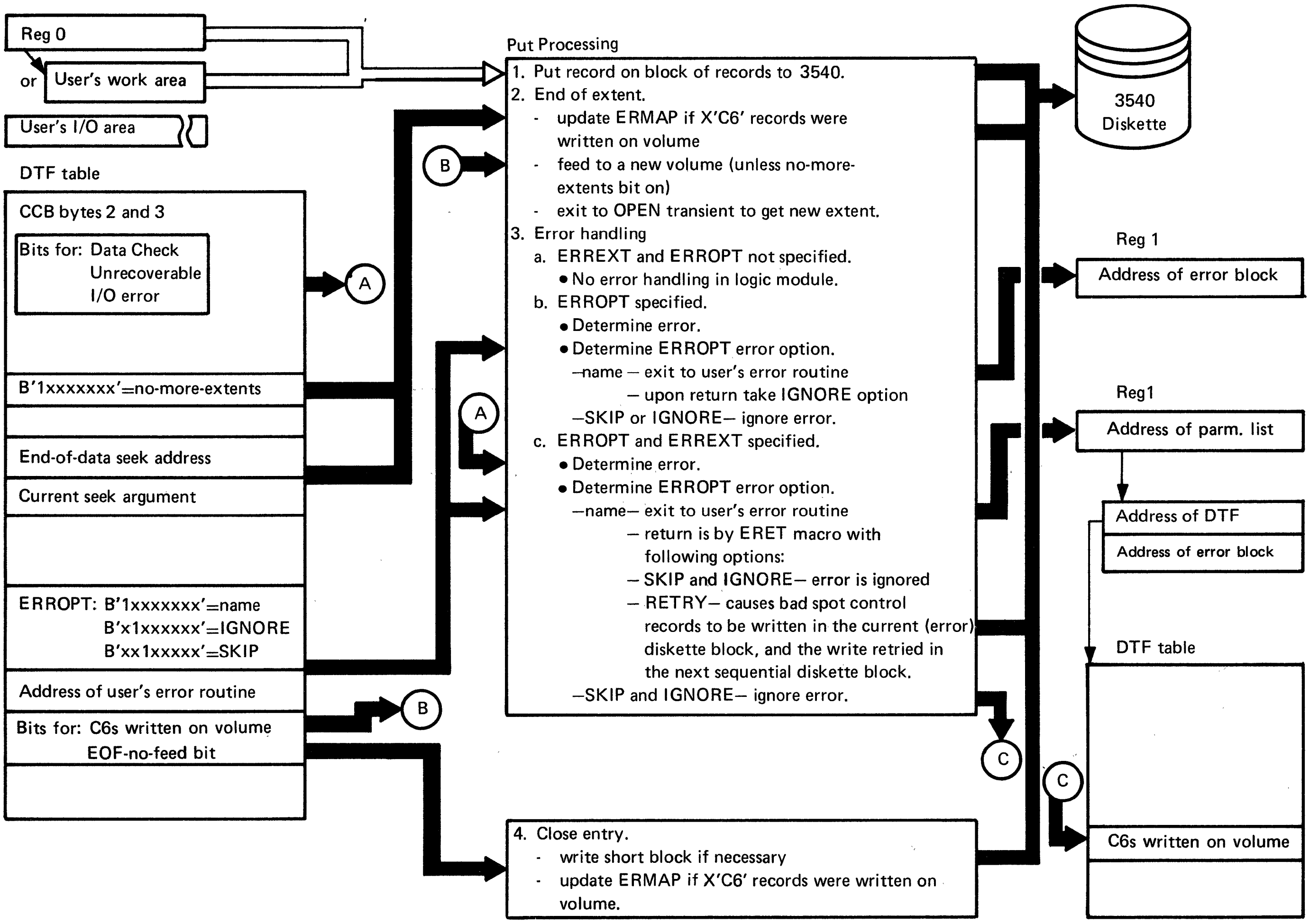

Reg 0
or
User's work area
User's I/O area
DTF table
CCB bytes 2 and 3
Bits for: Data Check
Unrecoverable
I/O error
B'1xxxxxxx'=no-more-extents
End-of-data seek address
Current seek argument
ERROPT: B'1xxxxxxx'=name
B'x1xxxxxx'=IGNORE
B'xx1xxxxx'=SKIP
Address of user's error routine
Bits for: C6s written on volume
EOF-no-feed bit
A
B
Put Processing
1. Put record on block of records to 3540.
2. End of extent.
- update ERMAP if X'C6' records were written on volume
- feed to a new volume (unless no-more-extents bit on)
- exit to OPEN transient to get new extent.
3. Error handling
a. ERREXT and ERROPT not specified.
• No error handling in logic module.
b. ERROPT specified.
• Determine error.
• Determine ERROPT error option.
–name – exit to user's error routine
– upon return take IGNORE option
–SKIP or IGNORE– ignore error.
c. ERROPT and ERREXT specified.
• Determine error.
• Determine ERROPT error option.
–name– exit to user's error routine
– return is by ERET macro with following options:
– SKIP and IGNORE– error is ignored
– RETRY– causes bad spot control records to be written in the current (error) diskette block, and the write retried in the next sequential diskette block.
–SKIP and IGNORE– ignore error.
4. Close entry.
- write short block if necessary
- update ERMAP if X'C6' records were written on volume.
C
3540
Diskette
Reg 1
Address of error block
Reg1
Address of parm. list
Address of DTF
Address of error block
DTF table
C6s written on volume

Chart UD. Index to 3540 OPEN Phases in Transition Sequence

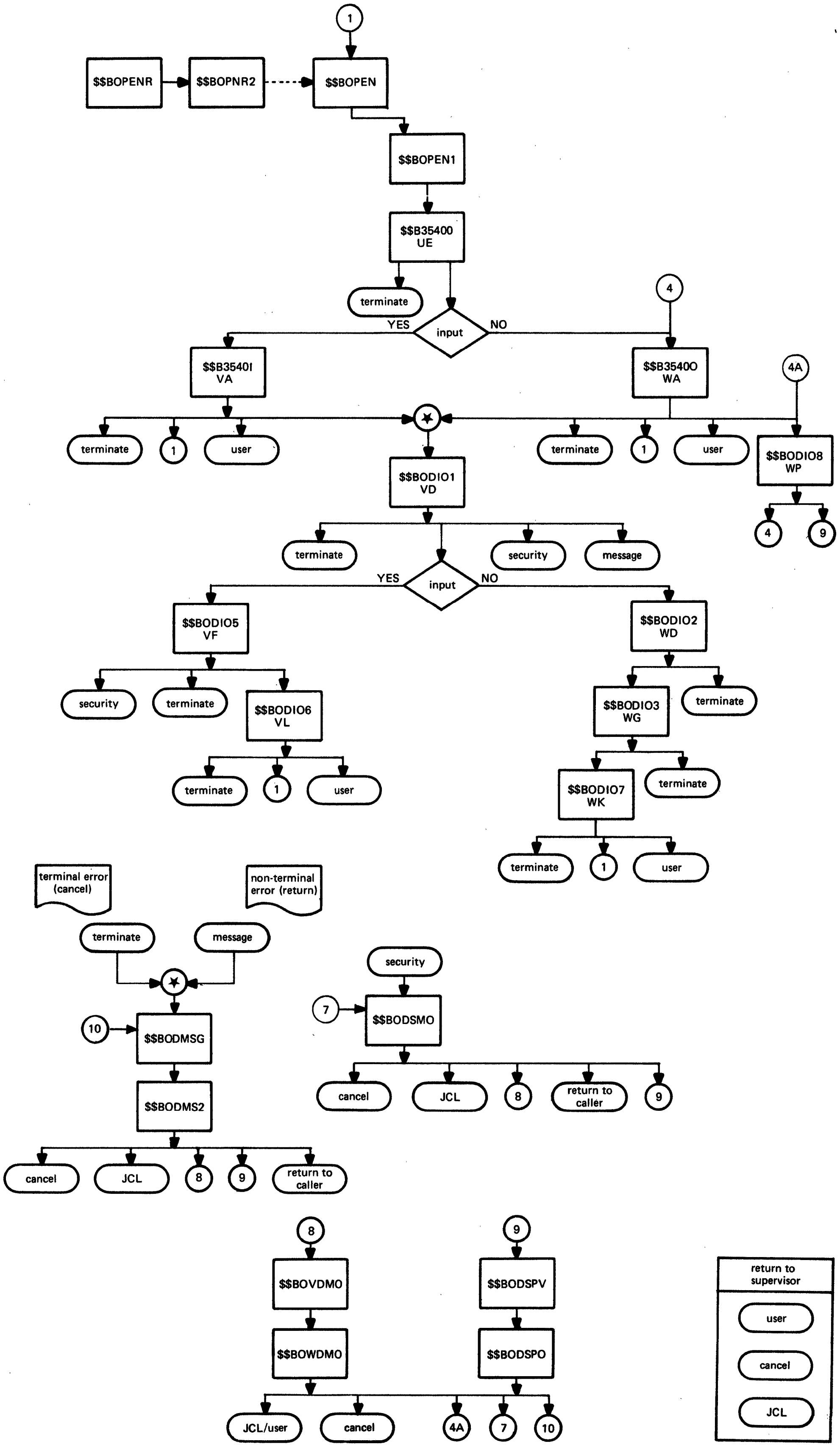
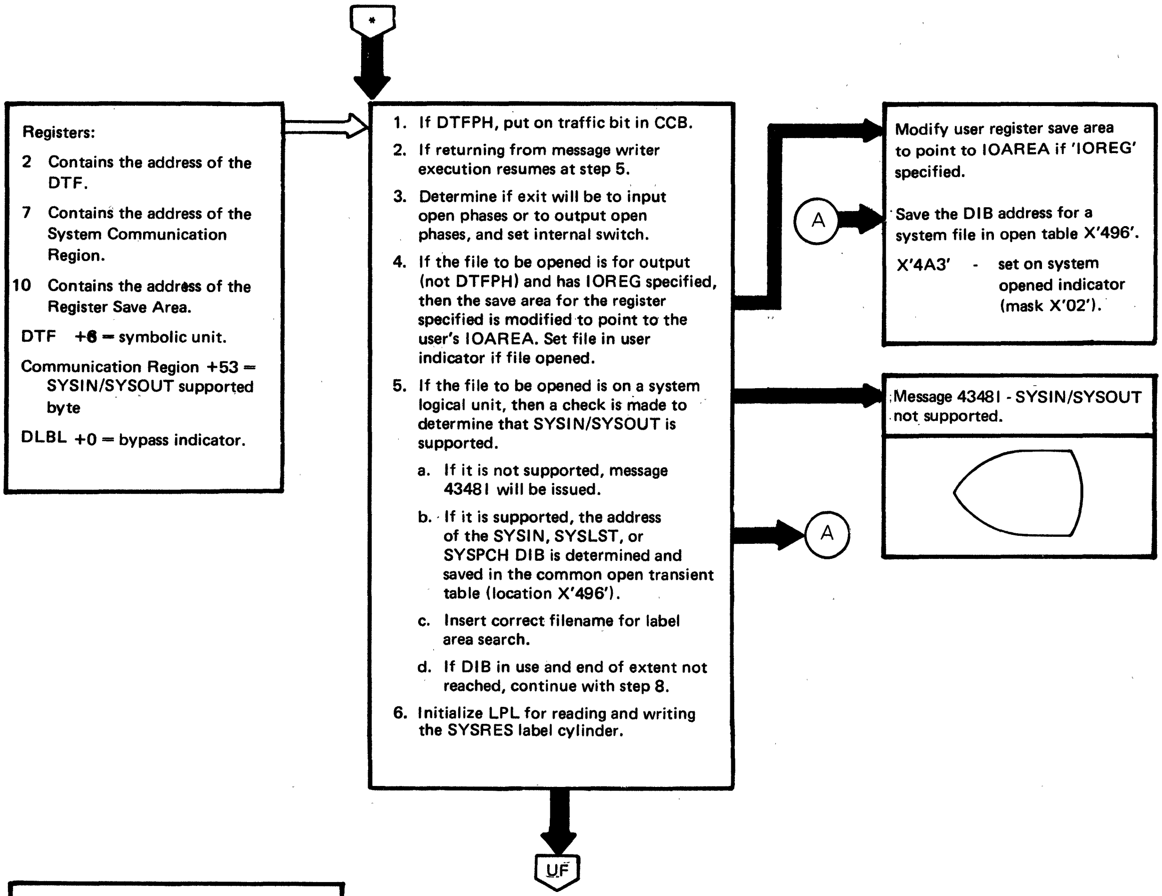


Chart UE. $$B35400: Diskette Open, Initialization (Part 1 of 3)

Input **Processing** **Output**



Registers:

2 Contains the address of the DTF.

7 Contains the address of the System Communication Region.

10 Contains the address of the Register Save Area.

DTF +6 = symbolic unit.

Communication Region +53 = SYSIN/SYSOUT supported byte

DLBL +0 = bypass indicator.

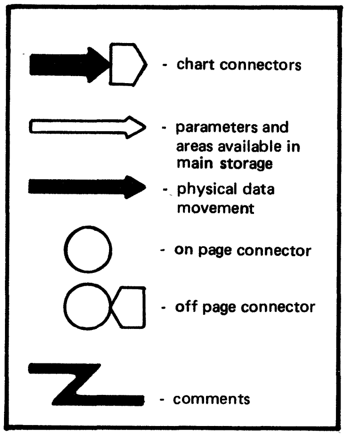
1. If DTFPH, put on traffic bit in CCB.
2. If returning from message writer execution resumes at step 5.
3. Determine if exit will be to input open phases or to output open phases, and set internal switch.
4. If the file to be opened is for output (not DTFPH) and has IOREG specified, then the save area for the register specified is modified to point to the user's IOAREA. Set file in user indicator if file opened.
5. If the file to be opened is on a system logical unit, then a check is made to determine that SYSIN/SYSOUT is supported.
   a. If it is not supported, message 4348I will be issued.
   b. If it is supported, the address of the SYSIN, SYSLST, or SYSPCH DIB is determined and saved in the common open transient table (location X'496').
   c. Insert correct filename for label area search.
   d. If DIB in use and end of extent not reached, continue with step 8.
6. Initialize LPL for reading and writing the SYSRES label cylinder.

Modify user register save area to point to IOAREA if 'IOREG' specified.

Save the DIB address for a system file in open table X'496'.

X'4A3' - set on system opened indicator (mask X'02').

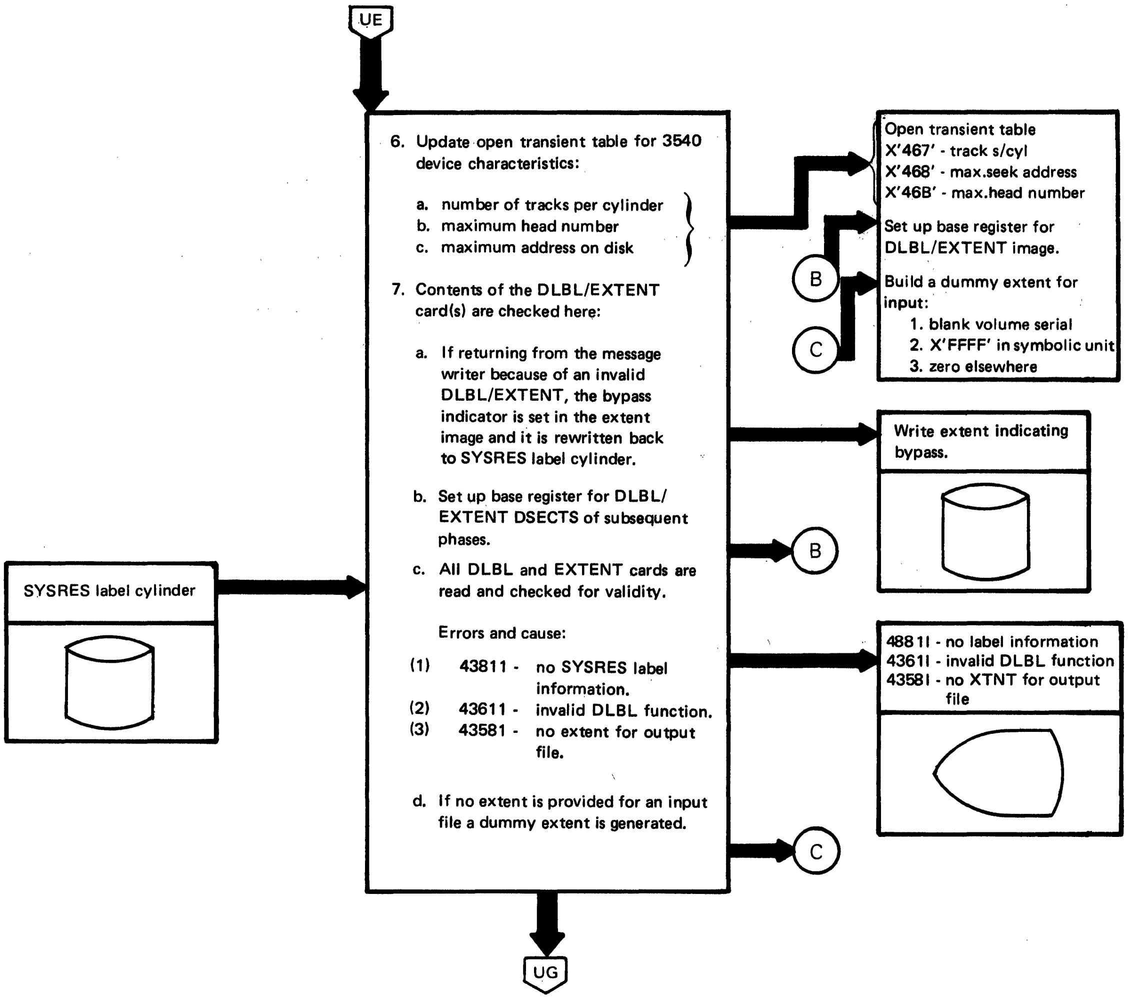
Message 4348I - SYSIN/SYSOUT not supported.



* See LIOCS Volume 1, $$BOPEN1 Monitor

Chart UF. $$B35400: Diskette Open, Initialization (Part 2 of 3)

Input **Processing** **Output**



SYSRES label cylinder

6. Update open transient table for 3540 device characteristics:
   a. number of tracks per cylinder
   b. maximum head number
   c. maximum address on disk
7. Contents of the DLBL/EXTENT card(s) are checked here:
   a. If returning from the message writer because of an invalid DLBL/EXTENT, the bypass indicator is set in the extent image and it is rewritten back to SYSRES label cylinder.
   b. Set up base register for DLBL/EXTENT DSECTS of subsequent phases.
   c. All DLBL and EXTENT cards are read and checked for validity.

      Errors and cause:

      (1) 43811 - no SYSRES label information.
      (2) 43611 - invalid DLBL function.
      (3) 43581 - no extent for output file.
   d. If no extent is provided for an input file a dummy extent is generated.

Open transient table
X'467' - track s/cyl
X'468' - max.seek address
X'46B' - max.head number

Set up base register for DLBL/EXTENT image.

Build a dummy extent for input:
1. blank volume serial
2. X'FFFF' in symbolic unit
3. zero elsewhere

Write extent indicating bypass.

48811 - no label information
43611 - invalid DLBL function
43581 - no XTNT for output file

Chart UG. $$B35400: Diskette Open, Initialization (Part 3 of 3)

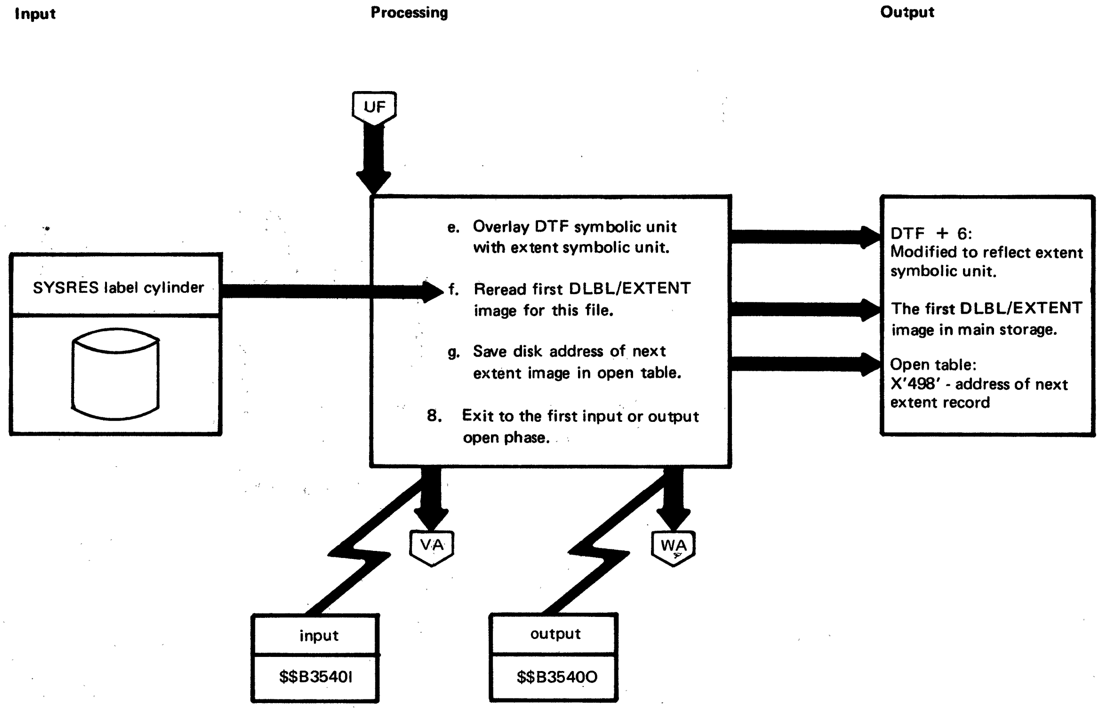
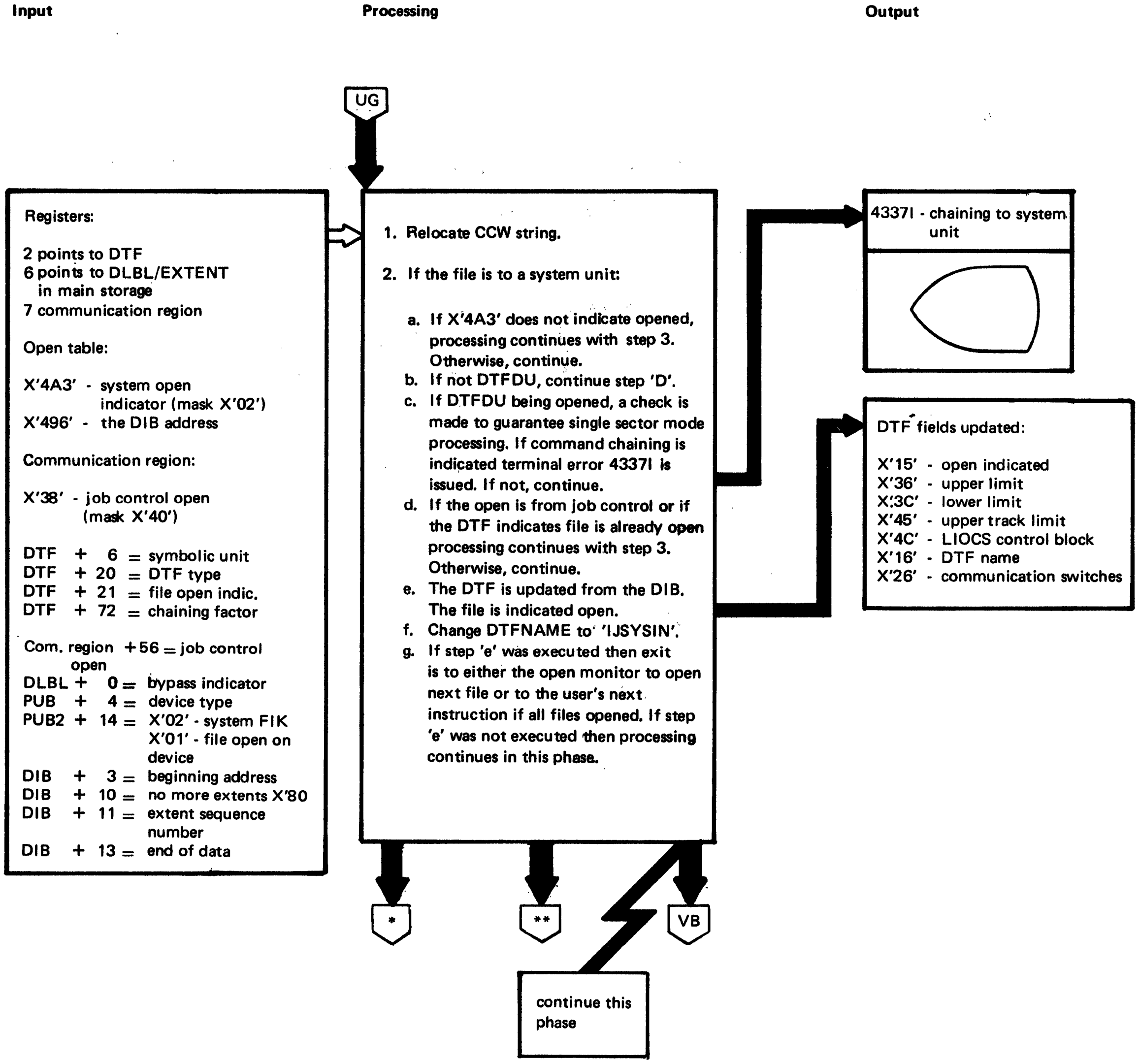

Input
Processing
Output
UF
SYSRES label cylinder
e. Overlay DTF symbolic unit with extent symbolic unit.
f. Reread first DLBL/EXTENT image for this file.
g. Save disk address of next extent image in open table.
8. Exit to the first input or output open phase.
DTF + 6: Modified to reflect extent symbolic unit.
The first DLBL/EXTENT image in main storage.
Open table: X'498' - address of next extent record
VA
WA
input
$$B3540I
output
$$B3540O

Chart VA. $$B3540I: Diskette Open Input (Part 1 of 3)



Registers:

2 points to DTF
6 points to DLBL/EXTENT
in main storage
7 communication region

Open table:

X'4A3' - system open indicator (mask X'02')
X'496' - the DIB address

Communication region:

X'38' - job control open (mask X'40')

DTF + 6 = symbolic unit
DTF + 20 = DTF type
DTF + 21 = file open indic.
DTF + 72 = chaining factor

Com. region +56 = job control open
DLBL + 0 = bypass indicator
PUB + 4 = device type
PUB2 + 14 = X'02' - system FIK
X'01' - file open on device
DIB + 3 = beginning address
DIB + 10 = no more extents X'80
DIB + 11 = extent sequence number
DIB + 13 = end of data

1. Relocate CCW string.

2. If the file is to a system unit:

   a. If X'4A3' does not indicate opened, processing continues with step 3. Otherwise, continue.
   b. If not DTFDU, continue step 'D'.
   c. If DTFDU being opened, a check is made to guarantee single sector mode processing. If command chaining is indicated terminal error 4337I is issued. If not, continue.
   d. If the open is from job control or if the DTF indicates file is already open processing continues with step 3. Otherwise, continue.
   e. The DTF is updated from the DIB. The file is indicated open.
   f. Change DTFNAME to 'IJSYSIN'.
   g. If step 'e' was executed then exit is to either the open monitor to open next file or to the user's next instruction if all files opened. If step 'e' was not executed then processing continues in this phase.

