Z24-5022-0

Program Logic

IBM System/360 Basic Operating System
System Control (16K Tape)
Program Logic Manual

Program Number 360M-CL-405

This manual was prepared by Programming Systems to provide detailed information on the internal logic of the IBM System/360 16K Tape System Control Programs. It is intended for technical personnel who are responsible for diagnosing the system operation and/or adapting the programming system to special usage.

PREFACE

The prerequisites for a thorough understanding of this publication are a basic knowledge of both System/360 programming concepts and Basic Operating System 16K Tape Control Programs. The publications providing this information are:

- IBM System/360 Principles of Operation, Form A22-6821;
- IBM System/360 Basic Operating System: System Control and System Service Programs (16K Tape), Form C24-3431;
- IBM System/360 Basic Operating System: Supervisor and Input/Output Macros (16K Tape), Form C24-3432;
- IBM System/360 Basic Operating System: System Generation and Maintenance (16K Tape), Form C24-5015.

Closely related publications are:

- IBM System/360 Basic Operating System: Assembler Language Specifications (16K Disk/Tape), Form C24-3414;
- IBM System/360 Basic Operating System: Data Management Concepts (16K Tape), Form C24-3430.

This publication is designed to help the reader in two ways:

1. Detailed flowcharts with accompanying narratives and a cross-reference label list provide a quick guide to the program listing.
2. Summaries of program flow, I/O flow, storage allocation, and constant areas provide a basic understanding of the program logic.

This publication describes the program logic at three levels:

System level. A brief description of the programs in the system and how they relate to one another.

Program level. A general description of the flow of each program describing the routines within the program and information common to these routines.

Routine level. A detailed description of the flow of each routine with reference to a detailed flowchart of the routine.

Information common to all programs in the system is found in the discussion of the Supervisor, and a detailed layout of system residence is found in the introduction to the Librarian.

Copies of this and other IBM publications can be obtained through IBM Branch Offices. A form has been provided at the back of this publication for readers' comments. If the form has been detached, comments may be directed to:
IBM Programming Publications, Endicott, New York 13760.

CHARTS

The 16K tape resident version of the IBM System/360 Basic Operating System is designed to minimize the time and effort required by the user to produce and process programs. Centralized control governs the execution of all other programs on the system. Services edit programs for execution under control and maintain libraries of programs for the system.

Control programs constitute the framework of System/360 BOS. The Supervisor handles all interrupts. Job Control provides program-to-program transition. IPL initializes the system, loading the Supervisor and Job Control.

Service programs prepare all programs for the system and maintain the system libraries. Linkage Editor edits language processor output for execution under the system Supervisor. Librarian maintenance includes items in, or deletes items from, the system libraries. Librarian service provides information on the contents of the system libraries.

Figure 1 shows the relationship between these system components.

SYSTEM RESIDENCE

The system control programs reside on a tape along with other IBM programs and user programs in the system. The tape records are divided into three libraries.

Core Image Library; Programs ready for execution; output from the Linkage Editor.

Relocatable Library; Program modules that have not been edited to run under the Supervisor; output from the language processors; input to the Linkage Editor.

Source Statement Library; Library routines in source language (e.g., macros); input to the language processors.

Figure 2 shows the organization of the residence tape. The tape unit is assigned to the logical unit SYSRES. The relocatable and source statement libraries may be on independent tapes assigned to SYSRLB and SYSSLB, respectively.

Along with the Supervisor in the core image library are a Job Control, IPL, Linkage Editor and Librarian edited to run with the Supervisor. A new Supervisor can be generated from the system generation macros in the source statement library. Then revised versions of the system control and service programs must be edited into the core image library from modules provided in the relocatable library.

SYSTEM CORE ALLOCATION

Under System/360 BOS control main storage is divided into two parts as in Figure 3. Low core is reserved for the Supervisor and transient routines. High core is called the problem program area.

The user can specify the point of division at system generation time.

SYSTEM I/O

All input and output under Supervisor control is done in reference to logical units with names of the form 'SYSXXX'. The I/O tables in the Supervisor define the relationship between logical units (LUB's) and physical devices (PUB's), according to the assignments made in the system or by the user. Figure 4 shows the logical units referenced by each of the system programs and the devices which may be assigned to them.

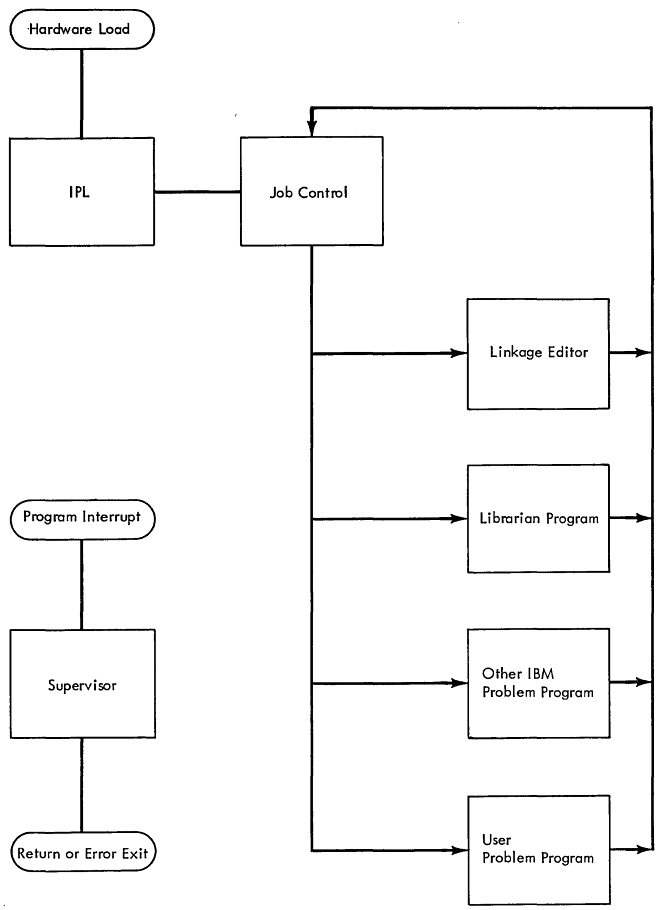


Figure 1. System Control Component Interrelationship

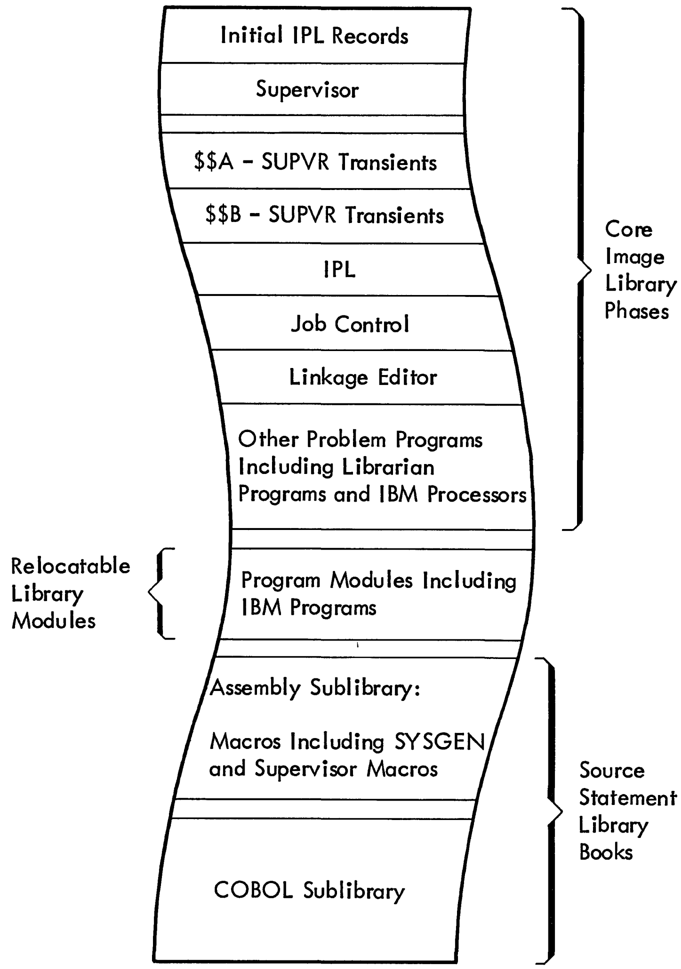


Figure 2. System Residence

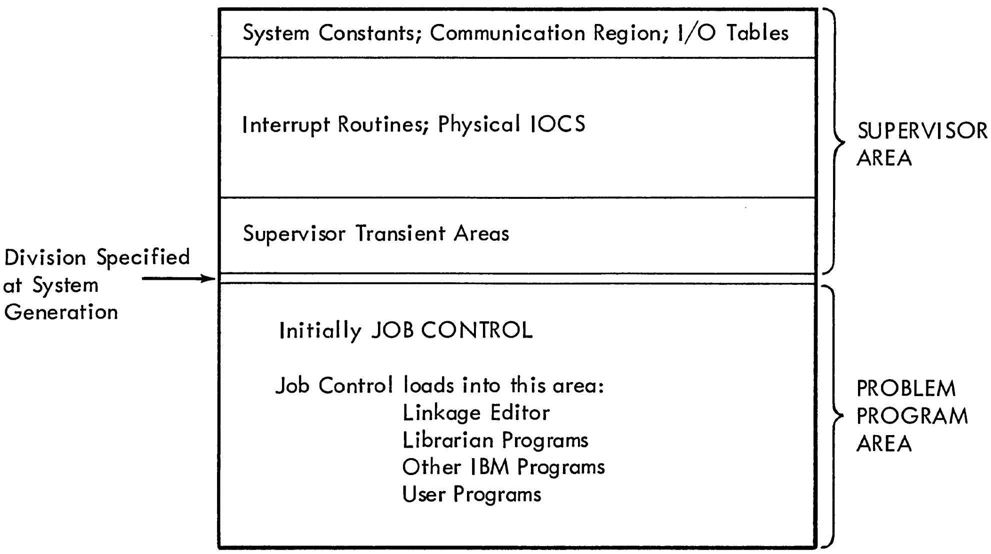


Figure 3. System Core Allocation

		SYSRDR	SYSIPT	SYSPCH	SYSLST	SYSLOG	SYSRES	SYSSLB	SYSRLB	SYS000	SYS001	SYS002	Other
	IPL	CR				K	T						
	Job Control	CR,T	CR,T		P,T	K,P				T			
	Linkage Editor				P,T	K,P	T		T	T	T	T	
Supervisor-Transients	CHKPT					K,P							T
	CNCL				P,T	K,P							
	DUMP				P,T								
	ILSVC				P,T	K,P							
	JCOPT	T	T	T	T								T
	MSGIN					K							
	PCHK				P,T	K,P							
	PDUMP				P,T								
	RSTRT					K,P							T
Librarian	MAINT	CR,T			P,T	K,P	T			T		T	
	MAINTR	CR,T	CR,T		P,T	K,P	T		T		T	T	
	MAINTS	CR,T	CR,T		P,T	K,P	T	T			T	T	
	DSERV	CR,T			P,T	K,P	T	T	T				
	RSERV	CR,T		CP,T	P,T	K,P	T		T				
	SSERV	CR,T		CP,T	P,T	K,P	T	T					

CR - Card Reader
CP - Card Punch
P - Printer
K - Console
T - Tape

Figure 4. System I/O: Possible Logical Unit and Allowable Device Usage

SUPERVISOR

The Supervisor is read into core at IPL time and remains there throughout system operation. The Supervisor area consists of four parts:

1. Constant areas to describe system operation.

2. Nucleus of coding to handle interrupts.

3. Type-A transient area to hold physical IOCS error-routine overlays.

4. Type-B transient area to hold Supervisor routine overlays.

Figure 5 shows the allocation of core to the various parts of the Supervisor.

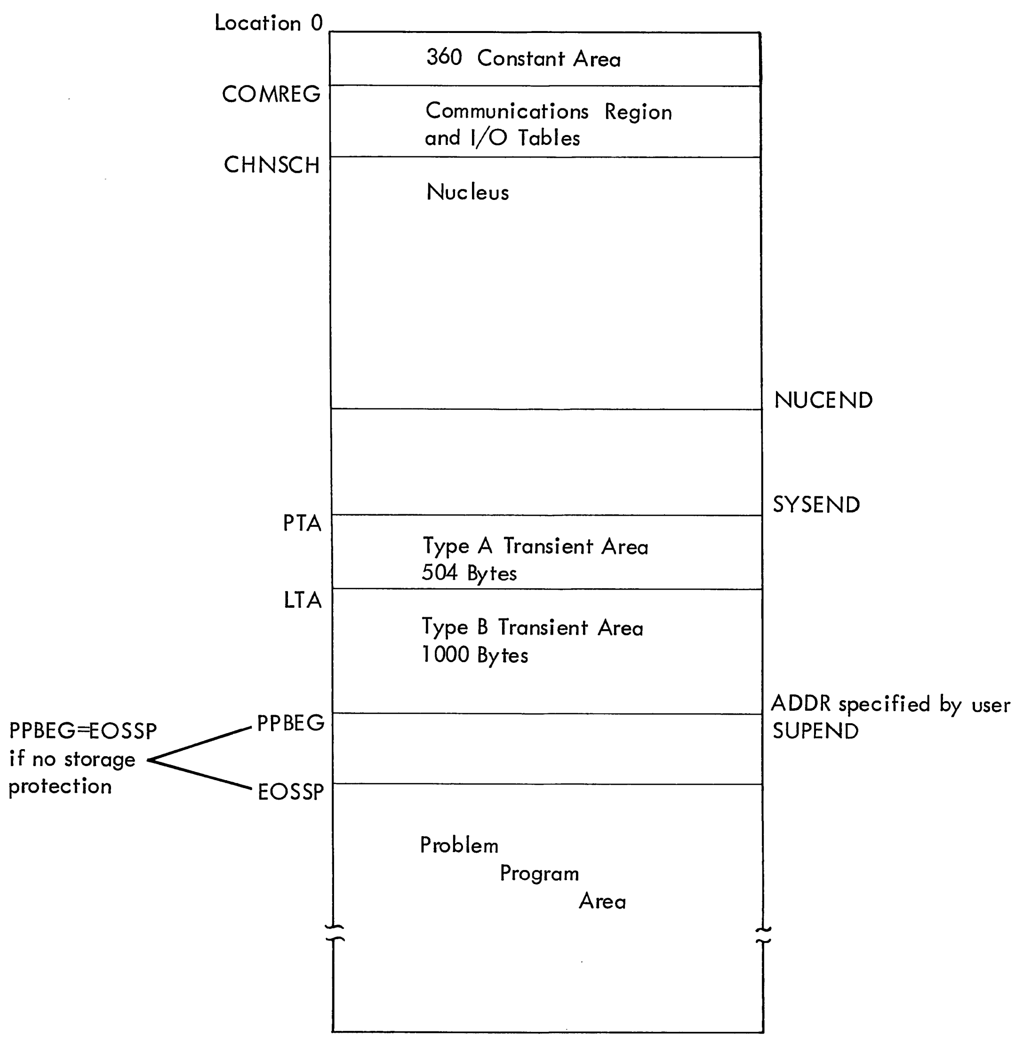


Figure 5. Supervisor Core Allocation

HEX	DECIMAL	DEFINITION	SYSGEN INITIALIZATION TO ZEROS EXCEPT FOR
00	00	IPL PSW	
08	08	IPL CCW_1	
10	16	IPL CCW_2	Zeros in This System-Location 22 (hex 16) contains address of communications region
18	24	External Old PSW	
20	32	SVC Old PSW	
28	40	Program Old PSW	
30	48	Machine Old PSW	
38	56	I/O Old PSW	
40	64	CSW	
48	72	CAW	
4C	76	——	
50	80	Timer	'FF FF FF FF' $\left.\right\}$ $\frac{\text{Time-of-day -Timer/256}}{\text{Actual Time-of-day}}$
54	84	Time-of-Day	'00 FF FF FF'
58	88	External New PSW	Addresses EXTINT; Bit 13 on, Bits 36-39 on
60	96	SVC New PSW	Addresses SVCINT; Bit 13 on
68	104	Program New PSW	Addresses PCHECK; Bit 13 on
70	112	Machine New PSW	Addresses SETACT (in Unit Check)
78	120	I/O New PSW	Addresses INTERR; Bit 13 on
80	128	Scan Out Area and Saved Double Word	

Figure 6. Supervisor Constant Areas

SUPERVISOR CONSTANT AREAS

An area in low-core is reserved by the Supervisor for a communications region and for I/O tables.

The communications region immediately follows the System/360 constant areas shown in Figure 6. The address of the communications region is always located in decimal address 22. The user has this address available through the macro COMREG. Figure 7 displays the information contained in the communications region.

The I/O tables follow the communications region. These tables are necessary for the execution of I/O by the Supervisor.

For every device on the system there must be a physical unit block (PUB). For every logical unit name (SYSXXX) used by the programmer, there must be a logical unit block (LUB).

When the programmer requests I/O, an entry is made in the channel queue table (CHANQ). The CCB provided names a logical unit.

The Supervisor processes the request when possible on the device assigned to the logical unit. If the user specifies TEB at system generation time, counts of tape errors are kept in tape error blocks (TEB's) for each tape unit.

PUB's are set up at system generation or IPL time. LUB's are set up at system generation. PUB's are assigned to LUB's at system generation or by the ASSGN routine in Job Control. Temporary or alternate assignments require that job information blocks (JIB's) be attached to LUB's.

CHANQ entries are set up by the Channel Scheduler routine and processed by the Actual I/O routine in the Supervisor. TEB's are used by the Tape Error routine in the Supervisor.

The elements of each I/O table are described in detail in the following sections. Figure 14 displays the interrelationship of tables and pointers. Figure 15 is a sample table. Figure 16 summarizes each table entry.

DECIMAL DISPLACEMENT	SYSTEM GENERATION INITIALIZATION	DEFINITION
00	'07/15/65'	Job date from DATE statement
08	Y (PPBEG)	Address of problem program label area; end of transient area +1
10	Y (EOSSP)	Address of problem program area; if storage-protect, first byte having a storage protection key of 1
12	Zeros	Problem program work area
23	'00'	UPSI byte
24	'NO NAME'	Job name from JOB statement
32	Hex 3FFF; Dec 16383	Address of last byte in problem program area
36	Hex 3F10; Dec 16144	Address of last byte placed in the problem program area by a FETCH or LOAD
40	Hex 3F10; Dec 16144	End of longest phase of problem program
44	'0000'	Length of problem program label area
46	'0000'	
48	Hex 3F10; Dec 16144	Address of last byte of data in SPOOL/Tele-processing area
52	'00' or as specified	Machine Configuration bit 0 - storage-protection feature 1 - decimal feature 2 - floating-point feature 3 - 0 4 - timer feature 5 - channel switchable device support 6 - burst/multiplex device support 7 - 0
53	'00' or '80'	Date configuration: MDY or DMY; Bits used as switches: 5 - burst/multiplex device in operation 6 - FETCH/LOAD in operation 7 - logical transient in operation

Figure 7. Supervisor Communications Region (Part 1 of 2)

DECIMAL DISPLACEMENT	SYSTEM GENERATION INITIALIZATION	DEFINITION
54	'00 00' or as specified	Standard job-control options for third and fourth job-control switch banks
56	'C0 00 00 00'	Job-control switch bytes: detailed description in Job Control
60	'00 00'	Disk address (CC HH) of label area
62	A (FOCL)	I/O table addresses
64	A (PUBTAB)	
66	A (FAVP)	
68	A (JIBTAB)	
70	'0000' or A (TEBTAB) if TEB specified	
72	A (FICL)	
74	A (NICL)	
76	A (LUBTAB)	
78	30 or as specified	Line count for SYSLST
79	' 07 15 65 166'	System date and day of year from SET statement
88	'00'	Cancel code
89	'00'	
90	A (ALLRGS)	Address of register save area for SVC
92	'0000'	Identification number of last checkpoint
94	'0000'	Address of SPOOL/Tele-processing data
96	A (DSKPOS)	Address of disk I/O position data
98	A (ERBLOC)	Address of channel scheduler error data

Figure 7. Supervisor Communications Region (Part 2 of 2)

CHANQ - CHANNEL QUEUE TABLE

The number of entries is specified at system generation time. The table must hold all the I/O requests that await execution at any one time. The entries are referenced by CHANQ pointers 0, 1, 2.

Each CHANQ entry consists of 4 bytes:
  0 - chain field; pointer to the next in queue. 'FF' marks the end of a queue.
  2-4 - CCB address.
Associated with each entry is one byte at LUBPTR and one byte at REQID. LUBPTR is a LUB pointer indicating the logical unit making the I/O request. REQID, if storage-protection, is a code identifying the program making the I/O request.

The byte FLPTR contains the pointer to the next free entry (free list pointer). The free entries are chained together in a queue to FLPTR as request entries are chained to a PUB.

CHANQ entries are referenced by PUB's identifying devices awaiting I/O. Channel queues are built by the Channel Scheduler routine in the Supervisor.

PUBTAB - PHYSICAL UNIT BLOCK TABLE

The number of entries is specified at system generation time. There must be one entry for each device on the system. Entries may be made to the table at system generation time. The operator may add or delete entries at IPL time. The entries are referenced by PUB pointers 0, 1, 2.

Each PUB entry consists of 8 bytes:
  0 - channel address; 0, 1,----6; 'FF' = null channel address.
  1 - unit address.
  2 - CHANQ pointer; 0, 1,----; 'FF' = null pointer.
  3 - TEB pointer; 0, 1,----; or error counter; '00' = null pointer or initial counter.
  4 - device type (Figure 8).
  5 - device options; '00' (non-tape); '93' or user mode (tape).
  6 - channel scheduler flag (Figure 9).
  7 - job control flag (Figure 10).

The end of the table is denoted by an 'FF'. The eight bytes labeled FOCL specify the first PUB on each of the eight possible channels (first on channel list). PUB's are referenced by the LUB's to which they are assigned. Bytes 2, 3, and 6 are modified by the Channel Scheduler routine. Bytes 5 and 7 are modified by Job Control. Other bytes remain constant.

USER-SPECIFIED MACRO PARAMETER	DEVICE TYPE BYTE IN PUB
1050A	X'00'
1050B	X'01'
1050C	X'02'
1050D	X'03'
1050E	X'04'
1050F	X'05'
1050G	X'06'
1050H	X'07'
2501	X'10'
2540R	X'11'
2520B2 or 2520B3	X'20'
2540P	X'21'
1442N2	X'22'
1442N1	X'30'
2520B1	X'31'
1403 or 1404	X'40'
1443 or 1445	X'41'
2400T7 or 2400T9	X'50'
UNSP or UNSPB	X'FF'

Figure 8. PUB Device Type

0 - Device busy
1 - Switchable device
2 - End of file for SYSRDR or SYSIPT
3 - I/O error recovery outstanding
4 - -------
5 - Device-end posting desired
6 - Burst/MPX device
7 - 7-Track tape device

Figure 9. PUB Channel Scheduler Flag

0-4 - Standard-assignment mode for 7-track tape; all bits one if device is not tape; all bits zero if device down.	
5 - Assigned to a system logical unit	
6 - Assigned to a programmer logical unit	
7 - -------	

Figure 10. PUB Job Control Flag

JIBTAB - JOB INFORMATION BLOCK TABLE

The number of entries is specified at system generation time. The table must hold all the temporary and alternate assignments that are needed at any one time. The entries are referenced by JIB pointers 0, 4, 8, etc.

Each JIB entry consists of 4 bytes:
  0 - alternate or stored standard PUB pointer.
  1 - '00' or stored standard LUB flag.
  2 - present assignment LUB flag and JIB flag (Figure 11).
  3 - chain field; bit-7-on defined as end-of-chain.

The byte FAVP contains the pointer to the next free entry (first available pointer). The free entries are chained together to FAVP as assignment entries are chained to a LUB.

JIB entries are referenced by LUB's to which temporary and alternate assignments are made. JIB chains are built by the ASSGN routine in Job Control.

0-3 - present assignment LUB flag bits
4 - -------
5 - stored standard JIB
6 - alternate JIB
7 - -------

Figure 11. JIB Flag Byte

TEBTAB - TAPE ERROR BLOCK TABLE

The TEB table appears if requested at system generation time. The number of entries is specified by the user. There should be one entry for each tape device in the system. The entries are referenced by TEB pointers 0, 1, 2, etc., in the PUB for each tape device.

Each TEB entry consists of 6 bytes:
  0 - error recovery retry count; 'FF' indicates a null TEB.
  1 - permanent read error count.
  2 - initial read error count.
  3 - initial write error count.
  4 - erase gap count.
  5 - noise record count.

The counters in the TEB are used by the tape error recovery routine. At end-of-job, Job Control resets each TEB to its null form 'FF 00 00 00 00 00'.

LUBTAB - LOGICAL UNIT BLOCK TABLE

The number of entries is specified at system generation time. A minimum of 15 and a maximum of 255 entries may be specified. To use a logical unit name, a corresponding LUB must appear in the table. PUB's may be assigned to LUB's at system generation time. At any time the programmer and operator may alter or add to these assignments through Job Control.

Each entry is referenced by a logical unit name, SYSXXX. The names are divided into two classes: system and programmer. Figure 12 shows the order of the names and the division between classes.

Each LUB entry consists of 2 bytes:
  0 - PUB pointer 0, 1, 2,----; 'FF' = null pointer.
  1 - LUB flag in first four bits; a JIB pointer 0, 4, 8,---- in the last four bits; 1 if no JIB attached.

The eight bytes labeled FICL and NICL specify the first LUB entry in each class and the number of entries in that class (first in class list and number in class list). Specifically:
- byte 0 - system first in class; must be '00'
- byte 1 - programmer first in class; must be '0A'
- byte 4 - system number in class; must be '0A'
- byte 5 - programmer number in class; specified by user.

System Logical Unit Class	SYSRDR SYSIPT SYSPCH SYSLST SYSLOG reserved SYSRES SYSSLB SYSRLB SYSUSE	required LUB's
Programmer Logical Unit Class	SYS000 SYS001 SYS002 SYS003 SYS004	
	SYS005 ...... SYS244	optional LUB's

Figure 12. Logical Unit Names

0 - with null PUB pointer, present assignment IGN rather than UA.
1 - present assignment standard.
2 - standard assignment UA.
3 - standard assignment IGN.

Figure 13. LUB Flag Byte

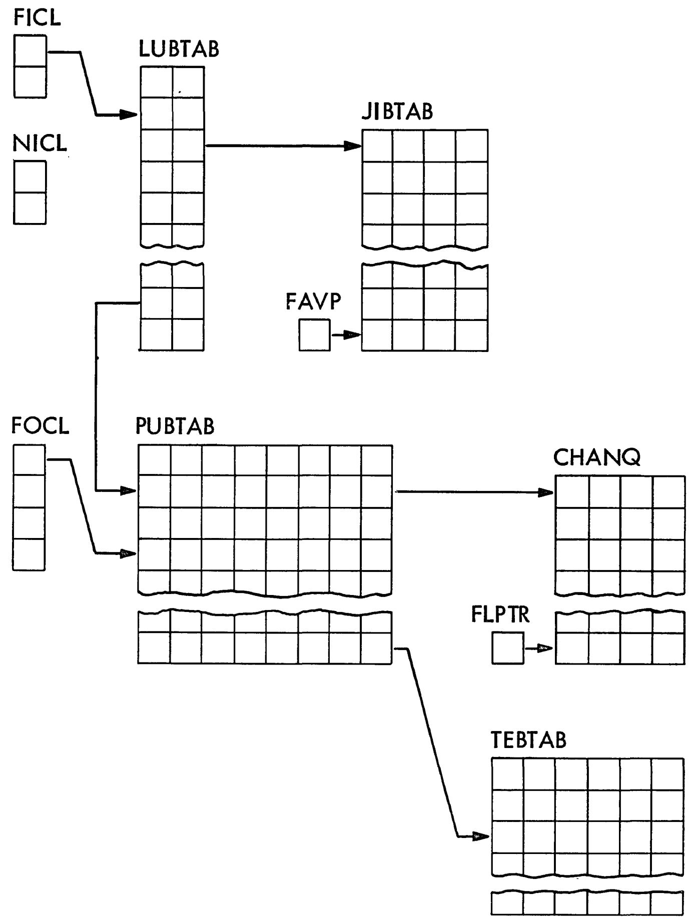


Figure 14. I/O Tables and Pointers

		System Generation Initialization of Tables		Sample Table	
	CHANQ	01 00 00 00	A request from SYSPCH	01 00 18 40	CHANQ
	1	02 00 00 00	awaits execution. There	03 00 18 70	1
	2	03 00 00 00	are no further requests from	FF 00 1E B0	2
6 Channel	3	04 00 00 00	SYSPCH waiting. Re-	04 00 00 00	3
Queue	4	05 00 00 00	quests from SYS002 and	05 00 00 00	4
Entries	5	FF 00 00 00	SYS003 have been com-	FF 00 00 00	5
	LUBPTR	00 00 00 00 00 00	pleted. REQID is not used	0C 0D 02 00 00 00	LUBPTR
	REQID	00 00 00 00 00 00	(no storage protection	00 00 00 00 00 00	REQID
	FLPTR	00	feature).	00	FLPTR
	FOCL	FF FF FF FF FF FF FF	All PUBs are being used to	00 04 FF FF FF FF FF	FOCL
	PUBTAB	FF 00 00 00 00 00 00 00	describe devices. There	00 0E FF 00 40 00 00 FC	PUBTAB
	1	FF 00 00 00 00 00 00 00	are six tape devices on	00 0C FF 00 11 00 00 FC	1
	2	FF 00 00 00 00 00 00 00	channel one - the user has	00 0D 02 00 21 00 80 FC	2
	3	FF 00 00 00 00 00 00 00	not changed the mode from	00 1F FF 00 00 00 00 FC	3
10 PUBs	4	FF 00 00 00 00 00 00 00	the assumed reset. SYSLOG	01 80 FF 00 50 93 00 92	4
	5	FF 00 00 00 00 00 00 00	is a 1052. SYSPCH is a	01 81 FF 00 50 93 00 92	5
	6	FF 00 00 00 00 00 00 00	2540. SYSLST is a 1403-	01 82 FF 00 50 93 00 92	6
	7	FF 00 00 00 00 00 00 00	1404. SYSRDR and SYSIPT	01 83 FF 00 50 93 00 92	7
	8	FF 00 00 00 00 00 00 00	are the same device, a	01 84 FF 00 50 93 00 9C	8
	9	FF 00 00 00 00 00 00 00	2540 (SYSIPT has been	01 85 FF 00 50 93 00 9C	9
		FF	temporarily assigned to a	FF	
			tape.)		
	FAVP	00	The standard assignment for	0C	FAVP
	JIBTAB	00 00 00 04	SYSIPT is stored in the first	01 41 04 01	JIBTAB
	4	00 00 00 08	JIB. Two alternate assign-	05 00 42 08	4
6 JIBs	8	00 00 00 0C	ments for SYS003 are in	06 00 02 01	8
	C	00 00 00 10	the second and third JIBs.	00 00 00 10	C
	10	00 00 00 14		00 00 00 14	10
	14	00 00 00 01		00 00 00 01	14
	FICL	00 0A FF FF 0A 05 00 00	SYSIPT and SYS003 have	00 0A FF FF 0A 05 00 00	FICL
15 LUBs	LUBTAB	FF 61 FF 61 FF 61 FF 61 FF 61	JIBs. SYSSLB, SYSRLB,	01 41 09 00 02 41 00 41 03 41	LUBTAB
	5	FF 61 FF 61 FF 61 FF 61 FF 61	SYSUSE, and SYS004 are	FF 61 08 41 FF 61 FF 61 FF 61	5
	A	FF 61 FF 61 FF 61 FF 61 FF 61	unassigned.	04 04 06 41 07 41 05 41 FF 61	A

Figure 15. I/O Sample Table

Entry	Byte	Definition
each CHANQ entry	0	CHANQ-pointer to next in queue; "FF" marks end of queue
	1	CCB address
	2	
	3	
each LUBPTR byte	0	LUB-pointer indicating logical unit requesting I/O
each REQID byte	0	code indicating program requesting I/O
each PUB entry	0	channel address; 'FF' is the null address
	1	unit address
	2	CHANQ-pointer; "FF" is the null pointer
	3	TEB-pointer or error counter; "00" is the null pointer
	4	device-type; see device-type table
	5	device-options; "00" (non-tape), "93" or user-mode (tape)
	6	channel-scheduler flag
	7	job-control flag
each alternate JIB entry	0	alternate PUB pointer
	1	"00"
	2	present assignment LUB-flag; alternate JIB flag
	3	JIB-pointer; bit 7 indicates end-of-chain
each stored standard JIB entry	0	standard-assignment LUB entry
	1	
	2	present assignment LUB-flag; stored-standard JIB flag
	3	JIB-pointer; bit 7 indicates end-of-chain
each TEB entry	0	retry count; "FF" is the null TEB indicator
	1	permanent read error count
	2	initial read error count
	3	initial write error count
	4	erase gap count
	5	noise record count
each LUB entry	0	PUB-pointer; "FF" is the null pointer
	1	LUB flag or, if JIB attached, JIB pointer

Figure 16. I/O Table Entries

SUPERVISOR NUCLEUS

The nucleus is entered whenever an interrupt occurs. The new PSW's shown in Figure 6 show the labels in the nucleus at which entry is made.

SVC Interrupt (SVCINT - Chart BA): An SVC 0 goes to the Channel Scheduler routine to make an entry in the channel queue table. Others fetch or load programs, transfer from routine to routine, alter the constant areas, and return to Job Control on a cancel or end-of-job condition.

External Interrupt (EXTINT - Chart BH): If provided by the user, the Supervisor gives control to a user routine.

Program Check Interrupt (PCHECK - Chart BH): If provided by the user, the Supervisor gives control to a user routine.

Machine Check Interrupt (SETACT - see Unit Check Exit Routines): The Supervisor enters a wait state with 'S' in location 1.

I/O Interrupt (INTERR - Chart AC): The Supervisor examines the status bits of the completed I/O and determines if another I/O operation should be started.

Furthermore, the Supervisor nucleus issues actual I/O commands after an SVC 0 if the I/O can be processed immediately, or after an I/O interrupt if more I/O awaits processing.

The routines are described in the order of their occurrence in the Supervisor:

- SVC 0 - Channel Scheduler
- Actual I/O
- I/O Interrupt
- Unit Check
- SVC 1-21, External, and Program Interrupts
- Tape Error.

Appendix D describes Physical IOCS fields that must be familiar to the reader: the 16K Tape CCB, the CCW, the PSW's, the CAW, and the CSW.

CHANNEL SCHEDULER - CHART AA

Objective: To indicate that an I/O request has been made by adding an entry to the channel queue table.

Entry: From the SVC Interrupt routine when an SVC 0 is identified.

Method: With the SVC 0 used to request I/O, a CCB is provided. The requesting program places the CCB address in register 1 and, in the CCB, sets:

- Options - byte 2, bits 3 and 5 and 7.
- Logical unit address - bytes 6 and 7.
- CCW address - bytes 9 and 10 and 11.

The Channel Scheduler resets the remainder of bytes 2, 3, and 4 to zeros.

This routine uses the logical unit address to locate the LUB and compute the LUB pointer. The LUB points to the PUB assigned to it. The I/O request is added to the channel queue for that PUB.

1. FLPTR replaces the end-of-queue pointer 'FF' in the PUB or in the chain field of the last entry in the queue for that PUB. This pointer references the new channel queue entry.
2. The chain field in the new entry replaces FLPTR.
3. The end-of-queue pointer 'FF' and the address of the CCB replace the fields in the new entry.
4. The LUB pointer is inserted in the correct byte of LUBPTR.

If the request is for SYSRES, bit 1 in JBCSW0 (displacement 56 of the communications region) is set on.

If the new I/O request is the first in the queue and the device is not busy (check bit 0 in the channel scheduler flag of the PUB), exit is to the Actual I/O routine to perform the I/O immediately. Otherwise, control returns to the requesting program. Registers 1-7 are saved and restored for that program.

ACTUAL I/O - CHART AB

Objective: To issue an SIO command, provided the channel is available.

Entries:

1. From the Channel Scheduler when an I/O request is made for an available device.
2. From the I/O Interrupt routine when a channel is freed for use and an I/O request for an available device on that channel is awaiting execution.

Method: On entry to this routine, the PUB table displacement for the device awaiting I/O must be in register 3. Register 2 is initialized to the device address (channel and unit) from the PUB.

A TCH command is issued to determine if the channel is available. If not, and the device is not switchable, an immediate exit is taken to the interrupted program. If the device is switchable (check bit 1 of the channel scheduler flag in the PUB), the next channel is also checked.

An SIO command is issued unless:

1. The device is for SYSRDR or SYSIPT and end-of-file has been encountered (input /& set bit 2 of the channel scheduler flag in the PUB). Instead control is given to the I/O Interrupt routine at GETCHQ to complete I/O and cancel the job.
2. The device is a burst/multiplex device in operation (check bit 5 of the configuration byte, displacement 53 in the communications region). Instead control returns to the interrupted program.

Before the SIO command is issued, the CAW must be set to the correct CCW address. If the device is tape, the set-mode CCW is the first to be executed (mode from PUB device-options byte). Otherwise, the CCW address is taken from the CCB addressed by register 1).

After the SIO command is issued, the condition code is tested:

- Condition code 0. The I/O operation is proceeding in a normal manner. This routine sets the device-busy indicator (bit 0 of the channel scheduler flag in the PUB). If the device is a burst/multiplex device, bit 5 of the configuration byte, displacement 53 in the communication region, is set on. Control returns to the interrupted program.

- Condition code 2. The channel is busy. Control returns to the interrupted program.

- Condition code 3. The channel is not operational. Control is given to the Unit Check routine at IONOP to signal an error and enter the wait state through the machine-check new PSW.

- Condition code 1. The I/O is complete and the status has been stored in the CSW. This routine tests all the status bits (32-47) of the CSW except 33, 35, 40, and 41. If none are on, control returns to the interrupted routine. If any are on, control goes to the I/O Interrupt routine (normal I/O completion).

Returning the interrupted program from this routine requires two checks:

1. Check for errors (ignore errors if interrupted program is the fetch routine - bit 1 in JBCSW0). If errors have been queued, complete pending I/O by putting the system in wait state and fetch the required Type-A transient routine.

2. Check for PIOCS cancel. If the first bit in the cancel code is on, turn it off and fetch the Cancel transient routine.

Return is through the I/O old PSW. Registers 1-7 are saved and restored for the interrupted program.

I/O INTERRUPT - CHART AC

Objective: To process the CSW bits from the completed I/O and to determine if another I/O operation should be started.

Entries:
1. From the interrupted program through the I/O new PSW.
2. From the Actual I/O routine when a command is completed immediately and status bits are set.

Method: If entry is through an I/O interrupt, registers 2 and 3 must be set. Register 2 must contain the device address (channel and unit) found in the I/O old PSW. Register 3 must contain the PUB displacement for the device; the section of the PUB table for the specified channel is searched for the specified device. If no PUB is found, no CSW bits are processed. An attempt is made to reschedule the channel at INITRG.

At GETCHQ, the CHANQ entry chained to the PUB is located and the address of the CCB placed in register 1. The CSW status bits are checked in a specified order (Figure 17). If no error exit is taken, an attempt is made to reschedule the channel at INITRG.