4337I - chaining to system unit

DTF fields updated:

X'15' - open indicated
X'36' - upper limit
X'3C' - lower limit
X'45' - upper track limit
X'4C' - LIOCS control block
X'16' - DTF name
X'26' - communication switches

continue this phase

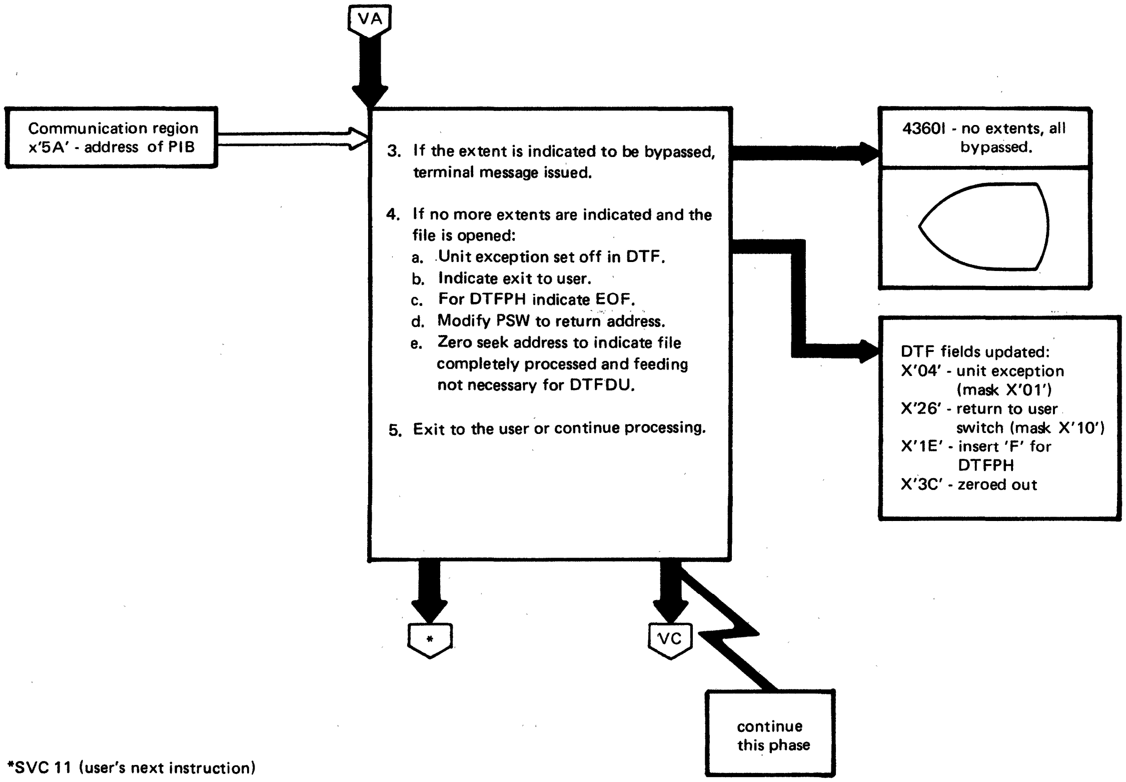
* See LIOCS 1, $$BOPEN1 Monitor
**SVC 11 (user's next instruction)

Chart VB. $$B3540I: Diskette Open Input (Part 2 of 3)

Input

Processing

Output



3. If the extent is indicated to be bypassed, terminal message issued.

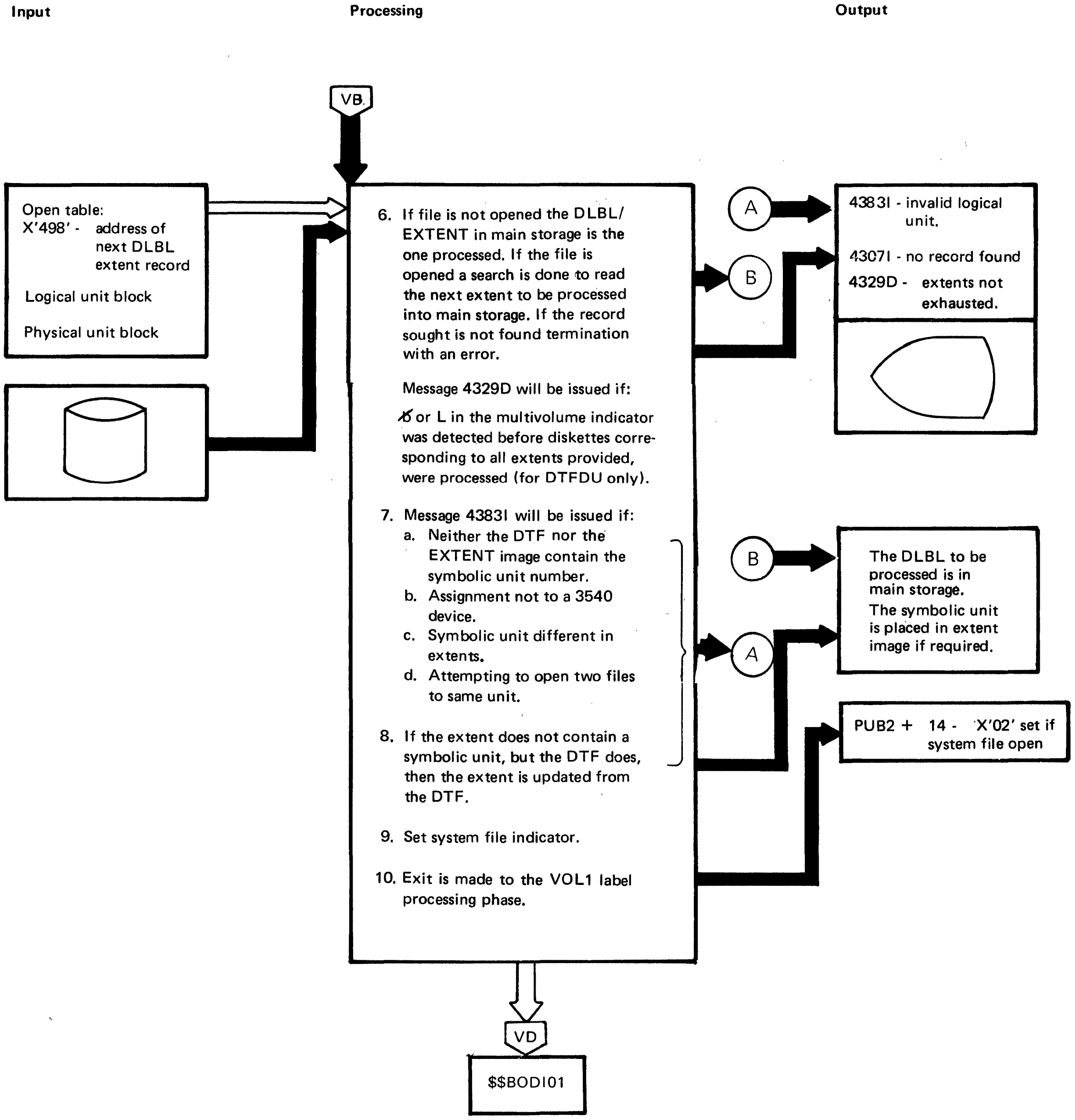
4. If no more extents are indicated and the file is opened:
   a. Unit exception set off in DTF.
   b. Indicate exit to user.
   c. For DTFPH indicate EOF.
   d. Modify PSW to return address.
   e. Zero seek address to indicate file completely processed and feeding not necessary for DTFDU.

5. Exit to the user or continue processing.

4360I - no extents, all bypassed.
DTF fields updated:
X'04' - unit exception (mask X'01')
X'26' - return to user switch (mask X'10')
X'1E' - insert 'F' for DTFPH
X'3C' - zeroed out
*
VC
continue this phase

*SVC 11 (user's next instruction)

Chart VC. $$B3540I: Diskette Open Input (Part 3 of 3)



Open table:
X'498' - address of next DLBL extent record

Logical unit block

Physical unit block

6. If file is not opened the DLBL/EXTENT in main storage is the one processed. If the file is opened a search is done to read the next extent to be processed into main storage. If the record sought is not found termination with an error.

   Message 4329D will be issued if:

   ƀ or L in the multivolume indicator was detected before diskettes corresponding to all extents provided, were processed (for DTFDU only).

7. Message 4383I will be issued if:
   a. Neither the DTF nor the EXTENT image contain the symbolic unit number.
   b. Assignment not to a 3540 device.
   c. Symbolic unit different in extents.
   d. Attempting to open two files to same unit.

8. If the extent does not contain a symbolic unit, but the DTF does, then the extent is updated from the DTF.

9. Set system file indicator.

10. Exit is made to the VOL1 label processing phase.

4383I - invalid logical unit.

4307I - no record found

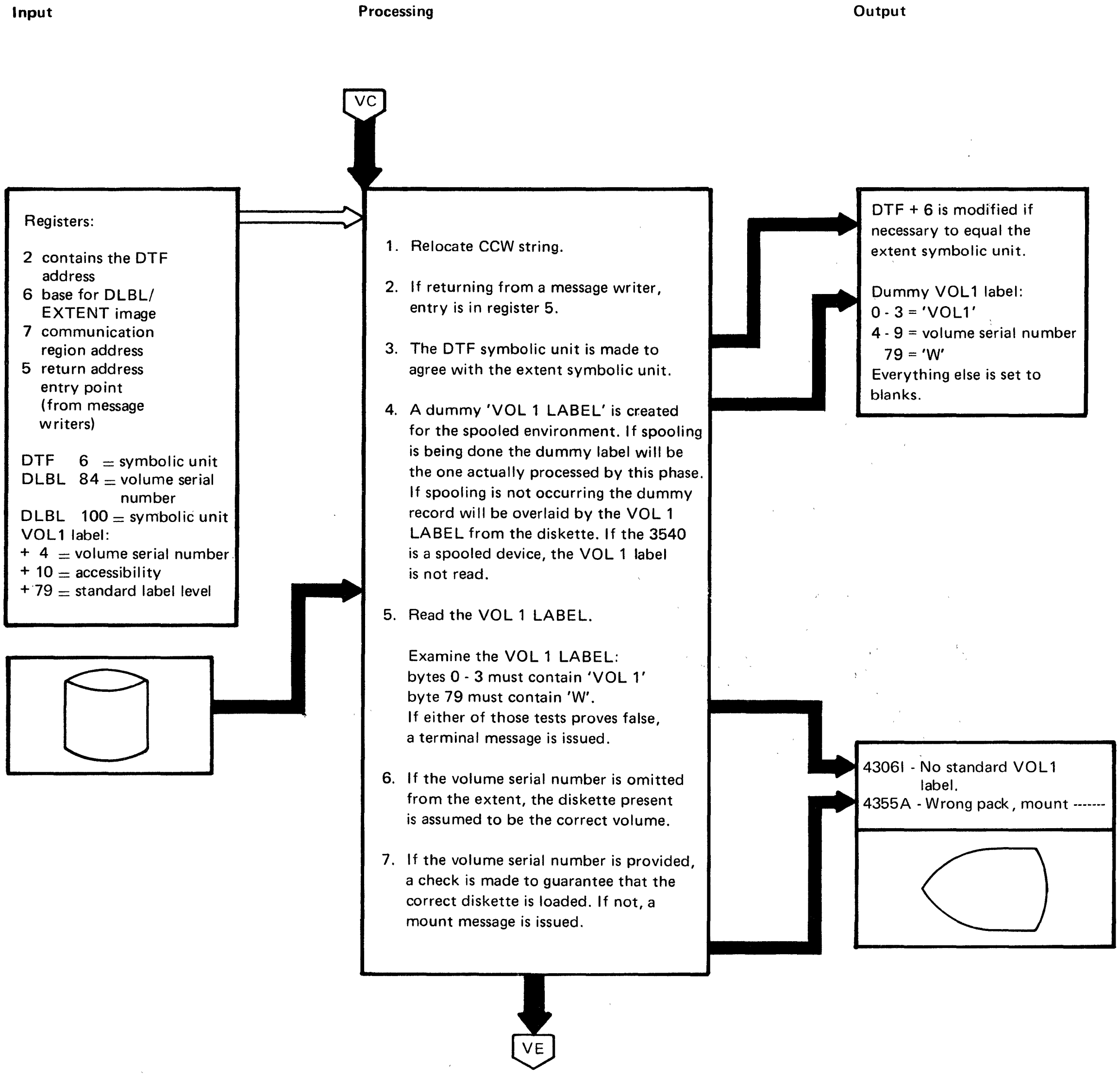
4329D - extents not exhausted.

The DLBL to be processed is in main storage.
The symbolic unit is placed in extent image if required.

PUB2 + 14 - X'02' set if system file open

$$BODI01

Chart VD. $$BODIO1: Diskette Volume Label Processor (Part 1 of 2)



Registers:

2 contains the DTF address
6 base for DLBL/ EXTENT image
7 communication region address
5 return address entry point (from message writers)

DTF 6 = symbolic unit
DLBL 84 = volume serial number
DLBL 100 = symbolic unit
VOL1 label:
+ 4 = volume serial number
+ 10 = accessibility
+ 79 = standard label level

1. Relocate CCW string.
2. If returning from a message writer, entry is in register 5.
3. The DTF symbolic unit is made to agree with the extent symbolic unit.
4. A dummy 'VOL 1 LABEL' is created for the spooled environment. If spooling is being done the dummy label will be the one actually processed by this phase. If spooling is not occurring the dummy record will be overlaid by the VOL 1 LABEL from the diskette. If the 3540 is a spooled device, the VOL 1 label is not read.
5. Read the VOL 1 LABEL.

   Examine the VOL 1 LABEL:
   bytes 0 - 3 must contain 'VOL 1'
   byte 79 must contain 'W'.
   If either of those tests proves false, a terminal message is issued.
6. If the volume serial number is omitted from the extent, the diskette present is assumed to be the correct volume.
7. If the volume serial number is provided, a check is made to guarantee that the correct diskette is loaded. If not, a mount message is issued.

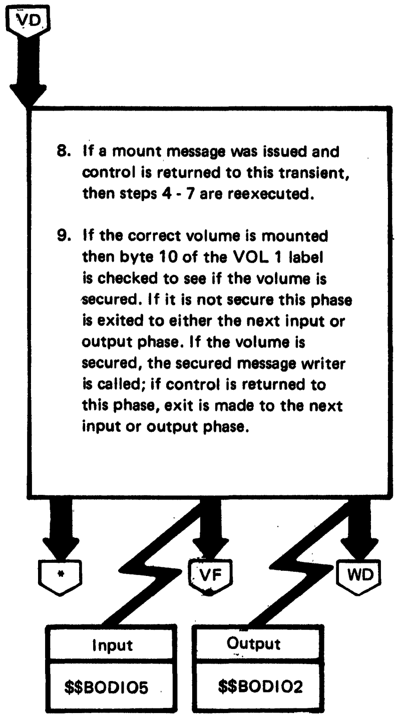
DTF + 6 is modified if necessary to equal the extent symbolic unit.

Dummy VOL1 label:
0 - 3 = 'VOL1'
4 - 9 = volume serial number
79 = 'W'
Everything else is set to blanks.

4306I - No standard VOL1 label.
4355A - Wrong pack, mount -------

Chart VE. $$BODIO1: Diskette Volume Label Processor (Part 2 of 2)

Input **Processing** **Output**

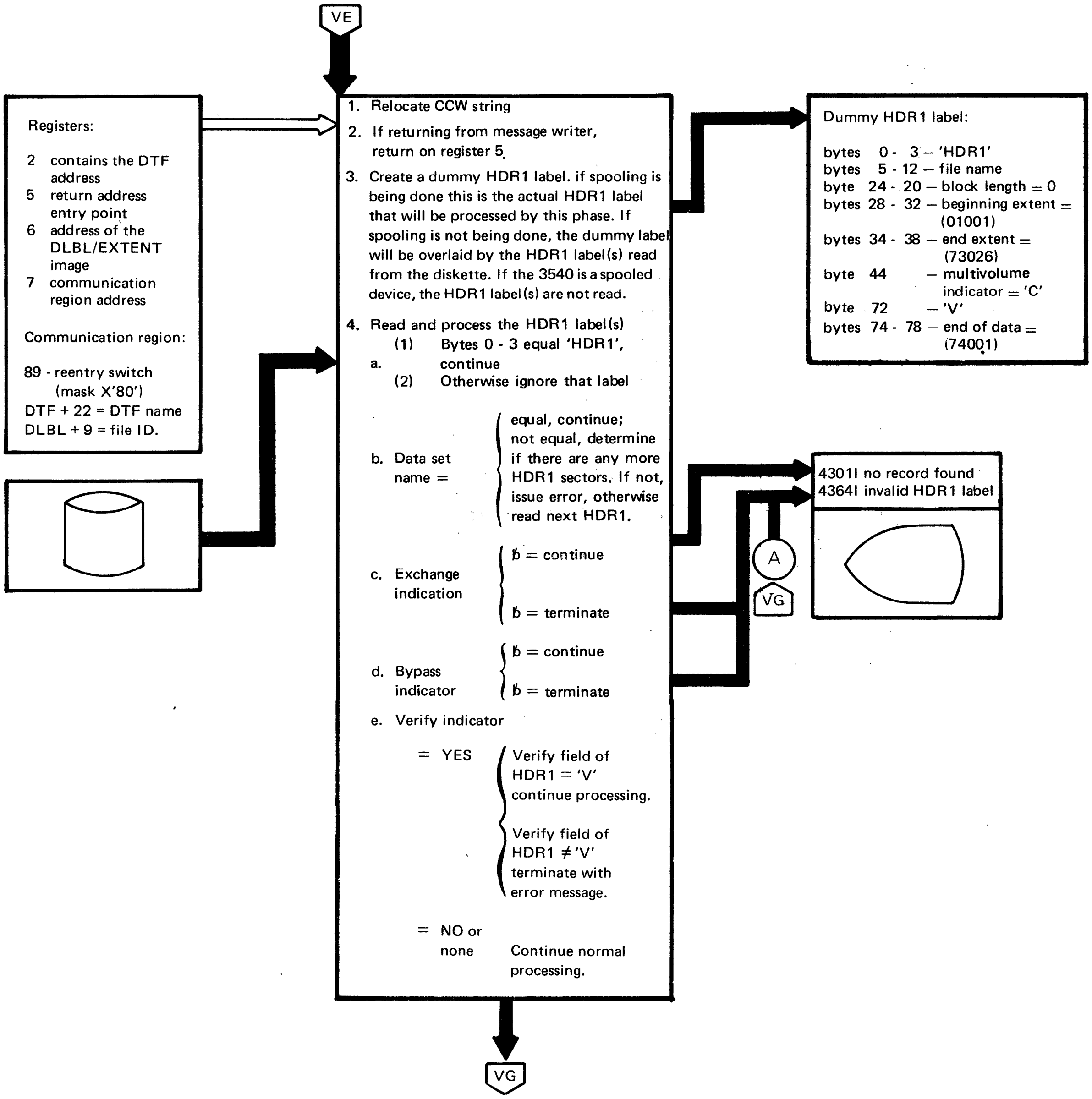


8. If a mount message was issued and control is returned to this transient, then steps 4 - 7 are reexecuted.

9. If the correct volume is mounted then byte 10 of the VOL 1 label is checked to see if the volume is secured. If it is not secure this phase is exited to either the next input or output phase. If the volume is secured, the secured message writer is called; if control is returned to this phase, exit is made to the next input or output phase.

* See LIOCS 1, $$BODSMO (secured message writer)

Chart VF. $$BCDIO5: Diskette Open Input, HDR1 Label Processor (Part 1 of 5)



Registers:

2 contains the DTF address
5 return address entry point
6 address of the DLBL/EXTENT image
7 communication region address

Communication region:

89 - reentry switch (mask X'80')
DTF + 22 = DTF name
DLBL + 9 = file ID.

1. Relocate CCW string
2. If returning from message writer, return on register 5.
3. Create a dummy HDR1 label. if spooling is being done this is the actual HDR1 label that will be processed by this phase. If spooling is not being done, the dummy label will be overlaid by the HDR1 label(s) read from the diskette. If the 3540 is a spooled device, the HDR1 label(s) are not read.
4. Read and process the HDR1 label(s)
   a. (1) Bytes 0 - 3 equal 'HDR1', continue
      (2) Otherwise ignore that label
   b. Data set name = equal, continue; not equal, determine if there are any more HDR1 sectors. If not, issue error, otherwise read next HDR1.
   c. Exchange indication
      b̸ = continue
      b̸ = terminate
   d. Bypass indicator
      b̸ = continue
      b̸ = terminate
   e. Verify indicator
      = YES
         Verify field of HDR1 = 'V' continue processing.
         Verify field of HDR1 ≠ 'V' terminate with error message.
      = NO or none
         Continue normal processing.

Dummy HDR1 label:

bytes 0 - 3 – 'HDR1'
bytes 5 - 12 – file name
byte 24 - 20 – block length = 0
bytes 28 - 32 – beginning extent = (01001)
bytes 34 - 38 – end extent = (73026)
byte 44 – multivolume indicator = 'C'
byte 72 – 'V'
bytes 74 - 78 – end of data = (74001)

4301I no record found
4364I invalid HDR1 label

Chart VG. $$BODIO5: Diskette Open Input, HDR1 Label Processor (Part 2 of 5)

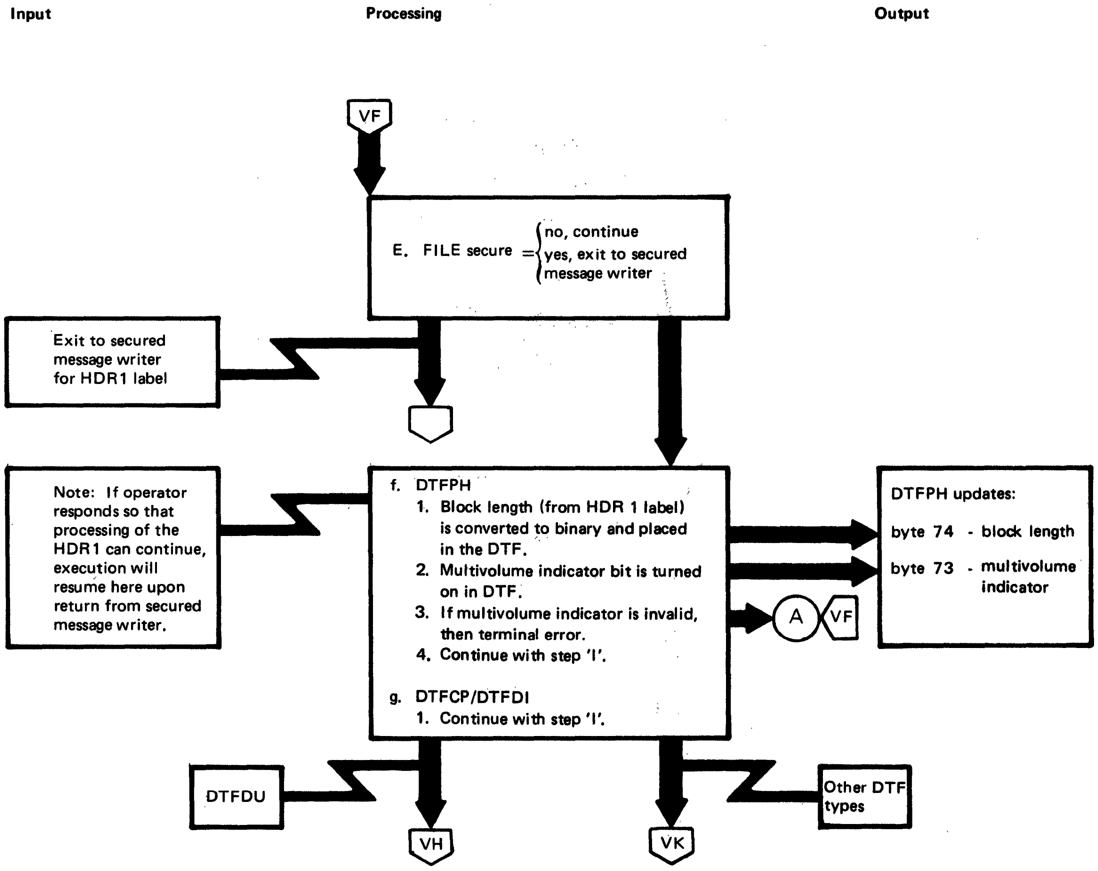
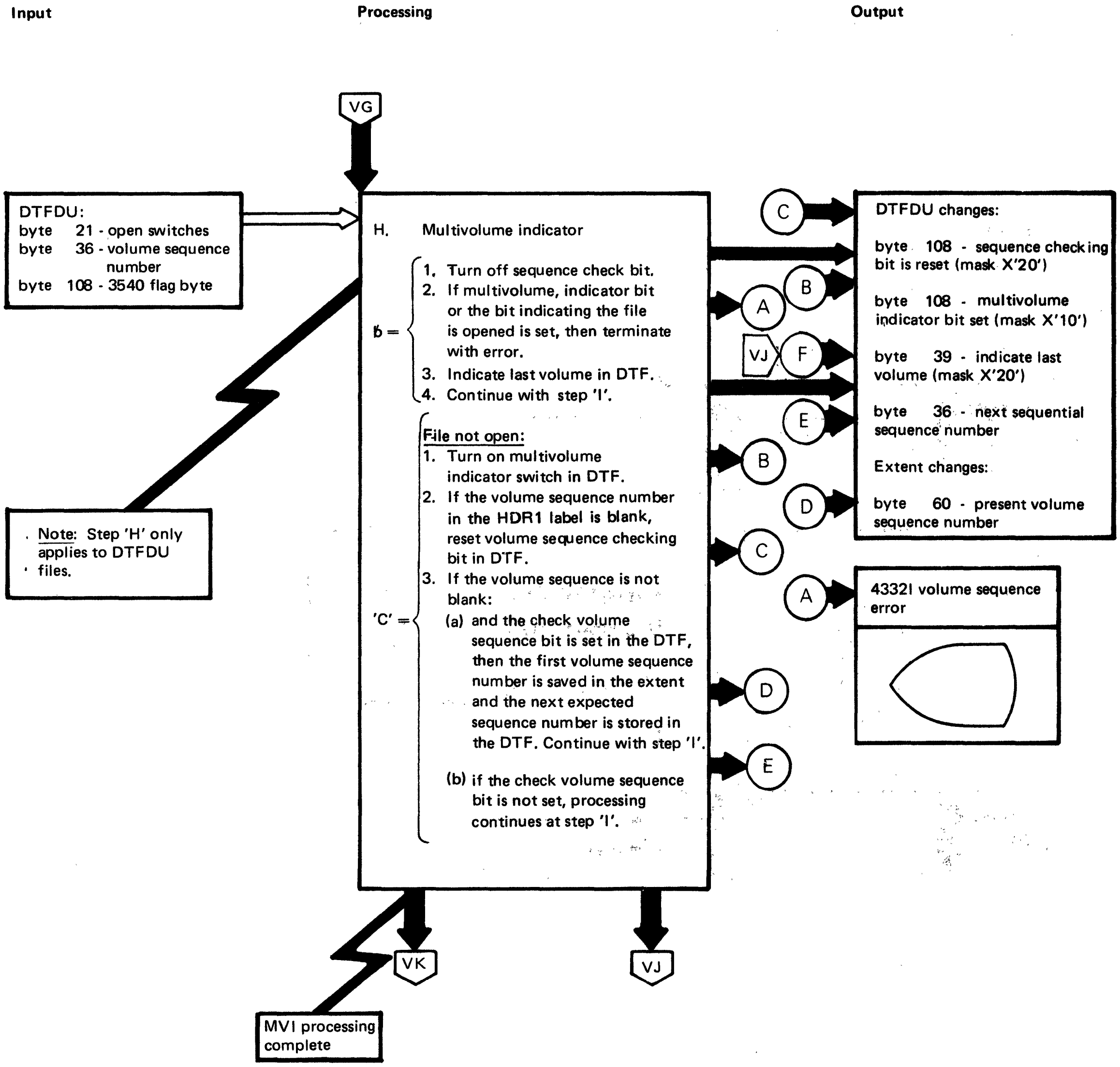

Input
Processing
Output
VF
E. FILE secure = no, continue
yes, exit to secured message writer
Exit to secured message writer for HDR1 label
Note: If operator responds so that processing of the HDR1 can continue, execution will resume here upon return from secured message writer.
f. DTFPH
1. Block length (from HDR 1 label) is converted to binary and placed in the DTF.
2. Multivolume indicator bit is turned on in DTF.
3. If multivolume indicator is invalid, then terminal error.
4. Continue with step 'I'.
g. DTFCP/DTFDI
1. Continue with step 'I'.
A
VF
DTFPH updates:
byte 74 - block length
byte 73 - multivolume indicator
DTFDU
VH
VK
Other DTF types

Chart VH. $$BODIO5: Diskette Open Input, HDR1 Label Processor (Part 3 of 5)



DTFDU:
byte 21 - open switches
byte 36 - volume sequence number
byte 108 - 3540 flag byte

Note: Step 'H' only applies to DTFDU files.

H. Multivolume indicator

ƀ =
1. Turn off sequence check bit.
2. If multivolume, indicator bit or the bit indicating the file is opened is set, then terminate with error.
3. Indicate last volume in DTF.
4. Continue with step 'I'.

'C' =
File not open:
1. Turn on multivolume indicator switch in DTF.
2. If the volume sequence number in the HDR1 label is blank, reset volume sequence checking bit in DTF.
3. If the volume sequence is not blank:
   (a) and the check volume sequence bit is set in the DTF, then the first volume sequence number is saved in the extent and the next expected sequence number is stored in the DTF. Continue with step 'I'.
   (b) if the check volume sequence bit is not set, processing continues at step 'I'.

MVI processing complete

DTFDU changes:

byte 108 - sequence checking bit is reset (mask X'20')

byte 108 - multivolume indicator bit set (mask X'10')

byte 39 - indicate last volume (mask X'20')

byte 36 - next sequential sequence number

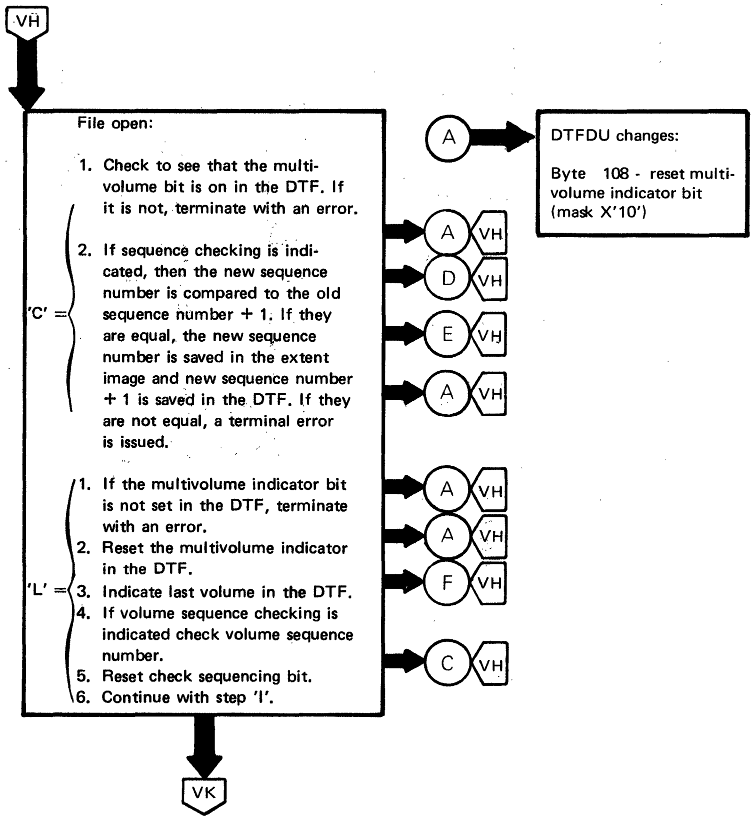
Extent changes:

byte 60 - present volume sequence number

4332I volume sequence error

Chart VJ. $$BOCIO5: Diskette Open Input, HDR1 Label Processor (Part 4 of 5)

Input **Processing** **Output**



File open:

1. Check to see that the multivolume bit is on in the DTF. If it is not, terminate with an error.
2. If sequence checking is indicated, then the new sequence number is compared to the old sequence number + 1. If they are equal, the new sequence number is saved in the extent image and new sequence number + 1 is saved in the DTF. If they are not equal, a terminal error is issued.

'C' =

1. If the multivolume indicator bit is not set in the DTF, terminate with an error.
2. Reset the multivolume indicator in the DTF.
3. Indicate last volume in the DTF.
4. If volume sequence checking is indicated check volume sequence number.
5. Reset check sequencing bit.
6. Continue with step 'I'.

'L' =

DTFDU changes:

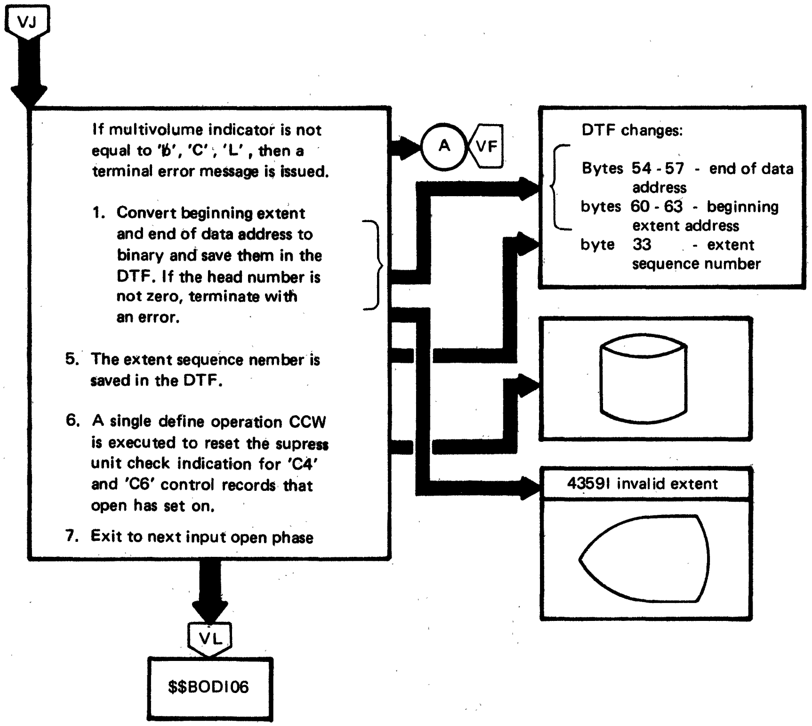
Byte 108 - reset multivolume indicator bit (mask X'10')

Chart VK. $$BODI05: Diskette Open Input, HDR1 Label Processor (Part 5 of 5)

Input

Processing

Output



If multivolume indicator is not equal to 'ƀ', 'C', 'L', then a terminal error message is issued.

1. Convert beginning extent and end of data address to binary and save them in the DTF. If the head number is not zero, terminate with an error.

5. The extent sequence nember is saved in the DTF.

6. A single define operation CCW is executed to reset the supress unit check indication for 'C4' and 'C6' control records that open has set on.

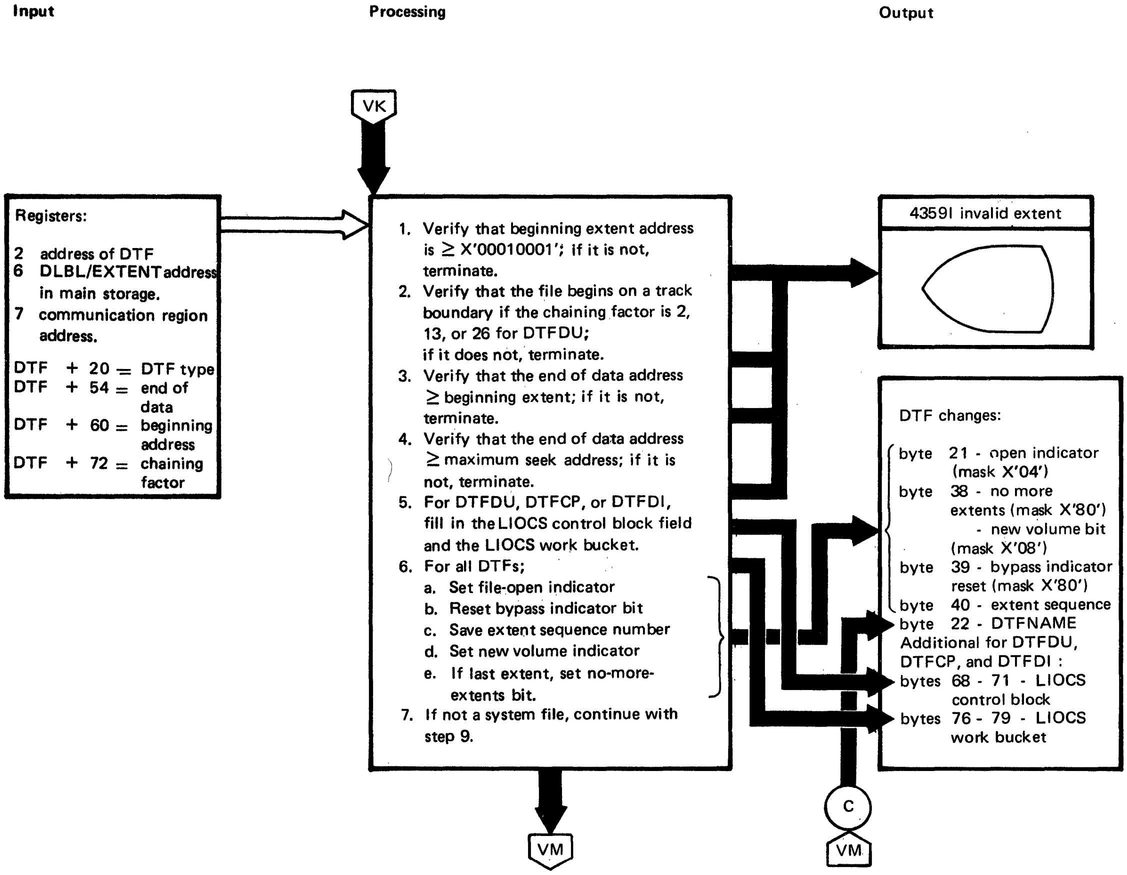
7. Exit to next input open phase

DTF changes:

Bytes 54 - 57 - end of data address
bytes 60 - 63 - beginning extent address
byte 33 - extent sequence number

43591 invalid extent

Chart VI. $$BODI06: Diskette Open Input, Initialize DTF Table (Part 1 of 2)



Registers:

2 address of DTF
6 DLBL/EXTENT address in main storage.
7 communication region address.

DTF + 20 = DTF type
DTF + 54 = end of data
DTF + 60 = beginning address
DTF + 72 = chaining factor

1. Verify that beginning extent address is ≥ X'00010001'; if it is not, terminate.
2. Verify that the file begins on a track boundary if the chaining factor is 2, 13, or 26 for DTFDU; if it does not, terminate.
3. Verify that the end of data address ≥ beginning extent; if it is not, terminate.
4. Verify that the end of data address ≥ maximum seek address; if it is not, terminate.
5. For DTFDU, DTFCP, or DTFDI, fill in the LIOCS control block field and the LIOCS work bucket.
6. For all DTFs;
   a. Set file-open indicator
   b. Reset bypass indicator bit
   c. Save extent sequence number
   d. Set new volume indicator
   e. If last extent, set no-more-extents bit.
7. If not a system file, continue with step 9.

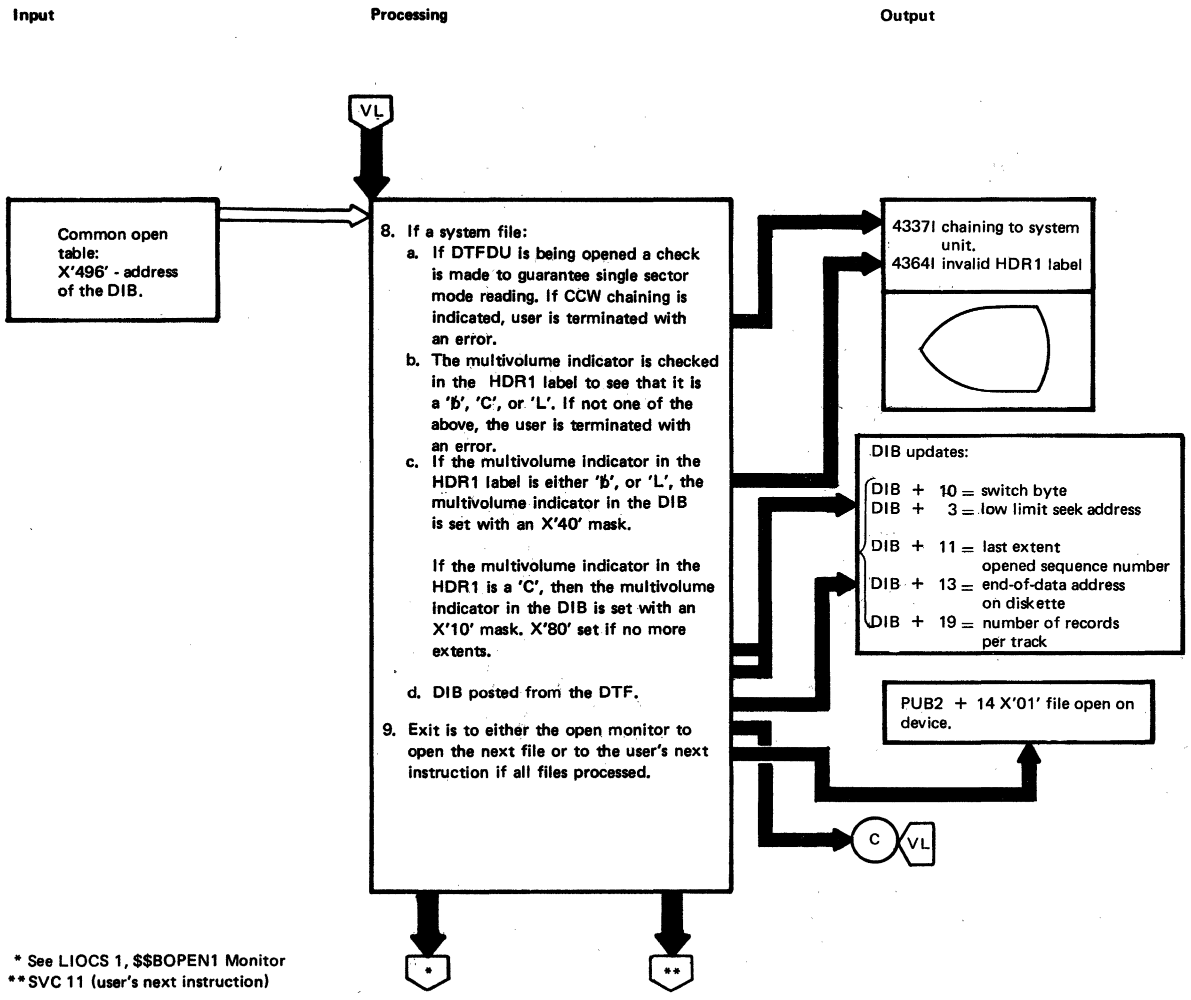
43591 invalid extent

DTF changes:

byte 21 - open indicator (mask X'04')
byte 38 - no more extents (mask X'80')
 - new volume bit (mask X'08')
byte 39 - bypass indicator reset (mask X'80')
byte 40 - extent sequence
byte 22 - DTFNAME

Additional for DTFDU, DTFCP, and DTFDI :
bytes 68 - 71 - LIOCS control block
bytes 76 - 79 - LIOCS work bucket

Chart VM. $$BCDI06: Diskette Open Input, Initialize DTF Table (Part 2 of 2)



Common open table:
X'496' - address of the DIB.

8. If a system file:
   a. If DTFDU is being opened a check is made to guarantee single sector mode reading. If CCW chaining is indicated, user is terminated with an error.
   b. The multivolume indicator is checked in the HDR1 label to see that it is a 'ƀ', 'C', or 'L'. If not one of the above, the user is terminated with an error.
   c. If the multivolume indicator in the HDR1 label is either 'ƀ', or 'L', the multivolume indicator in the DIB is set with an X'40' mask.

      If the multivolume indicator in the HDR1 is a 'C', then the multivolume indicator in the DIB is set with an X'10' mask. X'80' set if no more extents.
   d. DIB posted from the DTF.
9. Exit is to either the open monitor to open the next file or to the user's next instruction if all files processed.

4337I chaining to system unit.
4364I invalid HDR1 label

DIB updates:

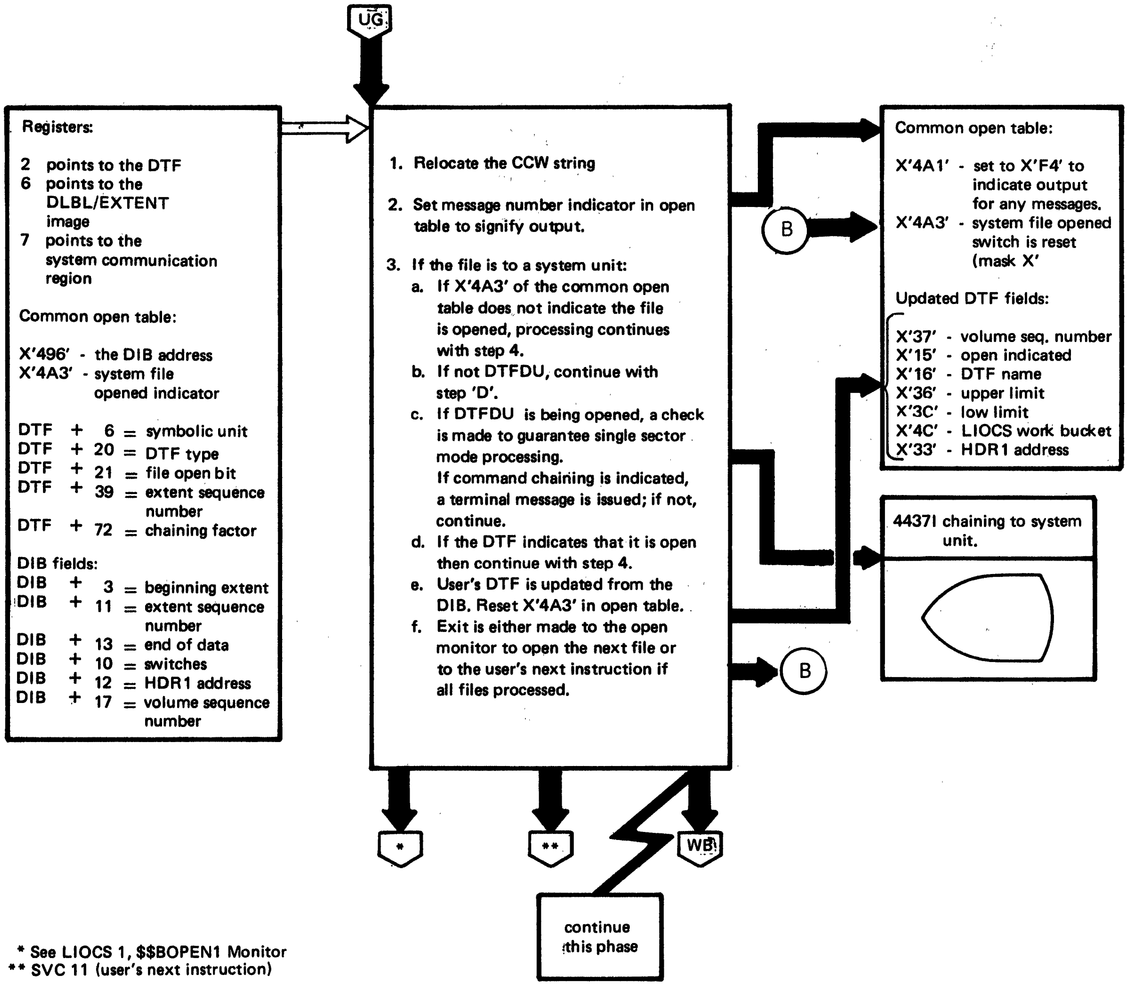
DIB + 10 = switch byte
DIB + 3 = low limit seek address
DIB + 11 = last extent opened sequence number
DIB + 13 = end-of-data address on diskette
DIB + 19 = number of records per track

PUB2 + 14 X'01' file open on device.

* See LIOCS 1, $$BOPEN1 Monitor
**SVC 11 (user's next instruction)

Chart WA. $$B35400: Diskette Open Output (Part 1 of 3)

Input **Processing** **Output**



Registers:

2 points to the DTF
6 points to the DLBL/EXTENT image
7 points to the system communication region

Common open table:

X'496' - the DIB address
X'4A3' - system file opened indicator

DTF + 6 = symbolic unit
DTF + 20 = DTF type
DTF + 21 = file open bit
DTF + 39 = extent sequence number
DTF + 72 = chaining factor

DIB fields:
DIB + 3 = beginning extent
DIB + 11 = extent sequence number
DIB + 13 = end of data
DIB + 10 = switches
DIB + 12 = HDR1 address
DIB + 17 = volume sequence number

1. Relocate the CCW string

2. Set message number indicator in open table to signify output.

3. If the file is to a system unit:
   a. If X'4A3' of the common open table does not indicate the file is opened, processing continues with step 4.
   b. If not DTFDU, continue with step 'D'.
   c. If DTFDU is being opened, a check is made to guarantee single sector mode processing. If command chaining is indicated, a terminal message is issued; if not, continue.
   d. If the DTF indicates that it is open then continue with step 4.
   e. User's DTF is updated from the DIB. Reset X'4A3' in open table.
   f. Exit is either made to the open monitor to open the next file or to the user's next instruction if all files processed.

Common open table:

X'4A1' - set to X'F4' to indicate output for any messages.
X'4A3' - system file opened switch is reset (mask X'

Updated DTF fields:

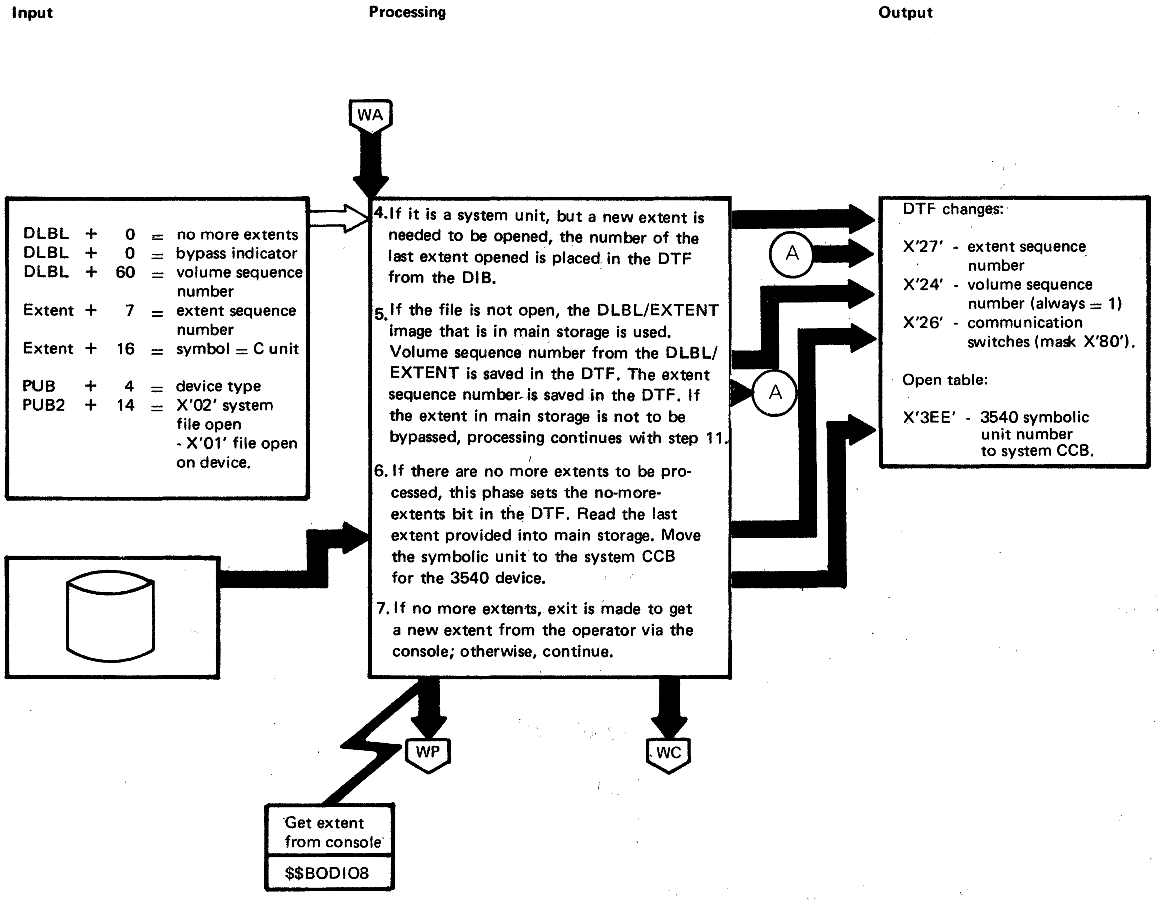
X'37' - volume seq. number
X'15' - open indicated
X'16' - DTF name
X'36' - upper limit
X'3C' - low limit
X'4C' - LIOCS work bucket
X'33' - HDR1 address

44371 chaining to system unit.

continue this phase

* See LIOCS 1, $$BOPEN1 Monitor
** SVC 11 (user's next instruction)

Chart WE. $$B3540C: Diskette Open Output (Part 2 of 3)



DLBL	+	0	=	no more extents
DLBL	+	0	=	bypass indicator
DLBL	+	60	=	volume sequence number
Extent	+	7	=	extent sequence number
Extent	+	16	=	symbol = C unit
PUB	+	4	=	device type
PUB2	+	14	=	X'02' system file open - X'01' file open on device.