The section of the PUB table for the freed channel is searched for an I/O request. If one is found for an available device, the CCB address from the CHANQ table is put in register 1 and control goes to the Actual I/O routine. Otherwise, return is to the interrupted program through the Actual I/O return routine.

STATUS BIT	STATUS	ACTION
44 45 46	Channel data check Channel control check Interface control check	Takes I/O channel failure exit (CHFAIL) to signal an error and enter wait state through the machine check new PSW.
38 42 43 47	Unit check Program check Protection check Chaining check	Exits to unit-check routine (UNTCHK). If error is recoverable, returns to TSTUEX or SELECT.
32	Attention	Exits to the attention routine (ATTRTN). Unless interrupted program is the fetch routine, executes the MSGIN transient routine and returns to the interrupted program.
39	Unit exception	Turns on bits 36 and 37. Continues CSW status test.
36	Channel end	At CHNDRT: 1. Saves CSW information in the CCB. 2. If device assigned to SYSRDR or SYSIPT, checks input and sets: a. end-of-file bit 2 in PUB channel scheduler flag on / &. b. flag bit 17 in CCB on /* or / &. c. unit exception bit 39 in CCB on /* or / &. 3. If device end, exits to that routine, step 2b. 4. Sets device busy bit 0 in the PUB channel scheduler flag. 5. If device-end-posting desired, exits to device-end, step 2d. 6. Sets device-end-posting desired bit 5 in the PUB channel scheduler flag. 7. Exits to attempt to reschedule the channel. No attempt is made for a multiplex channel unless this is a burst/multiplex device.
37	Device end	1. Turns off device busy bit 0 in the PUB channel scheduler flag. 2. If device-end-posting desired, a. saves CSW status bytes in the CCB. b. turns off device-busy bit 0 and c. turns off device-end-posting-desired bit 5 in the channel scheduler flag. d. zeros PUB error counter unless interrupted program is an error routine. e. turns on traffic bit 16 in the CCB. f. updates PUB CHANQ-pointer to next request. 3. Exits to attempt to reschedule the channel. Only an attempt to reschedule the device is made for a multiplex channel unless this is a burst/multiplex device.

Figure 17. I/O Interrupt: CSW Testing

UNIT CHECK - CHART AD

Objective: To determine action taken on an I/O interrupt with:

- unit check
- program check
- protection check
- chaining check.

Entry: From the I/O Interrupt routine when any of these checks occur.

Method: Before determining the error exit, device-end and channel-end conditions are tested. If device-end or control-unit end and not channel-end, the error flag indicates "ignore" rather than "retry".

If the user has provided his own unit check routine, control returns to the I/O Interrupt routine with bit 18 on in the CCB. PIOCS ignores the error.

If PIOCS must process the error, an error queue entry is built. Only 5 entries can be handled at once; an attempt to build more entries creates a cancel condition. Each entry has 18 bytes:

- 0-7: CSW, first byte zero.
- 8-9: address of PUB table entry for device in error.
- 10-15: sense information.
- 16: error flag (ignore/retry/message-writer).
- 17: message number.

If the error was either program check or protection check, exit is to the common routine EXPRG. The error flag is set to call the message writer and, unless the error is on SYSRES, control returns to the I/O Interrupt routine to await completion of pending I/O requests before fetching the message writer. If the error is on SYSRES or from a fetch on SYS000, the wait state is entered.

If the error was either unit check or chaining check, exit is either to the resident error routine (device is tape) or back to the I/O interrupt routine to await completion of pending I/O requests before fetching the device error routine.

Unit Check and Error Recovery Exit Routines

EXCON, EXCON 1, EXCON 2 - continue exit:

1. Restore CSW from error queue entry.
2. Release error entry.
3. If device-end or control-unit end, ignore error and continue testing CSW in I/O Interrupt routine (at TSTUEX).
4. If not, set device-end and channel-end. Ignore error and continue (at TSTUEX).

EXRTY - retry exit:

1. If error flag in error queue entry does not indicate retry, go to EXCON1.
2. Release error entry.
3. Save registers 2 and 3 and go to STRTIO in the Actual I/O routine to retry I/O request.

EXSIO - start I/O exit:

1. Initialize CAW from register 6.
2. Release error entry.
3. Go to STRTIO+6 in the Actual I/O routine to issue an I/O request.

EXPRG - program check exit:

1. Turn off retry/ignore bits in error flag byte of error queue entry.
2. Take equipment check exit.

EXEQU - equipment check exit:

1. Set error flag of the error queue entry to call the Message Writer.
2. If error is on SYSRES or occurred during a fetch from SYS000, enter wait state with error codes in locations 0-3.
3. If not, restore registers and return to SELECT in the I/O Interrupt routine. After completing the I/O awaiting execution, the Message Writer is fetched to process the error.

IONOP - device not operational:

1. Set error code in location 0 ('02') and machine check new PSW ('3F').
2. Save device address in I/O old PSW.
3. Enter wait state.

CHFAIL - I/O channel failure:

1. Set error code in location 0 ('01') and machine check new PSW ('0F').
2. Enter wait state.

SETACT - machine check interrupt:

1. Put 'S' in location 1.
2. Enter wait state.

Exit Subroutines

GETENT: Returns address of desired error queue entry.

DEQUER: Removes entry from error queue.

RSTREG: Restores registers for return to I/O routines.

ERRSIO: Executes error-recovery channel programs disabled for all system interruptions. If error occurs, takes program check exit EXPRG with error-on-recovery message.

HEADQ (in Tape Error Recovery): Inserts an I/O request at the beginning of a channel queue.

SUPERVISOR CALL INTERRUPT - CHARTS BA-BF

Objective: To analyze the SVC code and transfer control to the proper SVC routine.

Entry: From the interrupted program through the SVC new PSW.

Method: The interrupt routine checks for a SVC of 0 and, if present, it branches directly to the Channel Scheduler to handle the I/O command. The SVC old PSW (bytes 32-39) is checked to determine which program issued the SVC and whether the SVC was legal.

The SVC code in the old PSW is used to enter a branch table of SVC codes and routine addresses. The address of the specified SVC routine is picked up and control goes to the SVC routine.

SVC 1 - Chart BB: Loads and executes a problem program phase. The Fetch subroutine (Chart BG) loads the problem program phase. SVC 1 determines the entry point from the phase header label or from a parameter in register 0 and places it in USRPSW. The entry point is checked to make sure it is beyond the supervisor area. The phase entry address is loaded as the current PSW, and execution of the new phase is started.

SVC 2 - Chart BB: Loads and executes a Type-B (synchronous) transient. Type-B transients may be called by the problem program, a Type-A transient, or another Type-B transient. The transient-active switch is turned on, and the old SVC PSW is saved. The Fetch subroutine (Chart BG) is entered at LTFETC, and it loads the Type-B transient routine called for. LTRPSW contains the starting address for all Type-B transients. This address is loaded as the current PSW, and execution of the transient is started.

SVC 3 - Chart BB: Loads and executes a Type-A (asynchronous) transient. Type-A transients may be called by another Type-A transient but not by a Type-B transient or the problem program. The old I/O PSW is saved, and the Type-A transient-fetch switch is turned on. After getting the phase name, the Fetch subroutine (Chart BG) is entered at PTFETC to load the Type-A transient routine. The Type-A transient-fetch switch is turned off and the old SVC PSW is restored. The load address of the physical transient is returned in base register 15 by the Fetch subroutine. Branching to 8 bytes beyond it starts execution of the Type-A transient.

SVC 4 - Chart BB: Loads a problem program phase, and returns to the calling routine without executing the phase loaded. The Fetch subroutine (Chart BG) is used to load the phase. The entry point of the new phase is returned to register 1. The SVC old PSW becomes the current PSW and control is returned to the calling routine.

SVC 5 - Chart BB: Places information designated by the user (MVCOM macro) into the communication region. The address of the information to be moved is placed in register 13. The address of the communications region is put in register 14. The information the user wanted to put in the communications region is moved into it. The SVC old PSW becomes the current PSW, and control is returned to the calling routine.

SVC 6 - Chart BC: Cancels the problem program. A cancel code of hex 12 is set in the communications region, displacement 88. The Type-B transient-active switch is turned off. An SVC of 2 fetches the $$BCNCL routine. On return, register 1 contains the starting address of Job Control which is placed in JCTPSW. JCTPSW becomes the current PSW.

SVC 7 - Chart BC; Enters the wait state until the completion of the I/O instruction in progress. The second byte of the CCB specified by the user in register 1 (WAIT macro) is tested to see if there is any I/O operation going on. If I/O is in operation, bit 14 of the old SVC PSW is turned on to enter the wait state. If I/O was not in operation, the old SVC PSW is not changed. Making the old SVC PSW the current PSW will either put the system in wait state until an interrupt is encountered (normally I/O) or return it to the calling routine.

SVC 8 - Chart BC: Transfers control to the user after execution of an OPEN, CLOSE, or EOV transient routine. The old SVC PSW is saved and the return entry point that was stored in register 14 is put in USRPSW. After checking the return address to make sure it is beyond the supervisor area, USRPSW becomes the current PSW, and control is returned to the user program.

SVC 9 - Chart BC: Transfers control from the problem program to the transient OPEN, CLOSE, or EOV routines. The address of the transient routine is put in RTNPSW by SVC 8 when control is transferred to the user. SVC 9 makes RTNPSW the current PSW, transferring control back to the transient routine.

SVC 10 - Chart BC: Requests an interrupt after a specified time interval. The user has the option of deciding when to have timer interrupts. When a SVC 10 occurs, the time for the next interrupt set by the user in register 1 (SETIME macro) is stored in hex 50. The new time of day is set in hex 54. The old SVC PSW becoming the current PSW transfers control back to the user's program.

SVC 11 - Chart BD: Gives the return to the problem program from a Type-B transient. The Type-B transient-active switch is turned off, and registers 2-15 are restored. LTAPSW, which contains the reentry address to the problem program, becomes the current PSW, and control is returned to the user.

SVC 12 - Chart BD: Turns off bits specified by register 1 in the second Job Control switch (JBCSW1), decimal displacement 57 in the communications region. Registers 8-15 are restored, and the old SVC PSW becoming the current PSW returns control to the calling sequence.

SVC 13 - Chart BD: Turns on bits specified by register 1 in the second Job Control switch (JBCSW1), decimal displacement 57. Registers 8-15 are restored, and the old PSW becoming the current PSW returns control back to the calling sequence.

SVC 14 - Chart BD: Used for end-of-job processing. The cancel code is reset to indicate a normal EOJ, and the Type-B transient-active switch is reset. An SVC of 2 fetches the $$BCNCL routine with a cancel code of 00. On return, register 1 contains the starting address of Job Control, which is placed in JCTPSW. JCTPSW becomes the current PSW, transferring control to Job Control.

SVC 15 - Chart BD: Used to test the tape position of SYSRES when accessing the relocatable and source-statement libraries. SVC 15 tests bit 1 of the first Job Control switch (JBCSW0), decimal displacement 56 in the communications region, to see if the tape-moved-bit indicator is on. If it is not on, the routine returns to the calling sequence to reposition the tape back to the library desired. If the tape-moved bit is on, the old SVC PSW is saved, and an SVC 0 is issued to execute the CCB the user sets up for SYSRES. The old SVC PSW becomes the current PSW and control is returned to the calling sequence.

SVC 16 - Chart BE: Sets up an entry to the user's program check routine.

SVC 17 - Chart BE: Provides the return to the problem program from the user's program check routine.

SVC 18 - Chart BE: Sets up the entry to the user's timer routine.

SVC 19 - Chart BE: Provides the return to the problem program from the user's timer routine.

SVC 20 - Chart BF: Sets up an entry to the user's external interrupt routine.

SVC 21 - Chart BF: Provides the return to the problem program from the user's external interrupt routine.

ILSVC - Chart BF: The routine that handles any errors in the supervisor calls. The transient-active switch is turned off and $$BCNCL is fetched to cancel the problem program and call Job Control.

FETCH SUBROUTINE - CHART BG

Objective: To load a transient routine or problem-program phase.

Entry: From SVC 1, SVC 2, SVC 3, or SVC 4.

Method: The tape that contains the phase (SYSRES or SYS000) is searched in record format, looking at the first 61 bytes of each record for the desired routine. When the correct header label is read, the phase is read into storage. When the phase is loaded, the program returns to the SVC routine.

EXTERNAL INTERRUPT - CHART BH

Objective: To establish the cause of the external interrupt and to service it.

Entry: From the interrupted program through the external new PSW.

Method: An external interrupt may be caused by:

- a timer interrupt.
- the external interrupt key on the console.
- an external interrupt signal.

Timer Interrupt: The timer is reset to its maximum value, and the new time of day is set. The addresses of the user's timer routine and save area are set in registers 12 and 13.

External Interrupt Key: The addresses of the user's key routine and save area are set in registers 12 and 13.

For either interrupt, the routine and save-area addresses are checked to be sure they are within the problem-program area. The address of the user's routine is set in

SVPSW, and making it the current PSW transfers control to the user's external-interrupt key routine.

If the user's routine address or save area are not within the problem program area, SVPSW contains the address of the interrupt, and control is transferred back to the problem program.

External Interrupt Signal: Control returns to the user. The external interrupt signal is not supported in this program.

PROGRAM CHECK INTERRUPT - CHART BH

Objective: To establish whether the user has a program check routine and transfer to it, or abort using the IBM routine.

Entry: From the interrupted program through the program check new PSW.

Method: Control goes to the IBM program-check routine if the user does not provide his own routine, or if his routine does not have a legal address. Otherwise, control goes to the user program-check routine.

User Routine: The addresses of the user's program check routine and save area are put in registers 12 and 13. These addresses are checked for legality at CHK1. The address of the user program-check routine is set in SVPSW. SVPSW becoming the current PSW transfers control to the user's program check routine.

IBM Routine: The program-check code corresponding to the interruption code is set in the cancel code, byte 88 of the communications region. The Type-B transient-active switch is turned off and a supervisor call of 2 fetches $$BCNCL. $$BCNCL calls the $$BPCHK program-check transient, which prints the interrupt and job cancel message. If requested, the dump transient will dump the registers and the problem program area. When all requested information is received, the program exits calling Job Control.

TAPE ERROR RECOVERY - CHARTS CA-CD

Objective: To test sense data to determine the tape error recovery procedures.

Entry: From the unit-check routine when a unit check or chaining check occurs on a tape device.

Method: Just before entry to this routine, a sense command is issued, initializing bytes 10-15 of the error queue entry with sense information. This routine saves the address of the CCW causing the interruption in preparation for retrying the I/O in error.

The sense bits are then tested in the order shown in Figure 18. If a retry can be executed, the retry exit (EXRTY) or the reposition and retry exit (EXSIO) is taken. Otherwise, the correct error exit is provided. If TEB has been specified, error counts are updated.

SENSE BIT	ERROR TYPE	RETRY PROCEDURE	ERROR EXIT (see Unit Check)
Byte 0 Bit 2	Bus Out Check	Reposition tape and retry eight times. Turn off retry flag.	EXEQU
Byte 0 Bit 3	Equipment Check		EXPRG
Byte 0 Bit 1	Intervention required		EXCON if channel end, or EXEQU
Byte 0 Bit 5	Overrun	Reposition tape and retry eight times. Turn off retry flag.	EXEQU
Byte 0 Bit 4	Data Check	1. Control Command - Retry 15 times.	EXEQU
		2. Read Command - Turn on "selected error" flag.	
		a. Retry if record length is less than 12 and the sense bit indicating noise (byte 1 bit 0) is off.	
		b. Otherwise retry 100 times with repositioning performing CRC correction and tape cleaning every eight retries. Turn on ignore flag. Turn off retry flag.	EXEQU
		3. Write Command - Backspace erase and retry 15 times. Turn off retry flag.	EXEQU
Byte 0 Bit 7	Data Converter Check	Post bit 4 in CCB byte 3.	EXCON
Byte 0 Bit 0	Command Reject		EXEQU if write command and file protect, or EXPRG
CSW status byte	Chaining Check	Reposition tape and retry eight times. Turn off retry flag.	EXEQU
no sense bits			EXEQU

Figure 18. Tape Error Recovery Procedures

PHYSICAL IOCS ERROR TRANSIENT ROUTINES

This group of Type-A transient routines is used by the Supervisor nucleus to process I/O errors on devices other than the resident device (tape). These routines issue an error message and determine the error exit.

The Supervisor communications region displacement 98 contains the address of a block of information in the unit-check routine providing these transients with the addresses of the error queue entry and the error exits.

The routines reside in the core image library. The name of each routine begins with the characters $$A. Each routine is read into the Type-A transient area by the Supervisor nucleus (SVC 3).

MESSAGE WRITER - CHARTS CE-CG

Objective: To write out error information after an I/O error has been processed.

Entry: From the Actual I/O routine when I/O has been completed and there are error queue entries to be processed.

Method: The Message Writer consists of four overlays which build an error message, write it on SYSLOG, and if required, accept an operator response.

Phase 1 ($$ANERRM) of the Message Writer stores message information in the last 20 bytes of the Type-A transient area as follows.

Bytes 0-3: the CCB address from the channel queue entry, if obtainable.

Bytes 4-5: a message number taken from the error queue entry and converted to decimal.

Byte 6: operator's action indicator I (information) or A (action).

Byte 7-8: target indicators C (cancel), R (retry), and I (ignore).

Bytes 9-18: 10-character message.

If neither the retry nor the ignore bits are on in the error queue entry, the indicators are set to I and C. If requested by the user, device-end and selected errors are posted in the CCB. Exit is to fetch Phase 2.

Phase 2 ($$ANERRN) uses the last 116 bytes of the Type-A transient area to build the error message. In addition to the information provided by Phase 1, the message contains the following.

- The logical unit name and the device address from the CCB.
- The command code from the CCW just executed.
- The command address, status, and count from the CSW.
- The sense bytes from the error queue entry.
- The address of the CCB.

Figure 19 is a sample error message. Exit is to fetch Phase 3.

Phase 3 ($$ANERRO) writes the error message on SYSLOG, if SYSLOG is assigned. If operator action is expected, exit is to fetch Phase 4. If the message is for information, the continue exit is taken (see EXCON in the Unit Check routine). If the target indicator is cancel rather than ignore, the cancel switch is set before exit.

Phase 4 ($$ANERRP) allows operator communication with the error recovery routine. Locations 0-3 are set with the error message number, the operator action indicator, and the device address. A new PSW that allows return to this phase is set in the external new PSW, and wait state is entered enabled only for external interrupts.

When an interrupt occurs, control returns to this phase. A test is made for the cause of the interrupt. Any cause other than a key interrupt is saved for later processing. (The I/O old PSW and the external old PSW are interchanged so that when the I/O interrupt is completed, the external interrupt will be processed before returning to the user.)

When a key interrupt occurs, a test is made to determine if SYSLOG is a 1052. If not, the operator is expected to have placed his response in location 4 during the wait state. Valid responses are:

X'01'	retry
X'02'	ignore
X'03'	cancel

If no valid response is present, the wait state is reentered.

If SYSLOG is a 1052, a response is accepted from the operator. Valid responses are retry, ignore, and cancel. An invalid response results in an error message, and a new operator response is accepted.

A cancel response sets the cancel switch and takes the continue exit (see EXCON in the Unit Check routine). An ignore response is invalid for certain errors. If valid, it takes the continue exit. The retry response is also invalid for certain errors. If valid, it clears the CCB communications bytes and the retry counter and takes the retry exit (see EXRTY in the Unit Check routine).

OP11A	IR DATA CHECK SYS001 = 190 CCSW = CCW1W2W3W4W5W6W7 SNS = S1S2S3S4S5S6 CCB = AAAAAA
OP	standard message code
11	message number for DATA CHECK
A	operator's action indicator
IR	ignore and retry responses allowed
SYS001	logical unit in error
190	address of device in error
CC	command in error
W-W	CSW information
S-S	sense information
A-A	CCB address

Figure 19. Error Message from the Message Writer

DEVICE ERROR RECOVERY - CHARTS CH-CK

Objective: To test sense data to determine the error recovery procedures for nontape devices.

Entry: From the Actual I/O routine when I/O has been completed and there are error queue entries to be processed.

Method: The device-error routines consist of two overlays that test the sense information and determine the error exit.

Phase 1 ($$ANERRU) tests for unknown devices and impossible sense information. If valid error information is found, exit is to Phase 2. Otherwise the program check error exit (see EXPRG in the Unit Check routine) is taken. Figure 20 shows the devices supported and the sense bits recognized on them.

Phase 2 ($$ANERRV) tests the sense information in the order shown in Figure 21. If a retry can be executed, the retry exit is taken (see EXRTY in the Unit Check routine). Otherwise the correct error exit is provided.

Device \ Sense Bits	0	1	2	3	4	5	6
1052	X	X	X	X			
2501	X	X	X	X	X	X	
2540R	X	X	X	X	X		X
2520P	X	X	X	X			
2540P	X	X	X	X	X		X
1442P	X	X	X	X			
1442 R/P	X	X	X	X	X	X	
2520 R/P	X	X	X	X	X	X	
1403	X	X	X	X	X		
1443	X	X	X	X			

Figure 20. Devices Supported by Device Error Recovery

Sense Bit	Error Type	Retry Procedure	Error Exit (see Unit Check)
3	Equipment Check	2540 Reader: 2540 Punch: Turn on ignore; turn off retry. 1052: Retry once; turn on ignore. 1403/1443: Turn on ignore. other:	EXEQU EXEQU EXEQU EXEQU EXEQU
1	Intervention Required	 1052: Turn on ignore.	EXCON if channel and device end; otherwise EXEQU EXEQU
2	Bus Out Check	1052: Retry once. Retry if neither channel nor device end. Otherwise exit.	EXEQU EXEQU
4	Data Check	1403: Turn on ignore; turn off retry. other:	EXEQU EXEQU
5	Overrun		EXEQU
0	Command Reject		EXPRG
6	Unusual Command Sequence		EXCON
CSW Status byte	Chaining Check		EXEQU
no sense bits			EXEQU

Figure 21. Device Error Recovery Procedures

SUPERVISOR TRANSIENT ROUTINES

This group of Type-B transient routines is used by the Supervisor nucleus to perform services on request. These services may be requested by the programmer (PDUMP), by Job Control (RSTRT), or by the Supervisor nucleus itself (PCHK).

The routines reside in the core image library. The name of each routine begins with the characters $$B. Each routine is fetched into the Type-B transient area by the Supervisor nucleus (SVC 2). Normal return to the Supervisor nucleus is an SVC 11. However, several routines exit to other areas of the nucleus.

The routines are described in alphameric order as they appear in the core image library. Note that some Type-B transients (for example, $$BOPEN) not included in this group are described in the IBM System/360 BOS Logical IOCS 16K Tape PLM, Form Z24-5018.

CHECKPOINT ($$BCHKPT) CHART DA

Objective: To write a checkpoint on an output tape.

Entry: Called by a CHKPT macro instruction.

Method: A checkpoint is a group of records containing all necessary information to duplicate the status of a problem program on request (RSTRT). The macro provides this routine with the field shown in Figure 22 (address in register 0).

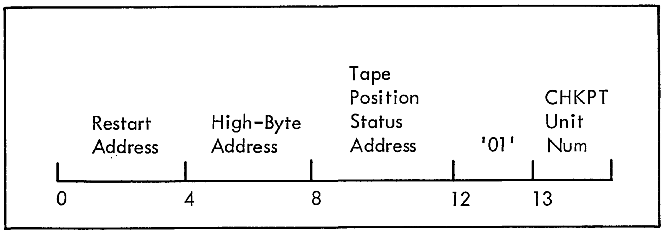


Figure 22. Checkpoint Field

The routine adds one to the serial number of the last checkpoint (displacement 92-93 of the communications region). The sum is used as the checkpoint identification number of the checkpoint being written.

The routine first writes a header on the device specified by the checkpoint unit number. The header is 20 bytes long.

1. Bytes 1-12 contain /// CHKPT //.
2. Bytes 13-14 contain the number of program records.
3. Bytes 15-16 contain the length of the last program record.
4. Bytes 17-20 contain the checkpoint record identification number (1, 2, 3, etc.).

The routine then writes a 123-byte record saving restart information.

1. Bytes 1-4 contain the address of the last byte in the checkpoint.
2. Bytes 5-64 contain saved general registers.
3. Bytes 65-68 contain the address of the beginning of the CHKPT macro expansion.
4. Bytes 69-123 contain information saved from the communications region.

Program records follow in blocks of 32,768 bytes (32K). If the last program record is less than 100 bytes, it is made to be 100 bytes long by padding. The last record is a trailer 20 bytes long. The trailer record is identical to the header record.

When a checkpoint is taken, the message; CHKPT XXXXX HAS BEEN TAKEN, is printed. If the unit for checkpoint is not a tape, the message: UNIT FOR CHKPT NOT A TAPE - CHKPT IGNORED, is printed. Return is to the calling routine.

CANCEL ($$BCNCL) CHART DB

Objective: To determine the message and exit for a preset cancel code in the Supervisor.

Entry: Called by the CANCEL macro or operator command (see $$BMSGIN), or by the Supervisor.

Method: When this routine is entered, a cancel code of one byte has been placed in the communications region (displacement 88). This code is used to determine the action taken by this routine. The null code is '00'. The legal codes are shown in Figure 22.

Before the cancel code is analyzed, all PUBS are checked for I/O completion. If the null code is present, there is an immediate exit to the calling routine after loading Job Control into the problem program area.

The routine then analyzes the cancel code and takes the action shown in Figure 23. The cancel code and the CATAL switch in the communications region are reset. If the dump switch in the communications region is on, exit is to the dump routine $$BDUMP. Otherwise, Job Control is loaded into the problem program area and exit is to the calling routine.

Code	Condition	Action
01 through 0F	Program check	Fetch $$BPCHK
10	Illegal SVC	Fetch $$BILSVC with register 0 at 1.
11	Phase not found	Fetch $$BILSVC with register 0 at 0.
12	Operator intervention	Write message
	Programmer request	Turn off dump switch and write message
14	Invalid CCB address	Write message
15	Undefined logical unit	Write message
16	Device not assigned	Write message
17	Reading past /& statement	Write message
18	I/O error	Write message
19	I/O operator option	Write message
1A	I/O error queue overflow	Write message

Figure 23. Cancel Codes

DUMP ($$BDUMP) CHARTS DC AND DD

Objective: To give a dump on SYSLST (printer or tape) of:

1. General registers
2. Floating point registers
3. Communications region
4. Supervisor and problem program area.

Entry: Called by a DUMP macro or by the Supervisor.

Method: After initialization for the specified device, the subroutine PGHD (Chart DD) writes a header. The number of lines per page and the heading-line information are specified in the Supervisor.

Print lines, in which all words are equal, are condensed as follows:

1. The first word is printed.
2. The second is replaced by --SAME--. The remainder of the line is blank.
3. No spacing or printing occurs for all equal lines that follow.
4. Printing and storage-location designations continue when unequal adjacent words are encountered.

The last line is printed, equal or not.

DUMP exits to Job Control for EOJ when SYSLST is unassigned or the dump is complete. In case of an end-of-reel condition before DUMP is complete on SYSLST, $$BEOVRT is called.

END OF VOLUME ($$BEOVRT) CHART DK

Objective: To cancel the job when a tape end-of-volume condition is encountered on SYSLST or SYSPCH.

Entry: Called by another transient or a problem program not using logical IOCS.

Method: A message is written on SYSLST indicating job cancellation because of an end-of-volume condition, and giving the device address of the unit at end-of-volume. The unit (SYSLST or SYSPCH) is closed by writing a tape mark, an end-of-volume trailer, and two tape marks, and by rewinding and unloading the tape. The unit at end-of-volume is unassigned.

A similar message is written on SYSLOG. The cancel bit and the operator-pause bit are set on in the Job Control switch bytes in the Supervisor communications region. Exit is to Job Control with an SVC 14.

ILLEGAL SVC MESSAGE ($$BILSVC) CHART DE

Objective: To write a message on SYSLOG and SYSLST when a phase has not been found or an invalid SVC code has been detected.

Entry: Called by $$BCNCL when a phase has not been found or an invalid SVC code is detected.

Method: This routine builds the phase-not-found message or the invalid-SVC message and prints it on SYSLOG and SYSLST (printer or tape) along with a job-cancel message. It sets the cancel switch for Job Control and exits calling Job Control or the dump routine ($$BDUMP).

JOB CONTROL OPEN FOR TAPE ($$BJCOPT) CHART DF

Objective: To open I/O tape units necessary for system operation.

Entry: Called by Job Control when:
1. SYSRDR, SYSIPT, SYSPCH, or SYSLST is assigned to a tape unit.
2. the option LINK is present to open SYS000. Called to open SYS001 and SYS002.

Method: After initializing the CCB and CCW, the PUB for the device to be opened is located for the device address. A rewind command is issued.

If the tape is an output file, the routine checks that the tape is not file-protected. In any case the first record is read. The first read is in standard mode. If an I/O error occurs, a read in user mode is issued. If an error occurs again, a read with retries is issued in the standard mode.

If a unit exception but not a tape mark occurs, control returns immediately to the calling routine. If anything else but a VOL1 label is read, the tape is rewound. If it is an input file, control returns to the calling routine. If it is an output file, an error has occurred.

When a VOL1 label is read on an input file, the tape is positioned beyond all labels and control returns to the calling routine.

When a VOL1 label is read on an output file, the tape is positioned beyond all VOL labels and checked for an HDR1 label and an inactive condition. An HDR1 record and a tape mark are written on the file. If there are no VOL1 labels on the file, no HDR1 record is written. Control returns to the calling program.

When an error occurs because an output tape is file-protected, an output tape has no label, or an output tape still active, control goes to the routine ERROUT, which lists a message on SYSLOG. If SYSLOG is not a 1052, the cancel switch is set and return is to EOJ in Job Control. If SYSLOG is a 1052, the operator may type a response of RETRY (rewind and return to sense file protect) or IGNORE (write HDR1 label and tape mark and return to the calling routine). IGNORE is not a valid response for the error condition of attempting to open a file-protected output file.

MESSAGE INPUT ($$BMSGIN) CHART DG

Objective: To process the following operator commands:
- Cancel
- Pause
- Log/Nolog

Entry: Called by the I/O Interrupt routine on an attention signal.

Method; SYSLOG must be assigned a 1052. If not, control returns to the calling routine. Otherwise, a ready message is written on SYSLOG and an operator entry is read.

If no entry is made, control is returned to the calling routine. A cancel entry fetches the cancel routine ($$BCNCL). Pause, log, and nolog entries condition a bit switch for EOJ pause, or a bit switch for SYSLOG printout, and return to read another operator command. All other entries are illegal. An error message is printed on SYSLOG and a correct command can be issued.

PROGRAM CHECK ($$BPCHK) CHART DH

Objective: To print a message on SYSLOG and SYSLST when a program check interruption occurs.

Entry: Called by $$BCNCL when a program check interruption occurs.

Method: This routine builds the interrupt and job-cancel message, and prints on SYSLOG and SYSLST (printer or tape). It sets the cancel switch for Job Control and exits calling Job Control or the dump routine ($$BDUMP).

PROGRAM DUMP ($$BPDUMP) CHARTS DC AND DB

Objective: To give a dump on SYSLST (printer or tape) of:
1. General registers
2. Floating point registers
3. Requested portion of main storage pointed to by register 0.

Entry: Called by a PDUMP macro.

Method: After initialization for the specified device, the subroutine PGHD (Chart DB) writes a header. The number of lines per page and the heading-line information are specified by the Supervisor.

Print lines, in which all words are equal, are condensed as follows:
1. The first word is printed.
2. The second is replaced by --SAME--. Remainder of the line is blank.
3. No spacing or printing occurs for all equal lines that follow.
4. Printing and storage location designations continue when unequal adjacent words are encountered.

The last line is printed, equal or not.

PDUMP exits to Job Control for EOJ when SYSLST is unassigned or the dump is complete. In case of an end-of-reel condition before DUMP is complete in SYSLST, $$BEOVRT is called.

RESTART ($$BRSTRT) CHART DJ

Objective: To load a checkpoint from tape, and to give control to the new problem program.

Entry: Called by Job Control.

Method: A checkpoint is a group of records containing all the necessary information to duplicate the status of the problem program at a given time. Job Control provides this routine with a PUB address in register 2 and the unit number and checkpoint identification number in register 3. The user is responsible for rewinding all tapes.

To locate the checkpoint, the tape unit specified in the PUB is read until a checkpoint header with the correct checkpoint identification number is encountered. The job name in the saved communication region record must be the same as the current job name. The size of the current problem program area must be equal to or larger than that of the checkpointed problem program area. The size of the current Supervisor must be equal to the size of the Supervisor for the checkpointed program.

When the correct checkpoint is found:
1. The communication region is restored.
2. The problem program is restored.
3. Each logical IOCS file is repositioned using the block count as specified in the DTF. A table of DTF addresses for the files to be repositioned must be maintained by the user.
4. Each physical IOCS file is repositioned using a logical unit address, tape mark count, and record count. A table of this information must be maintained by the user.
5. The PSW is prepared with the restart address.
6. The general registers are restored after returning to Supervisor control.

MACRO ROUTINES

Three types of macro instructions are included in the 16K tape BOS:

- Logical IOCS macros.
- System Generation macros.
- Supervisor Communication macros.

A description of the logical IOCS macros is given by the IBM System/360 BOS Logical IOCS 16K Tape PLM, Form Z24-5018.

SYSTEM GENERATION MACROS

These routines enable the user to build a Supervisor according to specifications. Required macros - SUPVR, CONFG, JCOPT, IOTAB, PIOCS, and SEND - are the various sections of the Supervisor. Optional macros - DVCGEN, ASSGN, and OPTION - initialize tables and flags. Figure 24 shows the order in which the macros are given and the sections of the Supervisor they build.

Macro	Action
SUPVR	Builds System/360 Constant Area
CONFG JCOPT	Build Communications Region
IOTAB	Builds Null I/O Tables
DVCGEN	Inserts PUB table entries
ASSGN	Inserts LUB table entries
OPTION	Sets flags to include specified interrupt routines
PIOCS	Builds Physical IOCS routines (Channel Scheduler, Actual I/O, I/O Interrupt, and Unit Check)
SEND	Builds other interrupt routines and the resident error routine. Sets the beginning of the problem program area.

Figure 24. Supervisor Generation Macros

Operation	Operand
SUPVR	TAPE

Generates PHASE and CSECT statements.

Generates constants to initialize the System/360 constant area. These are zeros except for address of communications region, timer and time of day, and PSW's.

Operation	Operand
CONFG	MODEL=nn, SP=YES/NO, DEC=YES/NO, FP=YES/NO, TIMER=YES/NO, CHANSW=YES/NO, BURSTMX=YES/NO, TEB=YES/NO

Determines the origin of the communications region.

Generates a section of the communications region through the machine configuration byte (displacement 52). The last byte is initialized from SP, DEC, FP, TIMER, CHANSW, BURSTMX.

Sets a switch to indicate if the TEB option is present.

Operation	Operand
JCOPT	DECK=YES/NO, LIST=YES/NO, LISTX=YES/NO, SYM=YES/NO, XREF=YES/NO, ERRS=YES/NO, CHARSET=48C/60C, LOG=YES/NO, DUMP=YES/NO, LINES=nn, DATE=MDY/DMY

From DATE sets configuration byte (displacement 53).

From DECK, LIST, LISTX, SYM, XREF, ERRS, and CHARSET provides standard setting of the third Job Control switch (see JBCSW2 in Job Control) in displacement 54.

From DUMP and LOG provides standard setting of the fourth Job Control switch (see JBCSW3 in Job Control) in displacement 55.

Generates the remainder of the communication region. The address of the TEB table is given only if TEB is specified in the CONFG macro. The number of lines for SYSLST is taken from LINES.

Operation	Operand
IOTAB	MTP=n1, NTP=n2, PGR=n3, JIB=n4, CHANQ=n5

Checks the parameter sizes.

Calls the macro INTEQU which generates equates for the I/O tables.

Generates:
1. a channel queue (CHANQ) table
2. a physical unit block (PUB) table
3. a job information block (JIB) table
4. if requested, a tape error block (TEB) table
5. a logical unit block (LUB) table.

There are n5 CHANQ entries, n1+n2 PUB entries, n4 JIB entries, n1 TEB entries, and n3+10 LUB entries.

Operation	Operand
DVCGEN	CHUN=X'CUU', DVCTYP=XXXXXX, CHANSW=YES/NO, MODE=X'SS'

Generates a PUB entry. PUB's are generated in the order in which the DVCGEN macros are given.

Bytes 0-1: channel and unit address from CHUN.
Byte 2: null CHANQ pointer 'FF'.

Byte 3: if tape and TEB option provided, the next sequential TEB pointer; otherwise '00'.

Byte 5: if not tape, '00'; if tape and no MODE given, '93'; otherwise 'SS' from MODE.

Byte 6: channel scheduler flag; bits 1 and 7 set from CHANSW and DEVTYP.

Byte 7: job control flag '00'.

Operation	Operand
ASSGN	SYSXXX, X'CUU'

Generates a LUB entry. A DVCGEN macro must have been given for the the device specified.

LUB PUB-pointer: from information generated in DVCGEN
LUB flag: '41'
Initializes PUB job control flag.

Operation	Operand
OPTION	OC=YES/NO, IT=YES/NO, PC=YES/NO

Sets switches to include coding for operator communications external interrupt, interval timer external interrupt, and program check interrupt.

Operation	Operand
PIOCS	SELCH=YES/NO, SETMOD=YES/NO, RWTAU=YES/NO

Sets switches to include coding for selector channel, 7-track tape, and read-while-write tape control unit.

Calls the macro SGTCHS to include coding for the Channel Scheduler, Actual I/O, and I/O Interrupt routines.

Calls the macro SGUNCK to include coding for the Unit Check routine.

Operation	Operand
SEND	ADDR

Calls macros SGTCON, SGFCH, SGSVC, and SGTPE to include the remainder of the Supervisor nucleus.

Calculates addresses:

NUCEND = last byte of coding in the Supervisor nucleus.
SYSEND = ADDR-1504; the beginning of the transient areas.
PTA = SYSEND; beginning of the Type-A transient area.
LTA = SYSEND+504; beginning of the Type-B transient area.
PPBEG = ADDR; beginning of the label area.
SUPEND = ADDR-1; end of Supervisor.
EOSSP = if no storage-protection, PPBEG; if storage-protection, the first byte not storage-protected 0; beginning of the problem program.

SUPERVISOR COMMUNICATION MACROS

These routines enable the programmer to enter or change the Supervisor and to control program flow.

Macros GETIME, SETIME, COMRG, and MVCOM provide access to the timer and communications region in the Supervisor.

- GETIME makes available the timer value in a specified form.
- SETIME sets the timer to a specified value through an SVC.
- COMRG makes available the address of the communications region.
- MVCOM places specified values in the communications region through an SVC.

Macros FETCH, LOAD, STXIT, EXIT, EOJ, and CANCEL communicate through an SVC with program-retrieval, user-interrupt, and end-of-job routines in the Supervisor.

Macros DUMP, PDUMP, and CHKPT provide access to an appropriate transient routine through an SVC.

Macros CCB, EXCP, WAIT, and CHNG communicate with the physical IOCS routines in the Supervisor.

- CCB generates a 16-byte channel command block.
- EXCP initiates the IOCS program through an SVC.
- WAIT tests the traffic bit in the channel command block, returning to the Supervisor through an SVC if it is not on.
- CHNG generates nothing; it is present for reasons of compatibility.

Macros CALL, SAVE, and RETURN provide direct linkage between routines within the problem program.

Name	Operation	Operand
NAME	CALL	P1, P2

Checks for the presence of P1. If P1 is absent, a message is issued and the macro is ignored.

Notes the number of elements in P2 (may be zero).

Checks if P1 is a register but not (15). If not (15), a warning is issued and the routine continues assuming P1 is (15).

Generates coding which sets register 15 to the routine address (P1), register 14 to the return address, and register 1 to the address of the parameter list. Generates the parameter list from P2 if P2 is present.

Name	Operation	Operand
NAME	CANCEL	

Generates coding which clears register 0 and issues a supervisor call of 6.

Name	Operation	Operand
CCBN	CCB	SYSXXX, CCWADD, OPTIONS

Checks for the presence of CCBN. If CCBN is absent, a warning is issued and the routine continues.

NAMOK: Checks type and form of SYSXXX. If incorrect a warning is issued and the routine continues, generating X'FFFF' for the logical unit address.

Calculates unit number and unit type (system or programmer).

FND: Checks for the presence of CCWADD. If CCWADD is absent, a warning is issued and the routine continues assuming CCWADD is zero. Checks type of CCWADD and issues warning if type is incorrect.

CKOPT: Checks for the presence of OPTIONS. If OPTIONS is absent, it is assumed zero. If OPTIONS is not in the proper form (X'nnnn'), a warning is issued and OPTIONS is assumed 0.

Sets a field of switch values from OPTIONS.

Generates a 16-byte field with the label CCBN.
- Bytes 0-1: zeros.
- Bytes 2-3: switch values from OPTIONS.
- Bytes 4-5: zeros.
- Byte 6: unit type from SYSXXX.
- Byte 7: unit number from SYSXXX.
- Byte 8: zero.
- Bytes 9-11: CCW address from CCWADD.
- Bytes 12-15: zeros

Name	Operation	Operand
LAB	CHKPT	SYS, RST, END, P

Checks form and range (SYS000-SYS245) of SYS. If incorrect, a message is printed and the macro is ignored.

Generates a 22-byte field:
- Bytes 0-3: from RST a four-byte address of 'FF' followed by a three-byte register number to specify a restart address.
- Bytes 4-7: from END a four-byte address or 'FF' followed by a three-byte register number to specify a high-byte address (zeros if END not present).
- Bytes 8-11: from P a four-byte address or 'FF' followed by a three-byte register number to specify a tape position-status address (zeros if P is not present).
- Byte 12: '01'
- Byte 13: the unit number from SYS.
- Bytes 14-21. $$BCHKPT

Generates coding which places the address of the 22-byte field in register 0, the address of the transient name (byte 14) in register 1, and a supervisor call of 2.

Name	Operation	Operand
	CHNG	SYSXXX

Generates no coding. CHNG is present in the 16K tape system to be compatible with the 8K system.

Name	Operation	Operand
NAME	COMRG	

Generates coding which places the address of the communications region into register 1.

Name	Operation	Operand
NAME	DUMP	

Generates coding that places into register 1 the address of a field containing the name $$BDUMP and issues a supervisor call of 2.

Name	Operation	Operand
NAME	EOJ	

Generates a Supervisor call of 14.

Name	Operation	Operand
NAME	EXCP	CCB

Checks for the presence of CCB. If CCB is absent, a message is issued and the macro is ignored.

Generates coding that places the CCB address in register 1 [it could already be there if CCB is (1)], and issues a Supervisor call of 0.

Name	Operation	Operand
	EXIT	TY

Checks that TY is PC, IT, or OC. If TY is none of these specifications, a message is issued and the macro is ignored.

Generates a Supervisor call of 17, 19, or 21 for PC, IT, or OC, respectively.

Name	Operation	Operand
NAME	FETCH	PHNM, ENTRY

Checks for the presence of PHNM. If PHNM is absent, a message is issued and the macro is ignored.

Generates coding that places the address of the phase name in register 1, the address of the entry point (zeros if ENTRY is not specified) in register 0, and issues a supervisor call of 2.

Name	Operation	Operand
NAME	GETIME	PAR

Checks that PAR is STANDARD, BINARY, TU, or blank (blank is same as STANDARD). If PAR is none of these, a warning is issued and the routine continues assuming PAR is STANDARD.

Generates coding that places in register 1 the time of day minus the timer divided by 256. If PAR is TU, no more coding is generated.

Generates coding that divides the contents of register 1 by 300 leaving the number of seconds in register 1. If PAR is BINARY, no more coding is generated.

Generates coding that:

1. converts to decimal and moves minutes and seconds into a save area in the form of 000MMSS-,
2. converts to decimal and moves to register 1 the hours,
3. combines the results in the form of HHHMMSS- in register 1.

Name	Operation	Operand
NAME	LOAD	PHNM, LDPT

Checks for the presence of PHNM. If PHNM is absent, a message is issued and the macro is ignored.

Generates coding that places the address LDPT in register 0 [it could already be there if LDPT is (0)]. If LDPT is not specified, zeros are put in register 0.

Generates coding that places the address PHNM in register 1 and issues a supervisor call of 4.

Name	Operation	Operand
NAME	MVCOM	TO, LNG, FROM

Checks that TO, LNG, and FROM are correct in form and magnitude. If incorrect, a message is issued and the macro is ignored.

Generates coding that places the address FROM in register 0, places the address of a move instruction containing TO and LNG in register 1, and issues a supervisor call of 5.

Name	Operation	Operand
NAME	PDUMP	START, END

Checks for presence of START and END. If either is absent, a message is issued and the macro is ignored.

Generates coding that places into register 1 the address of a field containing the name $$BPDUMP, places into register 0 the address of a field containing addresses START, END, and issues a supervisor call of 2.

Name	Operation	Operand
NAME	RETURN	REGS

Checks REGS for correct form. If incorrect, a warning is issued and the routine continues assuming the form (R1).

Checks the type and magnitude of the first register. If incorrect a warning is issued and the routine continues assuming the value of R1 is 14.

If REGS has only one element (R1), RETURN generates a store-register of R1 into a field relative to register 13. The macro generation is complete.

Checks the type and magnitude of the second register, considering the value of the first register. If incorrect a warning is issued and the routine continues assuming the value of R2 is 12.

Generates a store-multiple-register of R1, R2 into a field relative to register 13.

Name	Operation	Operand
NAME	SAVE	REGS

This macro is the same as RETURN except that a load register or a load multiple registers (rather than a store) is generated to move the contents of a field relative to register 13 to REGS.

Name	Operation	Operand
NAME	SETIME	SEC

Checks for the presence of SEC. If SEC is absent, a message is issued and the macro is ignored.

Generates coding that loads the address of the number of seconds into register 1, multiplies by 256*300 (result in registers 0 and 1), and issues a supervisor call of 10.

Name	Operation	Operand
	STXIT	TY, EN, SV

Checks that TY is PC, IT, or OC. If TY is none of these specifications, a message is issued and the macro is ignored.

Checks for the presence of EN and SV. If either is absent a message is issued and the macro is ignored. If both are absent, coding is generated to zero registers 0 and 1, indicating the user's routine is not to be used.

Generates coding that places the address of the save area (SV) into register 1, places the entry point (EN) into register 0, and issues a supervisor call of 16, 18, or 20 for PC, IT or OC, respectively.

Name	Operation	Operand
NAME	WAIT	CCB

Checks for the presence of CCB. If CCB is absent, a message is issued and the macro is ignored.

Generates coding that places the address of the CCB into register 1, tests the traffic in that CCB, issues a supervisor call of 7 if the bit is off, or goes on to the instruction following the macro if the bit is on.

JOB CONTROL

Job Control provides program-to-program transition under system control. Each program execution is called a job step. Between job steps programmer or operator statements are read by Job Control and processed to initialize the system for the new job or job step.

A series of job steps is called a job. A JOB statement marks the beginning of a job. A /& statement marks the end of a job.

Job Control is initially brought into storage by IPL. An EXEC or RSTRT statement brings in the new problem program for execution. At normal end-of-job (EOJ macro) or abnormal end-of-job (CANCEL transient), Job Control is brought back into the problem program area and a new job step begins.

JOB CONTROL - PROGRAM FLOW

Figure 25 shows the program flow of Job Control. If Job Control is brought into storage because of an abnormal end-of-job, the cancel routine is executed. Otherwise, input statements are read from SYSRDR (programmer input) or SYSLOG (operator input).

Control is given to the appropriate statement routine. If the statement is EXEC or RSTRT, a new problem program is brought into storage for execution. Otherwise, on completion of the statement routine, another input statement is read.

JOB CONTROL - I/O FLOW

Figure 26 displays the I/O flow in Job Control. Essentially operator input and output is on SYSLOG. Programmer input is from SYSRDR and programmer output is on SYSLST.

Job Control also performs services for Linkage Editor. On an INCLUD, data can be read from SYSIPT. Data and Linkage Editor control statements can be written on SYS000.

Furthermore, Job Control opens tape units assigned to SYSRDR, SYSIPT, SYSPCH or SYSLST. It opens SYS000 when a LINK or CATAL option is requested. It performs tape operations on any specified unit (MTC) and closes tape units on request (CLOSE).

JOB CONTROL - STORAGE ALLOCATION

Figure 27 displays main storage during execution of Job Control. Subroutines and input statement routines appear in alphabetical order in storage and are described in the same order in this publication.

The label area in Job Control is used to contain information for logical IOCS provided by the VOL and UPSI statements. Before fetching a problem program, the label information is moved up to the label area following the Supervisor.

Information that must remain in storage from one job step to another is kept in the Supervisor communication region, displacement 56-59. Figure 28 defines the significance of each bit in these four bytes. In Job Control they are referred to as JBCSW0, JBCSW1, JBCSW2, and JBCSW3.

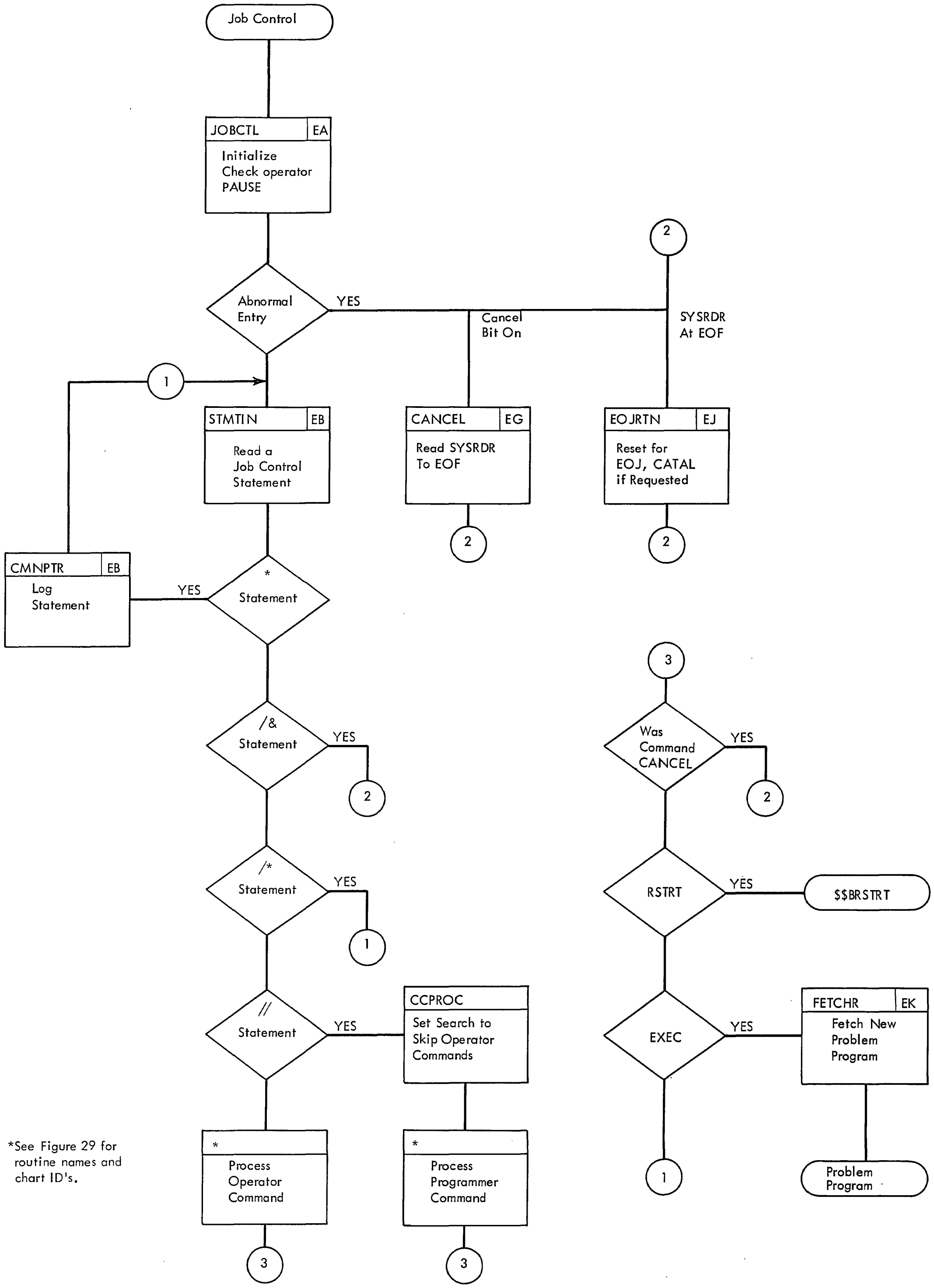


Figure 25. Job Control Program Flow

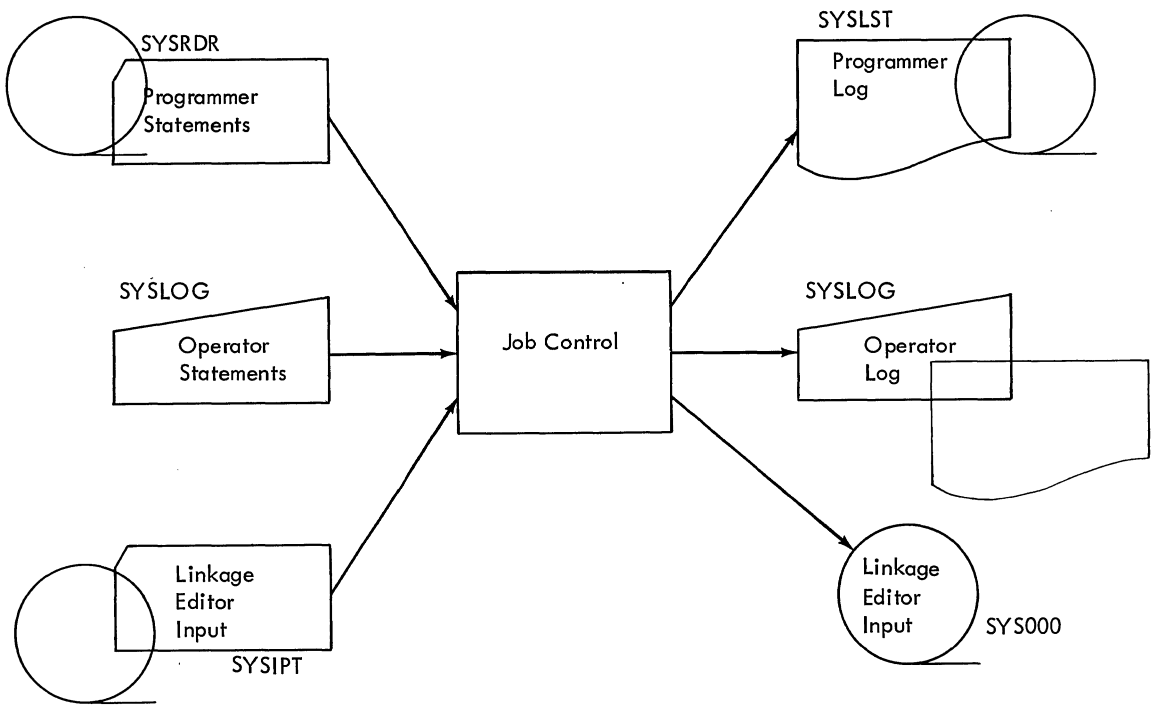


Figure 26. Job Control I/O Flow

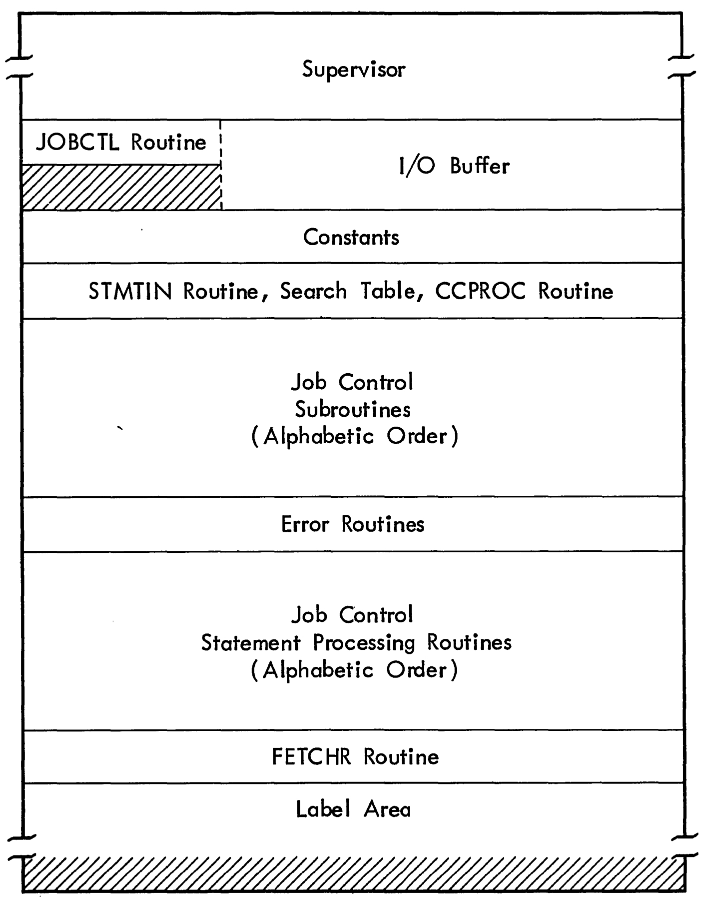


Figure 27. Job Control Core Allocation

	JBCSW0 Displacement 56	JBCSW1 Displacement 57	JBCSW2 Displacement 58	JBCSW3 Displacement 59
Bit 0 - '80'	FETCH from SYS000/SYSRES	LINK/NOLINK (Link Edit Bits)	DECK/NODECK	Job Status Bit: on by JOB Statement; off by /&
Bit 1 - '40'	SYSRES-Position-Changed; on by SVC 0 for SYSRES; off by FETCH or LOAD	Link Edit Permission Indicator (Link Edit Bits)	LIST/NOLIST	DUMP/NODUMP
Bit 2 - '20'	Job Control Input SYSLOG/SYSRDR	Execute-Catalog Permission Indicator (Link Edit Bits)	LISTX/NOLISTX	Reserved
Bit 3 - '10'	Operator LOG/NOLOG	CATAL Indicator	SYM/NOSYM	Programmer LOG/NOLOG
Bit 4 - '08'	Operator Command/Programmer Command	Supervisor Update Indicator	XREF/NOXREF	Cancel Indicator
Bit 5 - '04'	Operator PAUSE Indicator	Autotest Indicator	ERRS/NOERRS	LIOCS Switch First-Time OPEN
Bit 6 - '02'	SYSLOG a Printer (SYSLOG Bits)	Minimum System Indicator for Linkage Editor	48C/60C	LIOCS Switch DTFCP
Bit 7 - '01'	SYSLOG and SYSLST the Same Device (SYSLOG Bits)	GO Indicator	Reserved	Reserved

Note: Slash shows setting as on/off.

Figure 28. Job Control Switch Bytes

JOB CONTROL ROUTINES

JOB CONTROL INITIALIZATION (JOBCTL) CHART EA

Objective: To initialize between job steps.

Entry: From the Supervisor when Job Control is loaded into storage. Loading occurs at IPL or under EOJ conditions (dump, cancel, or normal end-of-job).

Method: JOBCTL is executed only once for each load of Job Control. After initialization the JOBCTL area is used for I/O buffers.

Initialization requires:

1. Setting register values.

2. Calculating the length of the label area for VOL/TPLAB statements.

3. Setting SYSLOG bits 6 and 7 in JBCSW0.

4. Checking the cancel bit in JBCSW3. This bit is set in the Supervisor transient routine $$BCNCL. If set, control goes directly to the cancel routine (CANCEL).

5. Checking the assignment for SYSRDR. If SYSRDR is unassigned, an immediate exit is taken. If SYSRDR is at EOJ (/& statement has been read), control is given to the end-of-job routine (EOJRTN).

6. Checking the operator pause bit in JBCSW0. This bit is set in the Supervisor transient $$BMSGIN. If set, a message is written on SYSLOG. The read statement routine (STMTIN) does not process another statement until the operator provides input on SYSLOG.

JOBCTL exits to the read statement routine (STMTIN).

JOB CONTROL INPUT (STMTIN) CHART EB

Objective: To read a statement and transfer control to the correct processing routine.

Entries:
1. Initially from the Job Control initialization routine (JOBCTL).
2. Thereafter, from any statement processing routine except EXEC or RSTRT.

Method: STMTIN reads a statement through the subroutine RDSTMT from either SYSLOG or SYSRDR, depending on the setting of the Job Control input bit in JBCSW0. The operator command bit in JBCSW0 is set on until a statement beginning with // is read.

The subroutine SCANR1 locates the first operand. If the whole statement is blank, control goes to the IGNORE routine where the Job Control input bit is set off indicating next input to be from SYSRDR. Otherwise, the registers are initialized for use by the BTLRTN subroutine to search for a matching operand in TABLE1.

If the first character of the statement is blank, the search begins at NOTFRS excluding the following statements.

- an asterisk. Go to the comment routine (CMNTPR).
- a slash-ampersand. Go to the end of job routine (EOJRTN).
- a slash-asterisk. Go back to STMTIN.
- a slash-slash. Go to CCPROC to revise search to look for a programmer command.

At CCPROC, which is located after the search table, the operator command bit in JBCSW0 is turned off, the second operand is located, and the search is revised to begin at TABLE2. The search results in a branch to the correct statement processing routine.

Note that some commands are both operator command and programmer command so that the ranges of the search overlap. The search must result in a branch to some routine because the argument of the search is placed in the search table. If the argument is not found in the permanent table, and exit to the error routine NVSTMT is taken.

Figure 29 reproduces the search table. The description of the statement routines is arranged in alphabetical order.

Scan for Operator Statement

Statement	Chart ID
ACTION	EC
CANCEL	EG
CLOSE	EG
DVCDN	EH
DVCUP	EH
ENTRY	EC
IGNORE	EJ
INCLUDE	EC
PHASE	EC
SET	ER
ASSGN	ED-EF
LISTIO	EM
LOG	EN
MTC	EN
NOLOG	EN
PAUSE	EQ
RESET	EQ
DATE	EG
EXEC	EK
JOB	EL
NMTLB	EN
OPTION	EP
RSTRT	EQ
TPLAB	ES
UPSI	ER
VOL	ES

Scan for Programmer Statement

Figure 29. Job Control Statement Search Table

ACTION - CHART EC

Objective: To copy the statement onto SYS000.

Entry: From the read statement routine (STMTIN).

Method: This is one of the statements being processed for Linkage Editor. It is logged if required.

The routine tests the LINK bit in JBCSW1. This bit is set on after the LINK or CATAL operand in an OPTION statement has been recognized and SYS000 has been opened. It must be on when a Linkage Editor statement is read.

The statement is copied onto SYS000. At COPYRT the link edit permission bit in JBCSW1 is turned on to indicate there is data present on SYS000. Return is to read another statement (STMTIN).

ASSGN - CHARTS ED, EE, EF

Objective: To assign a device to a logical unit.

Entry: From the read statement routine (STMTIN), when an ASSGN is given by the operator or programmer.

Method: Basically the ASSGN statement must provide a device, X'CUU', and a logical unit, SYSXXX. The ASSGN routine locates the PUB for the specified device, and inserts a pointer to the PUB in the LUB for the specified logical unit.

Instead of a device, X'CUU', one of the conditions UA or IGN may be assigned to a logical unit. Then, the ASSGN routine inserts a null PUB-pointer and sets the flag in the LUB for the UA or IGN specification.

Assignments are temporary or standard. All assignments made by the programmer (operator command switch in JBCSW0 off) are temporary. Assignments made by the operator are standard unless TEMP is specified. The current assignment must be considered when making an assignment. If a temporary assignment is made, the standard assignment may have to be saved in a JIB.

Finally, if a tape device is specified, the ASSGN statement may provide a DEVOPT field (X'SS') for the PUB, or it may specify that the assignment is alternate (ALT) to be saved in a JIB.

Analyze Operands - Chart ED

The statement is logged if required. The scan subroutine locates the first operand, which must be the logical unit (SYSXXX). It is checked by the subroutine CKCVLU, which initializes the following locations:

- LOGUNT - bits 5 or 6 on for system or programmer logical unit, respectively.
- LOGUNT+1 - logical unit number computed from XXX.
- UNCLOR - numerical combination of LOGUNT and LOGUNT+1.
- STRLUB - address of LUB for the specified logical unit.

The scan subroutine locates the second operand, which must be UA, IGN or X'CUU'. A device specification is checked by the subroutine CKCVCU. This operand initializes the following locations:

- LOGUNT - bit 0 on if IGN; bit 1 on if UA.
- UNT - channel and unit number computed from CUU.
- STRPUB - address of PUB for the specified device

More operands are not required. If present, they are saved at the following locations:

- LOGUNT - bit 3 on if ALT.
- COMTP7 - DEVOPT code for PUB set from X'SS' specification.
- JBCSW0 - operator command bit off if TEMP.

Check Device Type - Chart EE

If the assignment is not UA or IGN, the PUB pointer for the LUB is computed from the PUB address (STRPUB) and saved in STPPTR. Device type is saved in CFIELD.

If the logical unit is a programmer unit, it is necessary to check for SYS000. If the assignment is to SYS000, the link edit bits in JBCSW1 must be reset.

If the logical unit is a system unit, it is necessary to check the specified device against a table of possible devices that can be assigned. If the specified device is also assigned to another system unit, it is necessary to check the compatibility of the new assignment.

Make Alternate Assignment - Chart EE

The logical unit is checked to determine:

- If it is a programmer logical unit.
- If both the device assigned to it and the device about to be assigned as an alternate are tape units.
- If its present assignment and the alternate assignment are both standard or both temporary.

A JIB is created for the alternate assignment. If there is already a JIB pointer in the LUB, the new JIB is found by locating the end of the current chain, inserting FAVP to chain to the new JIB, and then calculating the address of the new JIB.

If there is no JIB pointer in the LUB, the new JIB is found by calculating its address from FAVP, saving the LUB flag in the JIB, and inserting the JIB pointer in the LUB.

In either case, FAVP is updated from the chain field in the new JIB. The new JIB is marked as the end of the chain. The ALT bit is turned on in the JIB flag. The PUB-pointer saved in STPPTR is stored in the first byte of the JIB.

The assignment is complete. Figure 30 shows the new LUB and JIB entries. Return is to the read statement routine (STMTIN).

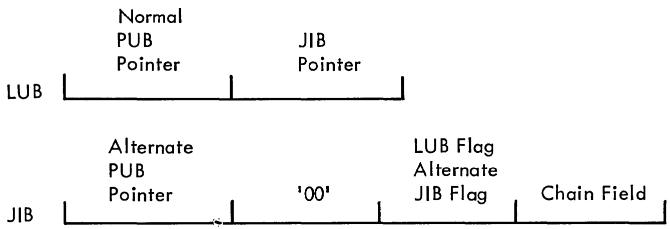


Figure 30. I/O Table Entries for an Alternate Assignment

Make Normal Assignment - Chart EF

To prepare for making the assignment, any JIB chains attached must be released. To release a JIB is to restore the LUB flag and make the JIB's in the chain available.

- If present assignment standard and making standard assignment, release current assignment.
- If present assignment standard and making temporary assignment, no need to release. If a JIB for an ALT is attached, restore LUB flag.
- If present assignment temporary and making standard assignment, release chain on standard assignment and then release and eliminate temporary assignment.
- If present assignment temporary and making temporary assignment, release temporary assignment.

The Job Control flag in the PUB for the specified device is reset (bits 5 and 6) by scanning all LUB's except the one being assigned for the specified PUB pointer.

When making a temporary assignment, a JIB is built unless the present assignment is UA or IGN. Figure 31 shows the JIB entry. In any case, bit 1 of the LUB flag is set off.

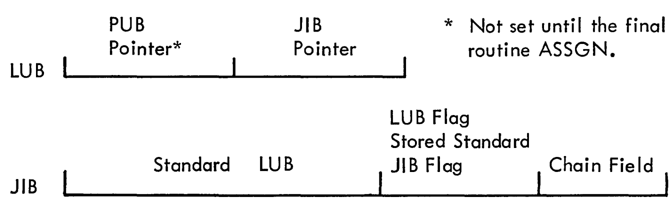


Figure 31. I/O Table Entries for a Temporary Assignment.

If the assignment is UA or IGN, it is completed by setting bit 0 in the LUB flag (UA - off; IGN - on) and putting a null PUB-pointer in the LUB. SYSLOG bits 6 and 7 of JBCSW0 are reset if necessary. Return is to the read statement routine (STMTIN).

When making a standard assignment, bit 1 of the LUB flag is set on. If the assignment is UA or IGN, it is completed by setting the LUB flag (UA - bits 0 and 3 off, bit 2 on; IGN - bits 0 and 3 on, bit 2 off) and putting a null PUB-pointer in the LUB. SYSLOG bits 6 and 7 of JBCSW0 are reset if necessary. Return is to the read statement routine (STMTIN).

If the assignment is a device, bits 0, 2, and 3 in the LUB flag are set off.

When completing a temporary or standard device assignment, bit 0 of the LUB flag is set off and the PUB-pointer (in STPPTR) is put in the LUB. Several checks are made:

- If assignment was made to SYSLST, the Job Control line count is reset to start a new page.
- If a tape device was assigned, the DEVOPT byte in the PUB is initialized. If the assignment is standard, bits 0-4 of the Job Control flag are also reset.
- If a tape device was assigned to SYSRDR, SYSIPT, SYSPCH, or SYSLST, and if the tape is at load point, the tape file is opened by the Supervisor transient $$BJCOPT.
- Bits 5 or 6 of the Job Control flag byte in the PUB are set from bits 5 and 6 of LOGUNT.
- Bits 6 and 7 of JBCSW0 (SYSLOG bits) are reset in case assignments have changed.

Return is to the read statement routine (STMTIN).

CANCEL - CHART EG

Objective: To prepare for entry to the end-of-job routine before normal end-of-job.

Entries:

1. From the initialization routine (JOBCTL) when Job Control is entered with the cancel bit on in JBCSW3.
2. From the read statement routine (STMTIN) when the operator issues a CANCEL command.
3. From the VOL routine at LBLEXH when the label area has been exhausted but more labels are required.

Method: If the CANCEL is an operator command, the cancel indicator in JBCSW3 is set off and the input indicator in JBCSW0 is set to SYSRDR. A message is written on SYSLOG and SYSLST.

The LUB assignments are reset to standard. If SYSRDR is not already at end-of-file (/& has been read), statements are read from SYSRDR until end-of-file is encountered. Then control goes to the end-of-job routine (EOJRTN).

CLOSE - CHART EG

Objective: To close tape files on request.

Entry: From the read statement routine (STMTIN) when the operator issues a CLOSE command.

Method: The statement is logged if required. The statement must contain an operand specifying a logical unit SYSXXX. The operand is checked and the specified LUB is located. If SYSXXX is a system unit, it must be SYSLST or SYSPCH.

The PUB pointed to by the LUB is located. It must be a PUB for a tape device. The routine writes a tape mark, an EOV trailer in user density and standard mode, and two tape marks on the device. The tape is rewound and unloaded.

If SYSXXX is a system logical unit, a standard assignment of IGN is made to the logical unit. Exit is to the read statement routine (STMTIN).

CMNTPR - CHART EB

Objective: To write a comment.

Entry: From the read statement routine (STMTIN) when an * statement is encountered.

Method: The statement is written on SYSLST if the programmer LOG bit is on (JBCSW3). The statement is written on SYSLOG unless it was read from SYSLOG. Return is to read another statement (STMTIN).

DATE - CHART EG

Objective: To initialize the job date field in the communications region.

Entry: From the read statement routine STMTIN when the programmer provides a DATE statement.

Method: The statement is logged if required. The statement must contain an eight-character operand. The eight characters are stored in the communications region (displacement 0). Return is to read another statement (STMTIN).

DVCDN - CHART EH

Objective: To adjust a PUB and LUB's to indicate a device is no longer physically available for system operations.

Entry: From the read statement routine (STMTIN) when the operator issues a DVCDN command.

Method: The statement is logged if required. All LUB's are reset to their standard assignments. The statement must contain an operand specifying a device. The operand is checked and the PUB for the device is located. The job control flag in the PUB is set to all zeros.

The entire LUB table is scanned for LUB's to which the specified device is assigned. If one is found, any JIB's chained to it are released. If the LUB is a system LUB, it is assigned IGN; if a programmer LUB it is assigned UA. The names of the modified LUB's are listed on SYSLOG.

Bits 6 and 7 of JBCSW0 (SYSLOG bits) are reset in case assignments have changed. Return is to read another statement (STMTIN).

DVCUP - CHART EH

Objective: To adjust a PUB to indicate a device is available for system operations after the device has been down.

Entry: From the read statement routine (STMTIN) when the operator issues a DVCUP command.

Method: The statement is logged if required. The statement must contain an operand specifying a device. The operand is checked, and the PUB for the device is located. The job control flag byte is reset to 'F8' or, if the device is tape, from the DEVOPT byte. Return is to read another statement (STMTIN).

ENTRY - CHART EC

See ACTION.

EOJRTN - CHART EJ

Objectives:
1. To catalog Linkage Editor output if required.
2. To set end-of-job conditions.
3. To list the TEB's (tape error blocks).

Entries:
1. From the initialization routine (JOBCTL) when Job Control is entered with SYSRDR at end-of-file (/& has been read).
2. From the read statement routine (STMTIN) when a /& statement is encountered.
3. From CANCEL.

Method: The job status bit in JBCSW3 is checked. This bit is turned on by a JOB statement and off by a /& statement. If it is not on when EOJRTN is entered, the end-of-file flag in the SYSRDR PUB is turned off and control returns to read another statement (STMTIN).

The CATAL bit in JBCSW1 is checked. This bit is set on after the CATAL operand in an OPTION statement has been recognized. It indicates that the Linkage Editor output on SYS000 is to be cataloged. Control goes to the MAINT program through a fetch.

Otherwise, the end-of-job conditions are set. All LUB's are reset to their assignments. If SYSIPT is not already at end-of-file, it is read until a /& is encountered. An end-of-job message is written on SYSLST and SYSLOG. The option bits in JBCSW2 and 3 are reset to normal. The job status bit in JBCSW3 is set off. The job name is reset to NO NAME.

If there are any TEB's, each PUB entry is checked for a corresponding TEB. If there are any nonzero counts other than the retry count in the TEB, the TEB is listed on SYSLOG with the counts converted to decimal.

Finally, all bits in JBCSW1 are set off except the Autotest indicator. Control goes to RSTCOM in the JOB routine.

EXEC - CHART EK

Objective: To determine the name and location of a phase to be fetched.

Entry: From the read statement routine (STMTIN) when the programmer provides an EXEC statement.

Method: The statement is logged if required. If the statement has no operand, the phase to be executed is Linkage Editor output on SYS000. The EXEC permission bit (bit 2 in JBCSW1) must be on. Bits 0 and 1 of JBCSW1 and bit 0 of JBCSW0 are set off. The program name is read from SYS000 and control goes to FETCHR.

If the statement has an operand, it must be a name consisting of at least one character and not more than eight. If the name is 'LNKEDT' or 'ATLEDT', the link-edit permission bit (bit 1 in JBCSW1) must be on. SYS000 is positioned for a read. In JBCSW1 bit 0 is set on and bit 1 set off. Control goes to RESFCH to fetch $LNKEDT or ATLEDT.

If the name is RPG, control goes to RESFCH to fetch RPG1.

If the name is not LNKEDT or ATLEDT or RPG, it is simply moved to the necessary field. Control goes to RESFCH.

At RESFCH bit 0 of JBCSW0 is set on to specify a fetch from SYSRES, and control goes to FETCHR to fetch the named phase.

IGNORE - CHART EJ

Objective: To allow further input from SYSRDR.

Entry: From the read statement routine (STMTIN) when the operator gives a response of IGNORE.

Method: The statement is written on SYSLST. The job control input bit 2 of JBCSW0 is set off to allow input from SYSRDR. Exit is to read another statement (STMTIN).

INCLUD - CHART EC

Objective: If no operand, to copy SYSIPT on SYS000; if operand, to copy statement on SYS000.

Entry: From the read statement routine (STMTIN).

Method: This is one of the statements being processed for Linkage Editor. If the scan subroutine locates an operand, control goes directly to the ACTION routine to copy the statement.

If no operand is found, the statement is logged if required. The routine tests the LINK bit in JBCSW1. This bit is set on after the LINK or CATAL operand in an OPTION statement has been recognized and SYS000 has been opened. It must be on when a Linkage Editor statement is read.

The routine reads card images from SYSIPT, and copies them onto SYS000 until it encounters a /* card in SYSIPT. When this end-of-file indicator is found, control goes to COPYRT in the ACTION routine to turn on the link-edit permission bit in JBCSW1. Return is to read another statement (STMTIN).

JOB - CHART EL

Objective: To prepare for starting a new job.

Entry: From the read statement routine (STMTIN) when the programmer provides a JOB statement.

Method: JBCSW1 is reset to zeros. The job status bit in JBCSW3 is checked. This bit is set on by a JOB statement and off by a /& statement. If the bit is on when the JOB statement is read, the routine simulates end-of-job. SYSRDR is set at end-of-file, and control goes to the end-of-job routine (EOJRTN).

All the LUB's are reset to the standard assignments. A job message is written on SYSLST and SYSLOG. The job name is put in the communications region (displacement 24). JBCSW2 and JBCSW3 are reset to the standard options.

The rest of the routine is common to JOB and EOJRTN. In the communications region the job date (displacement 0) is set to standard, the label area length (displacement 44) and the checkpoint identification number (displacement 92) and the user area (displacement 12) are set to zeros. If this point has been reached by simulating end-of-job, control returns to SIMEND to reset the LUB's and write the job message.

Otherwise the end-of-file bits (bit 2 of the channel scheduler flag) in all the PUB's are set off. If the operator-pause bit (bit 5 of JBCSW0) is not on, control returns to read another statement (STMTIN).

If the pause bit is on, a pause message is written on SYSLOG, and the job control input bit (bit 2 in JBCSW0) is set on. The pause bit is set off. Return is to read another statement (STMTIN).