4. If it is a system unit, but a new extent is needed to be opened, the number of the last extent opened is placed in the DTF from the DIB.

5. If the file is not open, the DLBL/EXTENT image that is in main storage is used. Volume sequence number from the DLBL/EXTENT is saved in the DTF. The extent sequence number is saved in the DTF. If the extent in main storage is not to be bypassed, processing continues with step 11.

6. If there are no more extents to be processed, this phase sets the no-more-extents bit in the DTF. Read the last extent provided into main storage. Move the symbolic unit to the system CCB for the 3540 device.

7. If no more extents, exit is made to get a new extent from the operator via the console; otherwise, continue.

Get extent from console

$$BODIO8

DTF changes:

X'27' - extent sequence number

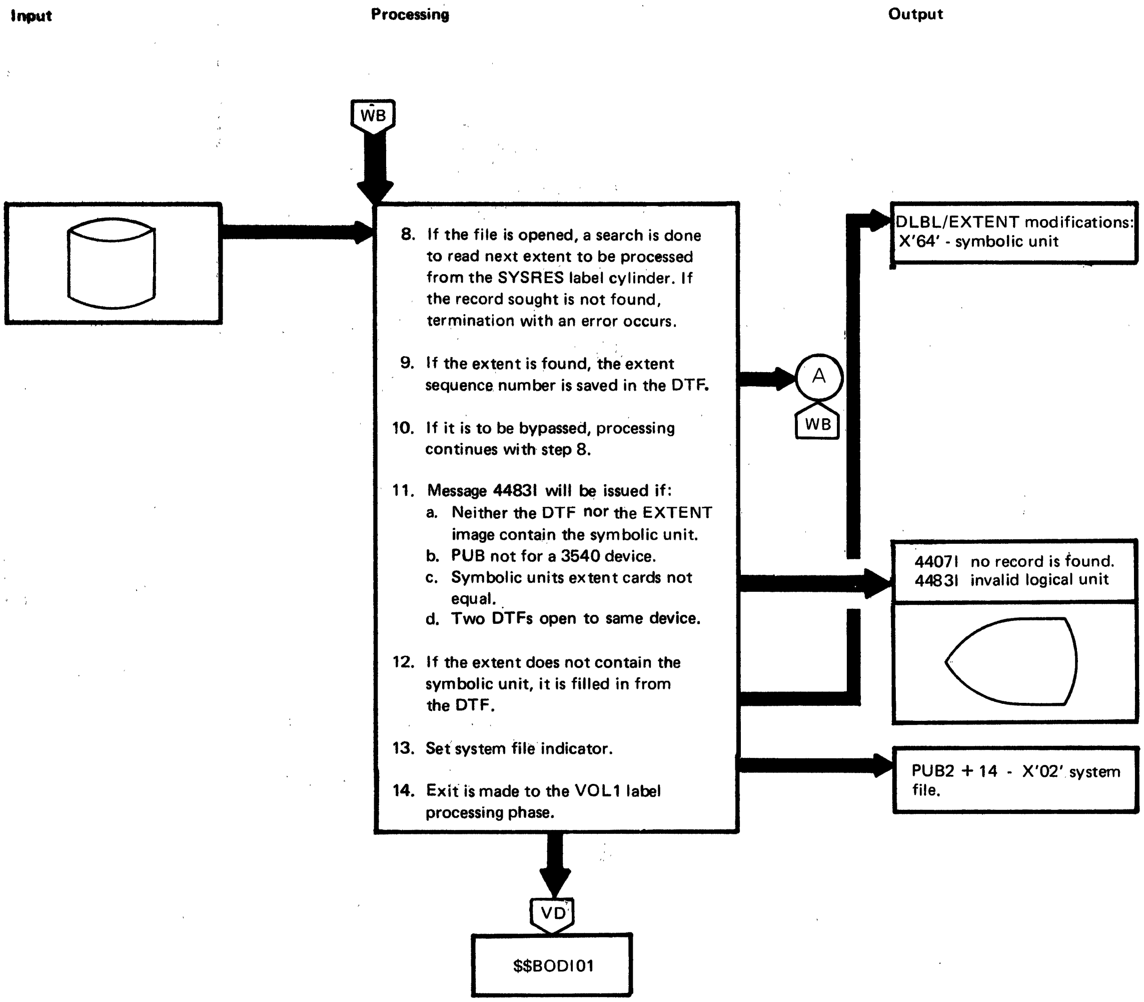
X'24' - volume sequence number (always = 1)

X'26' - communication switches (mask X'80').

Open table:

X'3EE' - 3540 symbolic unit number to system CCB.

Chart WC. $$B35400: Diskette Open Output (Part 3 of 3)



8. If the file is opened, a search is done to read next extent to be processed from the SYSRES label cylinder. If the record sought is not found, termination with an error occurs.

9. If the extent is found, the extent sequence number is saved in the DTF.

10. If it is to be bypassed, processing continues with step 8.

11. Message 4483I will be issued if:
    a. Neither the DTF nor the EXTENT image contain the symbolic unit.
    b. PUB not for a 3540 device.
    c. Symbolic units extent cards not equal.
    d. Two DTFs open to same device.

12. If the extent does not contain the symbolic unit, it is filled in from the DTF.

13. Set system file indicator.

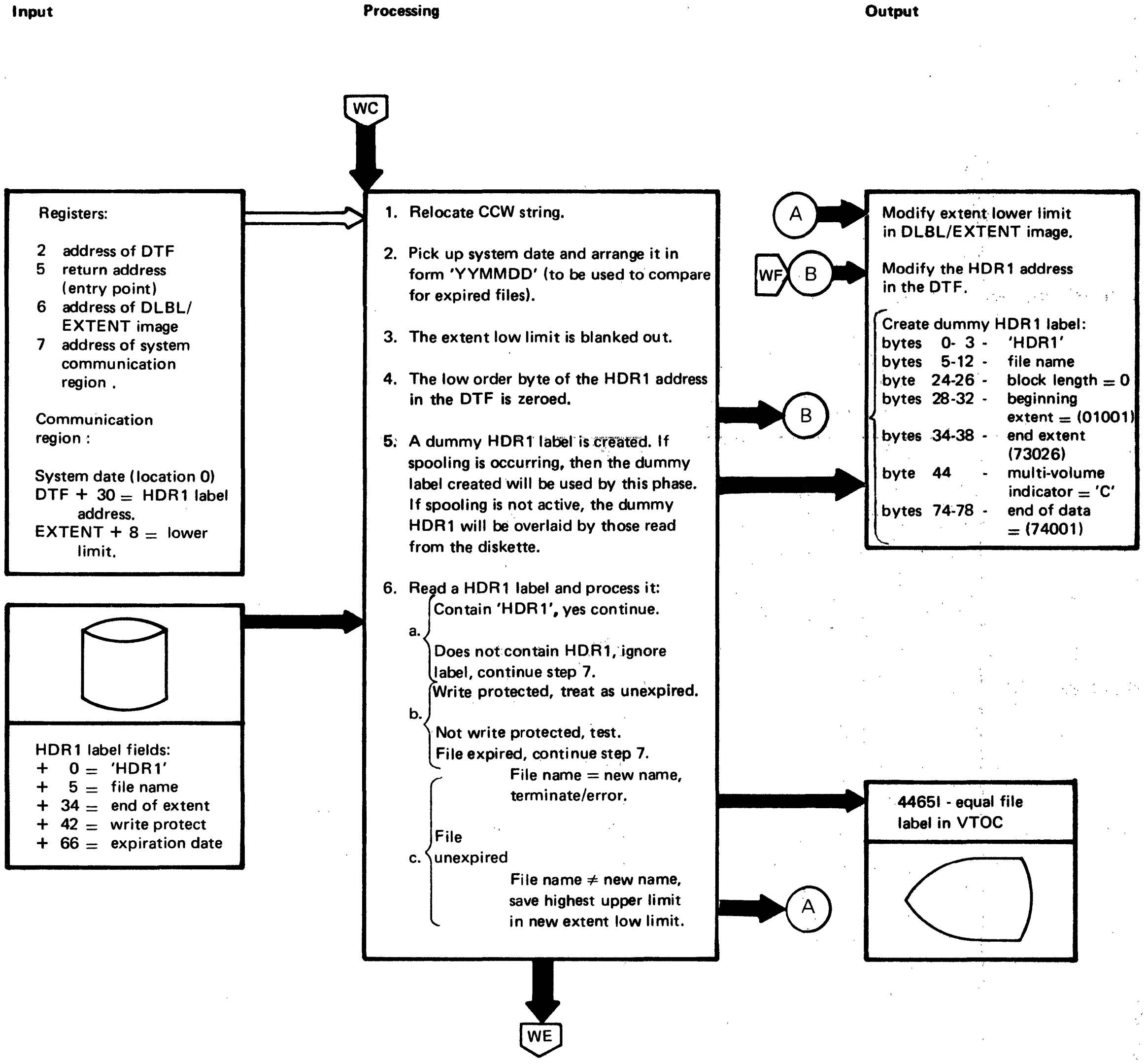
14. Exit is made to the VOL1 label processing phase.

DLBL/EXTENT modifications:
X'64' - symbolic unit

4407I no record is found.
4483I invalid logical unit

PUB2 + 14 - X'02' system file.

Chart WD. $$BCDIO2: Diskette Open Output, Determine Extents (Part 1 of 3)



Registers:

2 address of DTF
5 return address (entry point)
6 address of DLBL/EXTENT image
7 address of system communication region .

Communication region :

System date (location 0)
DTF + 30 = HDR1 label address.
EXTENT + 8 = lower limit.

HDR1 label fields:
+ 0 = 'HDR1'
+ 5 = file name
+ 34 = end of extent
+ 42 = write protect
+ 66 = expiration date

1. Relocate CCW string.
2. Pick up system date and arrange it in form 'YYMMDD' (to be used to compare for expired files).
3. The extent low limit is blanked out.
4. The low order byte of the HDR1 address in the DTF is zeroed.
5. A dummy HDR1 label is created. If spooling is occurring, then the dummy label created will be used by this phase. If spooling is not active, the dummy HDR1 will be overlaid by those read from the diskette.
6. Read a HDR1 label and process it:
   a. Contain 'HDR1', yes continue.
      Does not contain HDR1, ignore label, continue step 7.
   b. Write protected, treat as unexpired.
      Not write protected, test.
      File expired, continue step 7.
   c. File unexpired
      File name = new name, terminate/error.
      File name ≠ new name, save highest upper limit in new extent low limit.

Modify extent lower limit in DLBL/EXTENT image.

Modify the HDR1 address in the DTF.

Create dummy HDR1 label:
bytes 0- 3 - 'HDR1'
bytes 5-12 - file name
byte 24-26 - block length = 0
bytes 28-32 - beginning extent = (01001)
bytes 34-38 - end extent (73026)
byte 44 - multi-volume indicator = 'C'
bytes 74-78 - end of data = (74001)

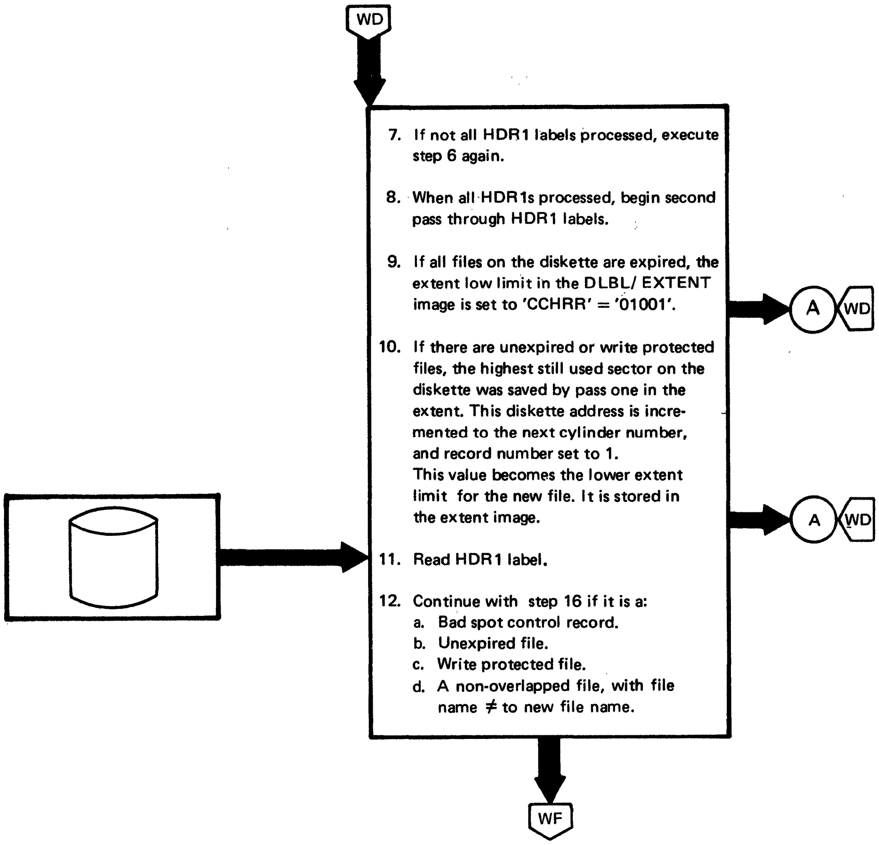
4465I - equal file label in VTOC

Chart WE. $$BODIO2: Diskette Open Output, Determine Extents (Part 2 of 3)

Input

Processing

Output



7. If not all HDR1 labels processed, execute step 6 again.

8. When all HDR1s processed, begin second pass through HDR1 labels.

9. If all files on the diskette are expired, the extent low limit in the DLBL/ EXTENT image is set to 'CCHRR' = '01001'.

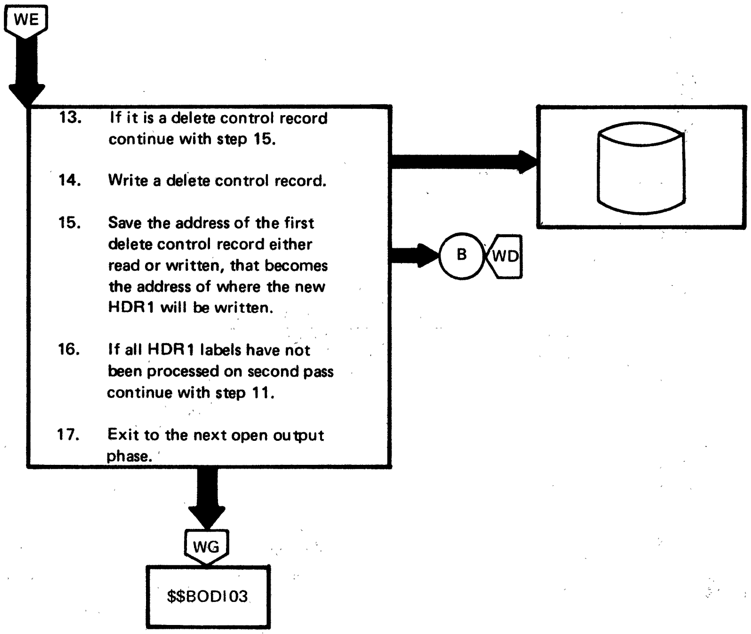
10. If there are unexpired or write protected files, the highest still used sector on the diskette was saved by pass one in the extent. This diskette address is incremented to the next cylinder number, and record number set to 1.
This value becomes the lower extent limit for the new file. It is stored in the extent image.

11. Read HDR1 label.

12. Continue with step 16 if it is a:
    a. Bad spot control record.
    b. Unexpired file.
    c. Write protected file.
    d. A non-overlapped file, with file name ≠ to new file name.

Chart WF. $$BODIO2: Diskette Open Output, Determine Extents (Part 3 of 3)

Input Processing Output



13. If it is a delete control record continue with step 15.

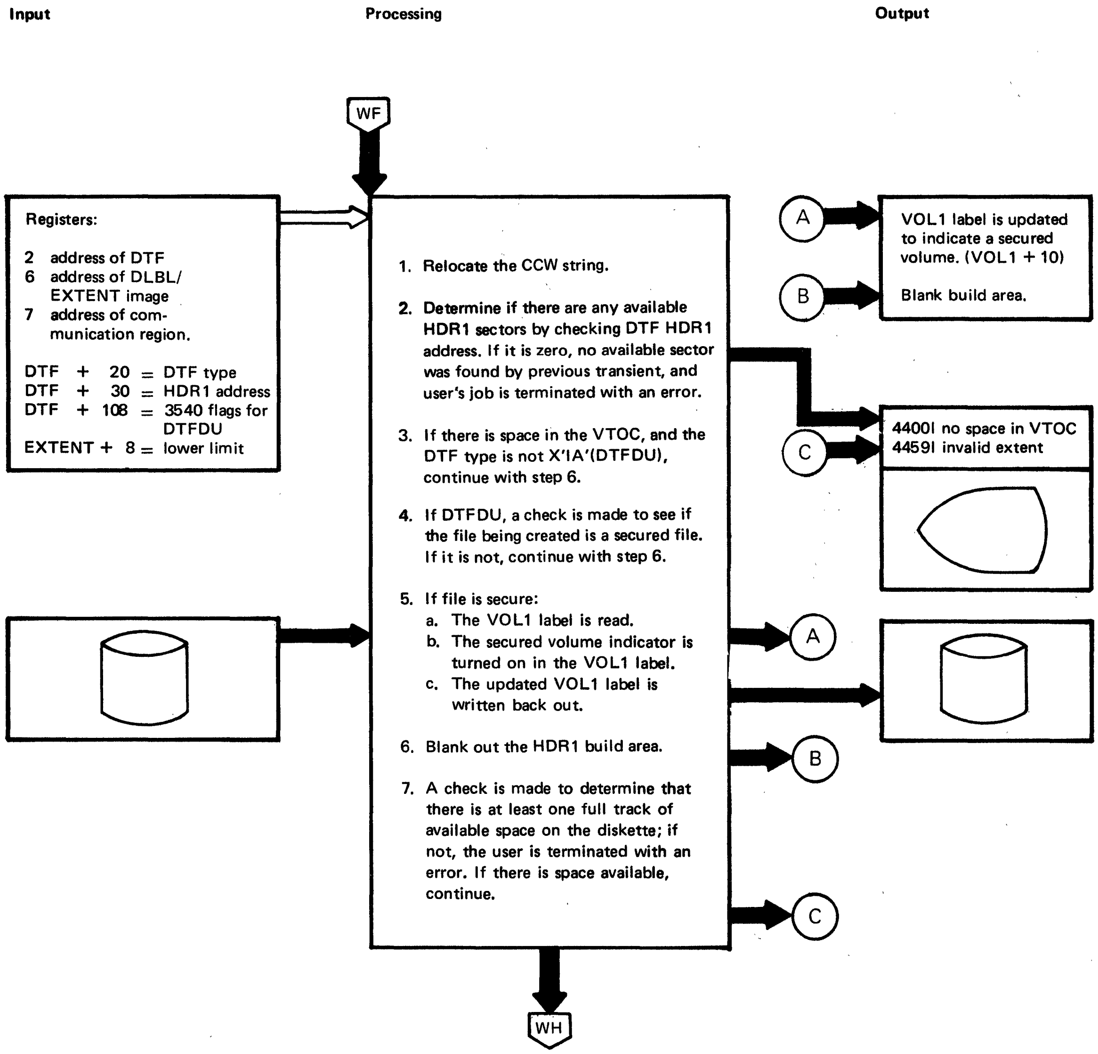
14. Write a delete control record.

15. Save the address of the first delete control record either read or written, that becomes the address of where the new HDR1 will be written.

16. If all HDR1 labels have not been processed on second pass continue with step 11.

17. Exit to the next open output phase.

Chart WG. $$BODIO3: Diskette Open Output, Create/Write New HDR1 Label (Part 1 of 3)



Registers:

2 address of DTF
6 address of DLBL/ EXTENT image
7 address of communication region.

DTF + 20 = DTF type
DTF + 30 = HDR1 address
DTF + 108 = 3540 flags for DTFDU
EXTENT + 8 = lower limit

1. Relocate the CCW string.
2. Determine if there are any available HDR1 sectors by checking DTF HDR1 address. If it is zero, no available sector was found by previous transient, and user's job is terminated with an error.
3. If there is space in the VTOC, and the DTF type is not X'IA'(DTFDU), continue with step 6.
4. If DTFDU, a check is made to see if the file being created is a secured file. If it is not, continue with step 6.
5. If file is secure:
   a. The VOL1 label is read.
   b. The secured volume indicator is turned on in the VOL1 label.
   c. The updated VOL1 label is written back out.
6. Blank out the HDR1 build area.
7. A check is made to determine that there is at least one full track of available space on the diskette; if not, the user is terminated with an error. If there is space available, continue.

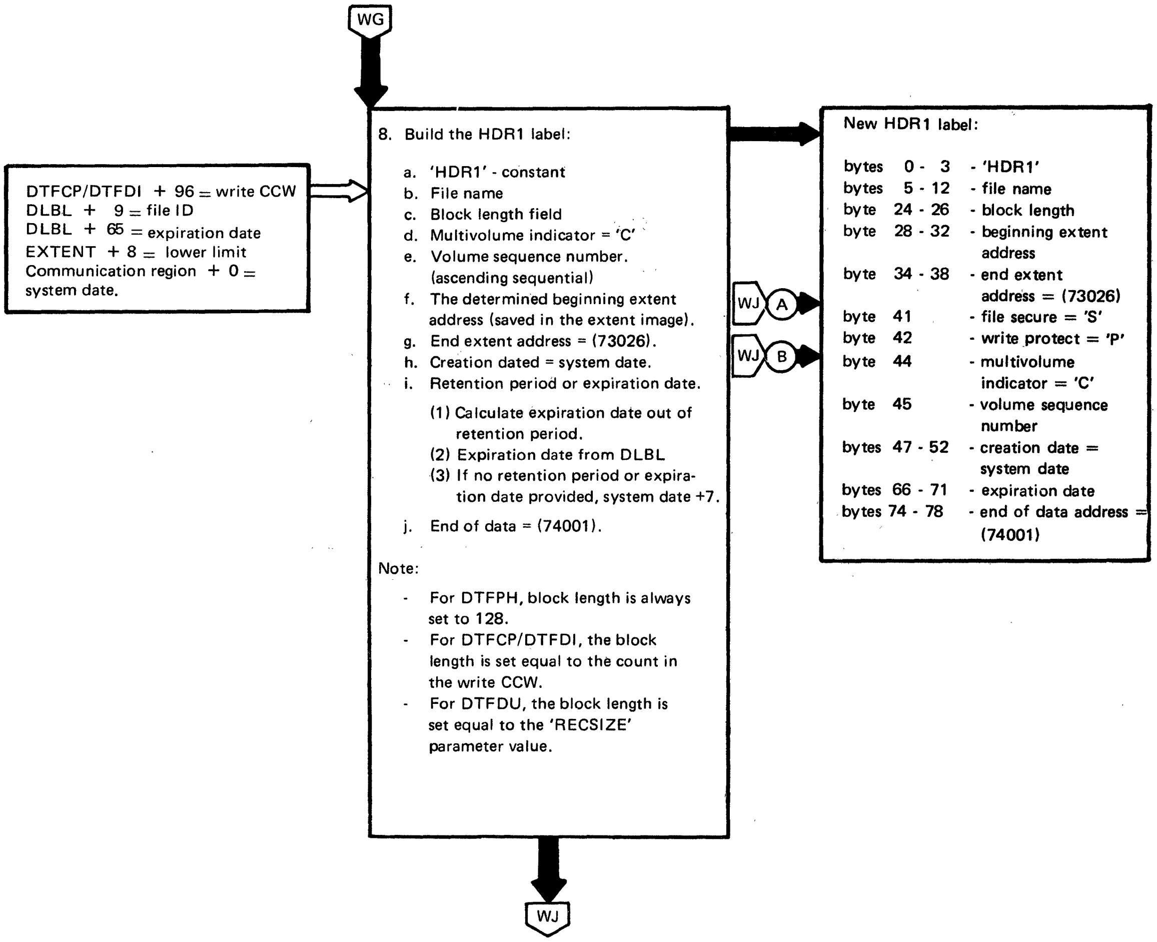
VOL1 label is updated to indicate a secured volume. (VOL1 + 10)

Blank build area.

4400I no space in VTOC
4459I invalid extent

Chart WH. $$BODIO3: Diskette Open Output, Create/Write New HDR1 Label (Part 2 of 3)

Input | Processing | Output



DTFCP/DTFDI + 96 = write CCW
DLBL + 9 = file ID
DLBL + 65 = expiration date
EXTENT + 8 = lower limit
Communication region + 0 = system date.

8. Build the HDR1 label:

a. 'HDR1' - constant
b. File name
c. Block length field
d. Multivolume indicator = 'C'
e. Volume sequence number. (ascending sequential)
f. The determined beginning extent address (saved in the extent image).
g. End extent address = (73026).
h. Creation dated = system date.
i. Retention period or expiration date.
   (1) Calculate expiration date out of retention period.
   (2) Expiration date from DLBL
   (3) If no retention period or expiration date provided, system date +7.
j. End of data = (74001).

Note:

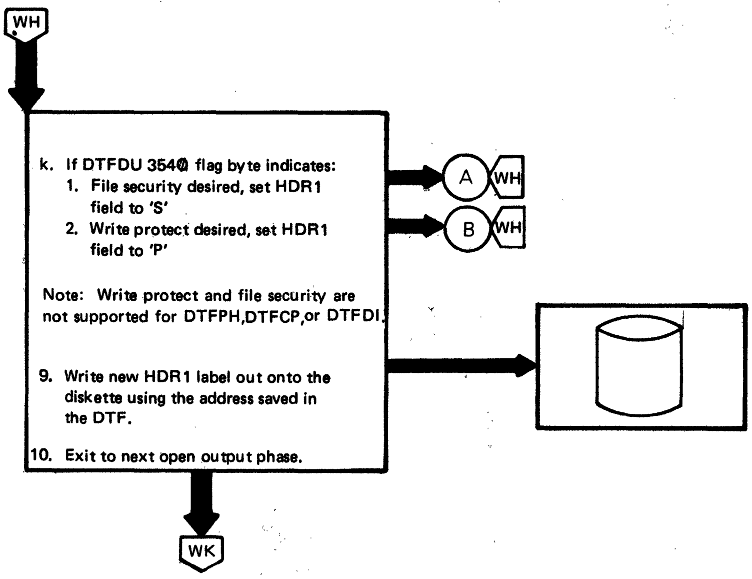
- For DTFPH, block length is always set to 128.
- For DTFCP/DTFDI, the block length is set equal to the count in the write CCW.
- For DTFDU, the block length is set equal to the 'RECSIZE' parameter value.

New HDR1 label:

bytes	0 - 3	- 'HDR1'
bytes	5 - 12	- file name
byte	24 - 26	- block length
byte	28 - 32	- beginning extent address
byte	34 - 38	- end extent address = (73026)
byte	41	- file secure = 'S'
byte	42	- write protect = 'P'
byte	44	- multivolume indicator = 'C'
byte	45	- volume sequence number
bytes	47 - 52	- creation date = system date
bytes	66 - 71	- expiration date
bytes	74 - 78	- end of data address = (74001)

Chart WJ. $$BODIO3: Diskette Open Output, Create/Write New HDR^1 Label (Part 3 of 3)

Input Processing Output



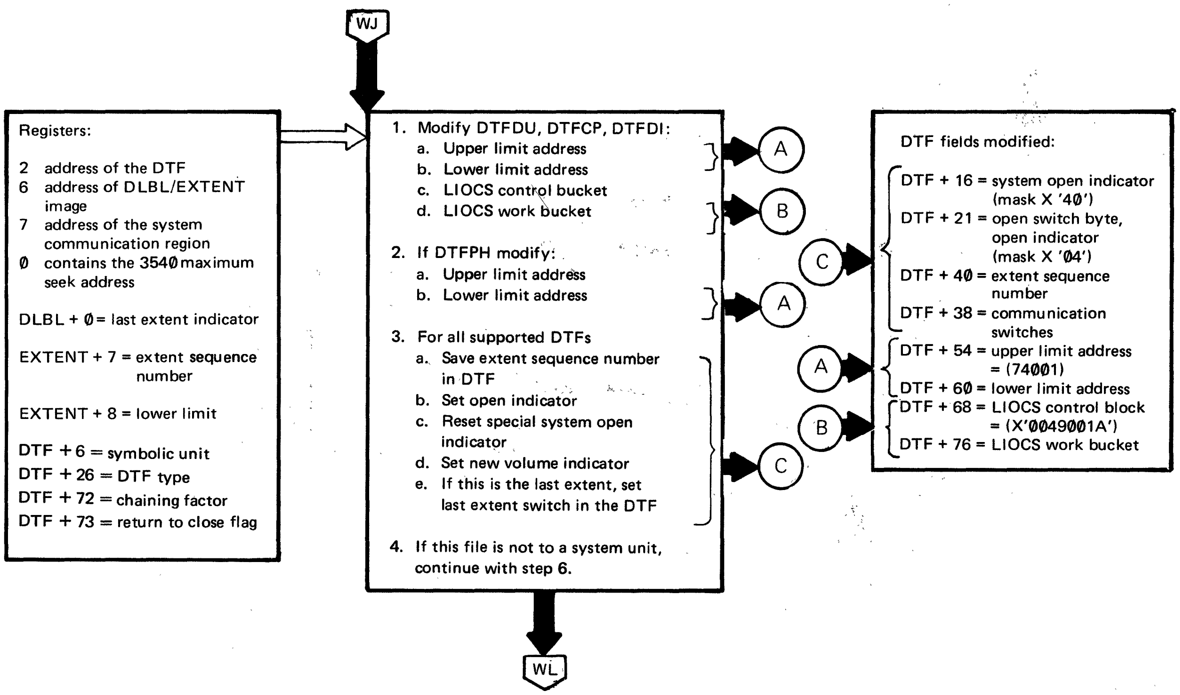
k. If DTFDU 3540 flag byte indicates:
1. File security desired, set HDR1 field to 'S'
2. Write protect desired, set HDR1 field to 'P'

Note: Write protect and file security are not supported for DTFPH,DTFCP,or DTFDI.

9. Write new HDR1 label out onto the diskette using the address saved in the DTF.

10. Exit to next open output phase.

Chart WK. $$BODIO7: Diskette Open Output, Initialize DTF Table (Part 1 of 2)



Registers:

2 address of the DTF
6 address of DLBL/EXTENT image
7 address of the system communication region
Ø contains the 354Ø maximum seek address

DLBL + Ø = last extent indicator

EXTENT + 7 = extent sequence number

EXTENT + 8 = lower limit

DTF + 6 = symbolic unit
DTF + 26 = DTF type
DTF + 72 = chaining factor
DTF + 73 = return to close flag

1. Modify DTFDU, DTFCP, DTFDI:
   a. Upper limit address
   b. Lower limit address
   c. LIOCS control bucket
   d. LIOCS work bucket

2. If DTFPH modify:
   a. Upper limit address
   b. Lower limit address

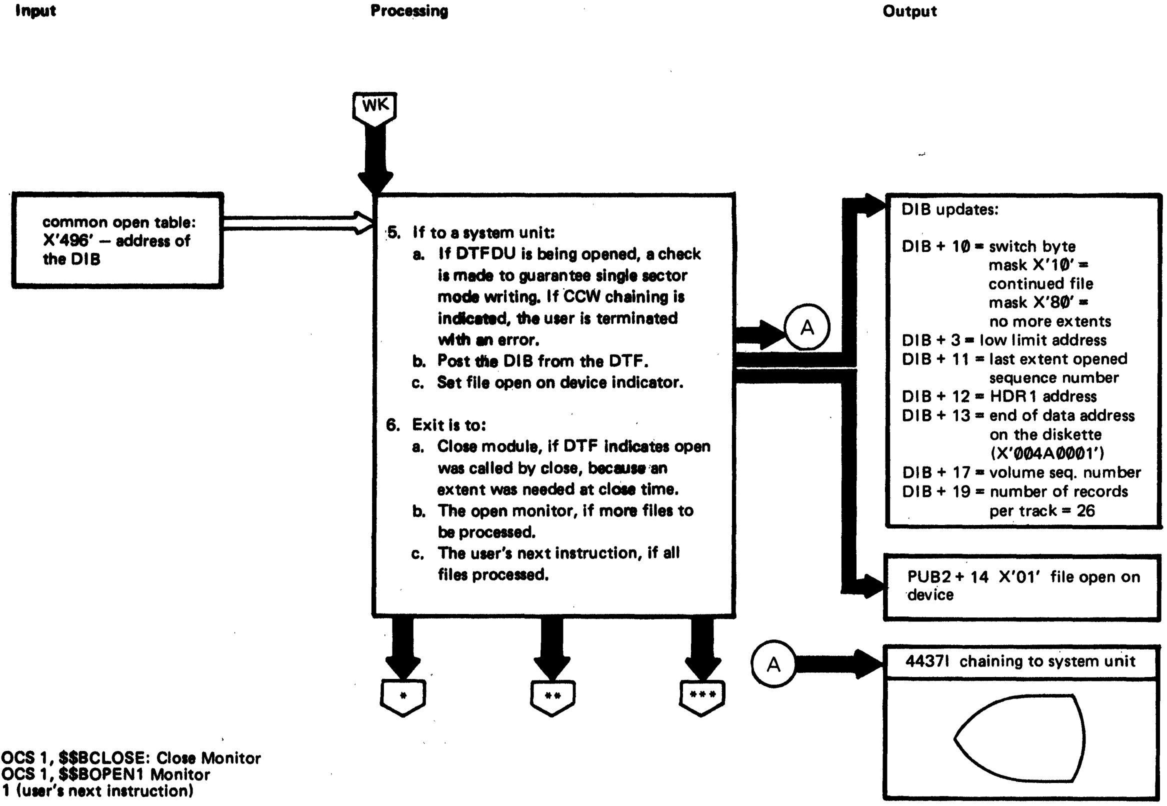
3. For all supported DTFs
   a. Save extent sequence number in DTF
   b. Set open indicator
   c. Reset special system open indicator
   d. Set new volume indicator
   e. If this is the last extent, set last extent switch in the DTF

4. If this file is not to a system unit, continue with step 6.

DTF fields modified:

DTF + 16 = system open indicator (mask X '4Ø')
DTF + 21 = open switch byte, open indicator (mask X 'Ø4')
DTF + 4Ø = extent sequence number
DTF + 38 = communication switches
DTF + 54 = upper limit address = (74ØØ1)
DTF + 6Ø = lower limit address
DTF + 68 = LIOCS control block = (X'ØØ49ØØ1A')
DTF + 76 = LIOCS work bucket

Chart WL. $$BODIO7: Diskette Open Output, Initialize DTF Table (Part 2 of 2)



common open table: X'496' – address of the DIB

5. If to a system unit:
   a. If DTFDU is being opened, a check is made to guarantee single sector mode writing. If CCW chaining is indicated, the user is terminated with an error.
   b. Post the DIB from the DTF.
   c. Set file open on device indicator.

6. Exit is to:
   a. Close module, if DTF indicates open was called by close, because an extent was needed at close time.
   b. The open monitor, if more files to be processed.
   c. The user's next instruction, if all files processed.

DIB updates:

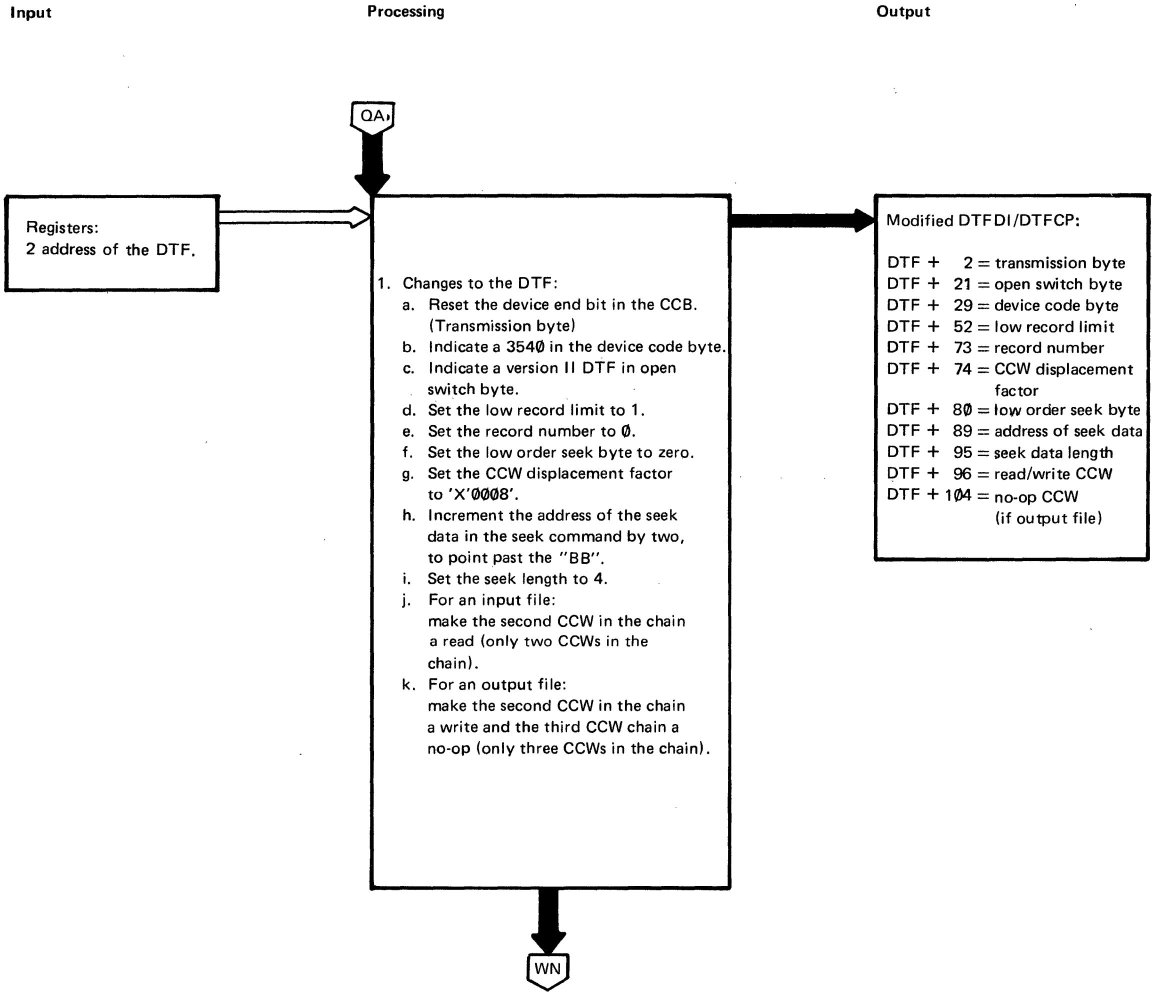
DIB + 1Ø = switch byte
mask X'1Ø' = continued file
mask X'8Ø' = no more extents
DIB + 3 = low limit address
DIB + 11 = last extent opened sequence number
DIB + 12 = HDR1 address
DIB + 13 = end of data address on the diskette (X'ØØ4AØØØ1')
DIB + 17 = volume seq. number
DIB + 19 = number of records per track = 26

PUB2 + 14 X'01' file open on device

44371 chaining to system unit

* See LIOCS 1, $$BCLOSE: Close Monitor
** See LIOCS 1, $$BOPEN1 Monitor
*** SCV 11 (user's next instruction)

Chart WM. $$BCDUCP: Diskette DTFCP/DTFDI Open (Part 1 of 2)



Input

Registers:
2 address of the DTF.

Processing

1. Changes to the DTF:
   a. Reset the device end bit in the CCB. (Transmission byte)
   b. Indicate a 354Ø in the device code byte.
   c. Indicate a version II DTF in open switch byte.
   d. Set the low record limit to 1.
   e. Set the record number to Ø.
   f. Set the low order seek byte to zero.
   g. Set the CCW displacement factor to 'X'ØØØ8'.
   h. Increment the address of the seek data in the seek command by two, to point past the "BB".
   i. Set the seek length to 4.
   j. For an input file: make the second CCW in the chain a read (only two CCWs in the chain).
   k. For an output file: make the second CCW in the chain a write and the third CCW chain a no-op (only three CCWs in the chain).

Output

Modified DTFDI/DTFCP:

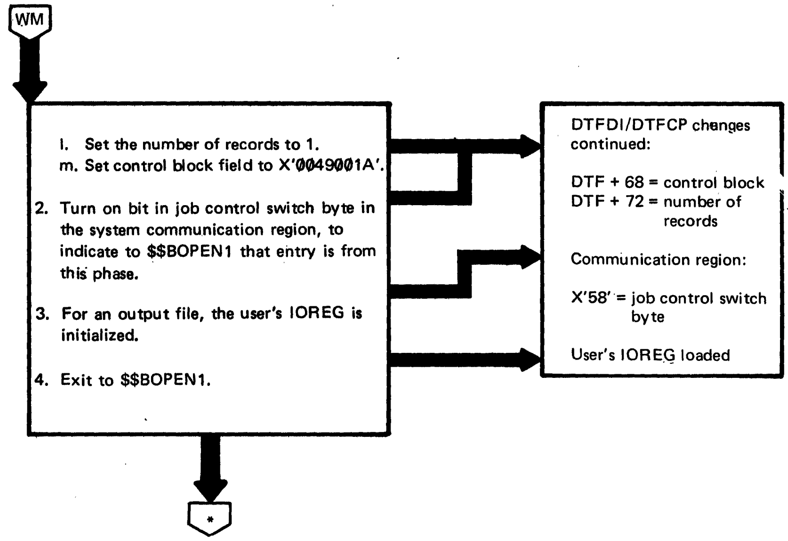
DTF + 2 =	transmission byte
DTF + 21 =	open switch byte
DTF + 29 =	device code byte
DTF + 52 =	low record limit
DTF + 73 =	record number
DTF + 74 =	CCW displacement factor
DTF + 8Ø =	low order seek byte
DTF + 89 =	address of seek data
DTF + 95 =	seek data length
DTF + 96 =	read/write CCW
DTF + 1Ø4 =	no-op CCW (if output file)

Chart WN. $$BODUCP: Diskette DTFCP/DTFDI Open (Part 2 of 2)

Input

Processing

Output



l. Set the number of records to 1.
m. Set control block field to X'0049001A'.

2. Turn on bit in job control switch byte in the system communication region, to indicate to $$BOPEN1 that entry is from this phase.

3. For an output file, the user's IOREG is initialized.

4. Exit to $$BOPEN1.

DTFDI/DTFCP changes continued:

DTF + 68 = control block
DTF + 72 = number of records

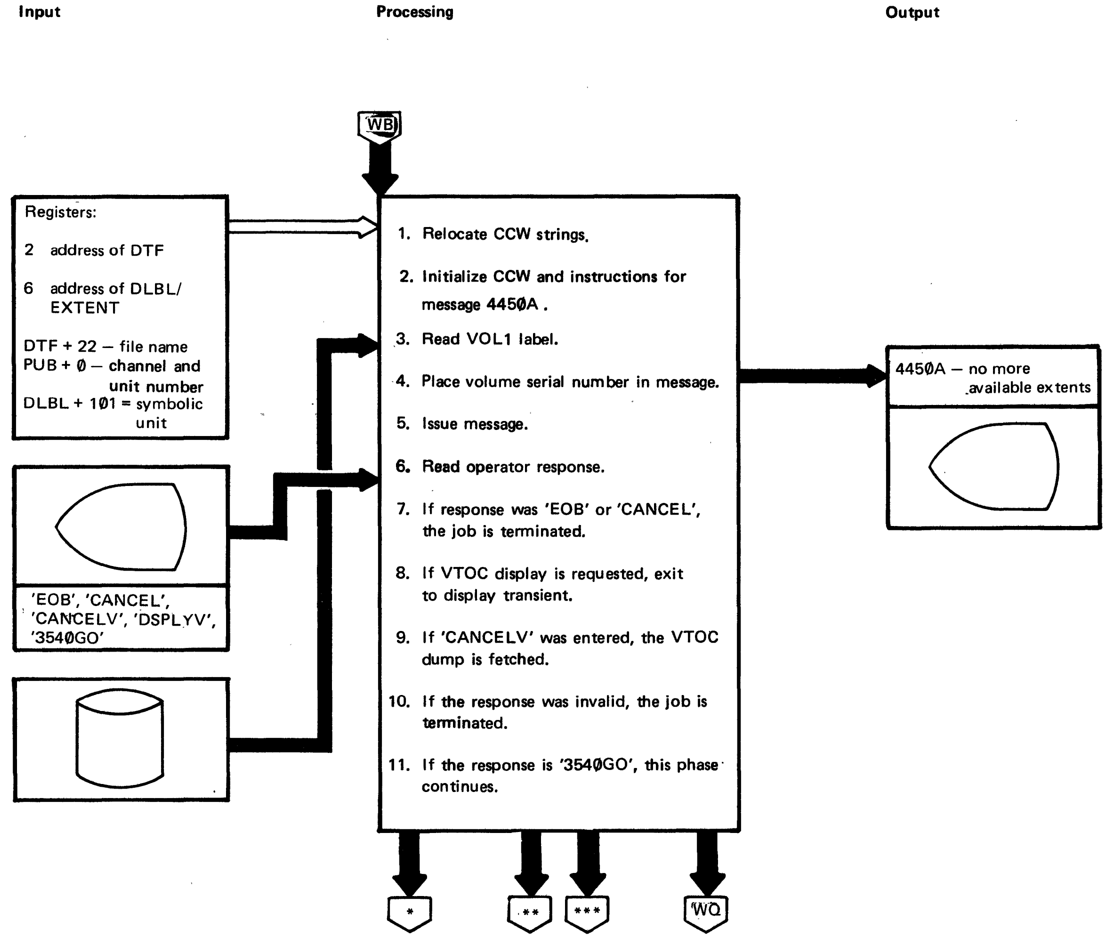
Communication region:

X'58' = job control switch byte

User's IOREG loaded

* See LIOCS 1, $$BOPEN1 Monitor

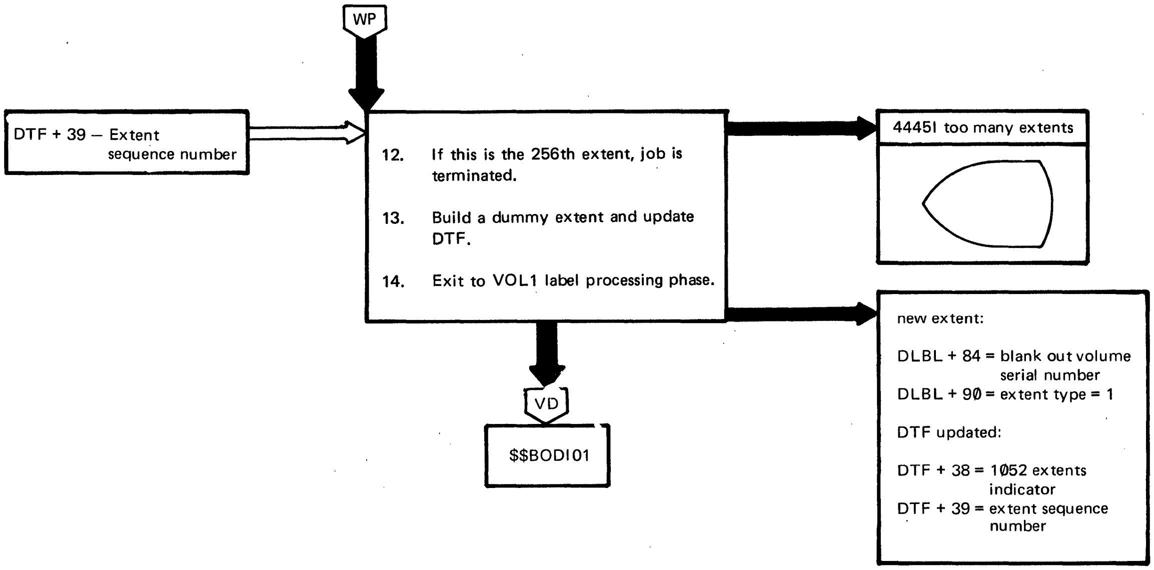
Chart WP. $$BODIO8: Diskette Open Output, Extents from Console (Part 1 of 2)


Input
Processing
Output
WB
Registers:
2 address of DTF
6 address of DLBL/ EXTENT
DTF + 22 – file name
PUB + Ø – channel and unit number
DLBL + 1Ø1 = symbolic unit
'EOB', 'CANCEL', 'CANCELV', 'DSPLYV', '354ØGO'
4450A – no more available extents
*
**
***
WQ

1. Relocate CCW strings.
2. Initialize CCW and instructions for message 445ØA.
3. Read VOL1 label.
4. Place volume serial number in message.
5. Issue message.
6. Read operator response.
7. If response was 'EOB' or 'CANCEL', the job is terminated.
8. If VTOC display is requested, exit to display transient.
9. If 'CANCELV' was entered, the VTOC dump is fetched.
10. If the response was invalid, the job is terminated.
11. If the response is '354ØGO', this phase continues.

* See LIOCS 1, $$BODSPV VTOC Display
** SVC 50 (program terminated)
*** See LIOCS 1, $$BOVDMO VTOC Dump

Chart WQ. $$BODI08: Diskette Open Output, Extents from Console (Part 2 of 2)



12. If this is the 256th extent, job is terminated.

13. Build a dummy extent and update DTF.

14. Exit to VOL1 label processing phase.

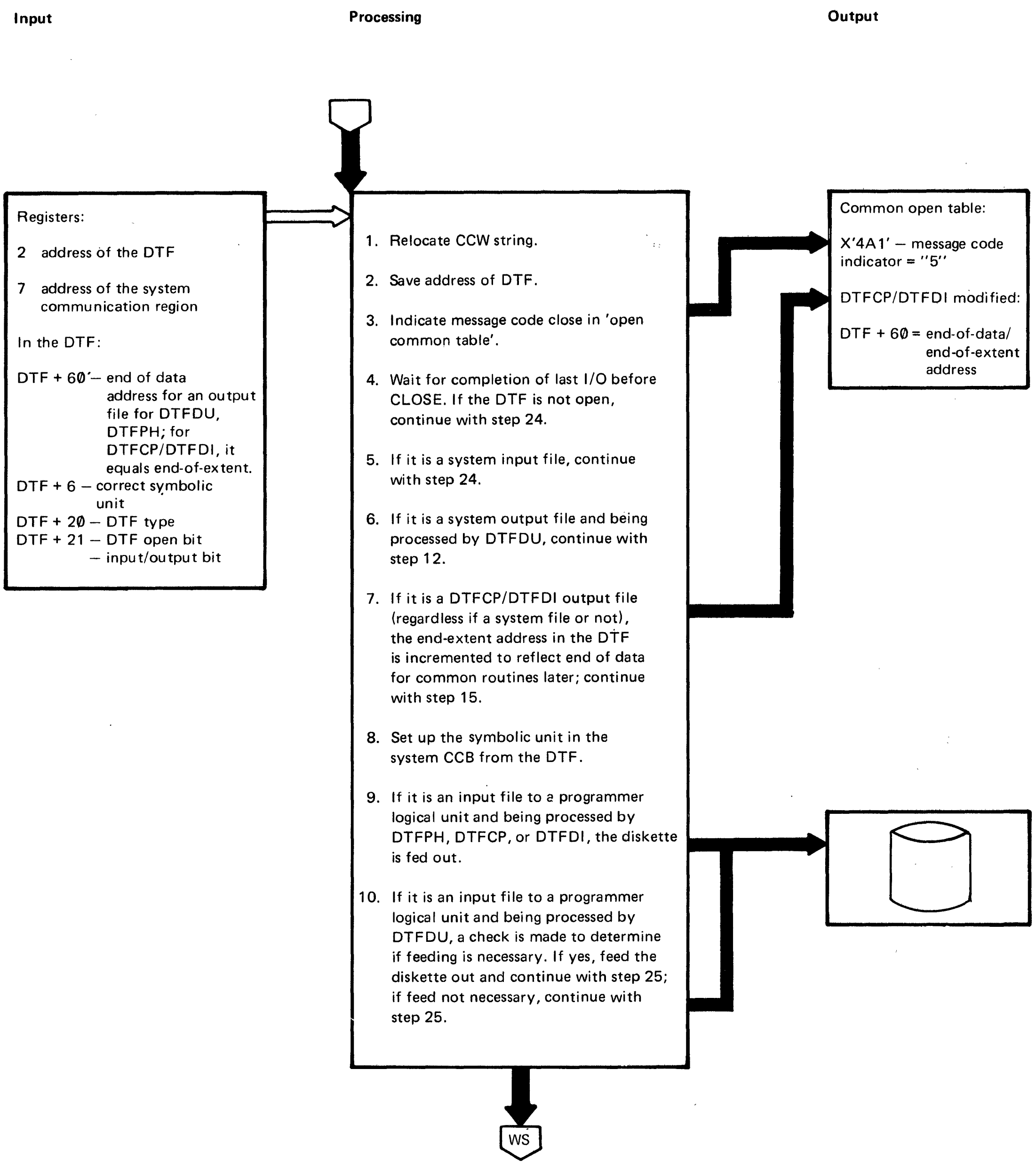
new extent:

DLBL + 84 = blank out volume serial number
DLBL + 90 = extent type = 1

DTF updated:

DTF + 38 = 1052 extents indicator
DTF + 39 = extent sequence number

Chart WR. $$BOCIO4: Diskette Close (Part 1 of 4)



Input

Registers:

2 address of the DTF

7 address of the system communication region

In the DTF:

DTF + 60 – end of data address for an output file for DTFDU, DTFPH; for DTFCP/DTFDI, it equals end-of-extent.
DTF + 6 – correct symbolic unit
DTF + 20 – DTF type
DTF + 21 – DTF open bit
– input/output bit

Processing

1. Relocate CCW string.
2. Save address of DTF.
3. Indicate message code close in 'open common table'.
4. Wait for completion of last I/O before CLOSE. If the DTF is not open, continue with step 24.
5. If it is a system input file, continue with step 24.
6. If it is a system output file and being processed by DTFDU, continue with step 12.
7. If it is a DTFCP/DTFDI output file (regardless if a system file or not), the end-extent address in the DTF is incremented to reflect end of data for common routines later; continue with step 15.
8. Set up the symbolic unit in the system CCB from the DTF.
9. If it is an input file to a programmer logical unit and being processed by DTFPH, DTFCP, or DTFDI, the diskette is fed out.
10. If it is an input file to a programmer logical unit and being processed by DTFDU, a check is made to determine if feeding is necessary. If yes, feed the diskette out and continue with step 25; if feed not necessary, continue with step 25.