LISTIO - CHART EM

Objective: To provide the programmer or operator with a list of I/O assignments.

Entry: From the read statement routine (STMTIN) when the programmer on the operator issues a LISTIO command.

Method: The statement is written on SYSLOG if LISTIO is an operator command but input is from SYSRDR. The statement is written on SYSLST if LISTIO is a programmer command.

The statement must contain an operand that specifies the assignments to be listed. Output is on SYSLST if LISTIO is given by the programmer, on SYSLOG if LISTIO is given by the operator.

The operand must be one of five forms:

- SYSXXX (a specific logical unit). Under the header "CH. UNIT", the assignment of the single unit is listed.
- SYS. Under the header "SYSTEM I/O UNITS CH. UNIT", the assignments of the eight named system logical units are listed. The sixth and tenth system logical units are reserved and do not appear in the listing. The names of the units are picked up from a table located at TBRDR.
- PROG. Under the header "PROGRAM I/O UNITS CH. UNIT", the assignments of all the programmer logical units are listed. The number of programmer LUB's and the location of the first one (SYS000) is picked up from PGNICL and PGFICL.
- DOWN. Under the header "DOWN CH. UNIT", the channel and unit of each PUB with device-down indicated (job control flag all zeros) is listed.
- UA. Under the header "UNASSIGNED CH. UNIT", the channel and unit of each PUB with the assignment bits off is listed. The assignment bits are bits 5, 6, and 7 in the job control flag.

Only one operand is expected in the statement. When the list is complete, return is to read another statement (STMTIN).

LOG - CHART EN

Objective: To indicate that statements and messages are to be logged on SYSLST or SYSLOG.

Entry: From the read statement routine (STMTIN) when the programmer or the operator issues a LOG command.

Method: If the statement is given by the programmer, it is logged if required. The log bit in JBCSW3 (bit 3) is set on to indicate logging is required on SYSLST. If the bit was off, the statement is logged on SYSLST. Return is to read another statement (STMTIN).

If the statement is given by the operator, it is logged unconditionally on SYSLOG and, if required, on SYSLST. The log bit in JBCSW0 (bit 3) is set on to indicate logging is required on SYSLOG. Return is to read another statement (STMTIN).

MTC - CHART EN

Objective: To execute a tape I/O control command on request.

Entry: From the read statement routine (STMTIN) when the programmer or the operator issues an MTC command.

Method: The statement is logged if required. The statement must contain operands specifying an I/O command and a logical unit (or device if operator command). The command is located in a table used to initialize the CCW for the channel program.

If SYSXXX is specified, the device assigned to it is found. If the operator specifies CUU, it is assigned to SYSUSE. In either case the device must be a tape.

If the statement also contains a duplication factor, the factor is converted to binary. The command is then executed the specified number of times. Exit is to the read statement routine (STMTIN).

NMTLB - CHART EN

Objective: To direct Linkage Editor to reserve a portion of the label area.

Entry: From the read statement routine (STMTIN) when the programmer provides an NMTLB statement.

Method: The statement is logged if required. The statement must contain an operand specifying the number of VOL/TPLAB pairs for which space is to be reserved. This operand must be from 1 to 3 decimal characters. It is converted to binary and multiplied by 80 to get the number of bytes that must be reserved. If this amount exceeds the space available, the job is canceled. Otherwise, return is to read another statement (STMTIN).

NOLOG - CHART EN

Objective: To indicate that statements and messages are not to be logged on SYSLST or SYSLOG.

Entry: From the read statement routine (STMTIN) when the programmer or the operator issues a NOLOG command.

Method: The statement is logged if required. If the statement is given by the programmer, the log bit in JBCSW3 (bit 3) is set off to indicate logging is not required on SYSLST. If the statement is given by the operator, the log bit in JBCSW0 (bit 3) is set off to indicate logging is not required on SYSLOG. Return is to read another statement.

OPTION - CHART EP

Objective: To record Job Control options requested by the programmer.

Entry: From the read statement routine (STMTIN) when the programmer provides an OPTION statement.

Method: The statement is logged if required. An operand is located in the statement and compared against a table to find a routine address. The routine is executed. Return is to locate another operand. When the routine for the last operand has been executed, return is to read another statement (STMTIN).

Each routine sets a bit on or off in one of the job control flags. Figure 32 shows the bit set by each option. The following requests provide further services:

- LOG logs the OPTION statement on SYSLST if, on entry to the routine, the programmer log bit was off.
- CATAL opens SYS000 through the transient $$BJCOPT, and sets bit 0 on and bits 1 and 2 off in JBCSW1. If the GO option is given before the CATAL option, these services are omitted.

- LINK opens SYS000 through the transient $$BJCOPT, and sets bit 0 on and bits 1 and 2 off in JBCSW1.
- GO sets bits 0 and 1 off and bit 2 on in JBCSW1.

LINK/NOLINK	bit 0 in JBCSW1 on/off
CATAL	bit 3 in JBCSW1 on
MINSYS	bit 6 in JBCSW1 on
GO	bit 7 in JBCSW1 on
DECK/NODECK	bit 0 in JBCSW2 on/off
LIST/NOLIST	bit 1 in JBCSW2 on/off
LISTX/NOLISTX	bit 2 in JBCSW2 on/off
SYM/NOSYM	bit 3 in JBCSW2 on/off
XREF/NOXREF	bit 4 in JBCSW2 on/off
ERRS/NOERRS	bit 5 in JBCSW2 on/off
C48/C60	bit 6 in JBCSW2 on/off
DUMP/NODUMP	bit 1 in JBCSW3 on/off
LOG/NOLOG	bit 3 in JBCSW3 on/off

Figure 32. Job Control Options

PAUSE - CHART EQ

Objective: To cause a request for operator input immediately (if issued by the programmer), or before the next job or job step (if issued by the operator).

Entry: From the read statement routine (STMTIN) when the programmer or the operator issues a PAUSE command.

Method: The statement is logged on SYSLOG and, if required, on SYSLST. If the command was from the operator, the operator pause bit in JBCSW0 (bit 5) is set on. If the command was from the programmer, the job control input bit in JBCSW0 (bit 2) is set on. Return is to read another statement (STMTIN).

PHASE - CHART EC

See ACTION.

RESET - CHART EQ

Objective: To reset all logical unit assignments to standard.

Entry: From the read statement routine (STMTIN) when the programmer or operator issues a RESET command.

Method: The statement is logged if required. The LUB assignments are reset. Return is to read another statement (STMTIN).

RSTRT - CHART EQ

Objective: To give control to the Supervisor transient $$BRSTRT in order to restart execution of a checkpointed program.

Entry: From the read statement routine (STMTIN) when the programmer provides a RSTRT statement.

Method: The statement is logged if required. The statement must contain two operands. The first must be the name of the logical unit on which the checkpoint was written. The second must be a 4-character serial number used to identify the checkpoint.

Information is passed to the transient routine in registers 2 and 3.

- Register 2 contains the address of the PUB for the device assigned to the logical unit specified in the first operand.
- Register 3 contains the address of UNCLOR. The subroutine CKCVLU (which checked and located the LUB for the specified logical unit) set UNCLOR at '00' for a system LUB or '01' for a programmer LUB. The subroutine set UNCLOR+1 to the logical unit number. RSTRT puts the checkpoint identification number in UNCLOR+2.

Control is given to the transient routine $$BRSTRT through a supervisor call of 2.

SET - CHART ER

Objective: To insert values in specified fields of the communications region and system timer.

Entry: From the read statement routine (STMTIN) when the operator issues a SET command.

Method: The statement is logged if required. An operand is located in the statement and control is given to the routine associated with one of the four valid operands. After the routine is executed, control returns to locate another operand. If there are no more operands, control returns to read another statement.

- CLOCK. Following the operand CLOCK must be a field of the form HH/MM/SS giving the actual time-of-day in hours, seconds, and minutes. These values are converted to binary and added together in units of 1/300 of a second.
  The timer is a negative value in units of 1/(300*256) of a second. The time-of-day and the timer divided by 256 are added to get the time-of-day. The result is stored in the time-of-day field of the system timer (Figure 33). The actual time-of-day can be calculated from the time-of-day minus the timer converted to hours, minutes, and seconds.

- LINECT. The two-character integer field following LINECT is converted to binary and stored in the line count field of the communications region (displacement 78). The Job Control field containing the remaining number of lines on the page must be adjusted for the new total number of lines on the page.

- DATE. Following the operand DATE must be a field of the form MM/DD/YY or DD/MM/YY giving the date in month, day, and year. These values are stored in the system date field of the communications region (displacement 79-84). The day of the year is computed in binary, then converted to decimal and stored in displacement 85-87.

- UPSI. The indicators following the operands UPSI (there must be from 1-8 indicators) are used to modify the bits of the UPSI byte in the communications region (displacement 23). The first indicator modifies bit 0 of the UPSI byte:
  indicator is 1 - bit is set on.
  indicator is 0 - bit is set off.
  indicator is X - bit is ignored.

  If there are fewer than 8 indicators, the corresponding bits in the UPSI byte are ignored. For example, an operand XX110 will modify an UPSI byte of 00001111 to 00110111.

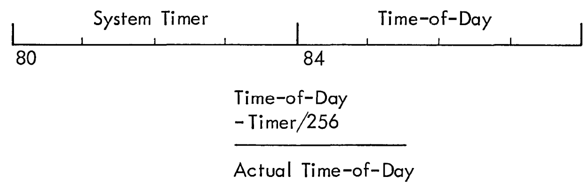


Figure 33. Calculation of Actual Time of Day

TPLAB - CHART ES

Objective: To build a tape label in the label area.

Entry: From the read statement routine (STMTIN) when the programmer provides a TPLAB statement.

Method: The VOL and TPLAB statements must come in ordered pairs. The byte PRVLBL is set at '01' after each VOL statement. If PRVLBL is not '01', the TPLAB statement is out of sequence.

The statement is logged if required. If the statement is on one card, it contains a 49-character label. If it is on two cards, 20 more characters must be read and added to the first 49. The new label information is moved to locations 10-78 in the label already initialized by the VOL routine.

The label area pointer and byte count are increased by 80. The byte PRVLBL is set '02'. If there is not space enough for another label, PRVLBL is set to '08'. Return is to read another statement (STMTIN).

UPSI - CHART ER

Objective: To initialize the UPSI byte in the communications region.

Entry: From the read statement routine (STMTIN) when the programmer provides an UPSI statement.

Method: The statement is logged if required. The statement must contain an operand specifying from 1-8 indicators. The routine uses the UPSI section of the SET routine to modify the UPSI byte with the indicators. Return is to read another statement (STMTIN).

VOL - CHART ES

Objective: To build the name of a tape file in the label area.

Entry: From the read statement routine (STMTIN) when the programmer provides a VOL statement.

Method: The VOL and TPLAB statements must come in ordered pairs. The byte PRVLBL is set at '02' initially and after each TPLAB statement (unless there is no space remaining in the label area). If PRVLBL is not '02', the VOL statement is out of sequence. If there is no space remaining in the label area, the job is canceled.

The statement is logged if required. The VOL statement must contain two operands. The first must be a logical unit, SYSXXX. The subroutine CKCVLU checks the operand and sets UNCLOR:

UNCLOR - '00' is system logical unit.
- '01' if programmer logical unit.
UNCLOR+1 - unit number.

The second operand must be a file name from 1-8 characters.

The next available 80-character label is initialized in the label area:

blank in location 1.
file name in locations 2-9.
UNCLOR and UNCLOR+1 in locations 79-80.

The byte PRVLBL is given the value '01' for the TPLAB routine. Return is to read another statement (STMTIN).

JOB CONTROL FETCH (FETCHR) CHART EK

Objective: To clear storage and issue fetch for a new problem program.

Entry: From EXEC.

Method: FETCHR is part of the EXEC routine located at the end of Job Control in order to set the limits for clearing storage. Before the fetch of the new phase is given, FETCHR:

- Moves the labels (generated by VOL and TPLAB statements) from the end of storage to the end of the Supervisor.
- Calculates the clear-start-address from the beginning of Job Control and the length of the label area.
- Calculates the clear-end-address from the end of storage and the length of the clear routine and phase name.
- Clears storage within the clear limits.
- Issues a fetch for the new phase.

JOB CONTROL SUBROUTINES - CHARTS FA-FG

ASGNLS - Chart FA

Given the address of a LUB in register POINT1, ASGNLS lists all the assignments (standard, temporary, alternate) associated with the LUB for the LISTIO routine.

1. At ASGNLS, NOASGN, and NANJIB the present assignment is listed.
2. At STDTST and NOJIB the standard assignment is listed if the present assignment is temporary.
3. At ALTTST any alternates to the present assignment are listed.

Output is on SYSLOG if the command (LISTIO) is given by the operator. Output is on SYSLST if the command is given by the programmer.

BINCON - Chart FB

Scans for an operand of the form XX/YY/ZZ in the input area. Checks that each character-pair is numeric. Converts each pair to a binary number, storing XX in WRKRG1, YY in WRKRG2, and ZZ in WRKRG3.

BTLRTN - Chart FB

Compares an eight-byte argument with a table of eight-byte keys. When a match is found, the two-byte function following the matching key is added to the address ORIGIN. Exit is to the routine addressed by the sum.

On entry to the routine, register POINT1 must contain the address of the table minus 10. Register POINT2 must contain the address of the key. The argument must also be the last key in the table to assure a match.

CHKNUM - Chart FB

Checks for a numeric character-pair in the input area. Packs the pair into a work area WRKFLD. Advances the input area pointer by three.

CKCVCU - Chart FB

Checks for a device address X'CUU' in the input area. Places '0CUU' in a four-byte field CHUNT. Computes the address of the PUB for this device:

1. Selects PUB pointer from the FOCL field.
2. Multiplies pointer by 8 to get displacement.

3. Adds displacement to PUBADD to get address of the first PUB on the channel.
4. Searches PUB's on the channel for the specified unit.

The PUB address is saved in STRPUB.

CKCVLU - Chart FB

Checks for a logical unit address 'SYSXXX' in the input area. The logical unit name is numeric for a programmer LUB (SYS003) or non-numeric for a system LUB (SYSLST).

If the LUB is a programmer LUB, the LUB number must be between 0 and the limit specified by PGNICL. Bit 6 is set on in LOGUNT. The number is stored in LOGUNT+1. To compute the LUB address, CKCVLU:

1. Adds the LUB number to the LUB pointer in PGFICL.
2. Multiplies the result by two to get the displacement.
3. Adds displacement to LUBADD.

If the LUB is a system LUB, the name must occur in the table TBRDR. Bit 5 is set on in LOGUNT. From the table position a LUB number and LUB displacement are known. The number is stored in LOGUNT+1. The displacement is added to LUBADD to get the LUB address.

For either a a programmer or system LUB, the LUB address is saved in STRLUB. UNCLOR is initialized from LOGUNT and LOGUNT+1. Figure 34 shows LOGUNT and UNCLOR.

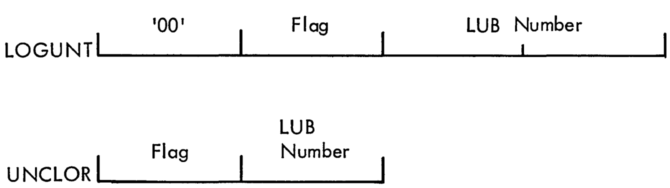


Figure 34. Work Areas LOGUNT and UNCLOR

CONCAT - Chart FC

Reads and logs a continuation card. Places both parts of the statement in a single buffer as in Figure 35.

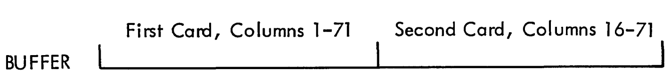


Figure 35. Concatenation

EXCPRG - Chart FC

Given a CCW address in register POINT1 and a symbolic unit address in register 0, EXCPRG initializes the Job Control CCB and issues an EXCP and a WAIT (if necessary). If a unit exception (not EOF) occurs and the unit is not SYSLOG or SYSLST, a message with the device address is issued (on SYSLOG for the operator, on SYSLST for the programmer). If the unit is SYSLOG, the I/O is reissued. If the unit is SYSLST, the unit exception is processed in STMTIN just before the next statement is read.

GETJIB - Chart FC

Given a LUB displacement (or address) in register POINT1, GETJIB computes a JIB address in register POINT2. Condition code 3 on return indicates that there was no JIB pointer in the LUB.

GETPUB - Chart FC

Given a LUB displacement (or address) in register POINT1, GETPUB computes a PUB address in register POINT3. Condition code 3 on return indicates there was a null PUB-pointer in the LUB.

JIBDCU - Chart FD

Given a LUB displacement (or address) in register POINT1, JIBDCU determines if there are any JIB's attached to the LUB. If there are, it releases the JIB chain. Furthermore, if JIBDCU is entered because a standard assignment is being made, and if the present assignment is temporary, JIBDCU must be executed twice: first to release any JIB's from the stored standard assignment; second to release any JIB's from the temporary assignment.

A JIB chain is released by:

1. Replacing each JIB flag in the chain with zeros.
2. Chaining the free list to this JIB chain by inserting the FAVP pointer into the chain field of the last JIB in the chain.
3. Adding this JIB chain to the free list by inserting the JIB pointer from the LUB into FAVP.

Finally, the standard LUB flag is restored to the LUB replacing the JIB pointer.

LOGCHK - Chart FD

Sets bits 6 and 7 of JBCSW0 depending on the assignments to SYSLST and SYSLOG.

LOGGER - Chart FE

Writes on SYSLOG and, if programmer LOG option is present, writes on SYSLST.

MOVERT - Chart FD

Given a from-address in register POINT1, a to-address in POINT2, and a field length in WRKRG1, MOVERT moves the field.

MSGOUT - Chart FD

Allows output on SYSLOG and SYSLST. Executes subroutine SYSLGR.

OUTPUT - Chart FA

Allows output on SYSLOG and SYSLST. If current command is from the operator, writes on SYSLOG. If not, writes on SYSLST.

PRPSY0 - Chart FE

Prepares SYS000 for input to Linkage Editor by writing an ENTRY statement and positioning the tape for a read.

QLOGER - Chart FE

If the operator LOG option is present, writes on SYSLOG. If the programmer LOG option is present, writes on SYSLST.

RDSTMT - Chart FE

Sets switches in SYSLGR and SYSPTR to allow output on SYSLOG and SYSLST. Reads from SYSRDR if job control input bit 2 if off. Reads from SYSLOG if bit is on.

RSTASG - Chart FF

Uses the RSTRTN to reset each of the LUB's to its standard assignment.

RSTRTN - Chart FF

Given a LUB address in register POINT1, RSTRTN determines if the present assignment is standard. If so, the routine is complete. If not, any JIB's are released from the present assignment.

If the standard assignment is IGN or UA, LOGUNT is initialized and the ASSGN routine is entered at MASGN (Chart EE) to complete the reassignment.

If the standard assignment is a device, the PUB pointer is restored to the LUB, and the ASSGN routine is entered at RSTNTR (Chart EE). On return, the LUB flag (or JIB pointer) is restored to the LUB to complete the reassignment.

SCAN - Chart FF

1. <u>SCANR1</u>: Scans for a nonblank character. If all characters are blank, return is to the main routine.
2. <u>SCANR2</u>: Picks up the pointer and limit from the previous scan. Scans for a nonblank character. If all characters are blank, return is to the main routine.
3. <u>SCANR3</u>: Picks up the pointer and limit from the previous scan.

After entry, each scan is for a stop-character: a comma, a blank, or an equal sign. When one is found (or at the end of the field if none are found), the condition code is set by comparing to a comma. The length of the parameter less one is saved in PARLGT.

SIZCHK - Chart FF

Assures the parameter length is between one and eight ($0 \le$ PARLGT ≤ 7).

SYSLGR - Chart FG

If output on SYSLOG is suppressed, an immediate return is taken to the main routine. If output is allowed, further requests for output are suppressed. If SYSLOG and SYSLST are the same, output is on SYSLST (see subroutine SYSPTR).

Before writing on SYSLOG, the output length is reduced by trailing blanks. SYSLGR writes on SYSLOG.

SYSPTR - Chart FG

If output on SYSLST is suppressed, an immediate return is taken to the main routine. If output is allowed, further requests for output areas are suppressed. SYSPTR writes on SYSLST, adjusting the line count as necessary.

TIMLOG - Chart FG

If the timer feature is present, the output buffer is cleared of everything but the time field. Output is allowed on SYSLOG. TIMLOG executes the subroutine SYSLGR.

TIMSTP - Chart FG

Computes the actual time-of-day by subtracting the timer (location 80) divided by 256 from the time-of-day (location 84). The result is placed in the time field of the output area in the form HH.MM.SS.

JOB CONTROL ERROR ROUTINES

CNIOAG - CONFLICTING I/O ASSIGNMENT
INDVTP - INVALID DEVICE TYPE
INIOAG - INVALID I/O ASSIGNMENT
NOMRJB - NO FREE JIBS
NVSTMT - INVALID STATEMENT FORMAT
OUTSQN - STATEMENT OUT OF SEQUENCE

For each of these errors, the statement in error is logged on SYSLST and SYSLOG if required. Then the error message is written on SYSLST and SYSLOG. The job control input bit 2 in JBCSW0 is set on to allow input from SYSLOG. Return is to read another statement (STMTIN).

IPTUNA - PLEASE ASSIGN SYSIPT
RDRUNA - PLEASE ASSIGN SYSRDR

For these errors in assignment, the message is written on SYSLOG. The job control input bit 2 in JBCSW0 is set on to allow input from SYSLOG. Return is to read another statement (STMTIN).

INITIAL PROGRAM LOAD

IPL is a three-phase program. Figure 36 shows the relationship of the phases. $$A$IPL1 is a 24-byte phase consisting of 1 PSW and 2 CCW's. It is the bootstrap routine that is used to load $$A$IPL2.

$$A$IPL2 clears main storage beyond IPL2, notes the size of main storage, reads the Supervisor into main storage, and moves the SYSGEN I/O tables (the tables on SYSRES) to high main storage. It builds an I/O table consisting of two devices, the SYSRES tape and a communications device, to be used in the operation of $IPLRT2. It issues a supervisor call to bring in $IPLRT2 from the core image library.

$IPLRT2 calculates the size of main storage and the number of PUB's and TEB's in use by the present system. Then it executes the read subroutine. The read subroutine (Chart GD) reads three types of control statements that the operator may put in the communications device. After a statement is read it is identified by the scan subroutine (Chart GE).

The ADD cards are processed by the ADD statement subroutine (Chart GF). After a PUB has been added, the program branches back to read another card. The DELETE cards are processed by the DELETE statement subroutine (Chart GG). After a PUB has been deleted, the program branches back to read another card. The SET card is processed by the SET statement subroutine (Chart GH), which sets the date and time of day in the Supervisor. The program branches back to restore the I/O tables to the Supervisor, load Job Control, and make final exit.

$$A$IPL1 (BOOTSTRAP)

Objective: To load $$A$IPL2

Entry: By operator action

Method: To initiate system operation, the operator sets the device address of SYSRES on the console and presses the load key. Hardware reads the first record from SYSRES into location 0 and transfers control to the CCW at location 8 (Figure 37).

In the 16K tape system, this CCW reads the next record on SYSRES, $$A$IPL2, into an area of storage that is higher than the end of the largest possible Supervisor. When the I/O is complete, hardware uses the PSW at location 0 as the new PSW. In this system the PSW transfers control to the beginning of $$A$IPL2 (Figure 38).

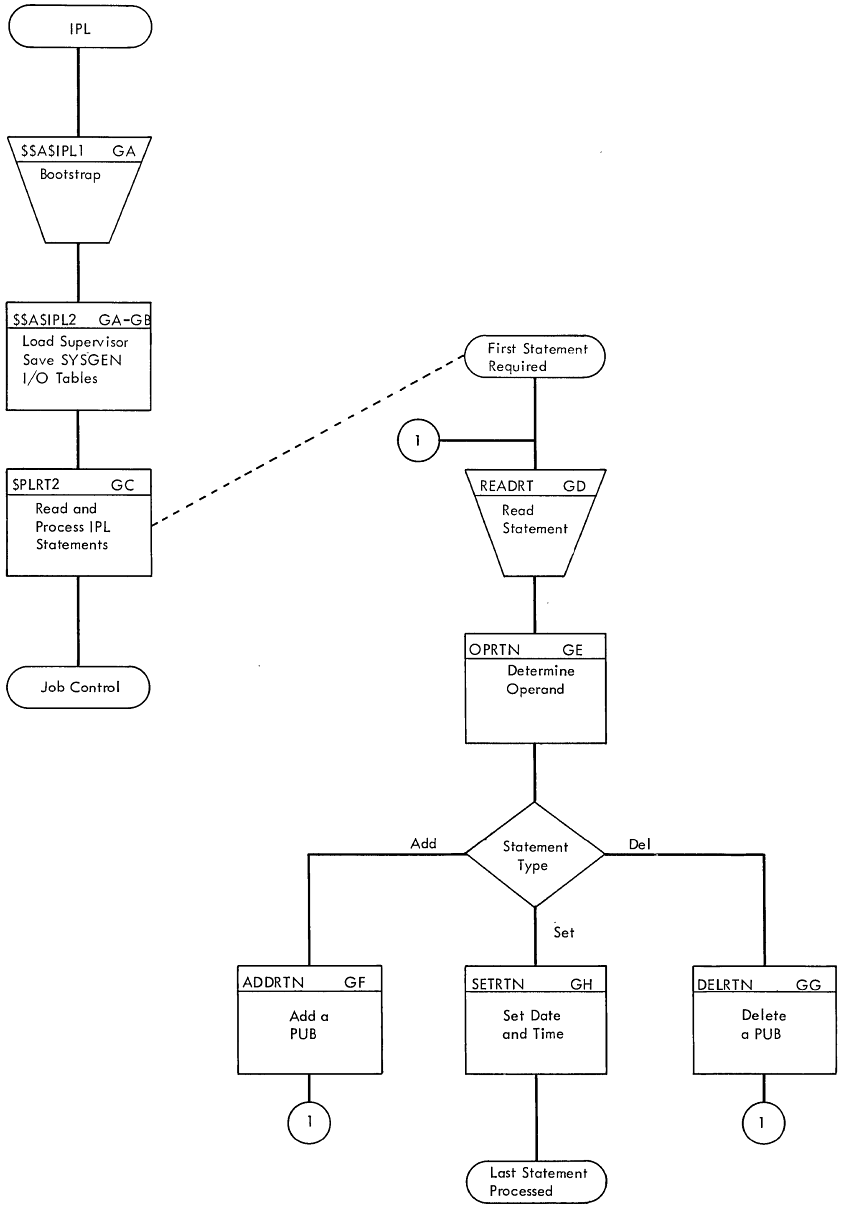


Figure 36. IPL Program Flow

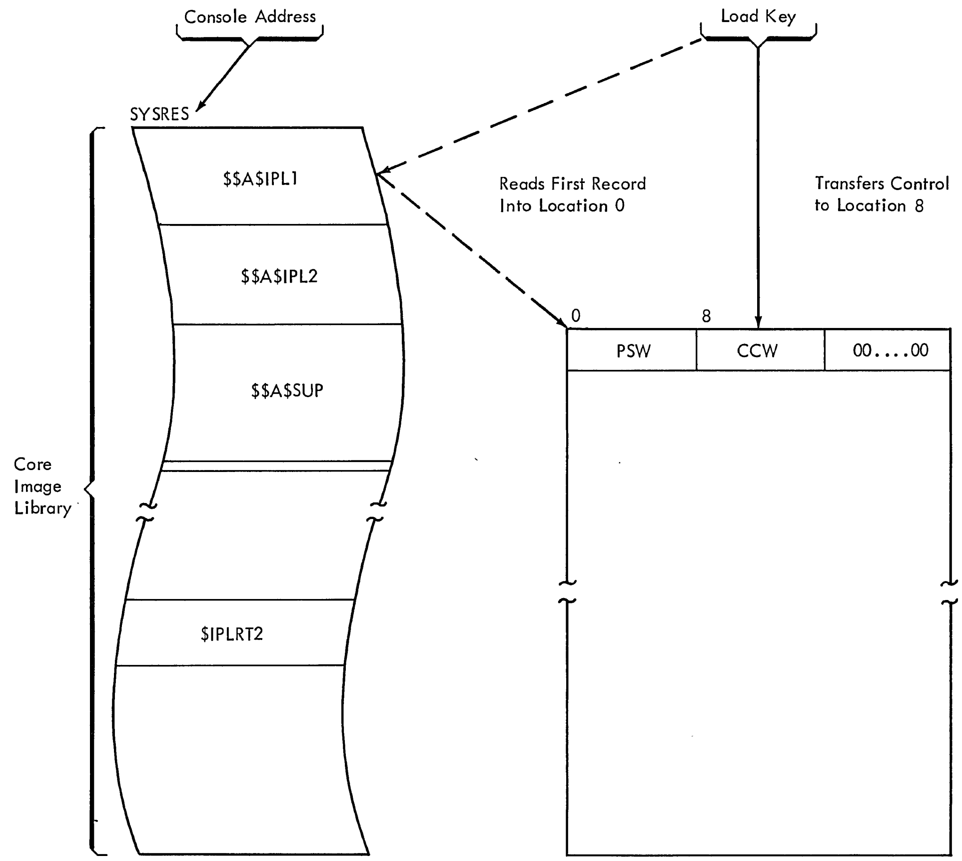


Figure 37. IPL Hardware Load

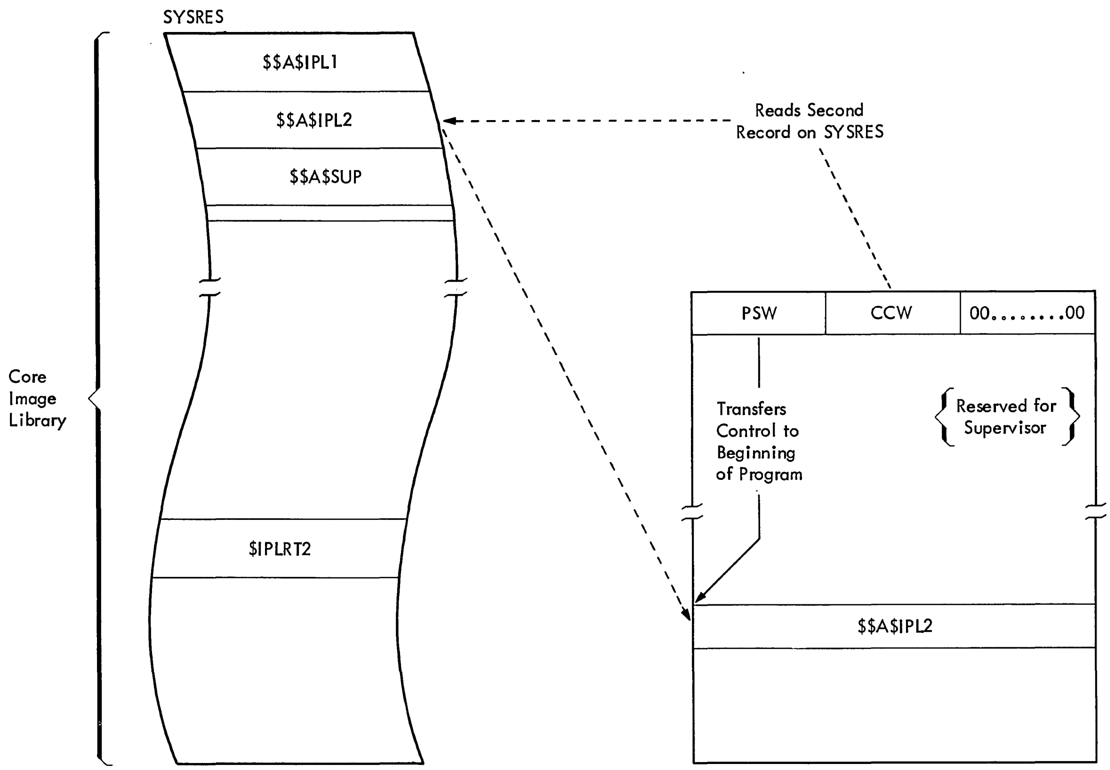


Figure 38. $$A$IPL1

$$A$IPL2 (LOAD) CHARTS GA AND GB

Objective: To load Supervisor and to prepare tables for SYSRES and a communication device.

Method: This routine clears storage from the end of this routine to the end of storage. It reads the next record (or records) into the Supervisor area from SYSRES. The address of SYSRES is still available from $$A$IPL1. The number of records in the Supervisor is specified in the Supervisor header.

If there are any read errors during the reading of the Supervisor, error codes are put in locations 0-3. The tape is rewound and the program enters a wait. The job must be restarted.

At ENDRD control is given to LUBRTN (Chart GB) to build a PUB for SYSRES in low storage. The first time through LUBRTN, the PUB, TEB, and LUB tables generated with the system are saved in high storage.

LUBRTN then builds the PUB, zeros the TEB if present, updates the FOCL and the PUB pointers in the LUB's, and assigns the PUB to the LUB for SYSRES.

On return from LUBRTN the program enters the wait state awaiting operator action. The operator takes one of the following actions to determine the IPL communications device:

- Presses the external interrupt key causing an external interrupt. SYSRDR is to be the communications unit and it is already assigned a device. The device address is located in the PUB table and placed in the I/O old PSW.
- Presses ready on a disabled reader causing an I/O device end interrupt. SYSRDR is to be the communications unit.
- Presses the 1052 request key causing an I/O attention interrupt. SYSLOG is to be the communications unit.

Again control goes to LUBRTN to build a PUB in low storage. The device address is taken from the I/O old PSW. The PUB is assigned to the LUB for SYSRDR or SYSLOG, depending on the type of interrupt.

The two-device I/O table is sufficient to complete the IPL process. If the storage protection feature is specified in the configuration byte (displacement 52) of the communications region, a key of 0 is given to the Supervisor area. The rest of storage is given a key of 1.

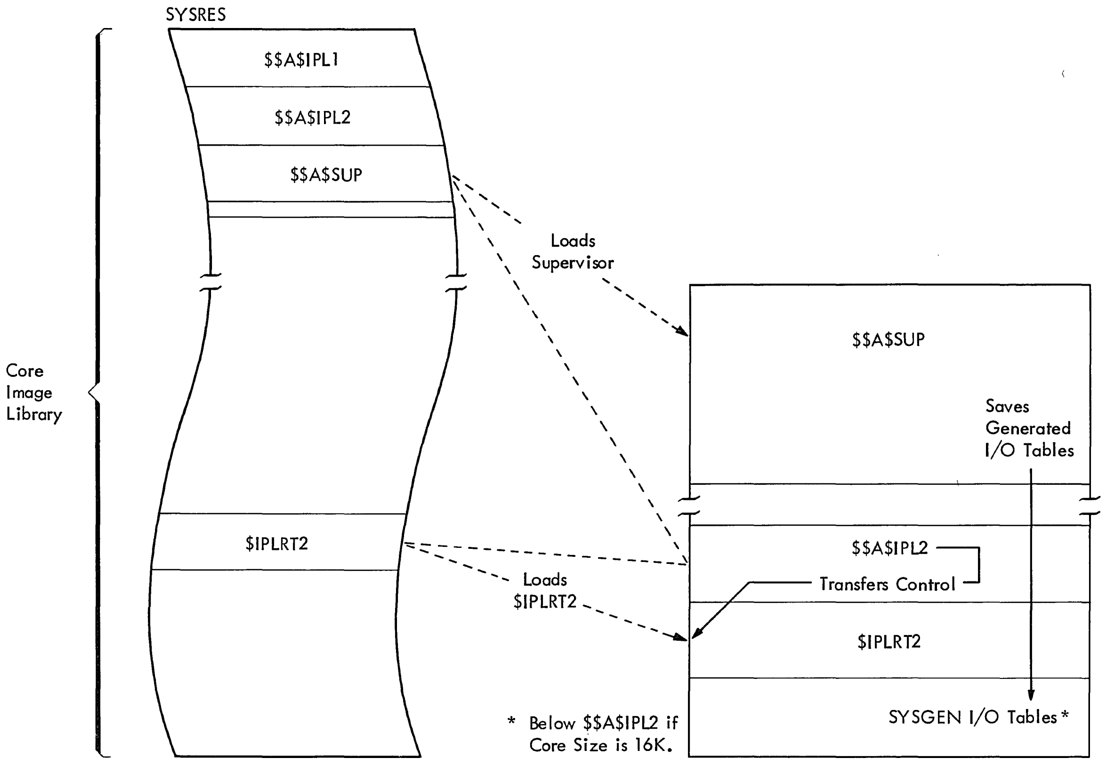


Figure 39. $$A$IPL2

Finally, $IPLRT2 is loaded by an SVC of 4. Figure 39 shows the layout of main storage at the end of $$A$IPL2. Control is given to $IPLRT2.

$IPLRT2 - CHART GC

Objective: To generate I/O tables and to transfer control to Job Control.

Entry: From $$A$IPL2.

Method: $IPLRT2 is the main routine controlling the generation of new I/O tables. After initializing, it calls the read and scan routines (Charts GD and GE), which in turn call the ADD, DELETE and SET routines (Charts GF, GG, GH).

After a SET statement is processed, this routine expects no further input on the communications device. It must resolve the I/O assignments in high and low storage and restore the updated high storage I/O tables to low storage.

First the communications device is resolved. The logical unit for the communications device is SYSRDR or SYSLOG. The routine checks whether the communications LUB in high storage is assigned to some PUB. If it is, and no other LUB has the same PUB-pointer, the Job Control flags in the associated PUB are cleared.

In any case, the PUB is located. There must be a PUB in high storage for this device. The device type in the PUB is checked to determine if it is the same as was used by IPL. The PUB is then permanently assigned to the LUB for SYSRDR or SYSLOG.

The process is repeated for SYSRES. The I/O tables are then complete except for possible reordering of the PUB and LUB table necessary for operating with a multiplex channel. The LUB's for SYSRDR, SYSLST, SYSLOG, and SYSIPT are unassigned if they are assigned to tapes. Job Control must make these tape assignments to open the files properly.

Finally, the I/O tables are moved to their normal position in low storage. If the communication device is a 1052, an exit message is written. All registers except 1 and 2 are set at zero. A program to load Job Control is moved into the transient area and given control.

READ SUBROUTINE (READRT) CHART GD

Objective: To read a statement from the communication device.

Entry: From $IPLRT2 when a control statement is required.

Method: A check is made to see if the communication device is a 1052 or a card reader. If it is a 1052, a test is made at LOGSTR to see if it is the first statement. If so, a message is written by MSGRTN: GIVE IPL CONTROL STATEMENTS. The CCW is then set to read, and the program branches to IOHLD.

If it was not the first statement, the buffer is blanked at LOGRED and the program branches to IOHLD.

At IOHLD the CCB address is saved in case a read error occurs. A supervisor call is issued to read a card from the communication device. The device can be assigned to SYSRDR or SYSLOG. After reading, a test is made for any I/O errors. If there was an error, SYSRES is rewound, if possible, and a wait is entered with the error code in storage location 0-3. The job must be restarted.

If there were no errors, the program returns to $IPLRT2 at REDRET, and the statement is ready to be evaluated.

SCAN SUBROUTINE (OPRTN) CHART GE

Objective: To scan the statement and determine if it is an ADD, DELETE, or SET statement.

Entry: From $IPLRT2 after a statement has been read.

Method: The statement is translated into internal machine codes for evaluation. A search is made for the first position of the op-code. At TRTBRC the op-code is tested to determine if the first character is a legal character. The function code of this translate-and-test instruction plus the location counter is used to control the next instruction. If the function code is 4, an error branch is performed. If the code is 8, the character is legal and a branch to TABRET is performed.

The beginning address of the op-code is stored, and a translate-and-test for the end of the op-code is done. At TRTBRC the delimiter character is tested for a blank. If it is not blank, an error branch occurs. If the delimiter character is a blank, the length of the op-code is calculated. The operation table field is initialized by MVCOP, and the op-code is moved into the end of the operation table.

After the length of the card is found, TRTFFD gets the first operand. It is tested for legality. After the address of the first operand is stored in FLDST, the registers are initialized to set up to search for the type of card being processed.

A search increment factor is set in register 4 at CDSCH to step through the table looking for the operation (op-code). A compare is made after each increment to see if the type of statement equals the factor in the table. When it does, the branch address is picked up from the factor table and a branch is made to the subroutine that will handle the statement. This subroutine will be the one specified in the card: ADD, DELETE, or SET.

ADD STATEMENT SUBROUTINE (ADDRTN) CHART GF

Objective: To add a device to the PUB table in high storage.

Entry: From the scan routine when an ADD statement is identified.

Method: A check is made at the start of the ADD routine to determine if there is space in the PUB table for one more device. If not, a message, CANNOT ADD PUB-INSUFFICIENT TABLE SPACE, is written. If there is space, FDSRTN gets the first operand.

If the device address operand includes a key, the operand registers are stored for later evaluation. The channel and unit numbers are checked and converted to hexadecimal by HEXRTN.

The PUB table is scanned to see if the PUB to be added already exists. If it is in the table, a message prints out PUB ALREADY EXISTS. If not, FDSRTN brings in the second operand, the device type specification. FNDTYP scans for the corresponding device type byte and stores it in the temporary PUB. It also initializes the TEB area if there is a TEB associated with this device.

The statement is tested to determine if there is another operand specifying device options. If so, it is brought in by FDSRTN. The options are converted to hexadecimal by HEXRTN, and stored in the PUB.

The key registers are picked up at KEYCHK and tested to see if there is a key field to establish. If there is not a key to establish, control goes to PUBMKE. This key is either a switchable device indicator or a priority-on-channel indicator.

If the key indicates a switchable device, the following values are set:

- Switchable device bit in the PUB channel scheduler flag
- Minimum priority key for the low-order channel
- Branch address for the build PUB routine.

If it is not a switchable device, the priority key is converted to decimal and, at PUBMKE, the branch address for the build PUB routine is set.

PUBMK1 sets the PUB table address for the build PUB routine.

BLDPUB builds the PUB for the device the operator has specified on the ADD card. After the PUB is built, CHURTN is used to update the FOCL. LUURTN updates the LUB table. This completes all the necessary updating of the I/O tables for the device that was added. The program returns to the main line at OPRET.

DELETE STATEMENT SUBROUTINE (DELRTN) CHART GG

Objective: To delete a device from the PUB table in high storage.

Entry: From the scan routine when a delete statement is identified.

Method: FSDRTN gets the first operand of the statement to be evaluated. It is checked for correct length and legal channel address. After the channel number and the corresponding FOCL entry are determined, the channel is tested to assure there is information on it.

The PUB's for the specified channel are searched for the specified device. When the device to be deleted is found, its address is saved.

When the PUB's have been searched, the program goes to SCNEND. There a check is made to make certain that the device to be deleted was found. If there is a TEB associated with the PUB, it is deleted by DEBLOP and the program goes to PUBDEQ. If there is no TEB for the PUB, control goes directly to PUBDEQ.

At PUBDEQ the address and the length of the portion of the PUB table following the PUB to be deleted are set up. Then SYSMVC deletes the PUB by moving that portion of the table down 8 bytes.

After the PUB is deleted, the FOCL bytes and the PUB-pointers in the LUB table must be updated.

On completion, control returns to $IPLRT2 at OPRET to read another card.

SET STATEMENT SUBROUTINE (SETRTN) CHART GH

Objective: To set the system date and time of day.

Entry: From the scan routine when a set statement is identified.

Method: After all the ADD and DELETE statements have been read, a SET statement must be read to set the time of day and the date.

There are two formats for the date. The American format is month followed by day. The European format is day followed by month. The date configuration byte, displacement 53 in the communications region, specifies the form used by the system.

The first operand of the card is brought in by FDSRTN and tested to see if it is the date field. If it is the date field, the system date is set in the Supervisor by DATERT.

A test is made to see if a timer is present. If not present, return is to $IPLRT2. If a timer is present, the next operand is brought in by FDSRTN and tested to see if it is the clock field. If it is not the clock field, it is in error. If it is the clock field, the hour, minute, and second of the day are set in time-of-day (location 84) by TIMERT. The program returns to $IPLRT2.

If the first operand is not the date field, it is tested to see if it is the time field. If it is not the time field, it is in error. If it is the time field, the hour, minute, and second of the day are set in time-of-day (location 84) by TIMERT. The next operand is brought in by FDSRTN and tested to see if it is the date field. If it is not, it is in error. If it is the date field, the system date is set in the Supervisor by DATERT and the program returns to $IPLRT2.

SUBROUTINES FOR $$A$IPL2 AND $IPLRT2

<u>BLDPUB</u>: Given the channel and desired position, inserts a PUB entry in the PUB table.

LUURTN: Given the PUB number of an inserted or deleted PUB entry, modifies PUB pointers in the LUB table as required.

CHURTN: Given the channel for which a PUB entry has been inserted or deleted, updates the FOCL.

SYSMVC: Given the length of a field, its location, and where it is to be moved, moves the field.

SUBROUTINES FOR $IPLRT2

COMCHK: Checks a device against a list of allowable devices. Error return if no entry found.

PBFRTN: Given a low-storage PUB entry, locates the high-storage PUB entry for the device. Error return if no entry found.

DBLRTN: Compares LUB's to determine if more than one LUB points to a particular PUB.

MPXRTN: Assures that the PUB table is correctly ordered and the LUB table modified for bursting on a multiplex channel.

OPNRTN: Unassigns any system I/O units (SYSRDR, SYSIPT, SYSPCH, SYSLST) that have a SYSGEN assignment to a tape unit.

ASNRTN: Assures that the PUB job control flag is correct for each assignment made at system generation time.

FDSRTN: Locates the next operand in the input area.

FNDTYP: Analyzes device type specified in an ADD statement to pick up tabled information.

DECRTN: Converts a field from decimal to binary.

HEXRTN: Converts a field from hexadecimal to binary.

DATERT and TIMERT: Update communications region date field and time.

ERROR HALTS FOR $IPLRT2

ILLCD, COMNFD, RESNFD, TEBEXD, PUBEXD, DBLADD, DELEXT, RESERR: Issues message. If communications device is SYSLOG, attempts to read another statement. Otherwise rewinds SYSRES and generates a wait, with error code in low storage.

PRCERR: If communications device is SYSLOG, issues message. If communications device is SYSRDR, rewinds, generates a wait, and puts error code in low storage.

IOHALT: Rewinds unless error is on SYSRES. Generates a wait.

LINKAGE EDITOR

The Linkage Editor prepares programs for execution on 16K Tape BOS. The Linkage Editor accepts as input the relocatable object modules produced by the language translators. It processes these modules into program phases, which may be immediately executed or cataloged into the core image library.

Linkage Editor control cards direct the program to read input module(s) and form phases from the control sections within the modules. Figure 40 shows how phases can be formed. Linkage Editor relocates the origin of each control section in the phase, assigns each phase an area of main storage and a transfer address, and modifies the contents of the address constants in the phase.

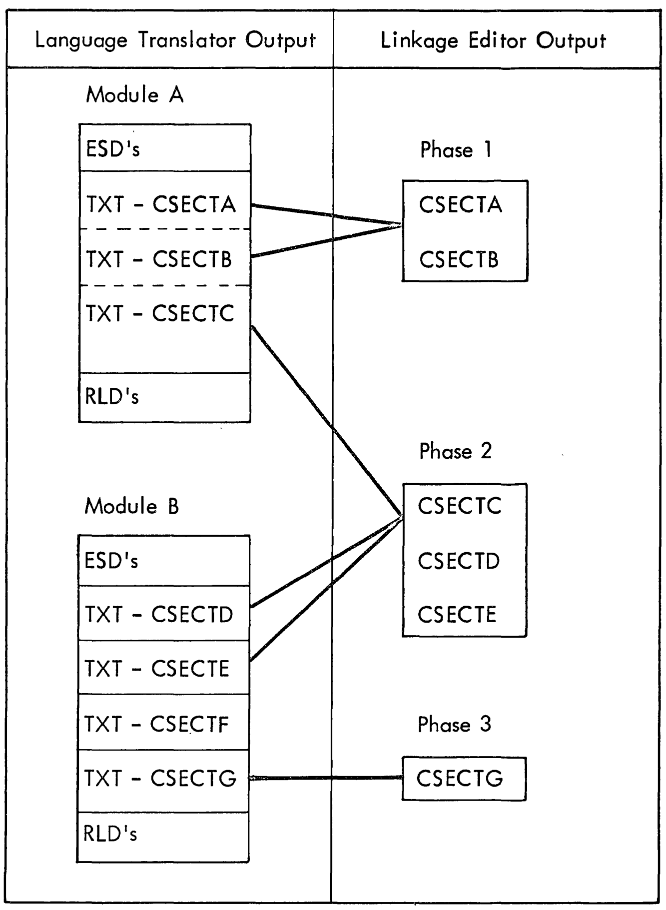


Figure 40. Module-Phase Relationship

The relocation factor for each control section is determined and saved by building a table called the control dictionary. This table contains the Linkage Editor phase definitions and the module ESD items. When complete, it provides sufficient information for determining the location of each control section and for resolving any references between control sections.

The module TXT items are then built into phase blocks. The RLD items (address constants) are modified and inserted into the text. A transfer address is determined for each phase.

LANGUAGE TRANSLATOR MODULES

The input to Linkage Editor consists of object modules and Linkage Editor control cards. Each module is the output of a complete language translator run. It consists of dictionaries and text for one or more control sections.

The dictionaries contain the information necessary for the Linkage Editor to resolve references between different modules. The text consists of the actual instructions and data fields of the module.

Five card types (described in Appendix E) are produced by the language translators or the programmer to form a module. They appear in the following order.

Card Type	Definition
ESD	External Symbol Dictionary
TXT	Text
RLD	Relocation List Dictionary
REP	Replacement to text made by the programmer
END	End of module

The External Symbol Dictionary contains control section definitions and intermodule references. When the Linkage Editor has the ESD's from all modules, it can relocate the sections and resolve the references. Five types of entries are defined in the control dictionary.

ESD Type	Definition
SD	Section Definition; provides control section name, assembled origin, and length
PC	Private Code; provides assembled origin and length for an unnamed control section
LD	Label Definition; specifies the assembled address and the associated SD of a label which may be referred to by another module
ER	External Reference; specifies the location of a reference made to another module
CM	Common; indicates the amount of main storage to be reserved for common use by different phases

The Relocation List Dictionary identifies portions of text that must be modified on relocation (address constants).

When Linkage Editor reads a module, it stores ESD's in its control dictionary, writes TXT and REP items on a TXT file, and writes RLD items on an RLD file. Each item, identified by the language translators with an ESID number, is identified by the Linkage Editor with a control dictionary number to avoid duplication of identification between modules.

LINKAGE EDITOR PROGRAM FLOW (FIGURE 41)

Linkage Editor is logically divided into four passes. Pass 1 reads the input, builds the control dictionary, checks for errors, and writes the TXT and RLD files. Pass 2 reads the TXT file and builds core image blocks. Pass 3 reads the RLD file, modifies the address constants, and merges them into the core image blocks. Pass 4 handles abort errors and assures that the Linkage Editor output is on SYS000.

Pass 1 actually consists of three phases:

- Overhead Processer
- Control Card Processor
- 12-2-9 (Module) Card Processor.

Job Control brings in the first phase when it gets Linkage Editor. The initialization section of this phase determines the end of the problem program area.

1. If 16K, the two card processors share the same area in storage. The Overhead Processor determines the input type and assures that the correct phase is present.
2. If greater than 16K, all three phases can be in storage at the same time.

The Overhead Processor reads Linkage Editor input and provides a logical record to the correct card processor. This phase contains the routines for the Linkage Editor control cards ACTION and INCLUDE.

ACTION must precede all other input and is grouped, therefore, with the initialization routines. It sets option bits in the Linkage Editor communications region.

INCLUDE determines the location of Linkage Editor input and is grouped, therefore, with the input routines.

This phase also contains subroutines common to the card processors.

The 12-2-9 Processor handles the input module cards.

ESD items are placed into the linkage table with a control dictionary number. Each item is entered into the control dictionary. For each ESD item of type SD or PC, a relocation factor is calculated. For each item of type LD or ER, an attempt is made to resolve the item into an LR.

TXT items are assigned control dictionary numbers corresponding to their ESID numbers and written on the TXT file (SYS001).

RLD items are assigned control dictionary numbers corresponding to their ESID numbers and written on the RLD file (SYS002).

REP items are converted to binary and treated like TXT items.

END items signal the end of a module. The linkage table is destroyed. The transfer address from the first END in a phase is saved in the Linkage Editor communications region.

The Control Card Processor handles the PHASE and ENTRY Linkage Editor control cards.

PHASE card processing constructs an entry in the control dictionary and writes a phase record on the RLD file. If not the first phase card, the routine writes a transfer record on the TXT file for the previous phase and performs AUTOLINK.

ENTRY card processing writes a transfer record on the TXT file for the last phase and performs AUTOLINK. It adjusts all the relocation factors by the length of common and, if a transfer address is given in the card, stores an overriding transfer address into the Linkage Editor communications region.

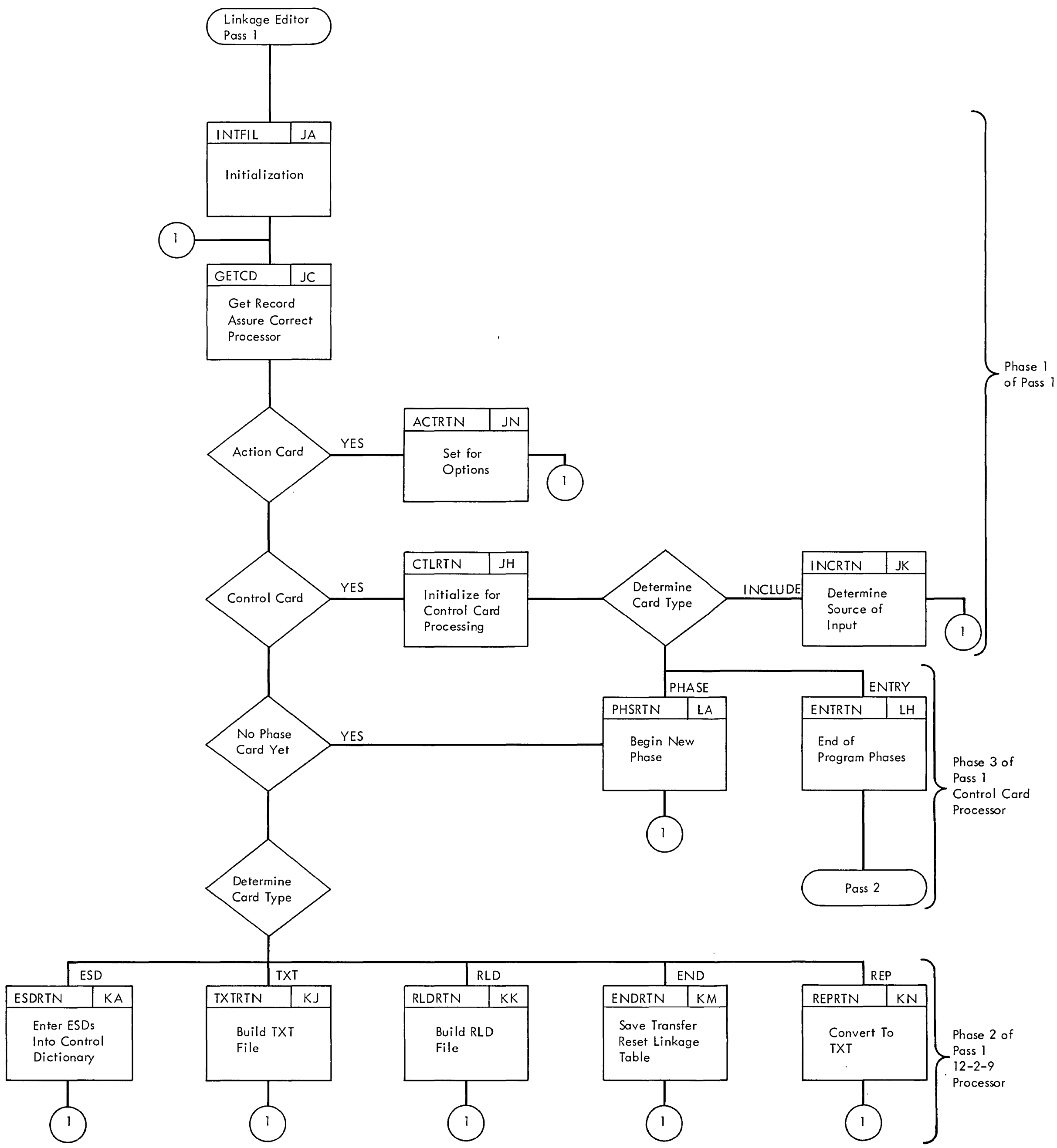


Figure 41. Linkage Editor Program Flow (Part 1 of 2)

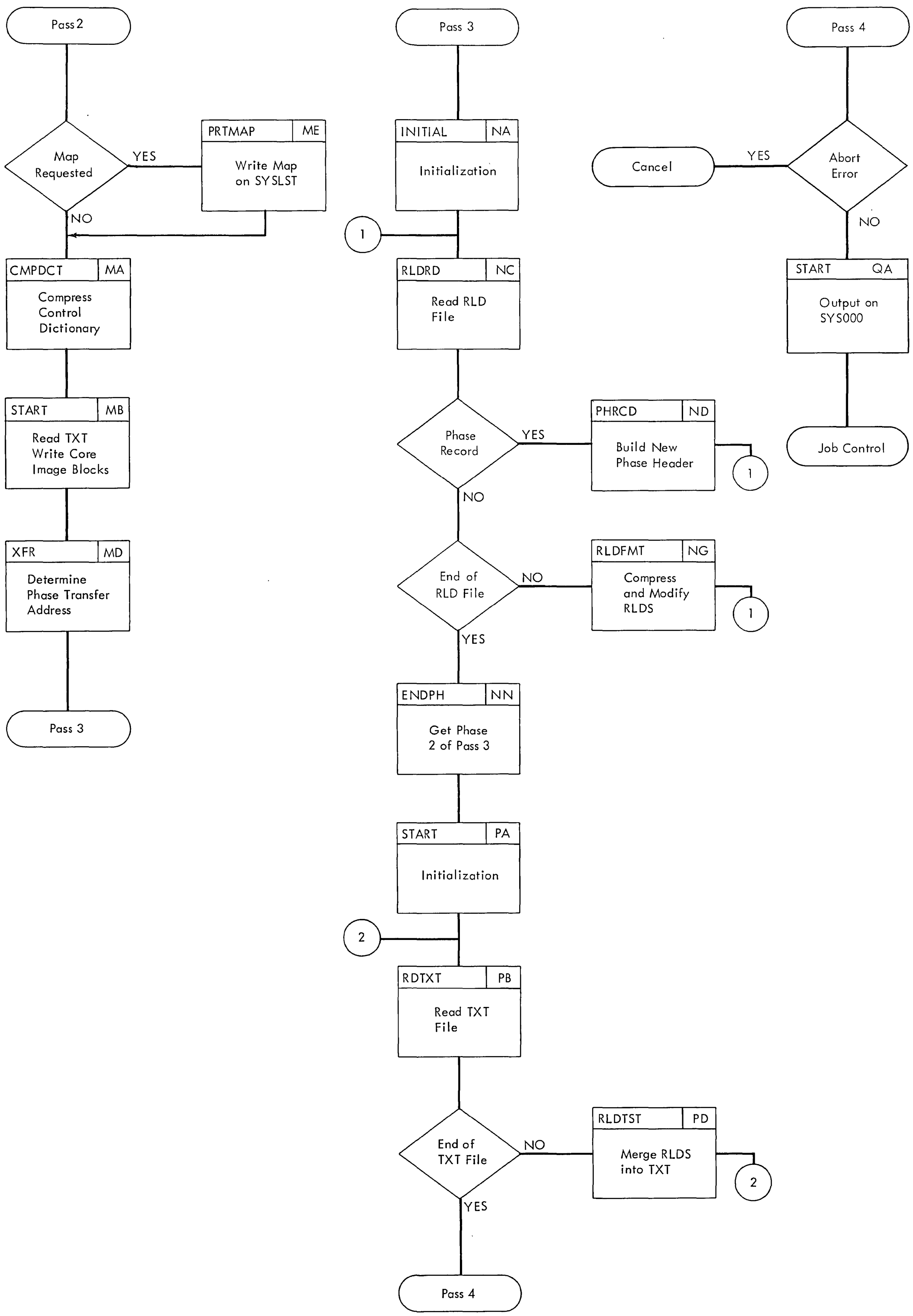


Figure 41. Linkage Editor Program Flow (Part 2 of 2)

Pass 2 is a single phase TXT processor. It writes a MAP on SYSLST, if requested, and compresses the control dictionary to half its Pass-1 size. It reads the TXT file (SYS001) and forms core image blocks, phase by phase, using the control dictionary to fix storage locations. The blocks are written on SYS002. Should a backward origin occur, an alternate unit (SYS000) is used to rewrite the blocks. At the end of the pass, output may be on either file.

Pass 3 is a two-phase RLD processor. The first phase determines if all the RLD's can be processed in storage or if they must be processed phase by phase. The routine reads the RLD file (SYS002) and modifies both the addresses and the text of the address constants from the control dictionary. If processing phase by phase, the modified and compressed RLD's are written on SYS001.

The second phase in Pass 3 reads core image blocks from SYS002 or SYS000, merges in the modified RLD's, and writes the completed phase on the alternate unit, SYS000 or SYS002, along with a phase header.

Pass 4 writes any warning messages and cancels the job if a serious error has occurred. assures that the final output is on SYS000 and calls in Job Control.

LINKAGE EDITOR CORE ALLOCATION (FIGURE 42)

Initially, Job Control brings the first phase of Linkage Editor into main storage. The first phase defines the Linkage Editor communications region, the Overhead Processor, the input area, and the initialization routines.

The initialization routines determine, from the size of main storage, how to allocate space to the TXT and RLD buffers and the linkage table and control dictionary area. They also determine if both of the remaining phases of Pass 1 fit into main storage at the same time. If this is a minimum system, these two phases must share main storage.

Entries to the control dictionary are made from high-to-low storage. Each entry is sixteen bytes long.

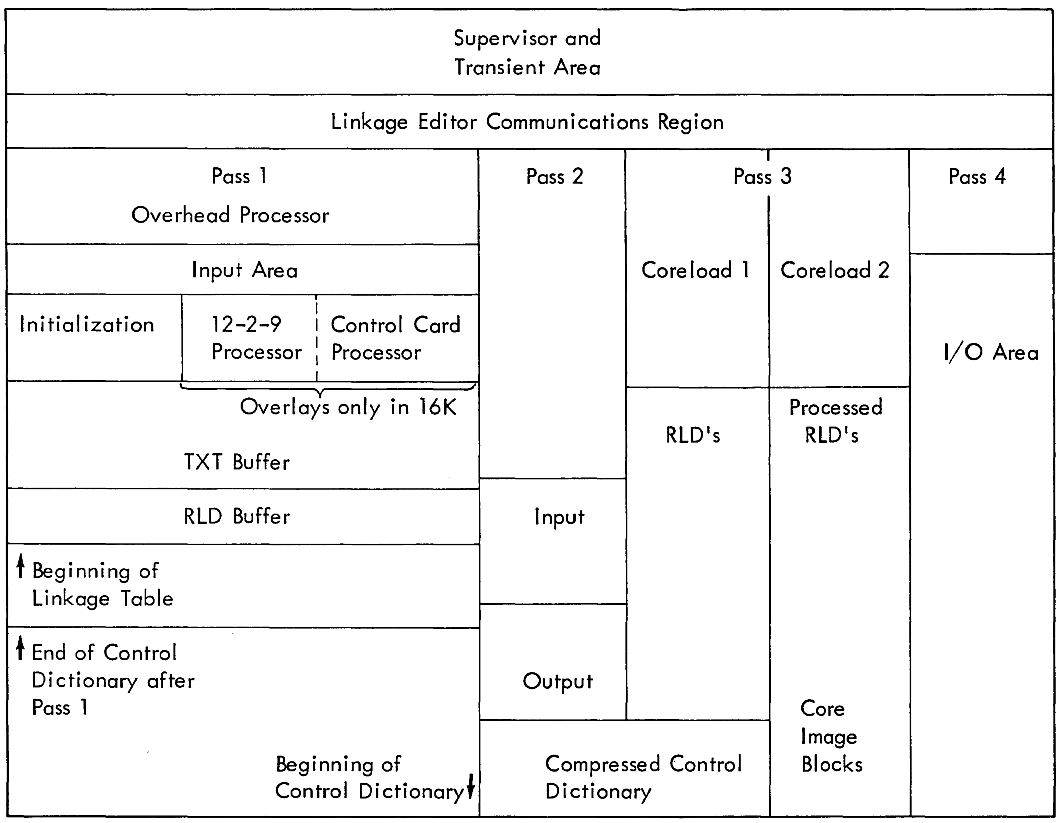


Figure 42. Linkage Editor Core Allocation

PHASE entry	0-7:	phase name
	8:	type = '07'
	9-11:	phase origin
	12:	--
	13-15:	phase end
ESD entry	0-7:	label
	8:	type = '01' for LD, '02' for ER, '03' for LR, '04' for PC, and '05' for CM
	9-11:	assembled origin
	12:	phase number
	13-15:	control section length for SD, PC, or CM; control dictionary number of section definition entry for LD or LR; blanks for ER.

Entries to the linkage table are made from low-to-high storage. The entries are ordered by ESD identification number by module. Each entry is three bytes long.

Linkage Editor	0-1:	control dictionary number
	2:	ESD type

PHASE entry	0:	ESD type = '07'
	1-3:	phase transfer address
	4-5:	number of blocks in phase
	6-7:	number of characters in last block
ESD entry	0:	ESD type
	1-3:	assembled origin
	4:	phase number
	5-7:	relocation factor for SD, PC, or CM; control dictionary number of section definition entry for LD or LR; blanks for ER.

Pass 2 compresses the control dictionary by eliminating the first eight bytes of each entry (as shown in the preceding chart).

LINKAGE EDITOR I/O FLOW (FIGURE 43)

Pass 1 reads the input (Linkage Editor control cards and language translator modules). Input is expected on SYS000, but a Linkage Editor INCLUDE card may direct the program to get a module from the relocatable library on SYSRES or SYSRLB. Appendix E shows the format of the input module cards.

Pass 1 writes text and transfer records on SYS001. The text records are unchanged from the text input except that the ESD identification number (ESID number) is converted to a control dictionary number. Pass 1 writes phase and RLD records on SYS002. The phase record is 20 bytes long:

- 0-1: phase number assigned by Linkage Editor
- 2-9: phase name
- 10: ESD type = '07'
- 11-13: phase origin
- 14-20: Reserved to eliminate noise record.

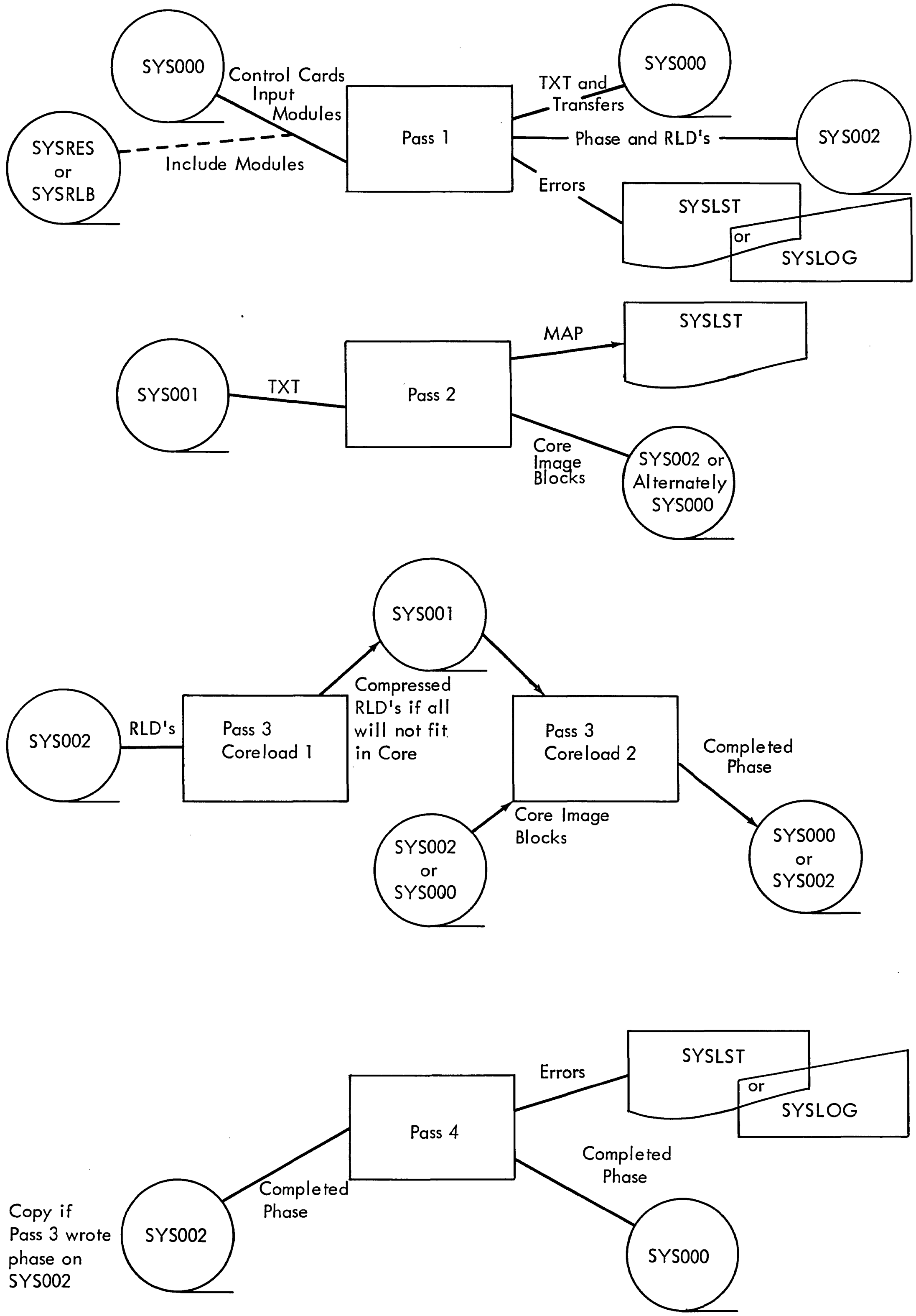


Figure 43. Linkage Editor I/O Flow

The RLD records are unchanged from the text input except that the ESID numbers are converted to control dictionary numbers. Four bytes precede the blocked RLD records, indicating number of records per block and number of bytes per record.

Pass 2 writes a Linkage Editor MAP if requested. Pass 2 reads the text records from SYS001 and writes core image blocks on SYS002 or SYS000. Each block is identified by a phase number. Pass 2 defines the phase transfer address.

Pass 3 reads the RLD records from SYS002 and compresses them. If they cannot all be processed in main storage at one time, the compressed RLD's are written on SYS001, phase by phase. The second part of Pass 3 then reads RLD's from SYS001 to merge with text from SYS002 or SYS000.

Pass 3 writes the completed phase on the third tape (SYS000 or SYS002) as it appears in the core image library. A 61-byte header/trailer record precedes the phase:

0:	"C"
1-30:	phase header
31-60:	trailer; header of the previous phase; zeros for the first phase.

The phase header contains:

0-7:	phase name
8-11:	phase origin
12-15:	phase end
16-19:	transfer address
20-23:	end of longest phase
24-25:	bytes per block
26-27:	number of blocks
28-29:	number of bytes in last block.

The core image records are written in 4K blocks for a 16K system or on an MINSYS option, 12K blocks for 32K, or 32K blocks for 64K.

PASS 1-CORELOAD 1

I/O INITIALIZATION (INTFIL) CHART JA

Objectives:
1. Determine the location of the relocatable library and initialize to use it.
2. Open tape files used by the Linkage Editor.
3. Rewind work tapes.
4. Initialize print control flags.

Entry: From a fetch of the Linkage Editor program by Job Control.

Method: This routine resets the error register and uses the facilities of the Supervisor to rewind tapes SYS001 and SYS002. It determines if a private relocatable library (SYSRLB) is assigned. If it is assigned, this routine:
1. Turns on the RLB flag.
2. Turns on the rewind SYSRLB switch.
3. Overlays the contents of location LIBLUB with SYSRLB.
4. Rewinds SYSRLB.

The routine next opens tape files SYS001 and SYS002 and performs these steps:
1. Turns on the appropriate print control flag if SYSLOG equals SYSLST.
2. Turns on the log print flag if SYSLOG is a printer.
3. Branches to initialize main storage (Chart JB) if SYSLST is not assigned.
4. Sets the line count if necessary.
5. Determines device type.
   a. Exits to initialize main storage (Chart JB) if the device is a tape.
   b. Turns off the ASA flag and the print control available flags before branching to initialize main storage, if the device is a printer.

<u>Note</u>: If the relocatable library is not a private library, it is located on SYSRES following the core image library.

STORAGE INITIALIZATION (INTCRE) CHART JB

Objectives:
1. Determine the end-of-supervisor address.
2. Align addresses on appropriate boundaries.
3. Determine the storage loads required by Pass 1.
4. Establish optimum sizes for TXT and RLD records, RLD buffer, and core image blocks.
5. Initialize linkage editor communications region with information about buffer dimensions.
6. Clear main storage used by linkage table and control dictionary.

Entry: From the I/O initialization routine, Chart JA.

Method: Using the supervisor communications region, this routine computes the address of the first usable byte. This is either the address of the end of the problem program label area plus one byte doubleword aligned, or the address of the beginning of the problem program area, whichever is greater.

The routine branches to a subroutine to align this address on a doubleword boundary, if necessary. It saves the aligned address for use in the first phase origin of the program to be linkage-edited, or for use in processing common areas.

The routine next initializes these locations in the linkage editor communications region:

1. CTLDAD with the address of the last entry made in the control dictionary (initially CDENT1+16).
2. CDENT1 with the address of the first control dictionary entry. The control dictionary is built from high- to low-storage. The first entry is located at the end of the problem program area.

The routine determines if minimum main storage is available by comparing the end-of-pass-3 address with the end-of-problem program address. If the end-of-pass-3 address is higher than the end-of-problem program address, it indicates less than minimum storage is available. In this case, the routine branches to location TRYLE to set up for a 16K configuration, and follows with a branch to location INIT1. If Pass 3 fits, the routine uses a table look-up technique to determine machine size, and the size of the tables and buffers associated with the machine configuration found.

At location INIT1, the routine initializes linkage editor communication region locations:

1. CIBLOC with a value equal to half the size of output core image blocks.
2. RLDSIZ with size of RLD input records to Pass 3.

It next tests for 16K machine configuration, and if found, branches to location INIT2 to split Pass 1 into three separate coreloads. If this is not a 16K configuration or if Pass 1 has been split into three coreloads, the routine continues at location INIT4.

Starting at INIT4 the routine initializes these locations:

1. TBFAD with the test buffer starting address.
2. WRTXT (CCW).
3. TBFEND with the text buffer ending address.
4. RBFAD with the RLD buffer starting address.
5. PHRLD (CCW).
6. RBFEND with the RLD buffer ending address.

It locates the starting address of the linkage table, determines the main storage available for the control dictionary, and executes a clear of the last 512 bytes of main storage. (Job Control clears remaining area prior to fetching Linkage Editor.) The area between the starting address of the linkage table and the end of the control dictionary is thus zeroed.

The routine resets the first four bytes of both the TXT buffer and the RLD buffer and performs these steps:

1. Branches to location GETCD if Pass 1 is more than one coreload.
2. Moves the load routine (LODORG, Chart JD) into the TXT buffer for use in the processor fetch routines.
3. Tests for a minimum size core image block request (MINSYS option given to job control).
   a. Branches to GETCD if no request has been made.
   b. Sets up location CIBLOC in the linkage editor communications region for minimum size output blocks (4K) before branching to GETCD, if the request has been made.

GET CARD PROCESSOR (GETCD) CHART JC

Objective: Determine and verify card operation, and call appropriate fetch and processor routines.

Entry:

1. Initially from the initialize core storage routine, Chart JB.
2. Thereafter from any of the card processor routines.

Method: Because this routine has multiple entries, it is tailored to fit different situations, which are discussed separately.

Initial Entry

The dummy phase card generated flag has been turned off by the main storage initialization routine. Input is not expected form SYSRES. A logical record is required for continued processing.

Dummy Phase Flag On

The first phase card is missing and a linkage-editor-supplied substitute phase card has been generated. The relocatable library is not a factor to be considered at this time. Another logical record is not required for continued processing.

Operations with the Relocatable Library

The dummy phase flag is off. The physical location of the relocatable library is determined:

1. When the relocatable library is on

SYSRLB, the monitor gets the logical record.
2. When the relocatable library is on SYSRES, the monitor branches to obtain the 12-2-9 processor, if necessary, before getting the logical record. Based on the high probability the 12-2-9 processor will be used, the fetch minimizes tape movement.

Starting at location GETGO all situations share common processing. The routine continues by performing these steps:
1. Branches to the action routine if action card images are expected.
2. Branches to the identify control card routine if the first column of the card image is blank.
3. Scans for a specific 12-2-9 card type.
4. Builds a dummy phase card, sets up the link register address, and branches to the control card fetch routine if no phase card has been read.
5. Sets up the link register address, and branches to the 12-2-9 card fetch if a match is found in the scan and a phase card has been read.

FETCH SUBROUTINES (CTLFCH OR TNTFCH) CHART JD

Objective: Fetch the card processor required by the calling routine.

Method: Depending on the entry point, this routine initializes to fetch either the control card processor or the 12-2-9 card processor. After this initial step has been completed, the remainder of the routine is common to both types of fetch.

The routine determines if the processor is already in main storage. If it is, the routine branches to the address in the link register because no fetch is necessary. If it is not present, the routine tests a program switch to determine if Pass 1 is to be a single or multiple coreload. Test results and actions taken are:

- All one's condition. Branch to load routine (LODORG, moved into the TXT buffer during core initialization) to fetch Pass 1 in a single coreload.
- All zero's condition. Get the name and origin of the card processor to be fetched and issue the fetch supervisor call.
- Mixed one's and zero's. Branch to the address in the link register, the desired processor is already present.

Note: The link register is used to properly route the program to the next sequential instruction. By modifying this address to the starting address of the desired processor, this sequence is established: call for fetch, issue fetch, and execute card processor.