Output

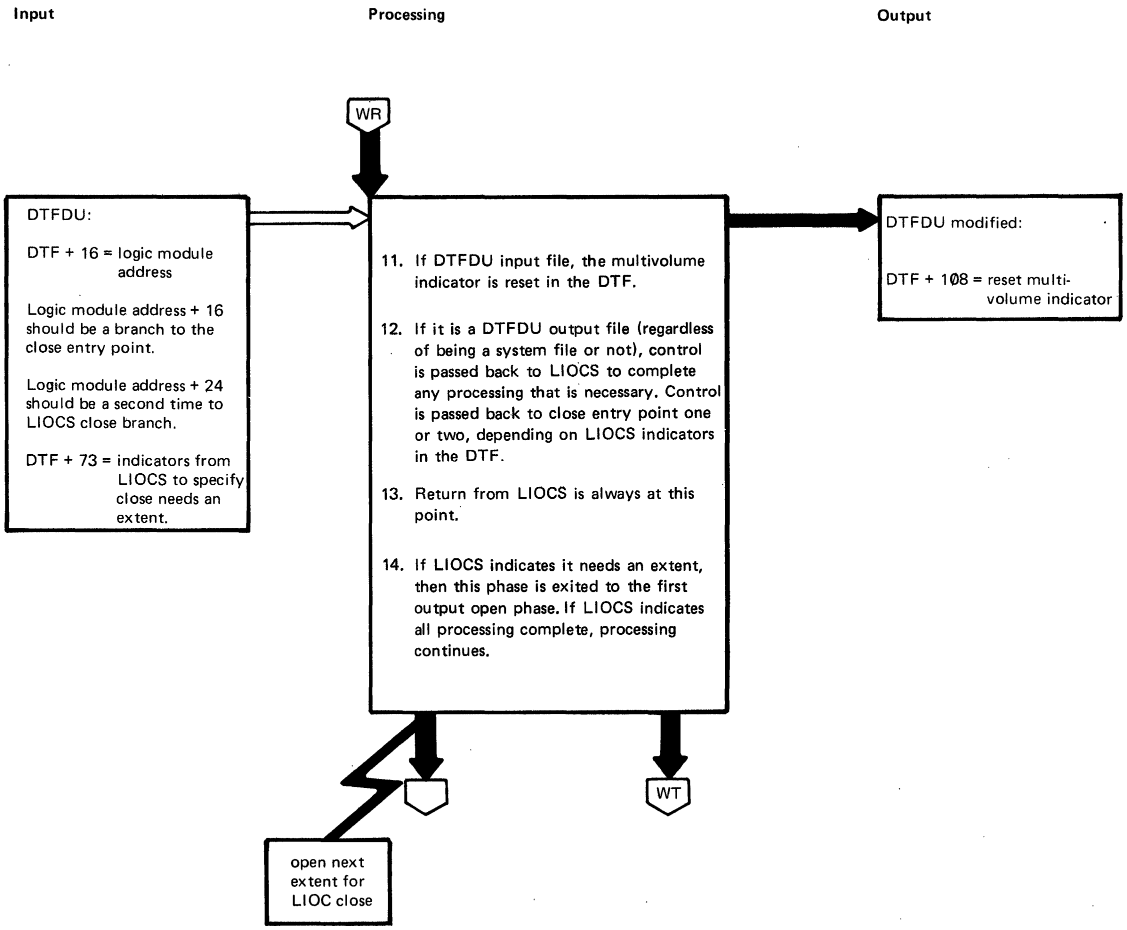
Common open table:

X'4A1' – message code indicator = "5"

DTFCP/DTFDI modified:

DTF + 60 = end-of-data/ end-of-extent address

Chart WS. $$BODIO4: Diskette Close (Part 2 of 4)



DTFDU:

DTF + 16 = logic module address

Logic module address + 16 should be a branch to the close entry point.

Logic module address + 24 should be a second time to LIOCS close branch.

DTF + 73 = indicators from LIOCS to specify close needs an extent.

11. If DTFDU input file, the multivolume indicator is reset in the DTF.

12. If it is a DTFDU output file (regardless of being a system file or not), control is passed back to LIOCS to complete any processing that is necessary. Control is passed back to close entry point one or two, depending on LIOCS indicators in the DTF.

13. Return from LIOCS is always at this point.

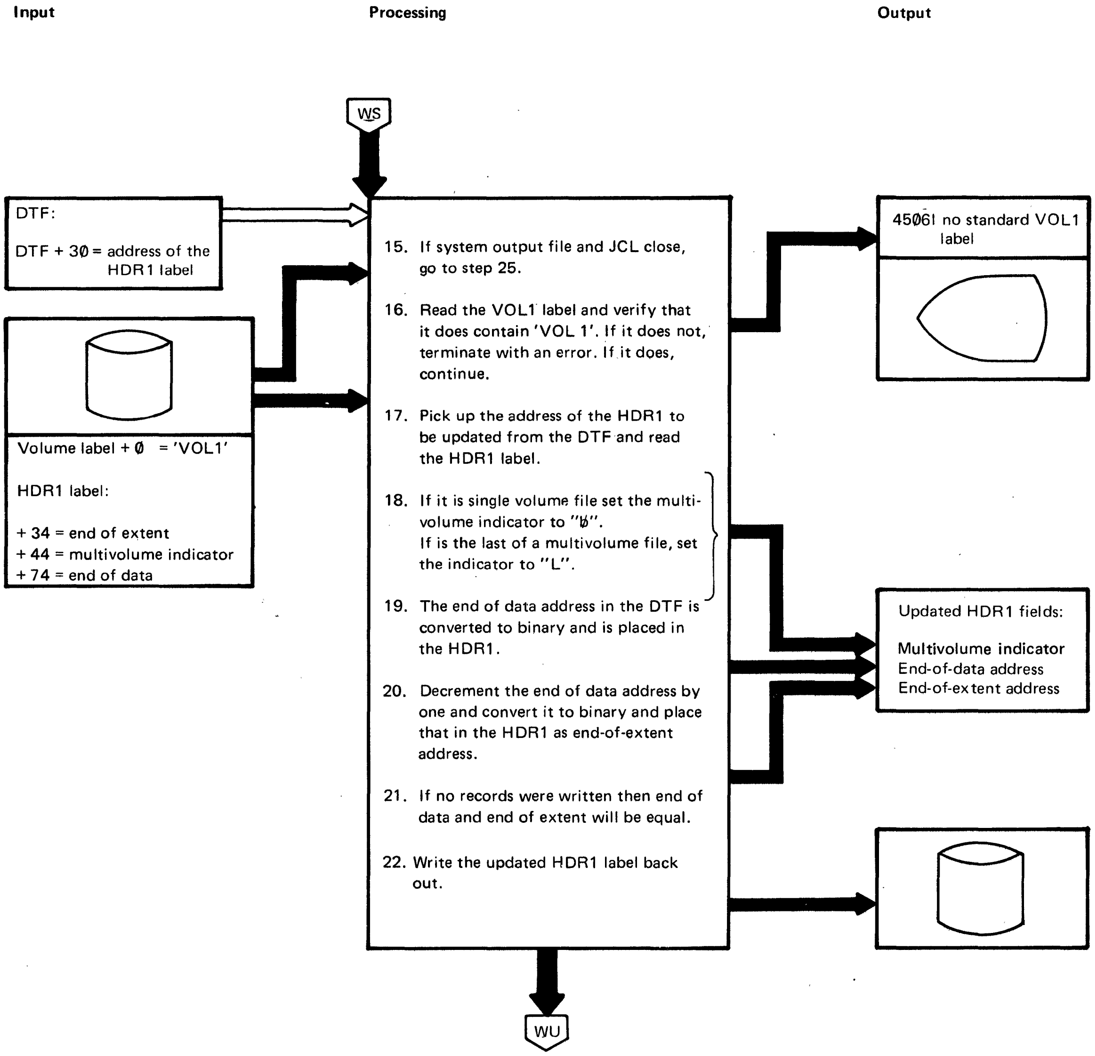
14. If LIOCS indicates it needs an extent, then this phase is exited to the first output open phase. If LIOCS indicates all processing complete, processing continues.

DTFDU modified:

DTF + 1Ø8 = reset multi-volume indicator

open next extent for LIOC close

Chart WT. $$BODIO4: Diskette Close (Part 3 of 4)



DTF:

DTF + 3Ø = address of the HDR1 label

Volume label + Ø = 'VOL1'

HDR1 label:

+ 34 = end of extent
+ 44 = multivolume indicator
+ 74 = end of data

15. If system output file and JCL close, go to step 25.

16. Read the VOL1 label and verify that it does contain 'VOL 1'. If it does not, terminate with an error. If it does, continue.

17. Pick up the address of the HDR1 to be updated from the DTF and read the HDR1 label.

18. If it is single volume file set the multivolume indicator to "b̸". If is the last of a multivolume file, set the indicator to "L".

19. The end of data address in the DTF is converted to binary and is placed in the HDR1.

20. Decrement the end of data address by one and convert it to binary and place that in the HDR1 as end-of-extent address.

21. If no records were written then end of data and end of extent will be equal.

22. Write the updated HDR1 label back out.

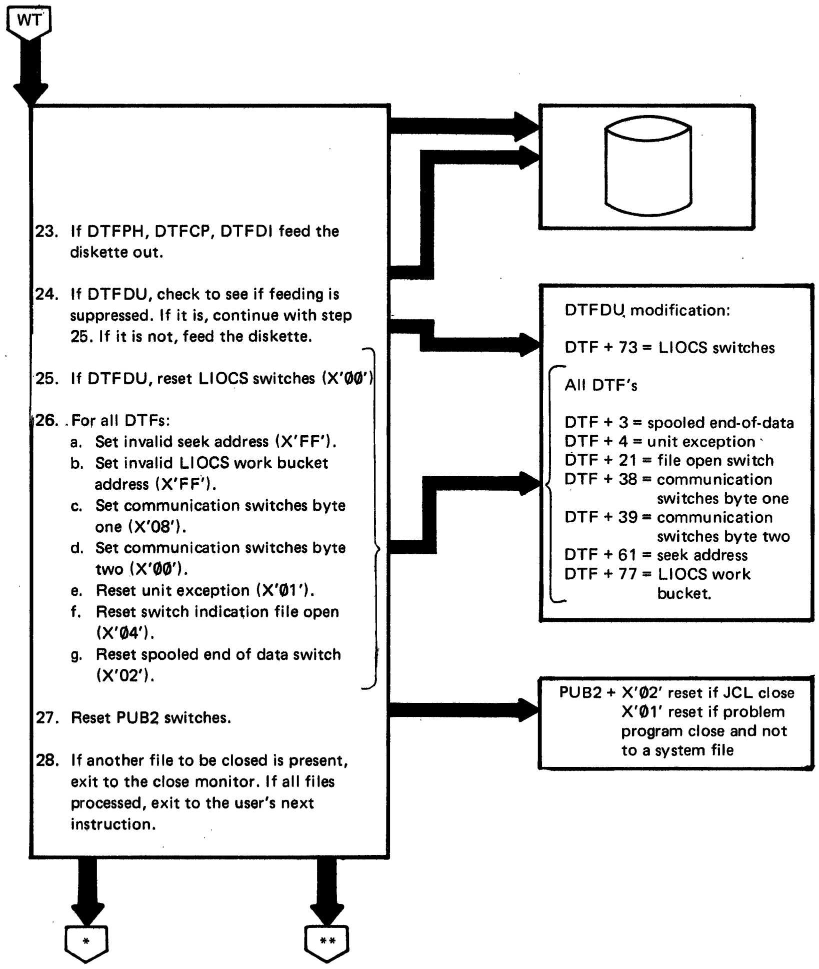
45Ø6I no standard VOL1 label

Updated HDR1 fields:

Multivolume indicator
End-of-data address
End-of-extent address

Chart WU. $$BCDIO4: Diskette Close (Part 4 of 4)

Input **Processing** **Output**



23. If DTFPH, DTFCP, DTFDI feed the diskette out.

24. If DTFDU, check to see if feeding is suppressed. If it is, continue with step 25. If it is not, feed the diskette.

25. If DTFDU, reset LIOCS switches (X'ØØ')

26. For all DTFs:
   a. Set invalid seek address (X'FF').
   b. Set invalid LIOCS work bucket address (X'FF').
   c. Set communication switches byte one (X'08').
   d. Set communication switches byte two (X'ØØ').
   e. Reset unit exception (X'Ø1').
   f. Reset switch indication file open (X'Ø4').
   g. Reset spooled end of data switch (X'02').

27. Reset PUB2 switches.

28. If another file to be closed is present, exit to the close monitor. If all files processed, exit to the user's next instruction.

DTFDU modification:

DTF + 73 = LIOCS switches

All DTF's

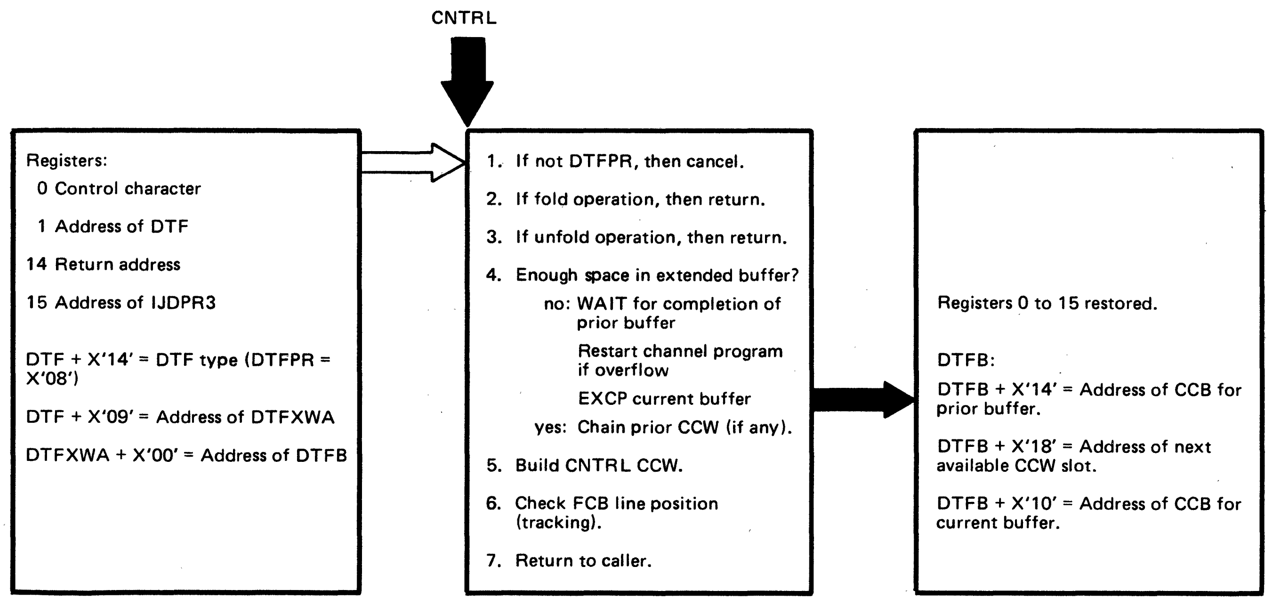
DTF + 3 = spooled end-of-data
DTF + 4 = unit exception
DTF + 21 = file open switch
DTF + 38 = communication switches byte one
DTF + 39 = communication switches byte two
DTF + 61 = seek address
DTF + 77 = LIOCS work bucket.

PUB2 + X'Ø2' reset if JCL close
X'Ø1' reset if problem program close and not to a system file

* See LIOCS 1, $$BCLOSE: Close Monitor
**SVC 11 (user's next instruction)

Chart YA. IJPDF3: CNTRL Macro

INPUT **PROCESSING** **OUTPUT**



CNTRL

Registers:
0 Control character
1 Address of DTF
14 Return address
15 Address of IJDPR3

DTF + X'14' = DTF type (DTFPR = X'08')

DTF + X'09' = Address of DTFXWA

DTFXWA + X'00' = Address of DTFB

1. If not DTFPR, then cancel.
2. If fold operation, then return.
3. If unfold operation, then return.
4. Enough space in extended buffer?
   - no: WAIT for completion of prior buffer
     Restart channel program if overflow
     EXCP current buffer
   - yes: Chain prior CCW (if any).
5. Build CNTRL CCW.
6. Check FCB line position (tracking).
7. Return to caller.

Registers 0 to 15 restored.

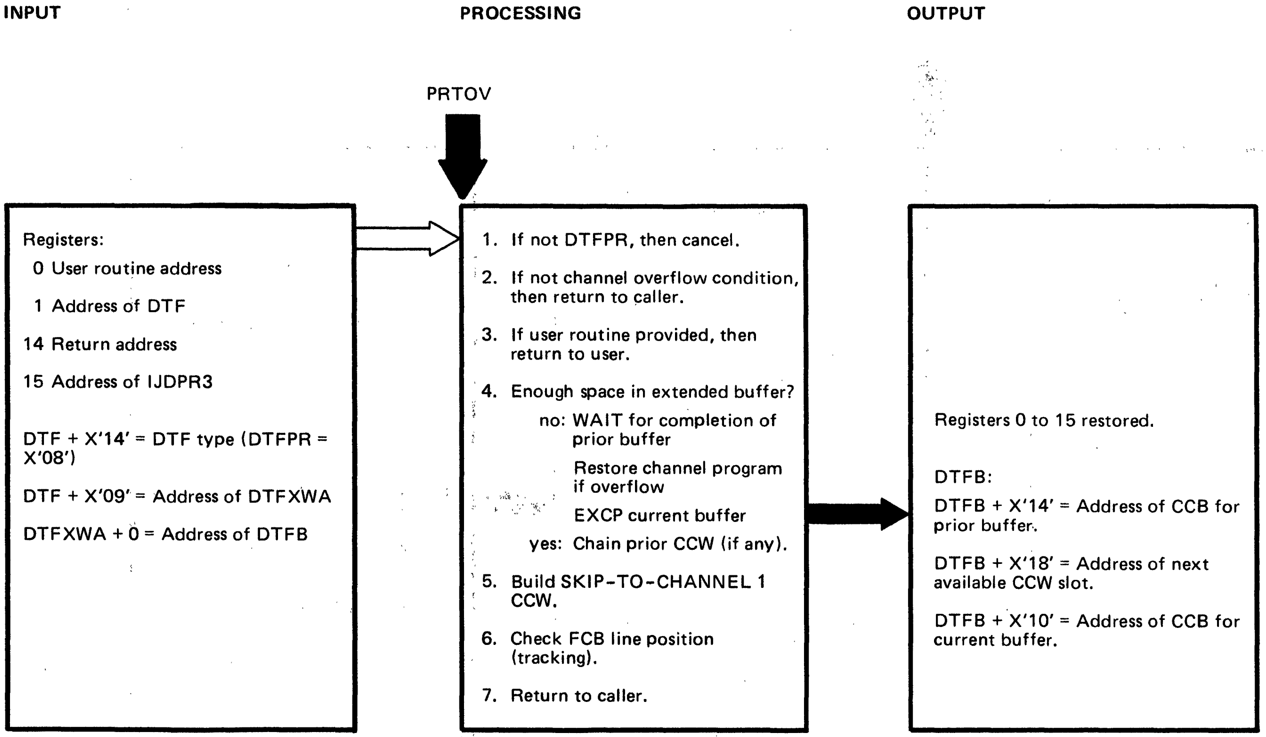
DTFB:

DTFB + X'14' = Address of CCB for prior buffer.

DTFB + X'18' = Address of next available CCW slot.

DTFB + X'10' = Address of CCB for current buffer.

Chart XB. IJDPR3: PRTOV Macro



INPUT

PROCESSING

OUTPUT

PRTOV

Registers:

0 User routine address

1 Address of DTF

14 Return address

15 Address of IJDPR3

DTF + X'14' = DTF type (DTFPR = X'08')

DTF + X'09' = Address of DTFXWA

DTFXWA + 0 = Address of DTFB

1. If not DTFPR, then cancel.
2. If not channel overflow condition, then return to caller.
3. If user routine provided, then return to user.
4. Enough space in extended buffer?
   - no: WAIT for completion of prior buffer
     Restore channel program if overflow
     EXCP current buffer
   - yes: Chain prior CCW (if any).
5. Build SKIP-TO-CHANNEL 1 CCW.
6. Check FCB line position (tracking).
7. Return to caller.

Registers 0 to 15 restored.

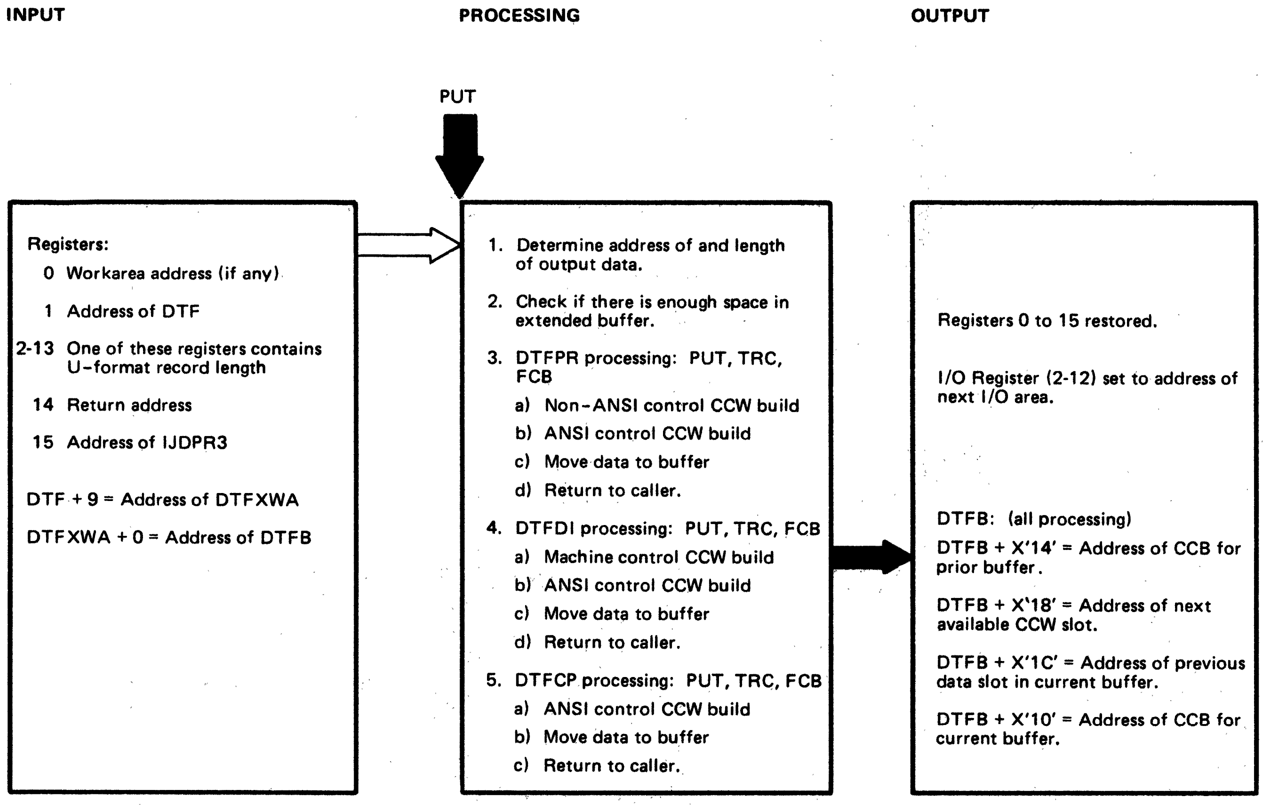
DTFB:

DTFB + X'14' = Address of CCB for prior buffer.

DTFB + X'18' = Address of next available CCW slot.

DTFB + X'10' = Address of CCB for current buffer.

Chart XC. IJPDR3: PUT Macro



INPUT

Registers:

0 Workarea address (if any)

1 Address of DTF

2-13 One of these registers contains U-format record length

14 Return address

15 Address of IJDPR3

DTF + 9 = Address of DTFXWA

DTFXWA + 0 = Address of DTFB

PROCESSING

PUT

1. Determine address of and length of output data.
2. Check if there is enough space in extended buffer.
3. DTFPR processing: PUT, TRC, FCB
   a) Non-ANSI control CCW build
   b) ANSI control CCW build
   c) Move data to buffer
   d) Return to caller.
4. DTFDI processing: PUT, TRC, FCB
   a) Machine control CCW build
   b) ANSI control CCW build
   c) Move data to buffer
   d) Return to caller.
5. DTFCP processing: PUT, TRC, FCB
   a) ANSI control CCW build
   b) Move data to buffer
   c) Return to caller.

OUTPUT

Registers 0 to 15 restored.

I/O Register (2-12) set to address of next I/O area.

DTFB: (all processing)

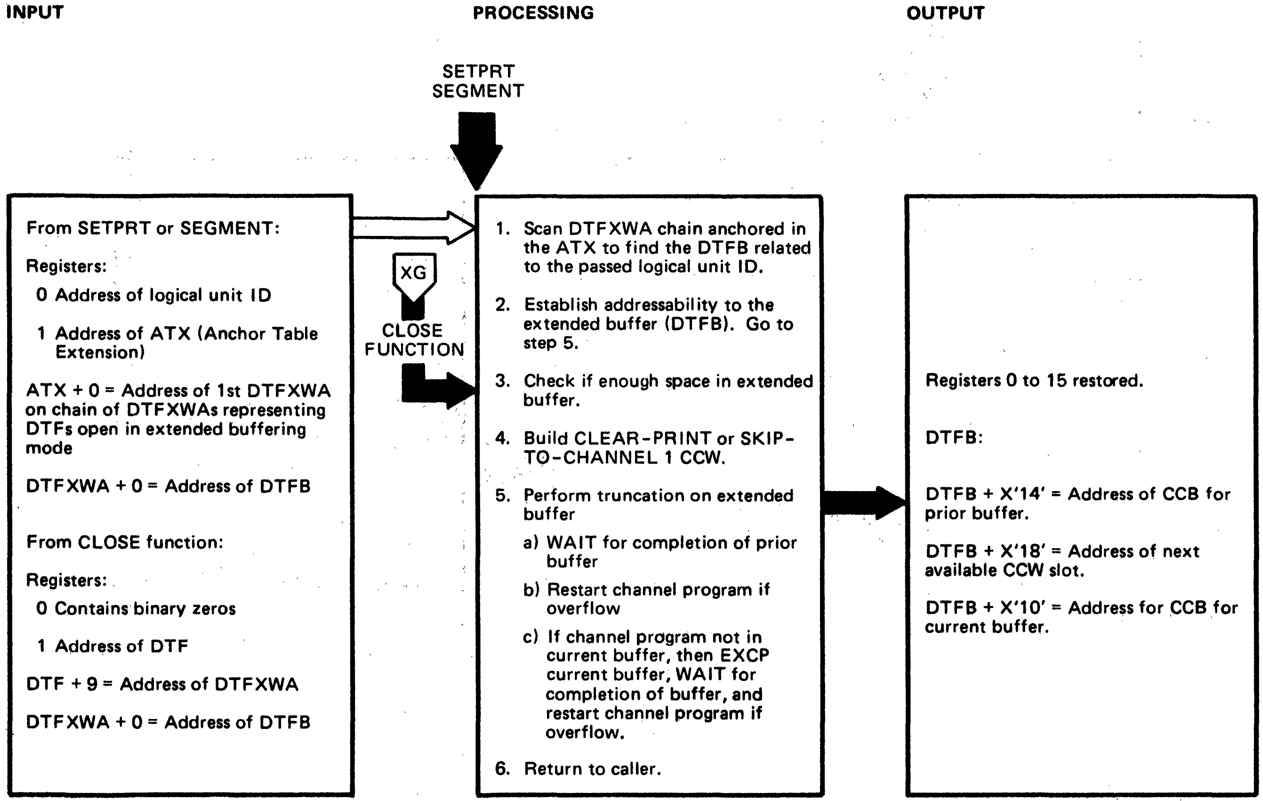
DTFB + X'14' = Address of CCB for prior buffer.

DTFB + X'18' = Address of next available CCW slot.

DTFB + X'1C' = Address of previous data slot in current buffer.

DTFB + X'10' = Address of CCB for current buffer.

Chart XD. IJDFF3: Truncation



INPUT

From SETPRT or SEGMENT:

Registers:

- 0 Address of logical unit ID
- 1 Address of ATX (Anchor Table Extension)

ATX + 0 = Address of 1st DTFXWA on chain of DTFXWAs representing DTFs open in extended buffering mode

DTFXWA + 0 = Address of DTFB

From CLOSE function:

Registers:

- 0 Contains binary zeros
- 1 Address of DTF

DTF + 9 = Address of DTFXWA

DTFXWA + 0 = Address of DTFB

PROCESSING

1. Scan DTFXWA chain anchored in the ATX to find the DTFB related to the passed logical unit ID.
2. Establish addressability to the extended buffer (DTFB). Go to step 5.
3. Check if enough space in extended buffer.
4. Build CLEAR-PRINT or SKIP-TO-CHANNEL 1 CCW.
5. Perform truncation on extended buffer
   a) WAIT for completion of prior buffer
   b) Restart channel program if overflow
   c) If channel program not in current buffer, then EXCP current buffer, WAIT for completion of buffer, and restart channel program if overflow.
6. Return to caller.

OUTPUT

Registers 0 to 15 restored.

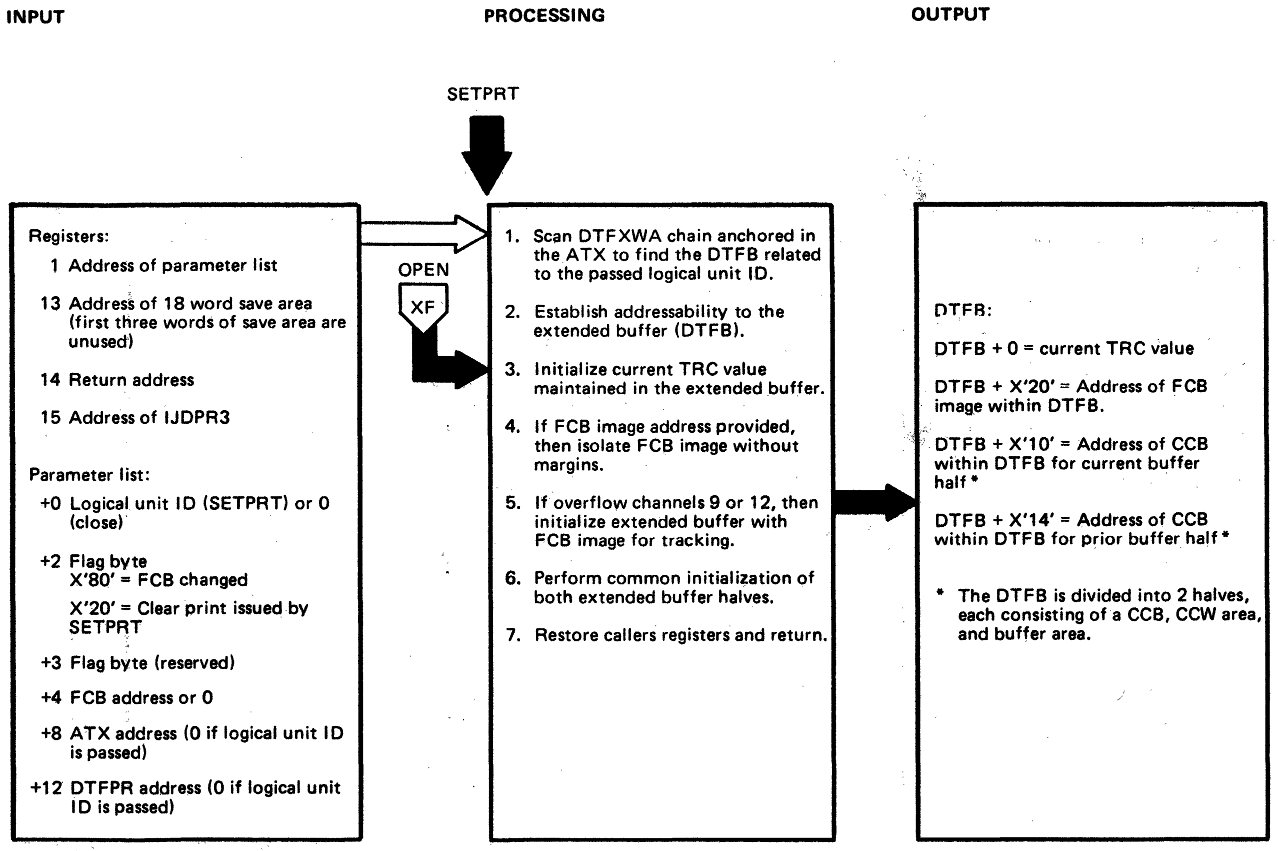
DTFB:

DTFB + X'14' = Address of CCB for prior buffer.

DTFB + X'18' = Address of next available CCW slot.

DTFB + X'10' = Address for CCB for current buffer.

Chart XE. IJDPR3: TRC/FCB UPDATE



INPUT

PROCESSING

OUTPUT

SETPRT

OPEN

XF

Registers:

1 Address of parameter list

13 Address of 18 word save area (first three words of save area are unused)

14 Return address

15 Address of IJDPR3

Parameter list:

+0 Logical unit ID (SETPRT) or 0 (close)

+2 Flag byte
X'80' = FCB changed
X'20' = Clear print issued by SETPRT

+3 Flag byte (reserved)

+4 FCB address or 0

+8 ATX address (0 if logical unit ID is passed)

+12 DTFPR address (0 if logical unit ID is passed)

1. Scan DTFXWA chain anchored in the ATX to find the DTFB related to the passed logical unit ID.
2. Establish addressability to the extended buffer (DTFB).
3. Initialize current TRC value maintained in the extended buffer.
4. If FCB image address provided, then isolate FCB image without margins.
5. If overflow channels 9 or 12, then initialize extended buffer with FCB image for tracking.
6. Perform common initialization of both extended buffer halves.
7. Restore callers registers and return.

DTFB:

DTFB + 0 = current TRC value

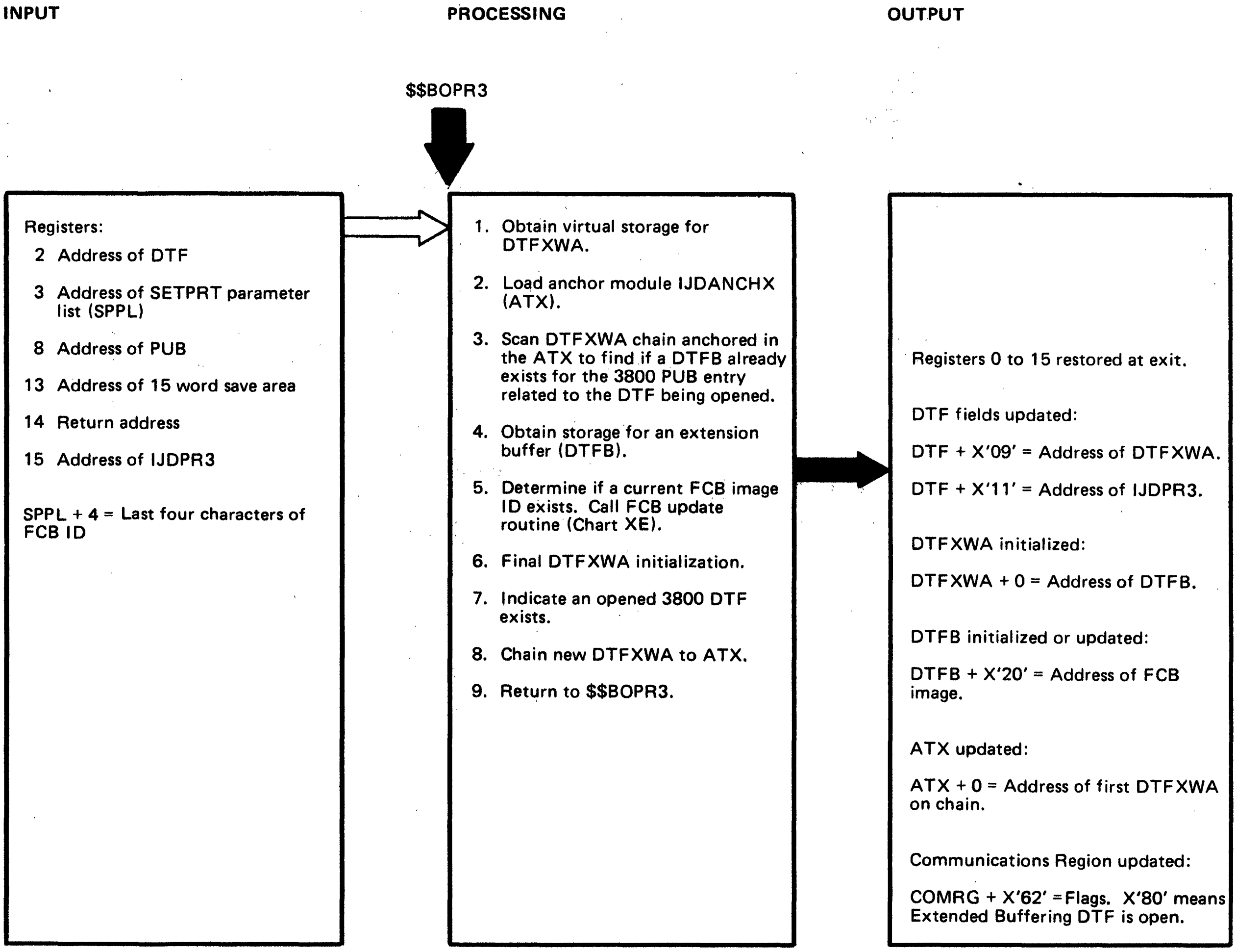
DTFB + X'20' = Address of FCB image within DTFB.

DTFB + X'10' = Address of CCB within DTFB for current buffer half *

DTFB + X'14' = Address of CCB within DTFB for prior buffer half *

* The DTFB is divided into 2 halves, each consisting of a CCB, CCW area, and buffer area.

Chart XF. IJDPR3: OPEN INIT



INPUT

Registers:
- 2 Address of DTF
- 3 Address of SETPRT parameter list (SPPL)
- 8 Address of PUB
- 13 Address of 15 word save area
- 14 Return address
- 15 Address of IJDPR3

SPPL + 4 = Last four characters of FCB ID

PROCESSING

$$BOPR3

1. Obtain virtual storage for DTFXWA.
2. Load anchor module IJDANCHX (ATX).
3. Scan DTFXWA chain anchored in the ATX to find if a DTFB already exists for the 3800 PUB entry related to the DTF being opened.
4. Obtain storage for an extension buffer (DTFB).
5. Determine if a current FCB image ID exists. Call FCB update routine (Chart XE).
6. Final DTFXWA initialization.
7. Indicate an opened 3800 DTF exists.
8. Chain new DTFXWA to ATX.
9. Return to $$BOPR3.

OUTPUT

Registers 0 to 15 restored at exit.

DTF fields updated:

DTF + X'09' = Address of DTFXWA.

DTF + X'11' = Address of IJDPR3.

DTFXWA initialized:

DTFXWA + 0 = Address of DTFB.

DTFB initialized or updated:

DTFB + X'20' = Address of FCB image.

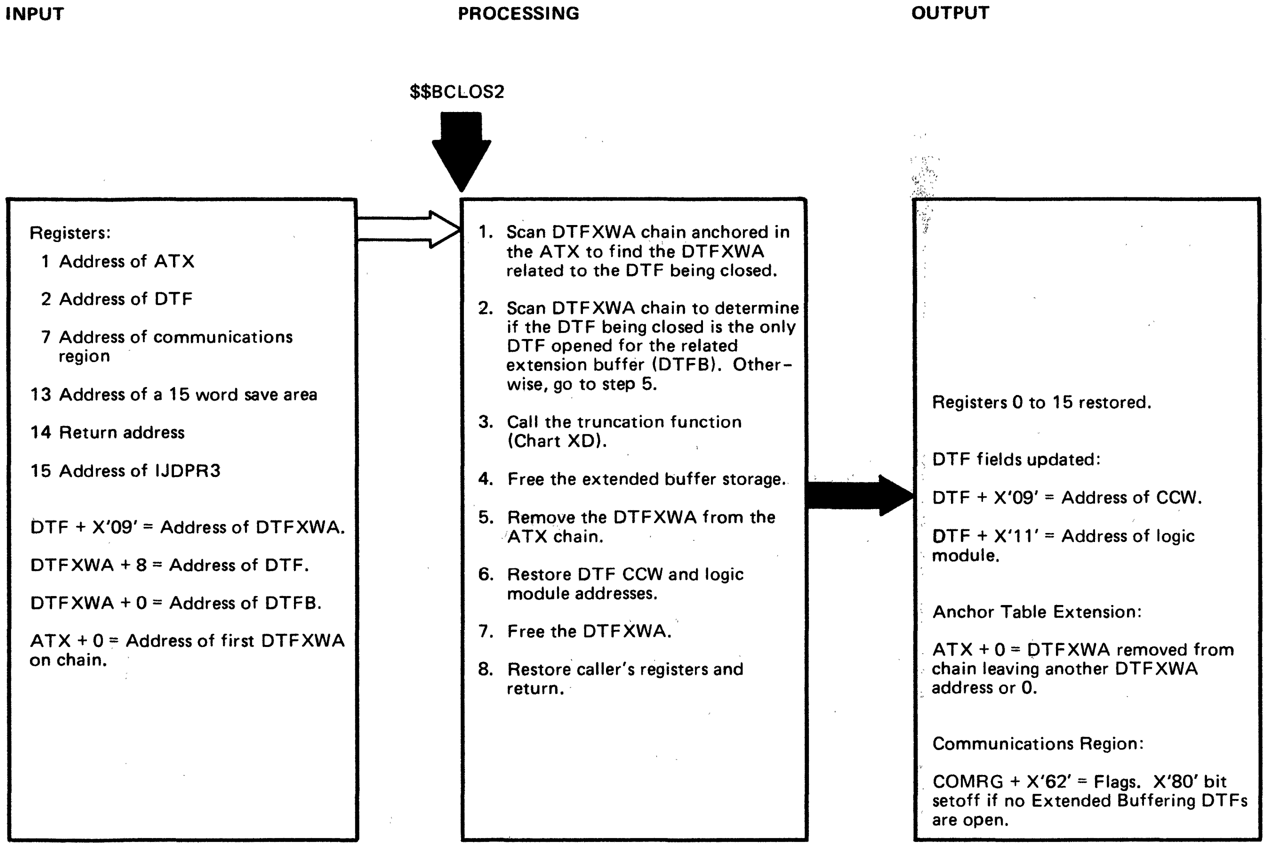
ATX updated:

ATX + 0 = Address of first DTFXWA on chain.

Communications Region updated:

COMRG + X'62' = Flags. X'80' means Extended Buffering DTF is open.

Chart XG. IJDPR3: CLOSE



INPUT

Registers:
1 Address of ATX
2 Address of DTF
7 Address of communications region
13 Address of a 15 word save area
14 Return address
15 Address of IJDPR3

DTF + X'09' = Address of DTFXWA.

DTFXWA + 8 = Address of DTF.

DTFXWA + 0 = Address of DTFB.

ATX + 0 = Address of first DTFXWA on chain.

PROCESSING

$$BCLOS2

1. Scan DTFXWA chain anchored in the ATX to find the DTFXWA related to the DTF being closed.
2. Scan DTFXWA chain to determine if the DTF being closed is the only DTF opened for the related extension buffer (DTFB). Otherwise, go to step 5.
3. Call the truncation function (Chart XD).
4. Free the extended buffer storage.
5. Remove the DTFXWA from the ATX chain.
6. Restore DTF CCW and logic module addresses.
7. Free the DTFXWA.
8. Restore caller's registers and return.

OUTPUT

Registers 0 to 15 restored.

DTF fields updated:

DTF + X'09' = Address of CCW.

DTF + X'11' = Address of logic module.

Anchor Table Extension:

ATX + 0 = DTFXWA removed from chain leaving another DTFXWA address or 0.

Communications Region:

COMRG + X'62' = Flags. X'80' bit setoff if no Extended Buffering DTFs are open.

APFENDIX A: IAEEL CRCSS-FEFERENCE LIST

.BK4	CG
.MF114	AM
.P38	CE
.P380	CD
.P44	CD
.P46	CE
.P47	CE
.P790	CE
.P81	CE
ABCRTJCE	FK, FM
ABOVEFE	AE
ACTMSG	FK, FL
ADJINPUT	SA
ADJNCECF	SA
AFTERSDR	EC
AGAIN	FM
AJALT	QC, QD
ALTCRM	FH
AUTOTERM	FM
BACKSPC2	ED, EF
BEGINRTN	PG, SC
BELCWFE	AE
BKSP	RH
BKSPACE	FJ
BUILDLBL	EF, RJ, SA
BYPASS	EF
BYSTDUTI	SA
CALLERP	SC
CALLMSG	QA
CALLMSG1	QE, QF
CALOPN	QE
CALOPN1	QE, QF
CBDUMP	EI
CCPTAPE	SA
CD25202	SD
CHECK	BE, EC, RJ, SA
CHECKDAT	EH
CHECKID	EA, EC
CHKALC	FE, FH
CHKFORM	AC
CHKIPT	EI
CHKLAB	RA, RD
CHKLP	FE, RE
CHKMOR	QA, QB, QD, QF
CHKNUMB	AD
CHKPCH	AA
CHKPRTR	AA
CHKRDR	AA
CHKTM	RA, RD
CHKTPM	EK
CHMOR	QC
CHRCARD	AC
CKERRCR	SC
CKMGTP	QA, QB, QE, QF
CKNXJB	FE, FH
CKPTR	QA, QB, QC, QF
CKRDR	QA, QB, QC
CKRST	QA, QC, QD
CKUDSM	EK
CLEAR	BE
CLEARREG	FC
CLOSECD	SD
CLOSEPT	SC
COBOLWRT	EF
COLPCINT	AC
COMMON	PJ
COMPARE	ED, RH
CONTINUE	EH, FD
CONTINU0	EA
CONTINU1	EA
CONTINU2	EB
CONTINU3	EB
CONTINU4	EB
CONTINU5	EB
CONT1	PG, SD
CONT2	PG, SD
CONT3	PG, SC
CPDUMP	EL
CPTAPE	RK
CPTAPEO	RD, RG
CP2520	SC
CUNAD	BF
DASDCHK	QA
DENSITY	EL
DETERM	RA, RD, SA
DETERMN	FK, FM
DETPATH	RA, RD, SA
DETPATH1	RA, RD
DISKERR	PG
DISKS	QA, QD
DISK2	SD
DISK5	SC
DONTBUMP	FB
DOREAD	RB, RE
DOREWIND	EG, EK
DOUBLMSG	FL
DTFPHFLE	EA
DUMHDR1	RJ
ENDFOR	AC
ENDOPD	AC
EOFBLDWT	SA
EOFSKIP	SA
ERRMSG	AC, AD, EL
ERROP	SA
ERROR	QA, QB, QE, QF
EXC	QB
EXCP	EM
EXDATE	FE
EXEC	QE
EXIT	CC, RC, RE, RH, RM, SB
EXITCLOS	FG
EXITLAB	RC, RH, RM
EXITMON	SB
EXITNLAB	RE, RH, RM
EXITNLB	RE, RH, RM
EXITPT	SC
EXITRDR	SC
EXITRTN	EH
EXMFCM	QC
EXPDATE	FD, RJ
EXPIRCHK	RH
FCHMSG	EM
FETCHALT	FA
FETCHIT	RG, RK
FETCH07	EA, EB
FINISH	AD
FIRSTCD	AC
FORTUNL	RH
FRMTBL	EM
FRSTINST	BE
FRSTINST	BF
FSTM	RL

GETADDRS	QG
GETDATE	ED
GETDEFLT	ED
GETOPN	QB, QE
GETPUB	AA, FJ
GOSENS	RD
GOTCIC	EK
HDRBLD	RJ
HDRCHK	RM
HDRCHKX	RM
HDRCHK2	RM
HDRMSG	RM
HDRWRITE	RJ
HDR1CHK	RA
IGNORE	AE
IJCA2XXX	AI
IJCA3XXX	AI
IJCA5XXX	AI
IJCA6XXX	AI
IJCBXXXX	AQ
IJCEXXX	AK
IJCMBXXX	AE
IJCMCXXX	AG
IJCMDXXX	AI
IJCMEXXX	AG
IJCMRXXX	AI
IJCPKXXX	AE
IJCPNXXX	AI
IJCPYXXX	AK
IJCPZXXX	AJ
IJCRAXXX	AF, AI
IJCRBXXX	AF
IJCRFXXX	AG
IJCRTXXX	AJ
IJCRVXXX	AF, AI
IJCR9XXX	AE
IJCXXXXX	AH
IJC00XXX	AF
IJC30XXX	AH, AM
IJC33XXX	AM
IJC34XXX	AH, AM
IJC35XXX	AM
IJC36XXX	AH
IJC38XXX	AH
IJC41XXX	AH
IJC42XXX	AH
IJC43XXX	AH
IJC44XXX	AH
IJC50XXX	AM
IJC96XXX	AH
IJC97XXX	AH
IJC98XXX	AH
IJDPKXXX	CD
IJDPMXXX	CE
IJDPRXXX	CF
IJDPUXXX	CG
IJDPWXXX	CG
IJDPYXXX	CG
IJDPZXXX	CE
IJDQ0XXX	CE
IJDQ1XXX	CE
IJDSRXXX	CD
IJD51XX	CH
IJD52XX	CH
IJD53XX	CH
IJD59XX	CH
IJD70XX	CH
IJE_END	CJ
IJE_ENDD	CM
IJE_IGN1	CM
IJE_RSTR	CM
IJE_TIST	CJ
IJE_WLR	CJ
IJEGLCCW	CQ
IJE0CHNG	CR
IJE0END	CR
IJE0EXIT	CR
IJE0P256	CR
IJE0RCLN	CR
IJE0WRT1	CR
IJE0WRT2	CR
IJE2CCMP	CJ
IJE3CLR2	CK
IJE3END	CK
IJE3EOFR	CK
IJE3LNXT	CK
IJE3MOVE	CK
IJE3MVRM	CK
IJE3READ	CK
IJE3RRED	CK
IJE3STSD	CK
IJE3STTR	CK
IJE3TRMV	CK
IJE4CLR2	CL
IJE4EXIT	CL
IJE4STSD	CL
IJE6TLST	CM
IJE6XXX	PB
IJE7CLR2	CN
IJE7END	CN
IJE7EXIT	CN
IJE7MOVE	CN
IJE7MVRM	CN
IJE7P256	CN
IJE7SCCW	CN
IJE7STTR	CN
IJE8BRKN	CP
IJE8ENDD	CP
IJE8EOFL	CP
IJE8EXIT	CP
IJE8P256	CP
IJE8RSTR	CP
IJE9IGN	CQ
IJE9NOTR	CQ
IJE9RMN	CQ
IJE9RSTR	CQ
IJE9TLST	CQ
IJE9WRT	CQ
IJF_BCK	DJ
IJF_BCNT	DQ
IJF_BCNT	DT
IJF_CHNG	DQ
IJF_CHNG	DT
IJF_CNT	DB
IJF_COMP	DW
IJF_CONV	DV
IJF_DBL	DK
IJF_EFE	DU
IJF_EFE	DC
IJF_EOFF	DC
IJF_EOFF	DU
IJF_EOFV	DC
IJF_EOFV	DU
IJF_ERR1	DP
IJF_ER1	DS, DT
IJF_ER2	DR, DU
IJF_EXDT	DT
IJF_EXIT	DF
IJF_EXOT	DS
IJF_EXT	DL
IJF_EX1	DQ, DT
IJF_EX3	DF
IJF_FDBL	DD
IJF_FEO	DC