INPUT SUBROUTINE (GETRCD) CHART JE

Objectives:
1. Branch to get a logical record.
2. Update record counters.
3. Test for end-of-file.
4. Test for wrong-size records.

Method: This routine determines if a logical record is needed by comparing the work count with the record count. If necessary, it branches to Chart JF to get the record. If end-of-file or wrong-size record is found, the routine issues the appropriate error message and aborts.

If the input is good, the routine updates the physical record count, and resets the logical record count.

If no physical record was required or if counter updating and error checking are complete, the routine is located at SKPRD. There it increases the logical record count by one and returns to the address in the link register.

GET RECORD SUBROUTINE (CHKRTN) CHART JF

Objectives:
1. Determine the tape movement required to obtain the desired physical record.
2. Call the I/O routine necessary to position tape or read a physical record.

Method: This routine restores the main input CCW and determines the relative location of the record desired on the tape. Based on the signed difference between the present position and the desired position, the routine performs these steps:

1. Plus difference - backspaces tape.
2. Minus difference - forward spaces tape.
3. Zero difference - does not move tape.

After orienting the tape, the routine initializes the CCW to read and branches to the I/O routine, Chart JR, to physically read the record. When I/O is finished, the routine increases the position count by one, indicating the record has been read, and returns to the address in the link register.

IDENTIFY CONTROL CARD (CTLRTN) CHART JH

Objective: Inspect the operation field of control cards to determine card type and appropriate action to be taken.

Entry: From the GETCD routine.

Method: This routine branches to the position routine to obtain the location of the operation field from the card image. Based on conditions found, the routine performs in this manner:

1. No operation field found. Branch to GETCD.
2. Entry card found. Branch to CTLCHK.
3. Program not in control mode. Print card in error and branch to GETCD.
4. Phase card found. Branch to CTLCHK.
5. Action card found. Branch to GETCD.
6. No control card found. Determine if a control card is expected.
   a. Print an error message and branch to GETCD if it is expected.
   b. Branch directly to GETCD if no control card is expected.
7. Include card found. Continue at CTLCHK.

Starting at location CTLCHK the routine tests the submodular flag, SUBFLG, to determine the condition of the search for control card and control card found switches.

1. Submodular flags both on. Branch to GETCD.
2. Submodular flags both off. If PHASE or ENTRY, and AUTOLINK is to be done, decrement the record count and branch to ALNKPR. Otherwise, print the card and branch to the INCLUDE routine, Chart JK, or CTLFCH, Chart JD, to get the control card processor.
3. Control card found switch off, search for control card switch on. Turn on control card found switch and branch to GETCD.

POSITION TO OPERAND SUBROUTINE (POSRTN) CHART JJ

Objective: Position a register address pointer to a desired field within a card image.

Method: This routine searches for the first nonblank character. If the entire card is blank, the routine branches to the address in the link register. If a nonblank character is found, the routine must search for a field-delimiting character. Valid delimiters are: blank, comma, left parenthesis, right parenthesis, minus sign, or plus sign.

If the list of field delimiters is exhausted and none has been found in the card image, the routine prints a message and branches to location GETCD. If the delimiting (stop) character is found, the routine:

1. stores the field origin
2. calculates the field length
3. saves the delimiting character
4. returns to the address in the link register.

Note: When the position routine is unable to find the field, it branches to the address in the link register. If the field is found, the position routine modifies the link register address so that the branch is made to the instruction following the unconditional branch.

INCLUDE CARD PROCESSOR (INCRTN) CHART JK

Objectives:

1. Check the operands of the INCLUDE card image to determine:
   a. INCLUDE level
   b. use of submodular structure.
2. Find a desired module in the relocatable library.

Entry: From the identify control card routine, Chart JH.

Method: This routine branches to the position routine to locate the operand field of the include card image. A totally blank field causes the card image to be ignored. Because the include card image can have optional operands, this routine determines the options used.

Option 1

,(namelist) shows the same level of include condition. This means the control sections specified in the namelist are either in the main input stream (SYS000) or are within the same module (specified by a previous include) in the relocatable library. When option 1 is identified, the routine:

1. Branches to the scan routine, Chart JM, to validity-check the operand format and put the control section names, specified in the namelist, into a work area.
2. Turns on a switch (same level), and sets the submodular level indicator at the appropriate value.
3. Branches to ESDCHK to insure that the 12-2-9 processor is resident in main storage because ESD processing can be expected.

4. Branches to GETCD for the next card image.

Option 2

Modulename, (namelist) shows this is not a normal include. This means the control sections specified in the namelist are in the relocatable library. Before testing for option 2 or option 3, the routine puts the module name (first operand) into a work area. When option 2 is identified, the routine:

1. Branches to the scan routine to validity-check the operand format and put the control section names, specified in the namelist, into a work area.
2. Branches to the nesting routine, Chart JQ, to update the nest list.
3. Sets the submodular level indicator at the appropriate level.
4. Continues at location SCHDT1.

Option 3

Modulename shows a normal include. This means the specified module is in the relocatable library. When option 3 is identified, the routine branches to the nesting routine, Chart JQ, to update the nest list. It continues processing at location SCHDT1.

Starting at SCHDT1, the routine is generalized to process either option 2 or option 3 because both options use the relocatable library. It insures that the 12-2-9 processor is resident in main storage because ESD processing is expected. It tests the library flag (IOSFLG) for one of three possible conditions.

1. No library. Go to LIBERR for error processing.
2. Library not open. Go to LIBRTN and LABCK2, respectively, to open the file.
3. Library open. Search for the module name in the relocatable library.

If the module name cannot be found in the relocatable library and end-of-file has been detected, the routine backspaces the file, down dates the nest list, and tests for AUTOLINK. If end-of-file has not been detected, the routine down dates the nest list and tests for AUTOLINK without backspacing tape.

If no AUTOLINK option is specified, the routine branches to send an error message and get another card image. If the AUTOLINK option is found, the routine branches to the AUTOLINK routine, Chart JP.

If the module is found in the relocatable library, the routine saves position and count information, prints an AUTOLINK message if required, and branches to GETCD for the next card image.

Note: Whenever a name list is specified, submodular processing (creation of phases from named control sections) will occur.

ACTION CARD PROCESSOR (ACTRTN) CHART JN

Objectives:

1. Determine the action to take place, and set print control flag switches accordingly.
2. Sense the first nonaction card.

Entry: From the Linkage Editor monitor routine when action cards are being collected.

Method: This routine branches twice to the position routine. The first branch gets the operation field for testing. The second branch gets the operand field for testing.

Operation Fields: If the current card is not a control card, or if it is a nonaction card, go to ACTGO to establish the necessary print routine for mapping. A blank operation field causes the routine to branch to GETCD.

Operand Field: No operand, or an operand of MAP with SYSLST unassigned, or an undefined operand cause an error message to be sent and the routine to go to GETCD. An operand of CLEAR causes a branch to GETCD. (CLEAR is a valid operand for disk only.) An operand of NOMAP causes the routine to turn off the action switch in the print control flag before branching to GETCD. An operand of MAP with SYSLST assigned causes the routine to turn on the action switch in the print control flag before branching to GETCD.

I/O SUBROUTINE (IORTN) CHART JR

Objectives:

1. Determine if a SYSRES reposition test is to be made.
2. Issue the proper supervisor call instruction to get the necessary I/O operation performed.

Entry: From any routine requiring I/O to be performed.

Method: This routine tests the flag at location IOSFLG to determine if a SYSRES reposition test is to be made. If not, it issues a supervisor call to execute the channel program, waits until I/O is finished, and returns to the address in the link register.

If a SYSRES reposition test is necessary, the routine issues a special call to determine if SYSRES has been moved. If it has not been moved, the special request causes the channel program to be executed. When I/O is complete, this routine exits to the address specified in the link register.

If the special request showed that SYSRES had been moved, the routine exits to location RPSRTN within CHKRTN, Chart JF.

Starting at RPSRTN the get record subroutine branches to location LABCK1, Chart JG, to locate the first record of the relocatable library (the header). It tests the position count to find the position of the desired record in relation to the beginning of the relocatable library. After positioning the tape, it reads the record and exits to the address specified in the link register.

PASS 1-CORELOAD 2

ESD PROCESSOR (ESDRTN) CHART KA

Objective: To process ESD items into the linkage table and the control dictionary.

Entry: From GETCD, via the routine that fetched the card processor.

Method: The routine tests for a submodular structure. If it is found, it either branches to down date the nest list (Chart JQ, if this is primary input, the point the submodular structure was requested), or it branches to GETCD after resetting the submodular flag.

If the initial test determined no submodular structure, this routine calculates and saves number of bytes to be processed. It checks the validity of values in the ESD type field. Types 0, 1, 2, 4, and 5 are valid. Other values are in error, and the routine branches to location ERR40 for error handling.

CONTROL DICTIONARY	ESD ITEM FOR PROCESSING				
	LD	SD	PC	CM	ER
LD	aa	ba	ca	da	ea
LR	ab	bb	cb	db	eb
SD	ac	bc	cc	dc	ec
PC	ad	bd	cd	dd	ed
CM	ae	be	ce	de	ee
ER	af	bf	cf	df	ef
NO MATCH	ag	bg	cg	dg	eg

Figure 44. ESD Control Dictionary Decision Table

The ESD item name or label is compared with control dictionary name fields. Figure 44 illustrates all possible combinations resulting from this comparison. You can determine the action taken by the routine for a given condition by looking first at the ESD item being processed. Next, look at the control dictionary entry whose name field matches the ESD item. The two alphabetic characters at the junction are a key to the actions taken by the routine. A provision is also made for a search that resulted in the no-match condition. Again the junction of the ESD item and the no-match condition contains a key that points you to the action taken by the routine.

aa: Assembled origins must agree and the ESD LD must be assigned. Compare the control dictionary number of the ESD LD with the control dictionary number of the control dictionary entry (an LD). Equal control dictionary numbers indicate exact duplicate LD's. Therefore, ignore this ESD item. If the control dictionary numbers do not match, compare the label fields of the control dictionary entries pointed to by the ESD LD and the control dictionary LD. If the name fields do not match, an error condition exists. If a match is found, the routine compares the phase number of the matching control dictionary entry with the current phase number. If these numbers do not match, the routine branches to ELBNCD to put the ESD LD into the control dictionary, and update the linkage table. This condition can occur when submodular structure is being used. If the phase numbers match, set a possible (duplicate entry) switch and exit to ESDRET.

ab: Same as aa.

ac: Assembled origins must agree. Compare the control dictionary number (pointer) of the ESD LD with the control dictionary number of the control dictionary item (SD). A no-match condition is an error. A match allows this ESD item to be ignored by causing the routine to branch to ESDRET.

ad: This condition is not possible because a private code has a blank name (label) field.

ae: This is an error-condition branch to ERR46 for error handling.

af: Assembled origins must agree. Force the ESD LD to become an LR type. Branch to location ELBINT to insert the ESD LD as an LR in the control dictionary (overlaying the old control dictionary entry). Update the linkage table (Chart KE), and branch to ESDRET.

ag: Update the control dictionary number and control dictionary address for a new entry. Insert the ESD LD in the control dictionary, update the linkage table, and branch to ESDRET (Chart KE).

ba: If the control dictionary item has been resolved (assigned to a previously processed SD), branch to ERR43 for error handling. If the ESID number of the LD/LR control dictionary entry matches the ESID number of the item being processed (ESD-SD) and the assembled origins also match, the routine branches to location ELBINT (Chart KE). At that location the control dictionary item (LD/LR) is replaced (overlaid) with the ESD-SD item. The routine updates the linkage table and exits via ESDRET. When the ESID numbers and assembled origins match, this control section has been previously defined as an entry in the assembly. Any condition other than matching ESID numbers and matching assembled origins causes a branch to ERR43 for error handling.

bb: Same as ba

bc: Check the phase number of this SD item. If it has been processed in this phase, ignore it. If it has not been processed, determine if it is in the root phase. If it is not in the root phase, move the ESD item (SD) into the control dictionary, replacing (overlaying) the control dictionary entry (an SD). Update the linkage table and exit via ESDRET (Chart KE). If the ESD item is in the root phase, set the control dictionary number equal to -1 (a switch in the linkage table that means: bypass all future references to this SD). Update the linkage table, and exit via ESDRET.

bd: This condition is not possible because the PC has a blank name (label) field.

be: This is an error condition. Branch to ERR46 for error handling.

bf: Branch to ELBINT to overlay (replace) the ER type control dictionary entry with the ESD-SD item. Update the linkage table and exit via ESDRET (Chart KE).

bg: Branch to ELBNCD to update the control dictionary number and the control dictionary address. Move the ESD-SD item into the control dictionary, update the linkage table, and exit via ESDRET (Chart KE).

ca: Not possible because LD entry in the control dictionary would have a blank name field.

cb: Not possible because LR entry in the control dictionary would have a blank name field.

cc: Not possible because SD entry in the control dictionary would have a blank name field.

cd: Branch to ELBNCD to update the control dictionary address. Move the ESD-PC into the control dictionary, update the linkage table, and exit via ESDRET (Chart KE).

ce: Ignored.

cf: Not possible because ER entry in the control dictionary would have a blank name field.

cg: Same as cd.

da: This is an error. Branch to ERR46 for error handling.

db: Same as da.

dc: Same as da.

dd: Bypass the control dictionary entry (PC) and continue to scan. If the scan ends without another match, process as described in dg.

de: Determine common with the longest length. Keep the longest-length value in the control dictionary. Update the linkage table and return via ESDRET (Chart KE).

df: Branch to ELBINT to replace (overlay) the ER type control dictionary entry with the ESD-Common. Update the linkage table, and exit via ESDRET (Chart KE).

dg: Branch to ELBNCD to update the control dictionary number and the control dictionary address. Move the ESD-Common item into the control dictionary, update the linkage table, and exit via ESDRET (Chart KE).

ea: Force the LD control dictionary entry to type LR. Branch to EUPDLT to update the linkage table, and return via ESDRET (Chart KE).

eb: Same as ea.

ec: Branch to EUPDLT to update the linkage table, and return via ESDRET (Chart KE).

ed: Not possible because the ER item cannot have a blank name field.

ee: Branch to EUPDLT to update the linkage table, and return via ESDRET (Chart KE).

ef: Branch to ELBINT to replace (overlay) the ER type control dictionary entry with the ESD-ER item. Update the linkage table, and return via ESDRET (Chart KE).

eg: Branch to ELBNCD to update the control dictionary number and control dictionary address. Move ESD-ER item into the control dictionary, update the linkage table, and exit via ESDRET (Chart KE).

The entire control dictionary is scanned for unresolved LD/LR entries. If any are found, they are tested to determine their status.

1. If the entry cannot be resolved, continue the scan.
2. If the entry is to be bypassed (negative control dictionary number), store the scan.
3. If the entry is resolved, flag it as assigned and continue the scan.

The scan ends when the dictionary is exhausted.

TXT PROCESSOR (TXTRTN) CHART KJ

Objective: Process a TXT card image.

Entry: From GETCD, via the routine that fetched the card processor.

Method: This routine branches to the subroutine at LTESID (Chart JS) to get control dictionary number, control dictionary address, and relocation factor. If the ESID has not been processed, the routine branches to ERR70 for error handling. If this ESID number is to be ignored, the routine branches to RDNEXT to get the next record. If this ESID has been processed, the routine tests for a zero length text card. A nonzero length causes the routine to add the assembled origin and the relocation factor.

If the text address does not fit within the phase boundaries, the routine branches to ERR50 for error handling. If the boundaries are not exceeded or if a zero length text card was found earlier, the routine continues at location BLKTXT.

Starting at location BLKTXT, the routine loads the text buffer address into a register. It compares the length of the incoming text records with the length of the text records in the output area. If the length is the same, the routine branches to location MVTXT.

If the length is not identical, the routine determines if the output area is empty. If it is not empty, it branches to location OPTXT to output the text in the buffer area, thereby emptying the buffer area. It restores control information and reenters this routine at location BLKTXT to repeat the processing sequence. If the output area is empty, the routine moves the length of the incoming record into the control information and branches to MVTXT+4.

Starting at location MVTXT, the routine finds the address of the next available position in the output area. At location MVTXT+4, it determines if the output area has space available. If no space is available, it performs the steps described previously as starting at location OPTXT. If space is available, the routine moves the incoming text record into the output, updates the record number, and branches to read the next record.

RLD PROCESSOR (RLDRTN) CHART KK

Objective: Process RLD cards.

Entry: From GETCD, via the routine that fetched the card processor.

Method: On entry to this routine, program switch RLSW1 is set to the NOP state to allow R and P pointer processing. Program switch RLWRIT is set to the branch state to show RLD's not wanted. When the routine finishes processing each RLD card, it branches to RLWRIT to write out RLD records on SYS002. If end-of-card has not been reached, the routine sets up a pair of registers. One register points to the address of the item to be processed, the other contains the count of bytes processed. When the routine falls through switch RLSW1, it resets it to force processing of the constant. RLSW1 is set to branch when the R and P pointers are the same as those previously processed.

P-pointer processing performs these steps:
1. Branch to LTESID to get the control dictionary number from the linkage table.
2. Branch to ERR70 for error handling if the ESID is not processed.
3. Branch to RLSTP to bypass this item, if the ESID number is negative.
4. Branch to ERR55 if the P-pointer does not point to SD/PC item.
5. Reset program switch RLWRIT to NOP indicating RLD's wanted at this time.
6. Save the control dictionary number in a register.

R-pointer processing performs these steps:
1. Repeat steps 1 and 2 of P-processing.
2. Force a positive control dictionary number from the negative value supplied, indicating the reference is to a nonprocess ESD.
3. Save the control dictionary number in the R-pointer field if the ESID has been processed and it is not to be bypassed.
4. Test the ESD type, and if it is not an ER, scan the RLD card image for the constant. If ER type is found, set a flag in the R-pointer field signifying to Pass 2 use the relocation factor plus the assembled origin as the relocation attribute.

Constants are processed by this routine starting at location RLCONS in this way:
1. Return to scan RLD card image, if there are more items belonging to these R and P pointers.
2. Return to scan RLD card image, if no more items are found, after resetting program switch RLSW1 to NOP (forcing pointer processing).

When the card image scan is finished, the routine branches to RLWRIT. If no RLD's are wanted at this time this program switch is set to branch to RDNEXT. When the switch is in the NOP state, the routine updates the total byte count of RLD's and continues at location BLKRLD. The instructions starting at BLKRLD perform the same functions for RLD records as the instructions at BLKTXT perform for TXT records (see TXT Processor).

END CARD PROCESSOR (ENDRTN) CHART KM

Objective: Process an END card to locate and save a transfer address for a phase.

Entry: From GETCD, via the routine that fetched the card processor.

Method: To initialize for END card processing, the routine first determines if a submodular control card has been found. If it has been found, the routine performs these steps:
1. Set up return position and new position.
2. Reset flags SUBLVL and SUBFLG.
3. Go to END card processing.

If the submodular card has not been found:
1. Go to END card processing when primary input is found.
2. Go to END card processing if the input is not primary (first INCLUDE card with a namelist field), and after the next list has been down-dated.

Actual END card processing begins at location ENDPRC. The routine determines if another transfer address has been accepted. If it has, branch to ENOXFR to bypass this END card. If a transfer address has not been accepted, the END card is examined for transfer information after canceling the previously established transfer address. If the card contains a label, that address becomes the transfer address. Any END card with no ESID number and a blank name field indicates no transfer address available for this card. An END card with an ESID number (relative to the control section in which the end occurs) causes the routine to pick up the related control dictionary number. The routine substitutes the control dictionary number for the ESID number. (Provision is made for handling not-processed and bypassed ESID numbers.)

If a label was present in the END card or if the control dictionary number substitution was successful, the routine:
1. Turns on the transfer switch for phase end.
2. Puts the transfer information into an area (X-area) for later use.

It next scans the control dictionary for an unassigned LD/LR. When found, the routine determines if the control dictionary number is negative. A positive number at this point is an error. When the scan is complete, the routine sets up and executes a loop to destroy the linkage table (built separately for each module).

The routine tests switch DERDSW to determine if control section length is required. If not, the routine tests the end-entry flag. If it is off, the routine exits to GETCD. If it is on, it turns it off and exits to the AUTOLINK processor, Chart JP.

If control section length is required, the routine tests card column 29. If column 29 is non-zero, the length is invalid. If zero, the next possible phase origin becomes the sum of the current possible phase origin and the control section length. Switch DERDSW is set to indicate that the control section length has been processed. The routine exits to either GETCD or to the AUTOLINK processor.

REP PROCESSOR (REPRTN) CHART KN

Objective: Process a REP patch card so that its card image resembles a TXT card. It can then be processed as another TXT card.

Entry: From GETCD, via the routine that fetched the card processor.

Method: After printing the card image, the routine obtains the hexadecimal origin specified by the card. It branches to the HEXRTN subroutine to convert the origin to binary, and stores this information in column 5 of the REP card image. Similarly, the ESID number in column 14 is converted and returned to column 15.

The text portion of the REP card is converted from digits, four at a time. Each set of four digits must be followed by a comma or blank. A comma indicates more text follows, and the routine loops through the conversion sequence. If a blank is found, the routine stores the byte count in column 11 (two bytes for each set of hexadecimal digits). A blank indicates no more text is available on the REP card image. Now, the REP card image is compatible with a TXT card image, and this routine branches to the TXT processor, Chart KJ.

PASS 1-CORELOAD 3

PHASE PROCESSOR (PHSPRO/PHSFIN) CHART LC

Objective: Process a phase card image.

Entry: From the phase scanning routine, Charts LA and LB.

Method: This routine checks phase names and takes the following actions:

1. $ type phase name. Sets a switch for Pass 2.
2. $$A type phase name. (Lowest form in collating sequence.)
   a. This phase name low. Branch to ERR21 for error handling.
   b. This phase name high or equal. Continue processing.

The origin field of the phase card image is processed next. The processing of origin depends on how origin is defined. The definitions and associated processing are:

1. ROOT: (First phase only). Move X'01' into ROOTNO, zero the control dictionary number, and set the root origin to equal the end-of-supervisor address.
2. LABEL: (Cannot be in first phase because there would be nothing to reference). Search the control dictionary for a matching label and take these steps:
   a. SD: Add relocation factor to assembled origin and go to ISDISP.
   b. Assigned LD/LR: Get the SD item pointed to by the ESID and add its relocation factor to the assembled origin. Go to ISDISP.
   c. Phase entry: Test for a qualifier (a pointer used to reference a phase). If not found, go to ISDISP. If found, get the phase number, and if it matches the qualifier, branch to ISDISP.
3. S: Origin is end of supervisor. Go to ISDISP.
4. ASTERISK: Origin is end of previous phase or, if first phase, end of supervisor. Go to ISDISP.
5. BLANK: Go to ISDISP.

At location ISDISP any plus or minus displacement is added to the base just determined. If a minus sum is found, the routine branches to ERR24 for error handling. Otherwise, the newly created phase origin is put on a doubleword boundary, a control dictionary image is built in location SYMBOL, and the routine tests for a first-phase condition. First phase causes a branch to NEWPHS. Phases other than the first cause a branch to TRFRAD, Chart LG, to determine the transfer address.

At location NEWPHS the new phase header is written on tape for use by the following passes, flags are reset, and the routine exits to GETCD.

ENTRY PROCESSOR (ENTRTN) CHART LH

Objective: Supply a transfer address if at this point a new transfer address is desired.

Entry: From GETCD, via the routine that fetched the card processor.

Method: This routine locates the operand field of the ENTRY card image via POSRTN. Control goes to TRFADR to determine the final transfer address. When control is returned to this routine, it searches for a common in the control dictionary. When found, it adds the length of the common to the assembled origin and to the previous address. After all commons have been processed, it updates the end of supervisor address and the linkage editor communications region.

It next searches the control dictionary and performs these steps, depending on the control dictionary entry found:

1. Phase entry. Add length of common to both high and low storage addresses and continue the scan.
2. SD/PC entry. Continue the scan.
3. Other entries. Add length of common to the relocation factor and continue the scan.

When the end of the control dictionary is reached, the routine sets the overriding transfer address to zero and tests the operand field.

1. Blank operand. Transfer address remains a zero. The routine positions the files for Pass 2 and fetches that pass.
2. Nonblank operand. Search the control dictionary for a matching label. If none is found, process as in step 1. If the label is invalid, continue the scan. If the label is valid, add the relocation factor to the assembled origin, make the sum the transfer address, and exit as described in step 1.

PASS 2

LINKAGE EDITOR ($LNKEDTF) CHARTS MA TO MG

Objectives:

1. To relocate addresses and assemble core image blocks from TXT input records.
2. To compress the control dictionary entries to 8 bytes.
3. To print the Linkage Editor MAP if it has been requested.

Entries: Fetched by Linkage Editor Pass 1. The entry point label is START1 on chart MA.

Method: If there are no entries in the control dictionary Pass 2 and Pass 3 are bypassed and Linkage Editor Pass 4 ($LNKEDTL) is fetched. Pass 2 prints the Linkage Editor MAP, if requested, and assembles the core image blocks.

MAP

The MAP is printed on SYSLST. SYSLST may be either a printer or a tape. First a heading line is printed. Then a scan is made from the beginning of the control dictionary, and information is listed from all common type entries. Then control-section and entry-point information for each phase is printed. To finish the MAP, the control dictionary is scanned again and all unreferenced symbols, EXTRN's, are listed. EXTRN's may be followed by any of these messages:

1. ROOT PHASE OVERLAID BY SUCCEEDING PHASE
2. POSSIBLE INVALID ENTRY POINT DUPLICATION IN INPUT
3. INVALID TRANSFER LABEL ON ENTRY OR END STATEMENT IGNORED
4. CONTROL SECTIONS OF ZERO LENGTH IN INPUT

When the MAP is complete, the routine branches to DECSON to compress the control dictionary entries.

Compress Control Dictionary

Each control dictionary entry is reduced from 16 bytes to 8 bytes. This is done by dropping the name field, or first 8 bytes, from the entry. The routine looks at the name field at this time to see if storage protection or Supervisor is specified. If the Linkage Editor output is to be cataloged, the end of supervisor address and the storage-protected problem-program address are changed in the supervisor communications region. This is the last time the name field is needed by Linkage Editor.

Assemble Core Image Blocks

TXT card images are contained in blocked records on SYS001.

The output block number that each logical TXT record will go into is computed. As long as the block number of the TXT record equals the number of the block being built in storage, the TXT record is moved into the block.

When the TXT record block number is higher than the number of the block in storage, the block in storage is written on the output tape. This condition is referred to as a forward origin. The output tape is SYS000 until the first backward origin, and alternates with SYS002 each time a backward origin is detected.

When the TXT record block number is less than the number of the block in storage, the block in storage is written on the output tape. This condition is referred to as a backward origin. The action to be taken is determined by what output tape has the most blocks written on it.

When the first backward origin occurs, the alternate output tape, SYS002, has no blocks written on it. Refer to Figure 45. The current output tape must be backspaced until it is positioned before the first block written on it. Then the output tapes are switched. Blocks are read into storage and copied on the alternate tape until the block in storage equals the block number of the input TXT record. Input TXT records and blocks already written on the output tape are merged and written on the alternate output tape.

When another backward origin occurs, two conditions can exist. Refer to Figure 45. The previous output tape may have more blocks on it than the current output tape. Blocks must be copied from the alternate tape to the current tape until they have the same block count. The action that is then taken is the same as when the current output tape has the most blocks. Refer to Figure 45.

Both tapes must be backspaced to a point where the blocks on both tapes are duplicates. This point is referred to as the minimum, MIN. The output tapes are switched and blocks are copied on the alternate tape from the previous output tape until the block in storage equals the block number of the input TXT record. The action taken is the same from this point as for the first backward origin.

An XFR record in the input from SYS001 indicates the end of a phase. If the transfer address of the phase is not in the XFR record, the relocated phase origin address is used as the transfer address for the phase. If there are more blocks on the alternate tape than on the current tape at this time, they must be copied on the current tape.

All phases are assembled into core image blocks in the same manner. The final output is on SYS000 if the total number of backward origins is even. It is on SYS002 if the total is odd. When a tape mark is encountered on SYS001, all phases are processed and the tapes are positioned for Pass 3.

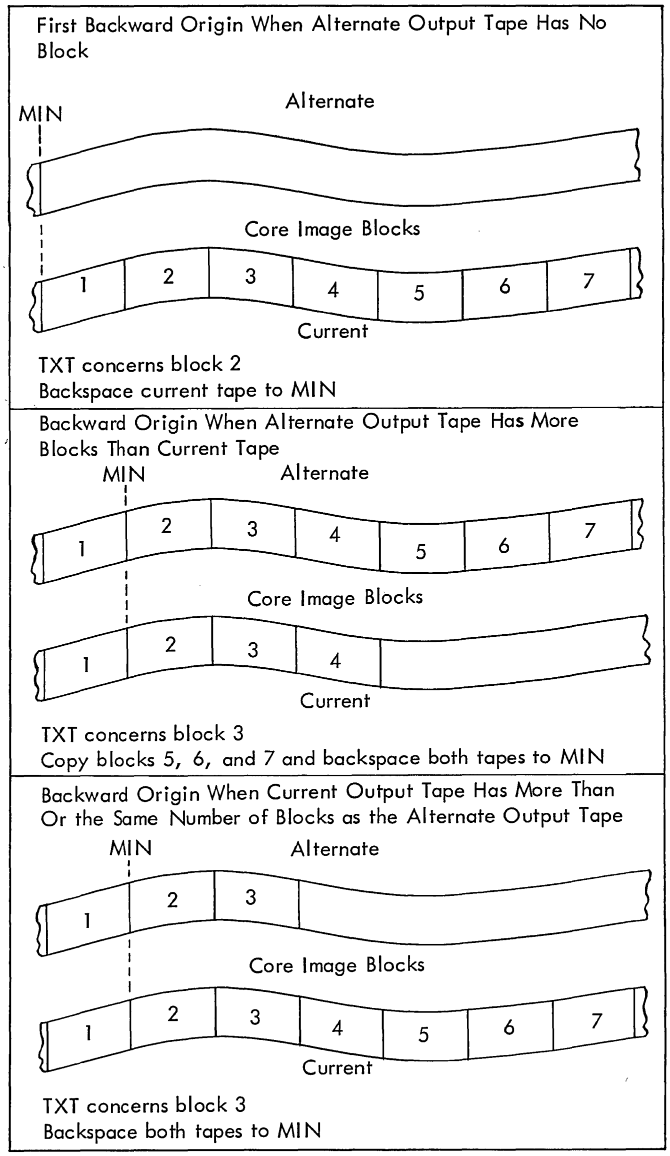


Figure 45. Backward Origin in Linkage Editor Pass 2

__Block Count Labels__

X - number of the output block to which this input TXT record belongs.

Y - number of the block in the output area of storage.

RIPT - number of blocks written on alternate output tape.

ROPT - number of blocks written on current output tape.

MIN - least number of blocks that would not have to be copied to alternate tape when a backward origin occurs.

OPTCT - total number of output blocks for all phases.

PASS 3-CORELOAD 1

INITIALIZATION (INITIAL) CHART NA

Objectives:
1. Save logical unit assignment for text.
2. Open the work and RLD tape files.
3. Print header.

Entry: Fetched by Linkage Editor Pass 2.

Method: After saving the logical unit assignment for TXT card information, this routine positions the RLD and work tapes to the beginning of the first file. If the user has not requested MAP, this routine branches to COMPUTE, Chart NB, to calculate the size and estimated usage of the RLD buffer.

If the user requested MAP, this routine determines the number of phases processed and lines needed for the header, and compares it to the number of print lines remaining for this particular page. If enough print lines remain, a blank line is printed, and SYSLST is checked for end-of-file (EOF). If EOF is reached on SYSLST, Pass 4 is fetched to process the error. If EOF is not reached, this routine branches to COMPUTE, Chart NB.

If there are not enough print lines available to print the header, the date is moved to the communications region. If SYSLST is not a tape, the CCW is set to perform a skip to channel 1, the paper is positioned on the printer, and the header is printed. SYSLST is then checked for EOF. If it is not EOF, the line count is updated by subtracting two from the maximum line count and storing the result in the remaining lines count. A blank line is printed and a branch is made to COMPUTE, Chart NB.

If SYSLST is assigned to a tape unit and MAP is requested, the ASA code is set to skip to channel 1 before printing, and the CCW is set for writing on tape. The header is then written on tape and, if EOF is not reached on SYSLST, the line count is updated, a blank line is written on tape, and a branch is made to COMPUTE, Chart NB. If EOJ is reached on SYSLST, Pass 4 is fetched to process the error.

COMPUTE BUFFER SIZE (COMPUTE) CHART NB

Objectives:
1. Compute size of RLD input buffer.
2. Determine amount of usage for RLD buffer.

Entry: From the Initialization routine for Pass 3, Coreload 1 (INITIAL), Chart NA.

Method: This routine uses the greater of either the true end address of this RLD processor phase or the end address of Pass 3-Coreload 2, as the starting address of the RLD input buffer for this pass.

To compute the end address of the RLD buffer and to determine the amount of usage for this buffer, several computations are performed.

TXT Buffer Computation

This routine subtracts the size of the area containing the RLD input records from the address of the last compressed control dictionary entry, and compares the result to the address of the last compressed control dictionary entry minus the text block size. If the first calculation is greater than the second calculation, the resulting address of the second calculation is used as the address of the TXT buffer. If the first calculation is less than or equal to the second calculation, the resulting address of the first calculation is used as the address of the TXT buffer.

Estimated RLD Buffer Usage Computation

This routine multiplies the number of bytes of RLD information processed by Pass 1 by 1.5 to determine the maximum increase for the RLD items. The result is the estimated RLD usage for this buffer. This routine then gets the number of phases processed by Pass 1, multiplies by 32 (the size of the phase header in the RLD buffer), and adds the result to the estimated RLD usage just calculated. The result is the estimated usage for the entire RLD buffer.

Estimated End of RLD Buffer Computation

This routine subtracts the address of the RLD buffer from the address of the text buffer to get the size of the RLD output. This information is stored in the CCW for the work tape. The number of bytes of RLD information processed by Pass 1 is then added to the address of the RLD input buffer to get the estimated end address for the RLD buffer.

If the RLD buffer is not large enough to hold all the RLD's for every phase processed in Pass 1, the write-out switch is set to indicate only the RLD's for one phase are to be read in and processed at one time. If the buffer is large enough, all the RLD's are read in and processed.

The RLD and work tape CCW's are then initialized for read and write, respectively, and a branch is made to RLDRD to read the RLD records from tape.

READ RLD TAPE (RLDRD) CHART NC

Objectives: Read the records from the RLD tape.

Entry: From the Compute Buffer Size routine (COMPUTE), Chart NB.

Method: This routine reads a record from the RLD tape. If there are no more RLD's to be read, this routine branches to the end-of-RLD's routine (ENDPH), Chart NN. If the record read was a phase record, this routine branches to PHRCD to process the record. If it was not a phase record and this is the first time through this routine, Pass 4 is fetched to process the error, indicating the first entry on the RLD tape was not a phase record. If a phase record was not read and this was not the first time through this routine, a branch to RLDFMT to format the RLD's is made.

PROCESS PHASE RECORD (PHRCD) CHART ND

Objectives:
1. Print phase name and transfer address.
2. Write RLD's for phase on SYS001.
3. Find address of phase entry in control dictionary.
4. Compute high address of phase in storage.
5. Build header.

Entry: From the Read RLD Tape routine (RLDRD), Chart NC.

Method: If this is the first time through this routine, a branch is made to PHMOVE to format the phase entry. If this is not the first time through this routine, the code indicating the end of the previous phase is moved to the RLD buffer. Using the PRTHDR subroutine, the phase name and transfer address are then printed.

If the RLD's are to be written on tape on a per-phase basis, they are written out on the work tape (SYS001), and the address of the RLD buffer register is reinitialized. If EOF is reached while writing on SYS001, Pass 4 is fetched to process the error. If EOF is not reached, the first time switch is turned off, and the phase name is moved to the RLD buffer. The size of the common area is added to the phase origin and the new phase origin is moved to the RLD buffer. The address of the phase entry in the control dictionary is obtained with the FNDIDX subroutine.

Next, the high address of the phase being processed is tested to determine if it is the longest phase processed up to this time. If it is, its address is saved for comparison to later phases. The phase-end address is then moved to the RLD buffer, together with the information for the phase entry from the control dictionary. Using the BUILDHDR and UPDATE subroutines, the header containing the phase name and transfer address is built, the RLD buffer register is increased by 4, and a branch to RLDRD is made to read the next record on the RLD tape.

RLD FORMATTING (RLDFMT) CHART NG

Objectives:
1. Compute end of RLD record.
2. Test for valid ESD type.
3. Get relocation factor and assembled origin for RLD.

Entry: From the Read RLD Tape routine (RLDRD), Chart NC.

Method: After obtaining the number of RLD records to be formatted and the number of bytes in each record, this routine sets the pointer to the first byte of the record. It picks up the number of bytes of information in the record and adds to it the address of the first R-pointer (points to the relocation factor of the contents of the load constant) in order to determine the end-of-record.

It saves the R-pointer and P-pointer (points to the relocation factor of the control sections in which the load constant occurs) and determines if this RLD is to be ignored. If this RLD is to be ignored, the pointer is moved to the next flag field that indicates the type of constant. If this entry has the same R- and P-pointer, the pointer is moved to the next flag and a check is made for end-of-record. If this entry does not have the same R- and P-pointer as the last entry, this routine gets the address of the R-pointer for this entry and branches to ENDTST to determine if this is the end of information for this record.

If this RLD is not to be ignored, the routine saves the address of the assembled origin byte and determines the address of the entry in the control dictionary, using the FNDIDX subroutine. It then determines if this entry is a Label Reference (LR) type of ESD. If it is, the routine checks to see if it could be a phase entry. If it could be a phase entry, an error is noted and processing continues. If the LR is not a phase entry, this routine saves the assembled origin address and, using subroutines ESDTST and GETRF, tests for valid ESD types and gets the relocation factor from the ESD and puts in a bucket. The routine then branches to MOVRFAO to move the relocation factor and assembled origin to the RLD buffer.

If the entry was not an LR, a test for valid ESD types is performed using subroutine ESDTST. The relocation factor is obtained using subroutine GETRF. After getting the assembled origin, a branch is made to MOVRFAO to move the relocation factor and assembled origin to the RLD buffer.

MOVE R/F AND A/O TO BUFFER (MOVRFAO) CHART NK

Objectives: Move relocation factor and assembled origin to RLD buffer.

Entry: From the RLD Formatting routine (RLDFMT), Chart NG.

Method: This routine first moves the relocation factor and the assembled origin of the constant to the RLD buffer area. The RLD buffer pointer is increased by eight, and the P-pointer is analyzed. This routine then computes the address of the entry in the control dictionary using subroutine FNDIDX. It determines whether this is either a Section Definition (SD) or Private Code (PC) entry. If it is neither, an error indicating an invalid control entry from P-pointer is noted, and Pass 4 is fetched to process the error. If the entry is either an SD or PC, this routine, using subroutine GETRF, gets the relocation factor from the ESD and stores it in a bucket, and then updates the pointer to the next flag field on the record. A branch is then made to RAO to process this flag, indicating the type of constant to be analyzed.