IJF_FEC1	DC
IJF_FEC2	DC
IJF_FTI	DP, DT
IJF_FT1	DQ
IJF_GBC	DI
IJF_GET	DD
IJF_GETB	DD
IJF_HRU	DI, DT
IJF_HRUS	DQ
IJF_HU2	DI, DQ, DT
IJF_IGN	DQ, DT
IJF_IGNR	DP, DS
IJF_INFL	DP
IJF_ICCK	DQ, DT
IJF_MAI	DI
IJF_MCVE	DW
IJF_MVCR	DW
IJF_NERX	DP
IJF_NICI	DQ
IJF_NCPR	DQ
IJF_NRXT	DS
IJF_PARI	DP
IJF_PUT	DG
IJF_PUTB	DG
IJF_PUTW	DG
IJF_PUT1	DG
IJF_RCK	DV
IJF_REL	DJ
IJF_REST	DW
IJF_SKI	DP
IJF_SKIP	DP, DS, DU
IJF_TEST	DD
IJF_TR	DV
IJF_TRI	DV
IJF_TRR	DV
IJF_TRU	DJ
IJF_VER1	DV
IJF_VER2	DV
IJF_WMV	DW
IJF_WCRK	DW
IJF_WRK	DW
IJF_WRK1	DW
IJF_WSR	DW
IJF_YSOT	DP
IJFFBCKW	DI
IJFFCON	DI
IJFFCSVC	DN
IJFFIGN	DI
IJFFRCK	DN
IJFFRCT	DI
IJFFSTD	DI
IJFFTET	DI
IJFFVRS1	DN
IJFFVRS2	DN
IJFNOPK	DT
IJFNRDWR	DW
IJFRDCBN	DD
IJFREX2	DQ
IJFRNICC	DG
IJFRNOPD	DD
IJFRNCPR	DQ
IJFRYSPD	DD
IJFSCNTS	DE
IJFSCNTT	DE
IJFSCNTU	DE
IJFSCNTV	DK
IJFSECVS	DR
IJFSEOVT	DR
IJFSECVU	DR
IJFSEOVV	DR
IJFSECVW	DK
IJFSGETS	DD
IJFSGETT	DD
IJFSGETU	DE
IJFSGETV	DE
IJFSMOVS	DE
IJFSMCVT	DE
IJFSPUTC	DG
IJFSPUTS	DH
IJFSPUTT	DH
IJFSPUTU	DH
IJFSPUTV	DH
IJFSSHF1	DE
IJFSSHF2	DE
IJFSSHIF	DE
IJFSSQER	DE
IJFWCHEK	DA
IJFWCTL	DB
IJFWEXCP	DW
IJFWEXT	DF
IJFWFND	DF
IJFWFSF	DW
IJFWIGNR	DA
IJFWNOTE	DF
IJFWNRXT	DA
IJFWOUT	DA
IJFWPNTR	DF
IJFWPNTS	DF
IJFWPNTW	DF
IJFWRIT	DJ
IJFWSKIP	DA
IJFWSTOR	DW
IJFWTEST	DF
IJFXFCR	DM
IJFXFORW	DM
IJFXFWD	DK
IJFXIGN	DM
IJFXLCOP	DM
IJFXLTST	DM
IJFXPAD	DM
IJFXRCT	DL
IJFXRST	DK
IJFXTDCH	DM
IJFXTRA	DK
IJG1XXX	PC
IJJASCNQ	NE
IJJASCN1	NE
IJJASCN2	NE
IJJCAL2	NC, NE, NG
IJJCEXEX	NC
IJJCFOND	NC, NE, NG
IJJCNDSK	NB
IJJCNOEK	NB
IJJCPCAL	NC, NE, NG
IJJCPDSK	NB, ND, NF, NH
IJJCPEND	NA
IJJCPEOV	NC
IJJCPFA3	ND, NF, NH
IJJCPF02	ND, NF, NH
IJJCPF03	ND, NF, NH
IJJCPF04	ND, NF, NH
IJJCPF13	NA, NB
IJJCPF14	NA, NB
IJJCPGP	NA, NC, NE
IJJCPIO	NC
IJJCPIUP	NC, NE
IJJCPSLD	NC
IJJCPSLO	NC
IJJCPST	NC
IJJCPSTO	NG
IJJCPTOV	NC
IJJCPUNX	NB, NC
IJJCPWDE	NE, NG
IJJCP10	NB

IJJCP210	NA
IJJC1RT	NE, NG
IJJC2SLO	NE
IJJEXEC	NA
IJJE1XXX	PA, PB, PC
IJJE5XXX	PB, PE
IJJE6XXX	PC, PF
IJJF1XXX	PE
IJJF3	PC
IJJF3XXX	PE
IJJF5XXX	PB, PE
IJJF6XXX	PE
IJJF8XXX	PA, PD, PF
IJJF9XXX	PC, PE
IJJG3XXX	PA, PC, PE
IJJG6XXX	PC
IJJG8XXX	PC
IJJH4XXX	PC
IJJH5XXX	PB, PC
IJJH8XXX	PB, PE
IJJIOXXX	PA, PE
IJJI0XXX	PE
IJJJ1XXX	PA, PB, PC, PE
IJJJ7XXX	PE
IJJJ8XXX	PA, PB, PC, PE
IJJK5XXX	PA, PB, PC, PE
IJJK6XXX	PA, PB
IJJK8XXX	PA, PB, PC, PE
IJJL5XXX	PB, PE
IJJMOV	NC, NE, NG
IJJM2XXX	PD
IJJM3XXX	PB, PD, PF
IJJM4XXX	PB, PD, PF
IJJM7XXX	PD, PF
IJJNCNED	NE
IJJNCSK	NC, NE, NG
IJJPWT	NE, NG
IJJRTRN	NA, NC
IJJSAVE	NC, NE, NG
IJJTEST	NA
IJJUNITX	NE, NG
IJJWCHED	NC, NG
IJJWINK	NE
IJJXXXXX	PC
IJJ10XXX	PC
IJJ44XXX	PE
IJJ48XXX	PA
IJMABXXX	BS
IJMAIXXX	BJ
IJMA1XXX	BM, BQ
IJMA2XXX	BJ
IJMBAXXX	BM
IJMBDXXX	BM
IJMBEXXX	BM
IJMBFXXX	BM
IJMBIXXX	BM
IJMBJXXX	BM
IJMBMXXX	BM
IJMBRYXX	BS
IJMCEXXX	BG
IJMCLXXX	BT
IJMCM	CA
IJMCNXXX	BG
IJMC0	CA
IJMC2	CA
IJMC3	CA
IJMDRXXX	BQ
IJMDYXXX	BQ
IJMEC	CA
IJME0	CA
IJME1	CA
IJME2	CA
IJME3	CA
IJME3XXX	BM
IJME4	CA
IJME4XXX	BM
IJME5	CA
IJME6	CA
IJME7	CA
IJME8	CA
IJME9	CA
IJMGBXXX	BJ
IJMGOXXX	BS
IJMM2XXX	BS
IJMM3XXX	BS
IJMM4XXX	BS
IJMOGXXX	BH
IJMRD	CA
IJMRDXXX	BP
IJMRGXXX	BT
IJMRLXXX	BR
IJMRTXXX	BS
IJMRTXXX	BT
IJMRUXXX	BT
IJMRXXXX	BT
IJMRZXXX	BR
IJMR1XXX	BR
IJMR3XXX	BR
IJMR4XXX	BR
IJMR5XXX	BR
IJMR8XXX	BR
IJMR9XXX	BR
IJMSAXXX	BJ, BU
IJMSBXXX	BH, BJ, BU
IJMSCXXX	BH, BJ, BU
IJMSKXXX	BG
IJMSO	CB
IJMSQXXX	BJ, BM, BS
IJMS0	CB
IJMS1	CB
IJMS2	CB
IJMS3	CB
IJMS5	CB
IJMS6	CB
IJMS7	CB
IJMS8	CB
IJMS9	CB
IJMTAXXX	BK, BN, BS
IJMTBXXX	BK, BN, BS
IJMTCXXX	BK, BN
IJMTDXXX	BK, BN
IJMTEXXX	BK, BN
IJMTFXXX	BK, BN, BT
IJMTIXXX	BL
IJMTJXXX	BJ
IJMTKXXX	BJ
IJMTLXXX	BJ
IJMTMXXX	BK
IJMTNXXX	BL
IJMTOXXX	BJ, BL
IJMT0XXX	BS, BM
IJMT1XXX	BP
IJMT2XXX	BP
IJMT3XXX	BP
IJMT5XXX	BG
IJMT6XXX	BG
IJMT7XXX	BG
IJMT8XXX	BM, BS
IJMT9XXX	BK, BN, BS
IJMUAXXX	BT
IJMUBXXX	BT
IJMUCXXX	BT
IJMUSXXX	BT
IJMUTXXX	BT

LASTOPN	AB, AD
LDIOREG	QE, QE
LFCOMP	RM
LOADADDR	FE1, FF
LOOP	EJ, PJ
MDRSBY	FE, FH
MFCM	SC
MFCX	QC
MF1	QC
MF3	QC
MF4	QC
MF5	QC
MGTPOP	QA, QB, QE, QF
MODE9	EI
MOVBLNK	ED, FE
MOVE	EE, FD, FE
MOVESER	SA
MOVEUNIT	EA
MSGEXIT	RM
MSGRTN	QG
MSGWRITE	EK, FH, FJ, RF, RJ, RL
MSG10A	ED
MSG11A	EA
MSG12A	EA, ED, RH, RL
MSG13D	EA, RM
MSG14A	EE, RL
MSG15A	EA, RL
MSG16A	EA, RL
MSG17D	EC
MSG18D	EC
MSG19A	EE, EH, RH
MSG23D	EC, EG
MSG30A	FA
MSG31D	FA, FC
MSG32D	EA, RL
MSG33D	EE, RL
MSG4184D	RF, RJ
MSG83I	EA, RC, ED, EH, EK, FA, FH
MVDATE	ED, FE
NEWVCI	EF
NEXTDEL	AD
NEXTJIB	FE
NINETRK	EC, ED, EF
NOALT	FE, FH
NOANSWER	FM
NOBLNK	FD
NOCTLCHR	AA
NODEFALT	RJ, SA
NOHDR	EI
NOLBL	FC
NOMSG	QA, QB, QD, QF
NOP	FK, FM
NORET	EC
NORETURN	FH
NOREWUN	FH
NOTBLANK	ED, FD, FE, RJ, SA
NOTLEGAL	FM
NOTOPEN	EF
NOVARY	AA
NOVCIO	RJ
NOVOL1	EH, EK, RL
NOVSAM	MM
NOXPIRCK	RH
NREWIND	EH
NRF	SA
NSTDEXIT	FF
NSTDSTD	FF
NVOL	EI
NVOLCK	EH, RH
NVOLON	EK, RE
NXTALT	FE, FH
NXTFLD	RM
NXTJIB	FH
NXTPHS	RA, RD
OELFND	AD
OLDCARD	ED, FD
ONEIA	SD
OPENINIT	SA
OTHERCD	AC
OUT	BE
PACKIST	AD
PCHBLANK	SC
PCHBLNK	SD
PPRT	QB
PRINTIN	QB
PROCESS	EH, PH, PG
PROCRUN	RG, RK, SA
PROCRUN1	RG, RK, SA
PROCRUN2	RK, RL, SA
PTSIO	SC
PT1	SC
PT2	SC
PUBADR	QC
PUNCHBLK	SD
P50	AK
RDAGAIN	EH
RDFRWRD	CC
READ	EG, FJ
READAGAN	EC
READBK	EC
READLABL	EB
READTAPE	EA, FJ
READVOL	FO
READ3TM	EA, EB
RELNEXT	RA, RD, RK
RELOCATE	EA
REPCHCP1	SD
REPCHCP2	SD
REPUNCH	PG
REREAD	PH
RESET	EJ
RESTORE	RL
RETMON	QB, QC, QD, SC
RETRNPT	EL
RETRYMSG	PG, SD
RETURN	FB, FO, RB, SC
RETURNED	EA, EK, FH
REWIND	EA, ED, FG, FO
REWINDIT	EG
REWINDT	FG
REWRITE	PH
RST	QC, QD
RTRYNT	EK
SAVE	FO
SAVENUMB	AD
SAVSYS	QC, QD
SENSE	RB, RE
SENSEX	RB
SETBIT	CC
SETCCB	FO
SETDEV	AD
SETEOB	FM, SE
SETEXIT	FF
SETOMR	AC
SETOPEN	EC
SETOPNBT	AA, AB
SETPHASE	FO
SETPHS	RG
SETPNT	EM
SETPUB	FO
SETSEQP	AA
SETSW	QB, QD
SETSWCH	FC
SETUPRD	CC

SET3886	CC
SET7MODE	EA
SEVENTRK	EH
SHIFT	BE
SKIP	BE, EG, EH, EK, EL, FA, FC, FD, FE, FH, RJ, SA
SKIPBLS	FA
SKIPFWD	FJ
SKIPMSG	FB
SKIP0	FM
SKIP1	EH
SKIP2	FI
SKPVOL	EI
SP	SA
SPACEFWD	EB
SRCHIAB	BE, RE
STARTIO	SD
STKON	QA
STMOD10	AA
STOR	QE
STRTOP	QA, QB, QD, QF
SVCALL	EI
SVNTRACK	EG
SYSLOAD	QE
SYSNO	FH
SYSO	QD
SYS0	QA
TEST	EA
TESTASC	FD
TESTREW	FF
TJJM2XXX	PF
TMPPBF	FE
TMWRITE	RH, SA
TOEXIT	EB
TRANS	FL
TRANSENT	EH, FA, FC, FD, FE, FF
TRASCII	RJ, SB
TREWIND	SA
TSTLAB	EM
TSTPCH	PG
TSTRNG	QB, QE, QF
TYPE	BF
UNTPCP	QD
UNTRC	QB, QC, QD
UNTRCP	QA, QB, QC, QF
UPDATE	AB, EF, EJ, FB, FD
USDSTM	EK
USER	FE1, FF
USERRTNE	FA
USERTEST	EF
VOLSKIP	RH, RL
VOLSKIP1	RH, RL
VOL1CHK	RH, RL
VOL800	EL, RE
WHDRS	RB
WORKFILE	FG
WRITE	FJ, FO
WRITETAP	FJ
WRITHDR	EH
WRITMK	EH
WRITTM	EL
WTDEVED	SD
XYZ	PH
YESLDPT	ED
YESOPEN	EE
YESREWND	EH
YESTM	EA

For further detailed information on these messages, see _VSE/Advanced Functions Messages_, SC33-6098.

Message Number	Issuing Routine	Chart	Message
4000I	CDMOD $$BERRTN $$BCLOSP	AL PG SC SD	RETRY
4n05I	$$BCEOV1 $$BOCPT1	EM SE	UNRECOVERABLE I/O ERROR
4110A	$$BOMT03 $$BOCPT3	ED RJ	NO VOL1 LBL FOUND TLBL=xxxxxx filename SYSxxx=cuu
4111A	$$BOMT01 $$BOCPT4	EA RL	NO VOL1 LBL FOUND filename SYSxxx=cuu
4112A	$$BOMT01 $$BOMT03 $$BOCPT3 $$BOCPT4	EA ED RH RL	VOL SERIAL NO. ERROR TLBL=xxxxxx filename SYSxxx=cuu
4113D 4113I	$$BOMT01 $$BOCPT4	EA RM	NO HDR1 LABEL FOUND filename SYSxxx=cuu
4114A	$$BOMT01 $$BOCPT4	EB RL	FILE SEQ NO. ERROR filename SYSxxx=cuu
4115A	$$BOMT01 $$BOCPT4	EA RL	FILE SER. NO. ERROR TLBL=xxxxxx filename SYSxxx=cuu
4116A	$$BOMT01 $$BOCPT4	EA RL	VOLUME SEQ. NO. ERROR filename SYSxxx=cuu
4117D	$$BOMT02 $$BOMT05	EC EG1	NO TM FOUND ON READBK filename SYSxxx=cuu
4118D 4118I	$$BOMT02	EC	FILE ID ERROR, READBK filename SYSxxx=cuu
4119A	$$BOMT03 $$BOMT06 $$BOCPT3	EE EH RH	FILE UNEXPIRED filename SYSxxx=cuu
4120I	$$BOMT03	EE	TAPE POSITIONED WRONG filename SYSxxx=cuu
4121A	$$BOMT07	FH	NO ALTERN DRIVE ASSGN SYSxxx=cuu
4122I	$$BOMT07	FH	ECV ENCOUNTERED SYSxxx=cuu
4123D 4123I	$$BOMT02	EC	WRONG POSITN, READBK filename SYSxxx=cuu
4124I	$$BOMT04	EF	TOO MANY UHLS. filename SYSxxx=cuu
4125D 4125I	$$BOMT05	EG	VOL1 LBL FOUND filename SYSxxx=cuu

Figure 40. Message Cross-Reference List (Part 1 of 4)

Message Number	Issuing Routine	Chart	Message
4126I	$$ECMT02	FB	ECV ENCOUNTERED filename SYSxxx=cuu
4127A	$$BCMTC5	FE	EOV WHILE WRITING ECF
4130A	$$BCMT01	FA	ECF OR EOV INQUIRY filename SYSxxx=cuu
4131D	$$ECMT01 $$BCNT03	FA FC	BIOCK COUNT ERROR filename SYSxxx=cuu DFT=xxxxxx LBL=xxxxxx
4132D	$$BOMT01 $$BOCPT4	EA RL	EFROR IN FILE ID filename SYSxxx=cuu
4133D	$$BOMTC1 $$ECCPT4	EB RL	EFRCR IN HDR LBL filename SYSxxx=cuu
4140A	$$BCMTC2	FB	NO AITERN DRIVE ASSGN filename SYSxxx=cuu
4151I	$$BOMT01 $$ECMT04	EA EF	HDR1 LBL INFORMATION filename SYSxxx=cuu
4170A	$$EJCOPT	EK	FILE PROTECTED TAPE SYSxxx=cuu
4171A	$$BJCOP1	EL	UNEXPIRED FILE SYSxxx=cuu
4172A	$$BJCOP1	EL	INVALID LABEL SET SYSxxx=cuu
4183I	$$ECMT01 $$BOMT02 $$EOMT03 $$BOMTC6 $$EJCCPT $$ECMT01 $$BCMT05 $$BCMT07	EA EC ED EH EK FA FE FH	INVALID LOGICAL UNIT filename SYSxxx=cuu
4184D	$$BOCPT2 $$ECCPT2 $$BOCPT3	RE RF RJ	NEED FILE PROTECT RNG filename SYSxxx=cuu
4185I	$$BOMRCE $$BOMRCE	AC AD	INVALID FORMAT CARD(S)
4400I	$$BODIO3	WG	NO LABEL SPACE IN VTCC or NC RECORD FOUND
4301I	$$EODIC5	VF	NC FORMAT 1 LABEL FOUND or NO RECORD FOUND

Figure 40. Message Cross-Reference List (Part 2 of 4)

Message Number	Issuing Routine	Chart	Message
4306I	$$BODIO1	VD	NO STANDARD VOL 1 LABEL or NO RECORD FOUND
4506I	$$BODIC4	WT	
4307I	$$B35400	VC	NO RECORD FOUND
4407I	$$B35400	WC	
4329D	$$B3540I	VC	EXTENTS NOT EXHAUSTED
4332I	$$BODIC5	VH	VOLUME SEQUENCE ERROR
4337I	$$B3540I $$BODIC6	VA VM	CHAINING TO SYSTEM UNIT
4437I	$$B35400 $$BODIC7	WA WL	
4445I	$$BODIC8	WQ	TOO MANY EXTENTS
4348I	$$B35400	UE	SYSIN/SYSOUT UNSUPPORTED
4450A	$$BODIO8	WP	NO MORE AVAILABLE EXTENTS
4355A	$$BODIC1	VD	WRONG PACK, MOUNT nnnnnn
4358I	$$B35400	UF	NO EXTENT FOR OUTPUT FILE
4359I	$$BODIC5 $$BODIC6	VK VL	INVALID EXTENT
4459I	$$BODIC3	WG	
4360I	$$B35400	VB	NO EXTENTS, ALL BYPASSED
4361I	$$B35400	UF	INVALID DLBL FUNCTION
4364I	$$BODIO5 $$BODIC6	VF VM	INVALID HDR1 LABEL
4465I	$$BODIO2	WD	EQUAL FILE LABEL IN VTOC

Figure 40. Message Cross-Reference List (Part 3 of 4)

Message Number	Issuing Routine	Chart	Message
4n79I	$$BCEOV1 $$BCCPT1	EM SB	GETVIS FAILED
4880I	$$BCEOV1	EM	INVALID FILE TYPE
4881I	$$BCCPT1 $$BCECV1 $$B35400	SA EM UF	NO LABEL INFORMATION
4383I	$$B3540I	VC	INVALID LOGICAL UNIT
4483I	$$BCSDC1 $$B35400	LM WC	
4883I	$$BCUR01 $$BCCP01 $$BOCP02 $$BOCP11 $$BOCP12	AB QA QB QE QF	
4884D	$$BCCP02 $$BOCP11 $$BCCP12	QB QE QF	NEED FILE PROTECT RNG
4887I	$$BERRTN	PG	SYS FILE EXTENT EXCEEDED
4888I	$$BERRTN	PG	EOF ON SYSTEM FILE
4MR1I	MRMOD	BB	EXTERNAL INTERRUPT I/O ERROR
4MR2I	MRMOD	BB	SCU NOT OPERATIONAL
4P01I	$$BERRTP	PH	DATA CHECK SYSxxx=cuu
4P02D	$$BERRTP	PH	DATA CHECK SYSxxx=cuu
P200I	$$BOPR3	QG	3800 Printer Extended Buffering Mode Not Used - Reason Code = xx

Figure 40. Message Cross-Reference List (Part 4 of 4)

APPENDIX C: CONTROL CODES

CTLCHR=ASA

A control character must appear in each logical record if the ASA option is chosen. If the control character for the printer is not valid, a message is given and the job is canceled. If the control character for the card punch is not V or W, the card is selected into pocket 1. The codes are as follows:

Code	Interpretation
b	Space one line before printing
0	Space two lines before printing
-	Space three lines before printing
+	Suppress space before printing
1	Skip to channel 1 before printing
2	Skip to channel 2 before printing
3	Skip to channel 3 before printing
4	Skip to channel 4 before printing
5	Skip to channel 5 before printing
6	Skip to channel 6 before printing
7	Skip to channel 7 before printing
8	Skip to channel 8 before printing
9	Skip to channel 9 before printing
A	Skip to channel 10 before printing
B	Skip to channel 11 before printing
C	Skip to channel 12 before printing
V	Select stacker 1
W	Select stacker 2

CTLCHR=YES

The control character is the command-code portion of the Channel Command Word used in printing a line or spacing the forms. If the character is not one of the following characters, unpredictable events will occur.

8-Bit Code	Punch Combination	Function
Stacker Selection on 1442 and 2596		
10000001	12,0,1	Select into stacker 1
11000001	12,1	Select into stacker 2
Pocket Selection on 2540		
00000001	12,9,1	Select into pocket 1
01000001	12,0,9,1	Select into pocket 2
10000001	12,0,1	Select into pocket 3

8-Bit Code	Punch Combination	Function
Stacker Selection on 2520, 3504, 3505, 3525		
00000001	12,9,1	Select into stacker 1
01000001	12,0,9,1	Select into stacker 2
Printer Control (not for 3525)		
00000001	12,9,1	Write (no automatic space)
00001001	12,9,8,1	Write and space 1 line after printing
00010001	11,9,1	Write and space 2 lines after printing
00011001	11,9,8,1	Write and space 3 lines after printing
10001001	12,0,9	Write and skip to channel 1 after printing
10010001	12,11,1	Write and skip to channel 2 after printing
10011001	12,11,9	Write and skip to channel 3 after printing
10100001	11,0,1	Write and skip to channel 4 after printing
10101001	11,0,9	Write and skip to channel 5 after printing
10110001	12,11,0,1	Write and skip to channel 6 after printing
10111001	12,11,0,9	Write and skip to channel 7 after printing
11000001	12,1	Write and skip to channel 8 after printing
11001001	12,9	Write and skip to channel 9 after printing
11010001	11,1	Write and skip to channel 10 after printing
11011001	11,9	Write and skip to channel 11 after printing
11100001	11,0,9,1	Write and skip to channel 12 after printing
00001011	12,9,8,3	Space 1 line immediately
00010011	11,9,3	Space 2 lines immediately
00011011	11,9,8,3	Space 3 lines immediately
10001011	12,0,8,3	Skip to channel 1 immediately
10010011	12,11,3	Skip to channel 2 immediately
10011011	12,11,8,3	Skip to channel 3 immediately
10100011	11,0,3	Skip to channel 4 immediately
10101011	11,0,8,3	Skip to channel 5 immediately
10110011	12,11,0,3	Skip to channel 6 immediately
10111011	12,11,0,8,3	Skip to channel 7 immediately

8-Bit Code	Punch Combination	Function
11000011	12,3	Skip to channel 8 immediately
11001011	12,0,9,8,3	Skip to channel 9 immediately
11010011	11,3	Skip to channel 10 immediately
11011011	12,11,9,8,3	Skip to channel 11 immediately
11100011	0,3	Skip to channel 12 immediately
Printer Control on 3525 with Print Feature		
00000011	12,9,3	No operation
00001101	12,5,8,9	Print on line 1
00010101	11,5,9	Print on line 2
00011101	11,5,8,9	Print on line 3
00100101	0,5,9	Print on line 4
00101101	0,5,8,9	Print on line 5
00110101	5,9	Print on line 6
00111101	5,8,9	Print on line 7
01000101	12,0,5,9	Print on line 8
01001101	12,5,8	Print on line 9
01010101	12,11,5,9	Print on line 10
01011101	11,5,8	Print on line 11
01100101	11,0,5,9	Print on line 12
01101101	0,5,8	Print on line 13
01110101	12,11,0,5,9	Print on line 14
01111101	5,8	Print on line 15
10000101	12,0,5	Print on line 16
10001101	12,0,5,8	Print on line 17
10010101	12,11,5	Print on line 18
10011101	12,11,5,8	Print on line 19
10100101	11,0,5	Print on line 20
10101101	11,0,5,8	Print on line 21
10110101	12,11,0,5	Print on line 22
10111101	12,11,0,5,8	Print on line 23
11000101	12,5	Print on line 24
11001101	12,0,5,8,9	Print on line 25

INDEX

VSE/Advanced Functions Diagnosis Reference: LIOCS Volume 1, LY24-520, contains a master index to LIOCS Volumes 1, 2, 3, and 4.

LY24-5210-0

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 2
SAM
LY24-5210-0

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:

____
____
____

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.)

Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

Reader's Comment Form

Fold and tape	**Please Do Not Staple**	Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE:

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

Fold and tape	**Please Do Not Staple**	Fold and tape

IBM®

**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**