PROCESS FLAG (RAO) CHART NL

Objectives:
1. Get assembled origin of constant.
2. Compute relocated origin of constant.
3. Move flag and relocated origin of constant to RLD buffer.

Entry: From the Move R/F and A/O routine (MOVRFAO), Chart NK.

Method: Having saved the flag indicating the type of constant, this routine gets the assembled origin of the constant, adds to it its relocation factor, and stores the result as its relocated origin. It then determines if the relocated origin of this constant is greater than the relocated origin of the last constant processed. If it is not greater, a bit is set in the phase entry to indicate the RLD's in this phase are nonsequential.

If the relocated origin of this constant is greater than the last, it is saved for comparison to the next constant to be processed. The flag and relocated origin are then moved to the RLD buffer, and the RLD buffer pointer is increased by 4.

This routine then examines the next field in the record. If this field has the same R- and P-pointer as the last entry, a branch to the beginning of this routine is made to process the flag. If this field does not have the same R- and P-pointer as the last, a branch to ENDTST is performed to check for end-of-record.

TEST FOR END-OF-RECORD (ENDTST) CHART NM

Objectives:
1. Test for end of information for this record.
2. Test for last input record in storage.
3. Get starting address of new record.

Entry: From the RLD Formatting routine (RLDFMT), Chart NG, and the Process Flag routine (RAO), Chart NL.

Method: This routine determines if this is the end of information for this input record. If it is not the end of information, a branch to RPOINT to process the R-pointer is made. If it is the end of information for this record, this routine determines if it is the last RLD input record in main storage at this time. If it is the last record, a branch is made to RLDRD to read the RLD's for another phase, if required. If it is not the last record in storage, this routine gets the beginning address of the record just processed and adds to it the number of bytes in the record to get the starting address of the next record in storage. It then branches to NEWRCD to compute the end address for this new record.

END OF RLD'S (ENDPH) CHART NN

Objectives:
1. Save end address of longest phase in storage.
2. Write RLD's of last phase on tape.
3. Rewind work and RLD tapes.
4. Set copy switch for Pass 4.

Entry: From the Read RLD Tape routine (RLDRD), Chart NC.

Method: This routine first saves the end address of the longest phase in storage and then moves the end-of-phase code to the RLD buffer area. Using subroutine PRTHDR, it prints the phase name and transfer address. It determines if RLD records are to be written on tape. If so, this routine writes the RLD's for the last phase on SYS001 and checks for EOF. It then writes a tapemark on the work tape (SYS001) and rewinds the tape. If the TXT and RLD's are on the same tape, this routine fetches the second coreload for Pass 3. If the TXT and RLD's are not on the same tape, the copy switch for Pass 4 is turned on, the RLD tape (SYS002) is rewound, and the second coreload for Pass 3 is fetched.

If RLD's are not to be written on tape, this routine branches to TMWORK to write a tapemark on SYS001, and continues processing as just described.

SUBROUTINES FOR PASS 3, CORELOAD 1

MOVWRK: Loads the address of the CCB for the work tape, and requests an I/O operation to write on SYS001.

MOVRLD: Loads the address of the CCB for the RLD tape, and requests an I/O operation to read from SYS002.

FNDIDX: Subtracts the root number from the phase number, and multiplies the control dictionary number by 8 to get the displacement. It then loads the 2's complement of the displacement into a register and adds the negative displacement to the end address of the control dictionary to get the address of the control dictionary entry desired.

GETRF (Chart NJ): Moves the relocation factor of the constant to a bucket. It then determines if the sign of the relocation factor is plus. If it is plus, the subroutine branches back to the main routine via LINKRG. If the sign of the relocation factor is negative, the 2's complement of the relocation factor is stored in the bucket, and the subroutine branches back to the main routine via LINKRG.

UPDATE: Increases the RLD buffer pointer by four, and determines if the buffer is full. If it is full, an error indicating that the buffer was full when not expected is noted, and Pass 4 is fetched to process the error. If the buffer was not full, this routine branches back to the main routine via LINKRG.

ESDTST (Chart NH): Determines if this entry is an SD or PC entry. If it is an SD or PC entry, the subroutine branches to the main routine via LINKRG. If it is not an SD or PC entry, the subroutine determines if the ESD number is valid. If the ESD number is invalid, the total number of unresolved address constants is increased by 1, and a branch to main routine is made via LINKRG.

If the ESD number is valid, the subroutine determines if it is a Common (CM) entry. If it is not, the error count is increased by 1, the total number of unresolved address constants is stored, and a branch to the main routine is made via LINKRG. If it is a CM entry, the length of the assembled origin field is zeroed, and a branch to the main routine is made via LINKRG.

BUILDHDR (Chart NF): Moves phase name to print area, and transfer address to output area. It saves registers 2-7 and gets the address of the print and output work areas. The subroutine then gets an 8-byte character from the output area and the address of the print character from the hexadecimal conversion table (TABLE). It converts half of a character at a time and moves it to the print area. After all the characters have been converted and moved to the print area, the subroutine restores register 2-7 and returns to the main routine via LINKRG.

PRTHDR (Chart NE): Determines if MAP has been requested. If MAP has not been requested, the subroutine branches to the main routine via LINKRG. If MAP was requested, it assumes SYSLST is a printer and sets the command code for printer and ASA code accordingly. At this time, the assignment for SYSLST is checked. If SYSLST is a tape, the CCW is set for tape write. The line count is decreased by one. If the line count is zero, it is reset, and a command to write and skip to channel 1 is issued. If SYSLST is a tape, the ASA code is set to skip to channel 1, and the CCW is set for tape write. If the line count is not zero, the line count is stored, the phase name and transfer address are printed, and SYSLST is checked for EOF. If EOF occurs, Pass 4 is fetched to process the error. If EOF does not occur, the subroutine branches to the main routine via LINKRG.

Error Message Routines: If an error occurs during Pass 3, Coreload 1, the proper error code is stored in register 2, and Pass 4 is fetched to process the error.

PASS 3-CORELOAD 2

INITIALIZATION (START) CHARTS PA-PC

Objectives:

1. Change assignments for TXT and work tapes if necessary.
2. Read RLD's from SYS001.
3. Write header for phase entry.
4. Read a text block.
5. Insert address constant in text.

Entry: From Pass 3, Coreload 1

Method: After storing the address of the RLD buffer from Pass 3, Coreload 1, this routine determines if the compressed RLD's are on tape. If the RLD's are on tape, a skip to the first tapemark on SYS001 is made. If the RLD's are not on tape, it is assumed that all the RLD's for the phases processed in Pass 1 could fit into main storage at one time and that Pass 3, Coreload 1, after formatting the RLD's, kept them in compressed format in storage.

This routine next determines if the assignments for the text and work tapes need to be changed. If they do, it changes the work tape assignment for SYS000 to SYS002, and the text tape assignment for SYS002 to SYS000. The routine then positions the two tape files, stores the address of the TXT buffer in the TXT CCW, and stores the size of the TXT block in the work tape CCW. It adds two to the text block size to allow for the phase number and stores the result in the TXT CCW . It sets the RLD buffer register to point to the start of the phase, and determines if it is the end-of-phase processing. If it is, the routine branches to ENDPHA to fetch Pass 4.

If it is not the end-of-phase processing, it again determines if the compressed RLD's are on tape. If they are not, the routine branches to NEWPHA to set up the phase header. If the RLD's are on tape, it reads the RLD's from SYS001 and tests for EOF. If EOF is reached, Pass 4 is fetched to process the error. If it is not EOF, the last-block switch is turned off and the phase name, origin, end address, and entry point for the RLD are moved to the header. It then moves to the header the highest address of the longest phase, the size of the text blocks, the number of blocks, and the number of bytes in the last block.

The routine stores the phase number, determines if the RLD's are sequential, and sets the appropriate switch to indicate their status. It writes the header, using the WRTHDR subroutine, and sets the pointer to the first RLD in the buffer. This routine then determines if all the RLD's for one phase have been read. If all the RLD's for one phase have been read, a switch is set to show no more RLD's, and processing continues. If all the RLD's for one phase have not been read, this routine reads a text block and gets the new low address of text in storage.

It determines if the phase number for the text block and the phase number in the header agree. If they do not, this routine fetches Pass 4 to process the error. If they do agree, a check for the last block of the phase is made. If it is not the last block of the phase, the routine gets the new high address of the text in storage by adding the text block size to the text end address, and branches to CONTST to determine if an address constant from a previous text block is to be inserted in this text block.

If it is the last text block of a phase, a switch indicating the last block of the phase is set, and a text is made to determine how many bytes of text are to be written on tape. If there are 61 bytes or less of text to be written, the byte count in the work tape CCW is set to 122, a switch to double text is set, and a branch to COMPNT is made to get the new high address of the text in storage. If there are more than 61 bytes of text to be written, this routine moves the number of bytes in the last block to the CCW for the work tape, and gets the new high address of the text in storage by adding the number of bytes in the last text block for this phase to the text end address plus one. The routine then branches to CONTST to determine if an address constant from a previous text block is to be inserted.

If no address constant from a previous text block needs to be inserted in this text block, this routine branches to RLDTST to match the RLD to the text block in which it belongs. If an address constant is to be inserted into this text block, it is moved into the text block, and a branch to RLDTST is made.

MATCH RLD TO TXT (RLDTST) CHART PD

Objectives:
1. Save relocation factor, assembled origin, and address of the R- and P-pointers for this RLD.
2. Save relocated origin of RLD.
3. Determine if RLD is within current phase and within text block.

Entry: From Initialization routine for Pass 3, Coreload 2 (START), Chart PA.

Method: This routine first determines if there are more RLD's to be processed for this phase. If there are no more RLD's to be processed, this routine branches to DOBBLK to determine if the text block is to be doubled. If there are more RLD's to be processed, the relocation factor, assembled origin, and address of the R- and P-pointers for this RLD are saved, the RLD buffer register is updated by 8 to point to the relocated origin of the RLD, and a test is made to determine if this RLD is to be ignored.

If this RLD is to be ignored, a branch is made to UPDATE to point to the next RLD. If this RLD is not to be ignored, the address of the relocated origin is saved, and a test is made to determine if the RLD is in the current phase. If the RLD is not in current phase, one is added to the count for RLD items not within the phase, an indicator is set to ignore the RLD in error, and the routine branches to UPDATE to get the next RLD.

If the RLD is in the current phase, this routine determines if the relocated origin is less than the starting address of this text block. If it is, a branch to UPDATE is made to point to the next RLD. If the relocated origin is higher than the starting address of this text block, the routine determines if it is greater than the end address of this text block. If it is not, a branch to SUBSTI is made to relocate the constant. If it is greater than the end address of this text block, this routine determines if the RLD's are sequential. If they are not, a branch to UPDATE is made to point to the next RLD. If the RLD's are sequential, a branch to TXTWRT is made to write the text block on tape.

RELOCATE CONSTANT (SUBSTI) CHART PE

Objectives:
1. Get relocated address of constant.
2. Determine end address of constant.
3. Place relocated constant in text block.

Entry: From the Match RLD to TXT routine (RLDTST), Chart PD.

Method: This routine computes the relocated address of the constant by first subtracting the starting address of the RLD buffer from the low address of text in storage, and then subtracting the resulting displacement plus two (to eliminate the header) from the relocated origin of the constant.

It then gets the end address of the constant by adding the length of the constant to the address of the constant. It determines if the end address of the constant is in storage. If it is not, a branch to EXTREAD is made to read the first six bytes of the next text block into storage.

If the end address is in storage, this routine loads the relocation factor of the constant into RFBUCK, and determines if this is an ER (External Reference) type constant. If it is not an ER type constant, the relocation factor is added to the assembled origin, the result is stored in the RFBUCK bucket, and the routine branches to TESTSUB. If this was an ER type constant, this routine branches to TESTSUB to determine if the contents of the RFBUCK bucket should be subtracted from the constant. If the contents of RFBUCK is not to be subtracted from the constant, it is added to constant, and the relocated constant is placed in the text block.

If the entire constant cannot fit in the text block, that part of the constant to be inserted into the first six bytes of the next block is saved, and the routine branches to UPDATE. If the entire constant fits into the text block, the routine branches immediately to UPDATE to point to the next RLD item.

EXTRA READ (EXTREAD) CHART PF

Objective:
1. Read first six bytes of next text block.
2. Move remaining bytes of constant from previous text block to first bytes of next text block.

Entry: From the Relocate Constant routine (SUBSTI), Chart PE.

Method: This routine determines if a constant is to be placed in the first bytes of the next text block to be read into storage. If there is no need to read the next text block at this time, the routine branches to PULLOUT to insert the constant into a register (TEMPRG).

If the constant needs to be placed in the next text block, this routine reads six bytes of the next text block into storage, turns on the constant-saved switch, and tests for EOF. If EOF is reached on the text tape, Pass 4 is fetched to process the error. If EOF is not reached, the record is backspaced, and the constant is moved over the phase number that is contained in the first two bytes of the record. The extra-read CCW is restored to read, and the routine branches to PULLOUT to insert the constant in the text block.

GET NEXT RLD (UPDATE) CHART PG

Objectives:
1. Point to next RLD.
2. Write text blocks for phase on tape.

Entry: From the Match RLD to TXT routine (RLDTST), Chart PD, and the Relocate Constant routine (SUBSTI), Chart PE.

Method: This routine increases the RLD register by four to point to the next RLD, and determines if this RLD has the same relocation factor and assembled origin as the last. If it does, a branch to RIGRLD is made to determine if this RLD is to be ignored. If this RLD has a different relocation factor and assembled origin than the last, this routine determines if this is the end of the phase.

If it is not the end of the phase, a branch to NEWRP is made to save the relocation factor and assembled origin of this RLD. If it is the end of the phase, this routine determines if the RLD's are sequential. If they are sequential, the end-of-RLD's switch is turned on and a branch to DOBBLK is made. If the RLD's are not sequential, a test is made to see if the last text block is to be doubled.

If the last text block is not to be doubled, a branch to TXTWRT is made to write the text blocks on the work tape. If the last text block is to be doubled, the double-text switch is turned off, the last text block is doubled, and the text blocks for the phase are written on tape.

This routine then determines if all the text blocks for the phase have been read. If they have, it branches to GETRLD to read the RLD's for the next phase. If all the blocks for one phase have not been read, the routine determines if all the RLD's have been processed.

If all the RLD's have been processed, a branch to RDTXT is made to read a text block from tape. If all the RLD's have not been processed, the routine determines if the RLD's are sequential. If they are sequential, this routine gets the address of the last R- and P-pointer processed, and branches to RDTXT to read the next text block. If the RLD's are not sequential, this routine sets the pointer to the beginning of the RLD's and branches to RDTXT.

END OF PROCESSING (ENDPHA) CHART PH

Objectives:
1. Backspace text file.
2. Rewind RLD tape.
3. Fetch Pass 4.

Entry: From the Initialization routine for Pass 3, Coreload 2 (START), Chart PA.

Method: After writing a tapemark on the work tape, this routine backspaces the text file and rewinds SYS001, the RLD tape. If it is not necessary to copy the work tape onto SYS000, this routine fetches Pass 4 to process any errors that may have occurred during linkage-editing. If it is necessary to copy the work tape onto SYS000, this routine backspaces the file on the work tape, positions SYS000 for copy, and fetches Pass 4 to copy tape and process any errors that occurred during linkage-editing.

SUBROUTINES FOR PASS 3, CORELOAD 2

<u>MOVWRK</u>: Loads the address of the CCB for the work tape, and requests an I/O operation to write on tape.

<u>MOVRLD</u>: Loads the address of the CCB for the RLD tape, and requests an I/O operation to read from SYS001.

MOVTXT: Loads the address of the CCB for the TXT tape, and requests an I/O operation to read from tape.

WRTHDR: Loads the address of the header CCW, and requests an I/O operation to write header on work tape.

NOTIN: Adds one to the counter for RLD items outside of phase limits, and sets the first bit in the flag field to ignore the RLD. The subroutine then branches to UPDATE to point to the next RLD.

Error Message Routines: If an error occurs during Pass 3, Coreload 2, the proper error code is stored in register 2, and Pass 4 is fetched to process the error.

PASS 4

LINKAGE EDITOR ($LINKEDTL) CHART QA

Objectives:
1. To copy Linkage Editor output to SYS000 if it was on SYS002 and go to Job Control for EOJ.
2. To cancel Linkage Editor if an abort error occurs.

Entry: Fetched by Pass 3 for normal EOJ. Fetched by Pass 1, Pass 2, or Pass 3 when an abort error occurs.

Method: If register 2 does not indicate that Pass 4 was fetched to cancel the job, Pass 4 insures that the final Linkage Editor output is on SYS000. If a Linkage Editor MAP was requested, messages are printed on SYSLST if there are:
1. Any unresolved RLD's.
2. Any TXT or REP outside the limits of a phase.
3. Any RLD's outside the limits of a phase.

Pass 4 then fetches Job Control for normal end-of-job processing.

If Pass 4 was fetched to cancel Linkage Editor or if end-of-reel is reached on SYS000, the message, LINKAGE EDITOR CANNOT CONTINUE, is printed on SYSLOG. This message is also printed on SYSLST if a MAP has been requested. A supervisor call of 6 is issued and Linkage Editor is canceled.

LIBRARIAN

The Librarian is a series of programs that maintain and service the three libraries: Core Image, Relocatable, and Source Statement, that make up the 16K Tape System Residence (see System Residence and Figure 2).

The Librarian consists of four programs; MAINT, DSERV, RSERV, and SSERV. MAINT catalogs and deletes elements of the libraries or copies the libraries from one unit to another. DSERV displays the names of the elements of each library. RESERV displays and punches modules from the relocatable library. SSERV displays and punches books from the source statement library.

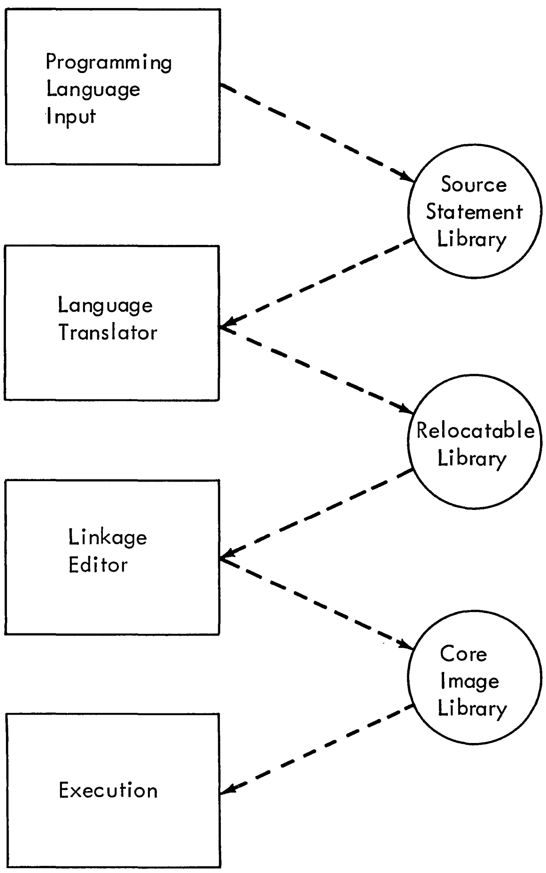


Figure 46. System Flow and System Libraries

LIBRARY FORMAT

Figure 46 shows the relationship of the libraries to the system. Programmer input to the language translators can be stored in the source statement library in elements called books. Output modules from the language translators can be stored in the relocatable library. Phases from the linkage editor can be stored in the core image library.

The elements of each library are arranged by name in collating sequence. After the Supervisor and after each library, directory, and sublibrary there is a tapemark. After the libraries there is a 26-byte trailer record beginning with BOSEOV and followed by a tapemark.

If the relocatable library or the source statement library is an independent tape, it is preceded by header records and a tapemark, and followed by a trailer label identical to the one on SYSRES and a tapemark. The header records consist of:

1. A VOL1 label.
2. A header in the form "HDR PRIVATE LIB--R or S".

SOURCE STATEMENT LIBRARY

Figure 47 shows the format of the source statement library. This library is divided into two sublibraries, Assembler and COBOL. Preceding the sublibraries is a directory. The source statement library is the only library with a directory.

The directory begins and ends with a directory label in the following format.

byte 0	S
bytes 1-2	number of records in directory
bytes 3-10	dummy name (zero's in header; one's in trailer)
byte 11	reserved
byte 12	dummy compression code '8F'
bytes 13-20	directory name DIRS BKS
bytes 21-26	dummy compression code '0F0F0FC00000'
bytes 27-171	not used

Each directory record has the following format.

byte 0	S
bytes 1-2	record number within directory
bytes 3-10	sublibrary name (ASSEMBLY or COBOL)
byte 11	reserved
bytes 12-21	bookname of entry in library; last two bytes reserved
bytes 22-171	up to 15 additional booknames

Each sublibrary begins and ends with a sublibrary label in the following format.

byte 0	S
bytes 1-2	record number within the sublibrary
bytes 3-10	dummy bookname (zero's in header; one's on trailer)
byte 11	reserved
byte 12	dummy compression code '8F'
bytes 13-20	sublibrary name (ASSEMBLY or COBOL)
bytes 21-26	dummy compression code '0F0F0FC00000'

Within the sublibrary, each data record has the following format.

byte 0	S
bytes 1-2	record number within the sublibrary
bytes 3-10	bookname
byte 11	reserved
bytes 12-171	compressed card images

Each card is compressed by deleting the blanks following the nonblank characters. Control bytes within the record indicate deleted blanks. The first four bits of a control byte indicate the number of nonblanks that follow. The second four bits indicate the number of deleted blanks. Each control byte can account for a maximum of 16 nonblanks followed by a maximum of 16 blanks.

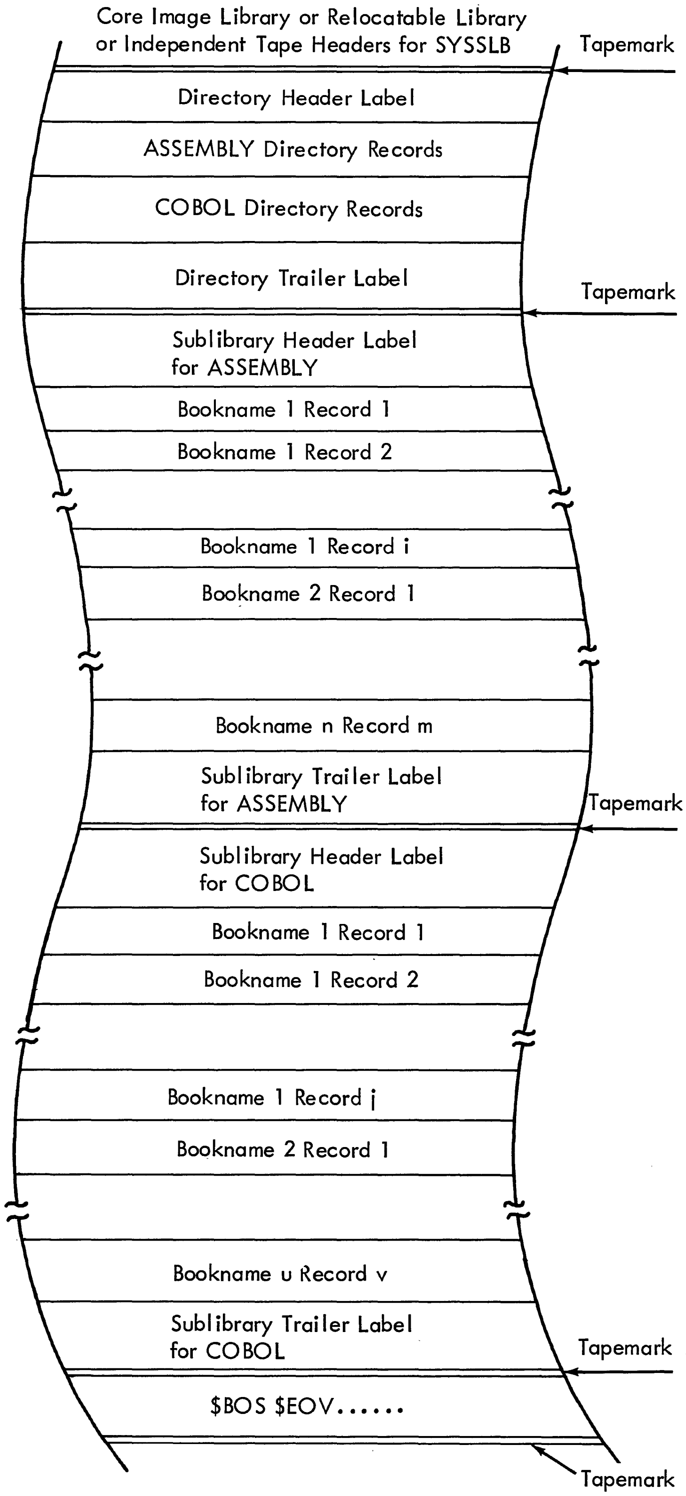


Figure 47. Source Statement Library

RELOCATABLE LIBRARY

Figure 48 shows the format of the relocatable library. This library is made up of module records from the language translators. Each module is preceded by a module directory record.

bytes 0-1	R0
bytes 2-3	number of records between this record and the last tape mark
bytes 4-5	number of records in this module plus one
bytes 6-13	module name

Each module record is preceded by a two-byte count. The first byte gives the number of logical records in the module record. The second byte gives the number of bytes in a logical record.

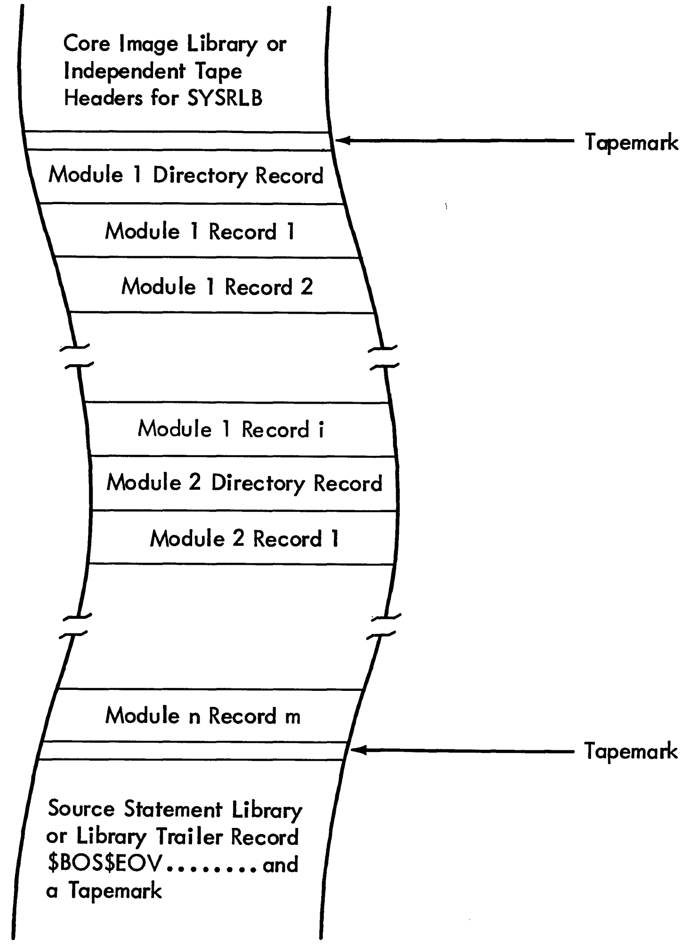


Figure 48. Relocatable Library

CORE IMAGE LIBRARY

Figure 49 shows the format of the core image library. This library is made up of phase blocks from the Linkage Editor. Phases are named in such a way that they are grouped in the library by program. The first four characters of a phase name are identical for each phase in a program. The last four characters identify the phase. System programs are named so that they will appear at the beginning of the library.

1. Initial IPL phases are $$A$IPL1 and 2
2. Supervisor is $$A$SUP
3. Type-A transients are $$AXXXXX
4. Type-B transients are $$BXXXXX
5. Job Control, IPL, and Linkage Editor phase names are prefixed by a $.

Each phase is preceded by 61-byte phase header. The first byte in the header is a C. The next 30 bytes are the header for the following phase. The last 30 bytes are the header for the previous phase.

The first header record contains a zero in byte 31. The last record in the library contains a zero in byte 1. Each phase header is in the following format.

bytes 1-8	phase name
bytes 9-12	starting address
bytes 13-16	ending address
bytes 17-20	transfer address
bytes 21-24	highest address of all phases in a program
bytes 25-26	number of bytes in a phase block
bytes 27-28	number of phase blocks in the phase
bytes 29-30	number of bytes in last record

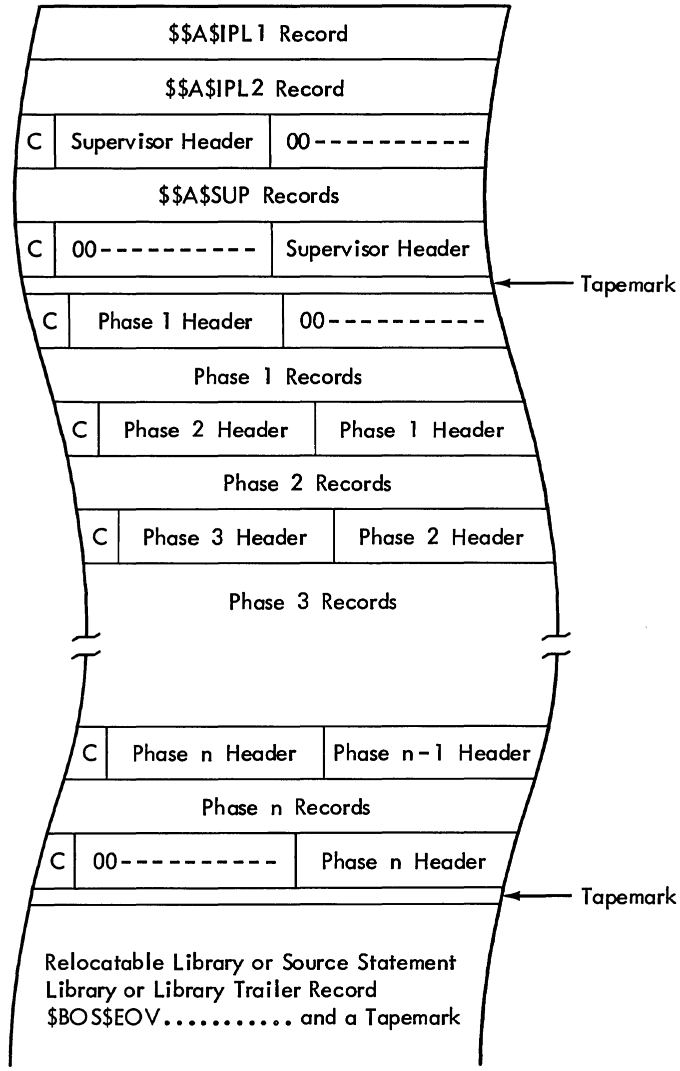


Figure 49. Core Image Library

LIBRARIAN MAINTENANCE - MAINT

MAINT catalogs and deletes elements of the three libraries or copies the libraries from one unit to another. Job Control calls MAINT when it reads a:

1. // EXEC MAINT control card.
2. /& control card after reading // OPTION CATAL control card.

In the latter case, the information that has been linkage-edited onto SYS000 during the job is to be cataloged into the core image library at end-of-job.

In the former case, not only will Linkage Editor output be cataloged on the CATAL option, but also Librarian control cards will be read and processed to copy, catalog to, or delete from the libraries:

- COPY CL or RL or SL.
- CATALR or CATALS.
- DELETC or DELETR or DELETS.

Librarian control cards must be given in the same order as the libraries and their elements are defined.

Figure 51 shows the program flow of MAINT. MAINT has two phase overlays (see Figure 50): MAINTR for relocatable library functions; MAINTS for source statement library functions. The section of MAINT that processes core image library functions is labeled MAINTC. Figure 52 shows the I/O flow of MAINT and of each of the library processors.

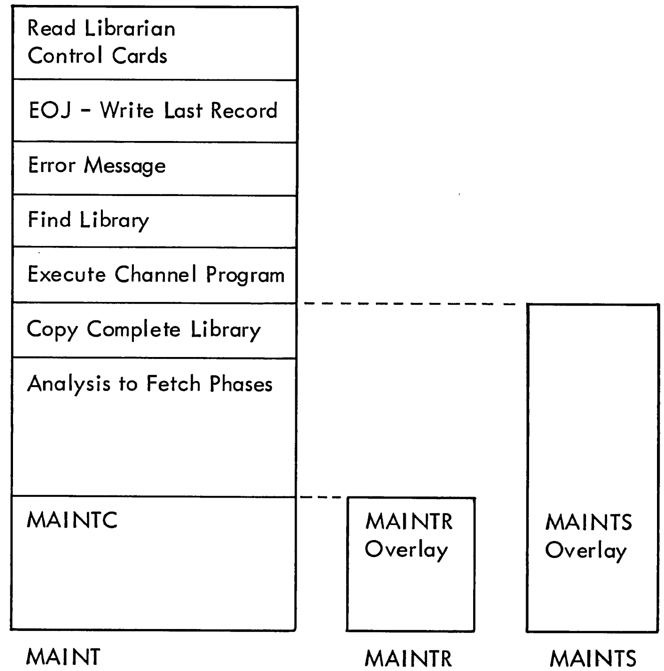


Figure 50. Librarian Maintenance Core Allocation

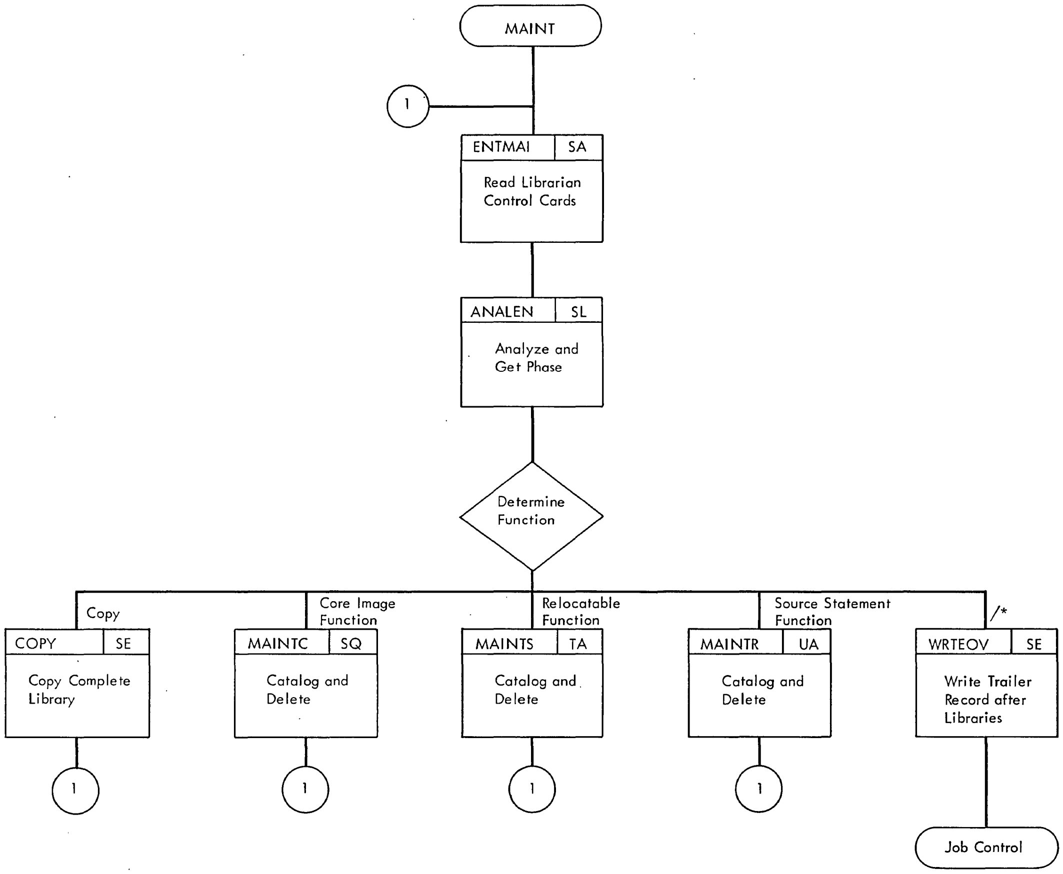


Figure 51. Librarian Maintenance Program Flow

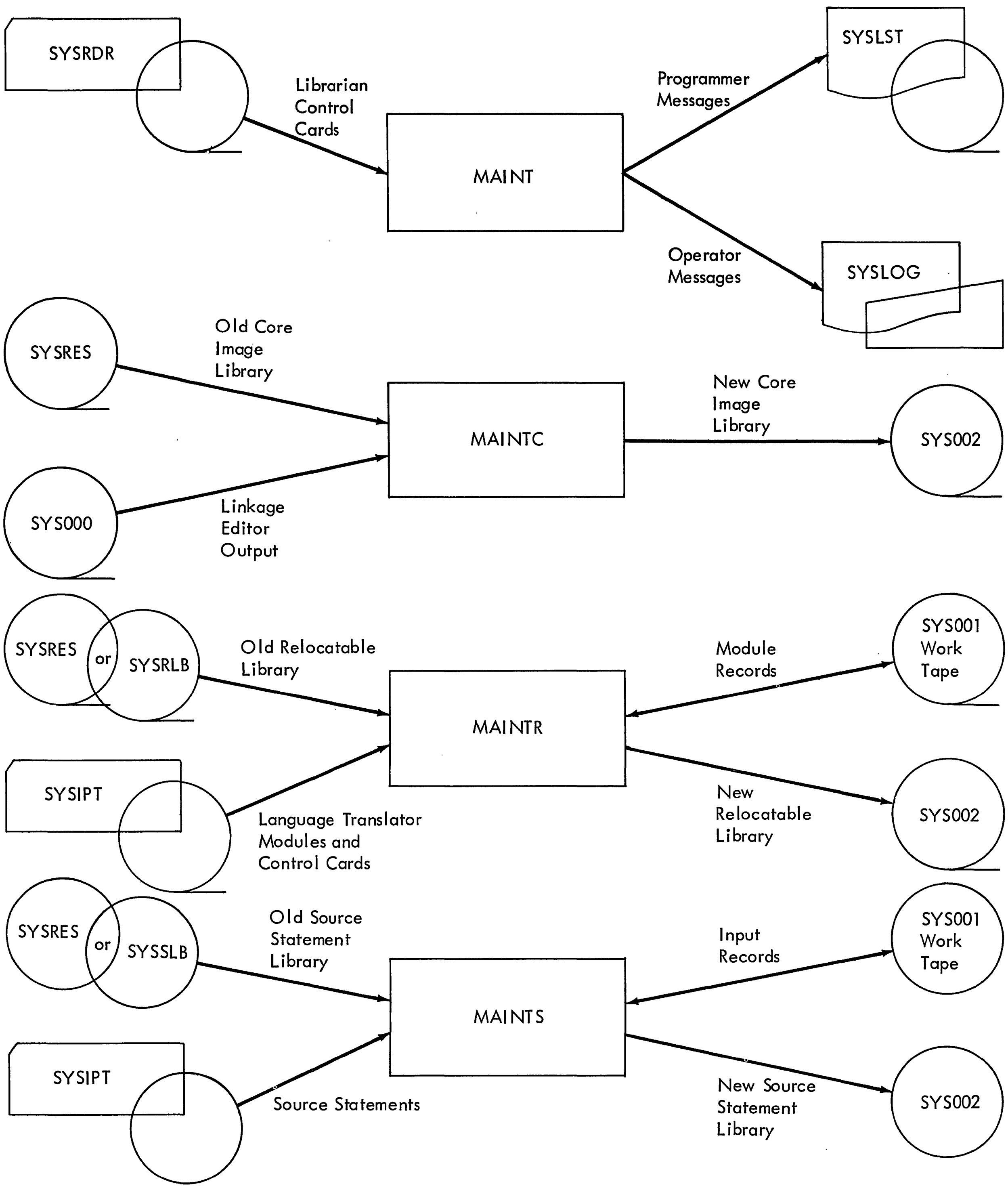


Figure 52. Librarian Maintenance I/O Flow

Byte \ Bit	0 X'80'	1 X'40'	2 X'20'	3 X'10'	4 X'08'	5 X'04'	6 X'02'	7 X'01'
CORBYT						SLCORE	RLCORE	CLCORE
CPYBYT						CPYSL	CPYRL	CPYCL
CRDBYT					NUVBIT	ENDBIT	CELBIT	CATBIT
EOVBYT						SLEOV	RLEOV	CLEOV
MNTBYT			SKIPI		PRVTPE		DELCL	
TESBYT	FRSTCD							
TAPBYT						SLBMSK	RLBMSK	

Figure 53. MAINT Switches

Supervisor Communications Region Switches

MAINT uses the communications region to determine if there is any Linkage Editor output which must be cataloged. Bits two and three of the linkage-control byte (JBCSW1), displacement 57, contain this information. Bit three informs MAINT that any Linkage Editor output must be cataloged.

MAINTC then checks bit two to see if there was any Linkage Editor output. Bit three is turned on if a //OPTION CATAL card is read. This bit is turned off if:

1. MAINT is executed successfully.
2. The abnormal end-of-job cancel routine is fetched.

MAINT Switches

Figure 53 shows the switch bytes initialized by MAINT.

CORBYT: Used by the Analysis to Fetch Phases routine. The flags are turned on as the phases are loaded into storage.

CPYBYT: Used by the Read Librarian Control Cards routine. The flags are turned on according to the operands on the COPY control card. In the absence of a COPY control card and a NEWVOL control card, all three flags are turned on. The Analysis to Fetch Phases routine then determines whether or not a library must be copied.

CRDBYT: Used by the Read Librarian Control Cards routine. The flags are turned on depending on the operation code: NUVBIT for a NEWVOL card, ENDBIT for /*, DELBIT for a DELET card, and CATBIT for a CATAL card.

EOVBYT: Used by the Analysis to Fetch Phases routine. The flags are turned on as the end of each library is reached.

MNTBYT: DELCL is turned on by the Analysis to Fetch Phases routine if a DELETC card is read. PRVTPE is turned on by the Analysis to Fetch Phases routine either if SYSRLB or SYSSLB is assigned, or if a NEWVOL card has been read. SKIPI is used by MAINTS to skip a book and not catalog it to the source statement library.

TESBYT: FRSTCD is used by the Read Librarian Control Cards routine to determine if the card being read is the first card.

TAPBYT: Used by the First-Card routine. RLBMSK is turned on if SYSRLB is assigned. SLBMSK is turned on if SYSSLB is assigned.

READ LIBRARIAN CONTROL CARDS CHARTS SA TO SD

Objective: To read librarian control cards from SYSRDR to SYSIPT, and to analyze the operation code and operands of the control cards.

Entries:
1. From Job Control when a // EXEC MAINT card is read.
2. From MAINTC, MAINTR, and MAINTS to ENTMAI for another control card operand.
3. From MAINTS to:
   - EMAINS on reading a librarian control card.
   - AEND on reading a /* card.

Method: This routine is the first to be executed when the control card // EXEC MAINT is read by Job Control. The librarian control cards are read from SYSRDR or SYSIPT and printed on SYSLST. The operation code is analyzed as:
- /* to indicate the end of librarian control cards.
- CATAL to catalog a library.
- DELET to delete a library.
- NEWVOL to define a new library.
- COPY to copy one or more complete libraries.
- IPTCTRL to read librarian control cards from SYSIPT.
- RDRCTRL to read librarian control cards from SYSRDR.

When a /* card is read the routine branches to the Analysis-to-Fetch-Phases routine at OPREN, or to MAINTS, depending on which routine is in storage.

When a CATAL or a DELET card is read:
1. The catalog or delete switch is set.
2. The operation code suffix is checked to see which library is to be updated.
3. The routine branches to the Analysis-to-Fetch-Phases routine at OPRCL, OPRRL, or OPRSL to load the proper phase.
4. The Analysis-to-Fetch-Phases routine returns to SCANFS to get the first operand.
5. The routine branches to the catalog or the delete entry of MAINTC, MAINTR, or MAINTS to update the library.
6. These three routines return to ENTMAI for the next operand. When all operands of the control card have been processed, another control card is read.

When a NEWVOL card is read:
1. The card is an error if it is not the first librarian control card.
2. The new volume switch is set.
3. The routine reads another control card.

When a COPY card is read:
1. The card is an error if it is not the first librarian control card.
2. The operands are checked to see which libraries are to be copied.
3. The routine reads another control card.

When an IPTCTRL or an RDRCTRL card is read, the symbolic unit name in the DTF for control-card input is altered to SYSIPT or SYSRDR.

The scan subroutines (Chart SD) extract the operation code and operands from the control card.
- INITA1 extracts the operation code.
- FRSTCH extracts the first operand.
- NXTOPR extracts the remaining operands.

EOJ (WRITE LAST RECORD) CHART SE

Objective: To write the end-of-volume label on SYS002 and return to Job Control.

Entry: From the Analysis-to-Fetch-Phases routine or from MAINTS when all maintenance functions are complete.

Method: This routine writes the end of volume label BOSEOV on SYS002 after the requested librarian maintenance functions are completed. A tapemark is written after the EOV label, and tapes used by MAINT are rewound. Control is given to Job Control by an SVC 14.

ERROR MESSAGE SUBROUTINE CHART SF

Objective: To print error messages on SYSLOG and SYSLST.

Method: This routine gets the address and length of the error message to set up the CCW. There are three types of messages:
- Information messages
- Decision messages
- Action messages

If it is an information type, the message is printed on SYSLST and the routine returns to the link register address.

If it is a decision type, the message is printed on SYSLST. If SYSLST and SYSLOG are different devices, the last control card read and the message are printed on SYSLOG. If SYSLOG is assigned as a 1052, the user has an option to ignore the message or cancel the job.

If it is an action type, the message is printed as for a decision type but no control card is written on SYSLOG.

FIND LIBRARY SUBROUTINE CHART SG

Objective: To find the beginning of the library to be processed and, if necessary, to open the output tape SYS002.

Method: If a new library is being built on SYS002, the NEWVOL switch is on and this routine does not locate the old library. Control skips immediately to OPN2SW to open SYS002 if necessary.

To locate the old library, this routine determines if it is on SYSRES or a private tape, SYSRLB or SYSSLB. If on SYSRES and the library to be found is the core image library, SYSRES is rewound. If on a private tape, the private tape is opened.

A search is made on SYSRES or the private tape for the library. The byte LIBNMG contains an indicator for the library to be found (C or R or S). The byte RESLBP contains an indicator for the last library found by SKIPTO (0 for no library found, C or R or S, $ for the end of the library tape found).

The routine tests whether the last library found is the right one. If not, it skips to the next tapemark and reads the next header. When the right library is found, the routine positions the tape back to the header. For SYSRES a count is kept of the number of tape marks preceding the library. The count of the number of records read in the library is set to zero.

Finally, at OPN2SW, SYS002 is opened if necessary. Control returns to the Analysis-to-Fetch-Phases routine.

EXECUTE CHANNEL PROGRAM SUBROUTINE CHARTS SH AND SJ

Objective: To reposition SYSRES if it has moved and to execute the requested channel program.

Method: If the entry point is EXCP, set a switch to wait after executing the channel program. If the logical unit determined by the Find Library routine is SYSRLB or SYSSLB, execute the channel program.

If the logical unit is SYSRES, test to see if the SYSRES tape has been moved by a fetch. If SYSRES has moved, reposition the tape using the tapemark count and record count. Then execute the channel program. Wait if the switch has been set. Reset the switch and return to the calling routine.

The tapemark count and record mark count must be updated after each execution of this routine.

COPY COMPLETE LIBRARY CHART SK

Objectives: To copy complete relocatable library from SYSRES or SYSRLB to SYS002, or to copy complete source statement library from SYSRES or SYSSLB to SYS002.

Entries: From the Analysis-to-Fetch-Phases routine to CPYARL to copy relocatable library, or to CPYASL to copy source statement library.

Method: The routine uses SKIPTO to find the start of the library to be copied. The complete library is copied to SYS002. When the end of the library is reached, the routine returns to ANALEN in the Analysis-to-Fetch-Phases routine.

ANALYSIS TO FETCH PHASES CHARTS SL TO SP

Objective: To determine if MAINTR or MAINTS has been fetched, and to fetch MAINTR or MAINTS if necessary.

Entries:
1. From the Read-Librarian-Control-Cards routine to:
   - OPRCL if the operation code has a C-suffix.
   - OPRRL if the operation code has an R-suffix.
   - OPRSL if the operation code has an S-suffix.
   - OPREN if a /* card is read and MAINTS is not in storage.
   - OPREN if MAINT was fetched by a /& Job Control card.
2. From MAINTC, MAINTR, and Copy-Complete-Library routines to ANALEN when library updating is complete.

Method: When more than one library is being updated in a single librarian run, the libraries must be modified in sequence:
1. Core image library
2. Relocatable library
3. Source statement library.

The relocatable library and the source statement library cannot be on the same private tape. SYSRLB and SYSSLB cannot be assigned the same tape unit.

OPRCL

If the control card concerns the core image library:
1. Check that MAINTR or MAINTS has not overlaid MAINTC.
2. Branch to SCANFS for the first operand.

OPRRL

If the control card concerns the relocatable library:
1. Set ANALEN switch to branch to OPRRL.
2. Check that MAINTS is not in storage.
3. Finish processing core image library.
4. Fetch MAINTR if it is not in storage.
5. Branch to SCANFS for the first operand.

OPRSL

If the control card concerns the source statement library:
1. Set ANALEN switch to branch to OPRSL.
2. Finish processing other libraries.
3. Fetch MAINTS if it is not in storage.
4. Branch to SCANFS for the first operand.

OPREN

If a /* card is read:
1. Set ANALEN switch to branch to OPREN.
2. Finish processing all libraries.
3. Branch to EOJ (Write-Last-Record) routine.

The Analysis-to-Fetch-Phases routine has the following exits:

- NEWRD to read next control card when a control card in error is being ignored.
- SCANFS to extract the first operand.
- MAINTC to copy the core image library.
- CPYARL to copy the relocatable library.
- CPYASL to copy the source statement library.
- MAINTC+8 to finish updating core image library.
- MAINTR+8 to finish updating relocatable library.
- WRTEOV to write EOV label and return to Job Control.
- OPN2SW to open SYS002.
- SKIPTO to find the beginning of a library.

MAINTC CHARTS SQ TO SS

Objectives:
1. To catalog phases in the core image library.
2. To delete phases or programs from the core image library.
3. To copy the core image library.

Entries:
1. From the Read-Librarian-Control Cards routine to:
   - MAINTC for cataloging Linkage Editor output.
   - MAINTC+4 on a DELETC statement.
2. From the Analysis-to-Fetch-Phases routine to MAINTC+8 to finish copying the core image library before processing another library.

Method: The first time MAINTC is entered, IPL and Supervisor programs are either cataloged from Linkage Editor output or are copied on the output tape SYS002. Only IPL and Supervisor cannot be deleted by a DELETC card. A tapemark is written after Supervisor.

Copy

If both the catalog bit in the communications region and the delete switch are off, the complete core image library will be copied to the output tape SYS002. Then the core image library trailer label and a tapemark are written, and the routine branches to the ANALEN entry point of the Analysis-to-Fetch-Phases routine.

Catalog

If there is output from the Linkage Editor to be cataloged, it is merged with the core image library on SYSRES in collating sequence. The output is on SYS002. If the delete switch is on, phases or programs specified by the operands of DELETC cards will not be copied from SYSRES to the output tape. If the phase to be cataloged has the same name as a phase that is already in the core image library, the new phase will be included on SYS002 and the old one will be deleted.

Delete

If there is no output from the Linkage Editor and the delete switch is on, phases or programs specified by the operands of DELETC cards will be deleted. All other phases and programs of the core image library will be copied from SYSRES to SYS002.

MAINTC Exits:

- ANALEN at the end of the core image library to process the other libraries.
- ENTMAI to get another control card operand.

MAINTR CHARTS TA TO TG

Objectives:
1. To catalog modules in the relocatable library.
2. To delete modules from the relocatable library.
3. To delete the entire relocatable library.

Entries:
1. From the Read-Librarian-Control-Cards routine to:
   - MAINTR on a CATALR statement.
   - MAINTR+4 on a DELETR statement.
2. From the Analysis-to-Fetch-Phases routine to MAINTR+8, to finish copying the relocatable library before processing another library.

Method: The first time MAINTR is entered, SYS001 is opened as a scratch tape. The Find-Library routine is used to locate the beginning of the relocatable library and open the output tape, SYS002.

Catalog

Modules that are specified by the operands of CATALR control cards are merged with the existing relocatable library in collating sequence. The output of this merge is on SYS002. The modules to be cataloged are on SYSIPT.

The cards permitted in a module to be cataloged are:
1. Linkage Editor control statements: PHASE, INCLUDE and ENTRY
2. Symbol (SYM) cards
3. External symbol dictionary (ESD) cards
4. Text (TXT) cards
5. Replace (REP) cards
6. Relocation dictionary (RLD) cards
7. End (END) card.

ESD, TXT, and RLD cards are packed into records that are 162 bytes long. The first two bytes of the record contain the number of logical records and the logical record length. The packed record is written on SYS001.

When the END card is read from SYSIPT:
1. It is written on SYS001.
2. SYS001 is rewound.
3. The module header is written on SYS002.
4. The module is copied from SYS001 to SYS002.
5. The routine returns to the ENTMAI entry of Read-Librarian-Control-Cards routine for another operand.

SYM, REP, INCLUDE, PHASE, and ENTRY statements are written on SYS001. The first and second operands of the PHASE statement are scanned to assure they are present.

Delete

If the operand of the DELETR control card is ALL, the entire relocatable library will be deleted from the output tape SYS002. If the operand has a suffix of .ALL, all consecutive modules that have the same three-character prefix as the DELETR operand will be deleted. If the operand is a module name, the module is deleted.

Copy

To finish copying the relocatable library when all operands of CATALR and DELETR cards have been processed:
1. Copy the remainder of relocatable library to SYS002.
2. Write a tapemark on SYS002 (if there are any records in the relocatable library).
3. Use the Find-Library-Routine to locate the source statement library if necessary.
4. Branch to the ANALEN entry point of the Analysis-to-Fetch-Phases routine.

MAINTR Exits

- ANALEN at the end of the relocatable library to process the source statement library.
- ENTMAI to get another control card operand.
- AEND on a SYSIPT end-of-file or out-of-cards condition to end the MAINT procedure.

MAINTS CHARTS UA TO UJ

Objectives:
1. To catalog books in the source statement library.
2. To delete books from the source statement library.
3. To delete a sublibrary from the source statement library.

Entries:
1. From the Read-Librarian-Control-Cards routine to:
   - MAINTS on a CATALS statement.
   - MAINTS+4 on a DELETS statement.
   - MAINTS+8 on a /* statement to merge input into the source statement library.

Method: The first time MAINTS is entered, SYS001 is opened as a scratch tape. The Find-Library routine is used to locate the beginning of the source statement library and open the output tape, SYS002. The first block of the directory is read from the source statement library.

Catalog - Pass 1

Each book name in the directory block is compared to the operand of the CATALS card until the place is found where the name fits in. A new directory block is being assembled up to this point. If the block becomes full, it is written on the output tape, SYS002.

The book read from SYSIPT is written on the scratch tape, SYS001, in compressed format. Input on SYSIPT may be in compressed or card image format. The book begins with a BKEND or a MACRO card and ends with a BKEND or a MEND card. After the book is written on SYS001, the routine branches to the Read Librarian Control Cards routine for another operand.

Delete - Pass 1

Each book name in the directory block is compared to the operand of the DELETS card until the one to be deleted is found. A new directory block is being assembled up to this point. If the block becomes full, it is written on the output tape, SYS002.

The book name to be deleted is added to a list-of-deletes block. When this block becomes full or if a CATALS card is read, the block is written on SYS001. The first byte is X'00' to identify a list-of-deletes block. The routine branches to the Read-Librarian-Control-Cards routine for another operand.

Copy and Merge - Pass 2

When all operands of CATALS and DELETS cards have been processed, the remainder of the directory is written on SYS002. The scratch tape SYS001 is merged with the existing source statement library in collating sequence. When a list-of-deletes block is read from SYS001, these books will be bypassed on the existing source statement library. When a book to be cataloged has a book name equal to one already in the existing source statement library, the existing one is deleted. The output of this merge is on SYS002.

When end-of-file on both SYS001 and the existing source statement library is reached, the updated library is complete on SYS002, and the routine branches to WRTEOV in the EOJ-Write-Last-Record routine.

Figure 54 summarizes the flow of Passes 1 and 2.

MAINTS Exits

- ENTMAI to get another control card operand.
- WRTEOV to write end-of-volume record on SYS002.

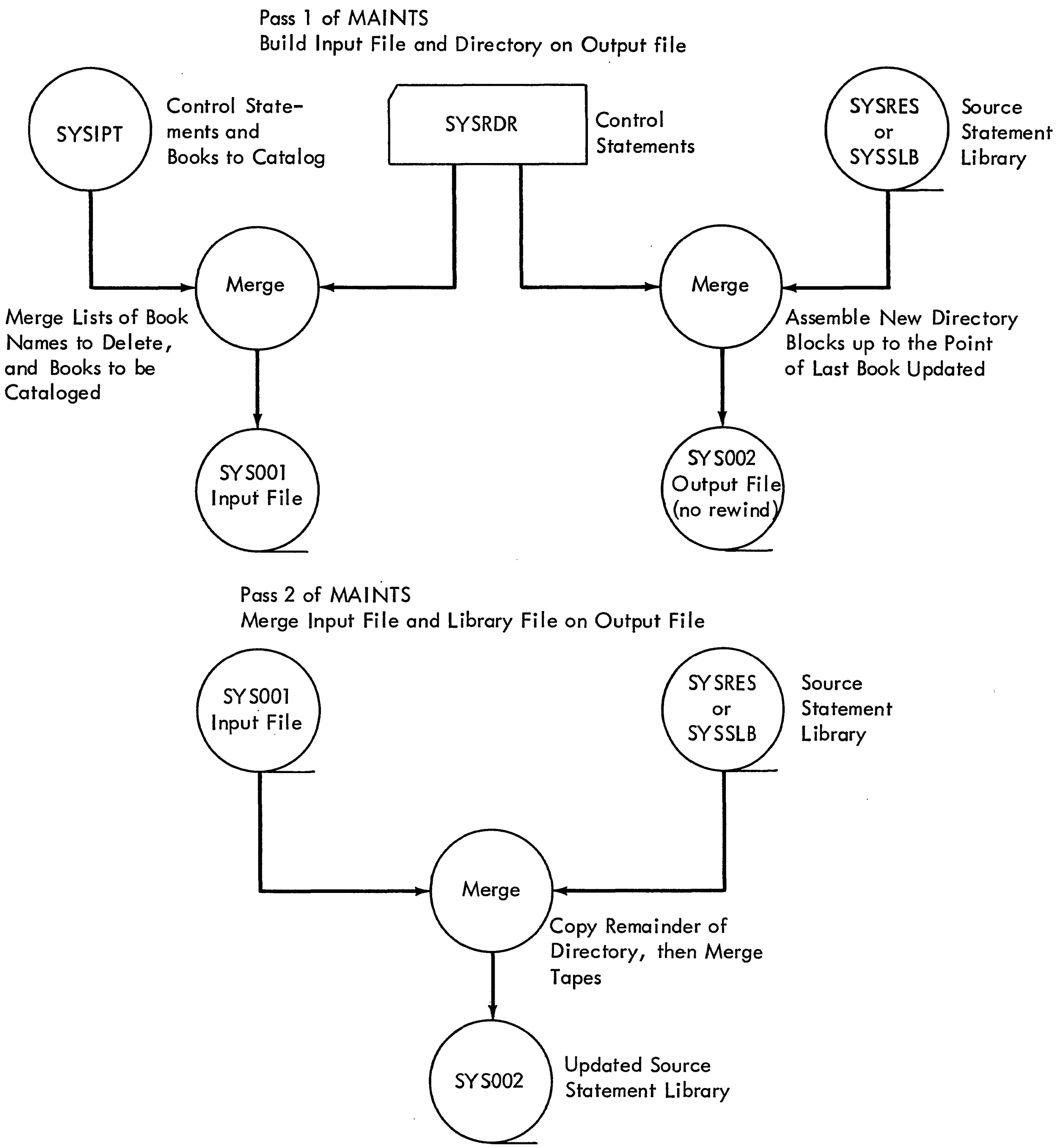


Figure 54. Passes 1 and 2 of MAINTS

LIBRARIAN DIRECTORY SERVICE (DSERV)

DSERV displays the names of the elements of the three libraries. DSERV is composed of one phase and is fetched when a // EXEC DSERV card is read by Job Control.

The directory information for the core image and relocatable libraries is taken from the header records preceding each phase or module. The directory information for the source statement library is contained in a directory preceding the sublibraries of the source statement library.

Figure 55 shows the I/O and program flow of DSERV. DSERV switches are:

1. First card switch, TESBYT
   - X'00' - first card has not been read
   - X'80' - first card has been read
2. Display switch, DSPBYT
   - X'01' - display core image directory
   - X'02' - display relocatable directory
   - X'04' - display source statement directory

READ CONTROL CARDS CHARTS VA AND VB

Objective: To read librarian control cards from SYSRDR, and analyze the operation code and operands of the control cards.

Entry: From Job Control when a // EXEC DSERV card is read.

Method: Librarian control cards are read from SYSRDR. The operation code is analyzed as DSPLY or it is invalid. The operands are analyzed as:

- ALL, display all three directories.
- CD, display core image directory.
- RD, display relocatable directory.
- SD, display source statement directory.

These operands may be in any order or in any combination. They may be entered on more than one consecutive DSPLY card. They may also be repeated and only one listing will be created.

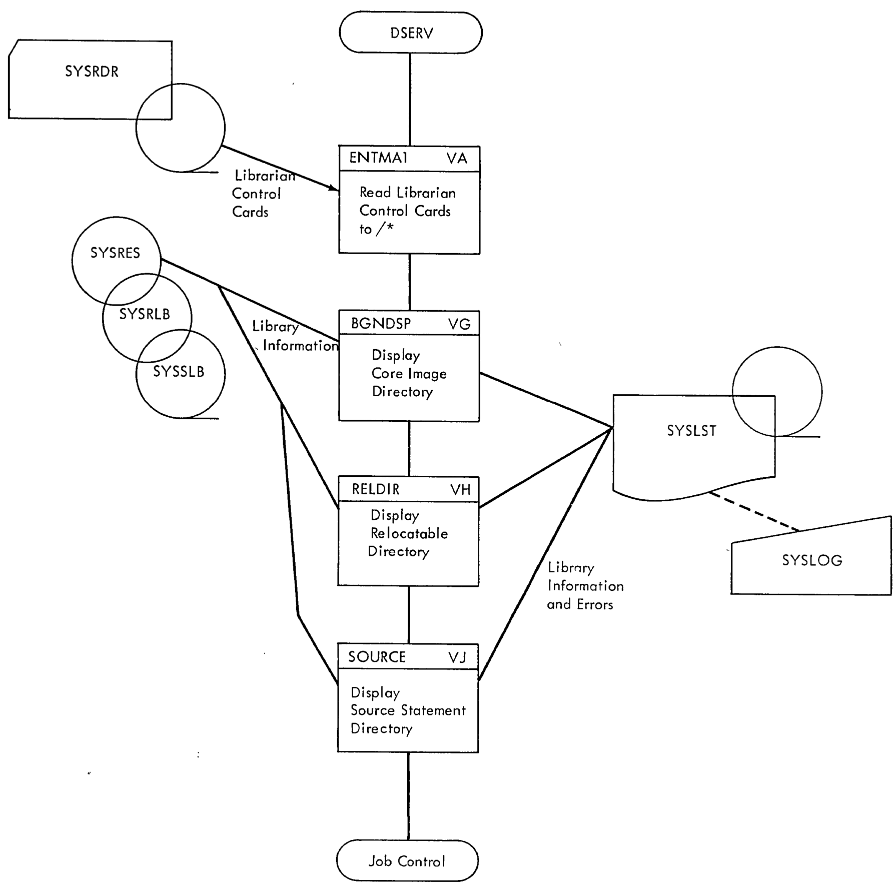


Figure 55. DSERV Program and I/O Flow

After all operands of the DSPLY card are analyzed, the routine reads another control card. If it is a /* end card, the routine exits to BGNDSP in the Display CD routine.

The scan subroutines extract the operation code and operands from the control card.

- INITSHL extracts the operation code.
- FRSTCH extracts the first operand.
- NXTOPR extracts the remaining operands.

ERROR MESSAGE SUBROUTINE CHART VC

Objective: To print error messages on SYSLOG and SYSLST.

Method: This routine writes the last control card (unless /*) on SYSLST. It gets the address and length of the error message to set up the CCW. There are three types of messages:

- Information messages
- Decision messages
- Action messages.

If it is an information type, the message is printed on SYSLST and the routine returns to the link register address.

If it is a decision type, the message is printed on SYSLST. If SYSLST and SYSLOG are different devices, the last control card read and the message are printed on SYSLOG. If SYSLOG is not a 1052 the job is canceled. If SYSLOG is assigned as a 1052, the user has an option to ignore the message or cancel the job.

If it is an action type, the message is printed as for a decision type but no control card is written on SYSLOG.

FIND LIBRARY SUBROUTINE CHART VD

Objective: To find the beginning of the relocatable or source statement library.

Method: The logical unit is determined to be SYSRES, SYSRLB, or SYSSLB. The private tape label is checked if the library is on SYSRLB or SYSSLB.

Every time this routine is executed the library tape is positioned immediately before the first record of the library. If the library is on SYSRES:

1. The number of tapemarks preceding the library on SYSRES is saved.
2. The library record count is initialized to 0.

The current position of SYSRES must always be known in case the SYSRES tape is moved by a fetch.

EXECUTE CHANNEL PROGRAM SUBROUTINE
CHARTS VE AND VF

Objective: To reposition SYSRES, if it has moved, and execute the requested channel program.

Method: If the entry point is EXCP, set a switch to wait after executing the channel program. If the logical unit determined by the Find-Library Routine is SYSRLB or SYSSLB, execute the channel program.

If the logical unit is SYSRES, test to see if the SYSRES tape has been moved by a fetch. If SYSRES has moved, reposition the tape using the tapemark count and record count. Then execute the channel program. Wait if the switch has been set. Reset the switch and return to the calling routine.

The tapemark count and record mark count must be updated after each execution of this routine.

DISPLAY CD CHART VG

Objective: To display the core image library directory on SYSLST.

Entry: From the Read-Control-Cards routine when a /* card has been read.

Method: If the CD operand has been specified, <u>all</u> phase names are listed from the header labels in the core image library. IPL1 and IPL2 do not have header labels, so phase names for them are inserted by the program. Information that is printed for each phase is:

1. Phase name
2. Starting address
3. Ending address
4. Entry point address
5. Highest address used by the program
6. Maximum record size
7. Number of records including last record
8. Size of the last record.

The routine returns to BGNDSP to see if the relocatable or source statement library directory has been requested. The routine exits to:

- RELDIR to display the relocatable directory.
- SOURCE to display source statement directory.
- NDSPLY to return to Job Control.

DISPLAY RD CHART VH

Objective: To display the relocatable library directory on SYSLST.

Entry: From the Display CD routine if the display switch indicates display relocatable library directory.

Method: If the RD operand has been specified, all module names and the number of records in each module will be listed. This information is taken from the from the header labels at the beginning of each module.

The routine returns to BGNDSP in the Display CD routine to display the source statement library directory if it has been requested.

DISPLAY SD CHART VJ

Objective: To display the source statement library directory on SYSLST.

Entry: From the Display CD routine if the display switch indicates display source statement library directory.

Method: If the SD operand has been specified, all book names will be listed. The sublibrary name will be printed only at the beginning of a sublibrary or the beginning of a new page.

The source statement directory precedes the sublibraries in the source statement library. The first record of the directory is a header label. The records that follow contain the sublibrary name, as the first entry, and 16 book names. The last record is a trailer label.

When the source statement directory trailer label is read, the routine exits at NDSPLY to Job Control.

RELOCATABLE LIBRARY SERVICE (RSERV)

RSERV displays on SYSLST or punches on SYSPCH or displays and punches modules from the relocatable library. RSERV is composed of one phase and is fetched when a // EXEC RSERV card is read by Job Control.

Figure 56 shows the I/O and program flow of RSERV.

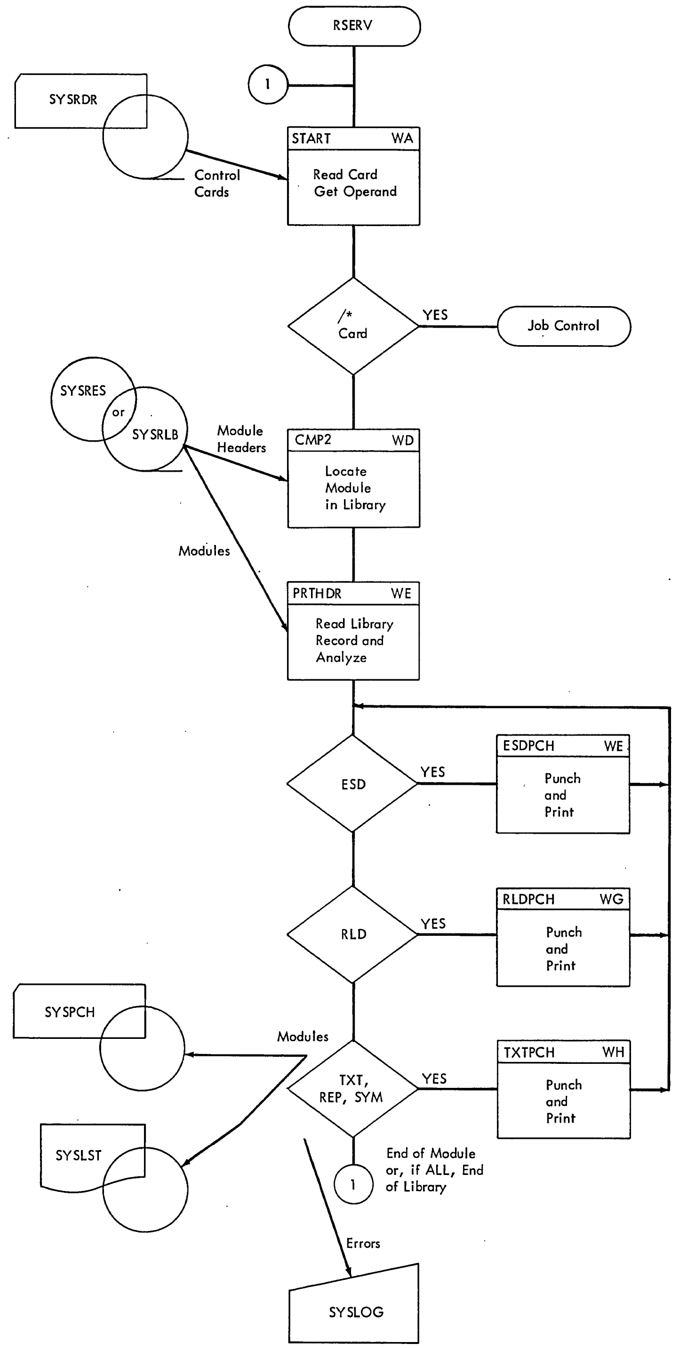


Figure 56. RSERV Program and I/O Flow

READ CONTROL CARDS CHARTS WA TO WC

Objective: To read librarian control cards from SYSRDR, and analyze the operation code and operands of the control cards.

Entries:
1. From Job Control when a // EXEC RSERV card is read.
2. From the analyze library record routine to:
   - AINITT to read another card when the entire library has been serviced for an ALL operand.
   - IFDOT to check on new operand for .ALL
   - CPLSOP to determine if all operands have been processed.
3. From the locate-module routine to analyze a new operand when the previous operand module cannot be located.

Method: After locating the beginning of the relocatable library, a control card is read from SYSRDR. The operation code is analyzed as PUNCH, DSPLY, or DSPCH. When a /* control card is read, the punch is closed, if it was used, and the routine returns to Job Control.

The operands of the control card are analyzed as:
- ALL - perform the requested service on the entire relocatable library. ALL must be the first operand.
- (First three characters of program name) .ALL - perform the requested service on all consecutive modules with the same 3-character prefix.
- (Module name) - perform the requested service on the individual module specified.

The scan subroutine GETFLD extracts the operation code and the operands from the control card. If the first operand is ALL, the routine sets a switch and branches directly to DRASFN in the Locate-Module-Header routine. If the first operand is not ALL, the operands are moved to a save area with an * marking the end of the operand list.

At IFDOT a switch is set if the operand has .ALL suffix. At CPLSOP a check is made to determine if any operands remain to be processed. Exit is to CMP2 unless the library tape must be backspaced to the beginning of the library at DRASFN.

LOCATE MODULE HEADER CHART WD

Objective: To search the relocatable library for the requested module header.

Entry: From the Analyze-Control-Cards routine to CMP2 or, for access to the beginning of the library, to DRASFN.

Method: If the ALL operand was specified, the entry point is DRASFN. SYSRES or SYSRLB is backspaced to the beginning of the relocatable library. The first relocatable library header is read, and the routine branches to PRTHDR.

For other than the ALL operand the point is CMP2. If an operand with a suffix of .ALL has been specified, only the first 3 characters of the module name from the relocatable library header are compared to the prefix of the operand. If the .ALL suffix was not specified, the first 8 characters of the module name are compared to the operand.

Modules are bypassed using the EXCP routine until there is an equal compare. If the module is not found, the operand pointer is updated and control returns to CPLSOP in the Read-Control-Card routine. If the module is found, control goes to PRTHDR in the Analyze-Library-Record routine.

ANALYZE LIBRARY RECORD CHART WE

Objective: To read a record from the relocatable library module and to determine the record type.

Entries:
1. Initially from the Locate-Module routine
2. To TSTPRT from the Process ESD routine after punching has been completed for any of the record types.
3. To PRT1 from each of the record-processing routines to complete printing for the record.

Method: If DSPLY or DSPCH has been requested, the module header is printed on SYSLST. If PUNCH or DSPCH has been requested, a CATALR card is punched on SYSPCH.

For punching, each record in the module is analyzed as a blocked ESD, TXT, or RLD record. The routine branches to the proper routine for handling these records. Other records are punched without special handling.

For display, each record in the module is analyzed as a blocked ESD, TXT, or RLD or as a REP or SYM record. This routine branches to the proper routine for handling these records. Other records are displayed without special handling.

When the end of the module is found, if the operand is not ALL or .ALL, the operand pointer is updated and control returns to IFDOT in the Read-Control-Card routine. If the operand is ALL, this routine is reexecuted until the end of the relocatable library is found; then control goes to AINITT to read another control card. If the operand has the suffix .ALL, no new operand is picked up; control returns to CPLSOP in the Read-Control-Card routine.

PROCESS ESD CHART WF

Objective: To punch or display ESD records.

Entries:
1. From the Analyze-Library-Record routine to ESDPCH to punch, or to ESDPRT to display, the ESD record.
2. From the RLD and TXT routines to punch the last card for each record.

Method: ESD records are 162 bytes long. The first 2 bytes are the number of logical records and the length of each logical record within the physical record.

From the ESD record, 48 bytes of information are punched into each card. The last card contains the remaining bytes if there are any (BBC or block byte count is greater than 0). The routine returns to the TSTPRT entry point in the Analyze Library Record.

From the ESD record, each ESD item is displayed on a new print line. After the end of the ESD block is found, the routine returns to the PRT1 entry point in the Analyze-Library-Record routine to print the last ESD item.

PROCESS RLD CHART WG

Objective: To punch or display RLD records.

Entry: From the Analyze-Library-Record routine to RLDPCH to punch, or to RLDPRT to display, the RLD record.

Method: RLD records are 162 bytes long. The first 2 bytes are the number of logical records and the length of each logical record within the physical record.

From the RLD record, 48 bytes of information are punched into each card. When less than 48 bytes remain to be punched, the routine branches to EDCPGP in the Process ESD routine to punch a card with the remaining bytes, or to REINT1 if no bytes remain (BBC or block byte count equal 0).

From the RLD record, each RLD item is displayed on a new print line. After the end of the RLD block is found (BBC equals 0), the routine returns to the PRT1 entry point in the Analyze-Library-Record routine to display the last RLD item.

PROCESS TXT CHART WH

Objective: To punch or display TXT records and to display REP or SYM records.

Entries: From the Analyze-Library-Record routine to:
- TXTPCH to punch TXT
- TXTPRT to display TXT
- REPPRT to display a REP card
- SYMPRT to display a SYM card.

Method: TXT records are 162 bytes long. The first 2 bytes are the number of logical records and the length of each logical record within the physical record. The TXT record from the relocatable library contains two text statements. Each text statement is punched into a card or printed on one line.

When punching TXT statements, this routine punches the first and branches to EDCPGP in the Process ESD routine to punch the second.

When displaying TXT statements, this routine prints the first and branches to PRT1 in the Read-Control-Card routine to print the second.

SYM and REP records contain only one logical record. It is displayed at PRT1 in the Read-Control-Card routine.

ERROR MESSAGE SUBROUTINE CHART WJ

Objective: To print error messages on SYSLOG and SYSLST.

Method: This routine writes the last control card on SYSLST. It gets the address and length of the error message to set up the CCW. There are three types of messages:

- Information messages
- Decision messages
- Action messages.

If it is an information type, the message is printed on SYSLST and the routine returns to the link register address.

If it is a decision type, the message is printed on SYSLST. If SYSLST and SYSLOG are different devices, the last control card read and the message are printed on SYSLOG. If SYSLOG is not a 1052 the job is canceled. If SYSLOG is assigned as a 1052, the user has an option to ignore the message or cancel the job.

If it is an action type, the message is printed as for a decision type but no control card is written on SYSLOG.

FIND LIBRARY SUBROUTINE CHART WK

Objective: To find the beginning of the relocatable library.

Method: The logical unit is determined to be SYSRES or SYSRLB. The private tape label is checked if the library is on SYSRLB.

Every time this routine is executed, the library tape is positioned immediately before the first record of the library. If the library is on SYSRES:

1. The number of tapemarks preceding the library on SYSRES is saved.
2. The library record count is initialized to 0.

The current position of SYSRES must always be known in case the SYSRES tape is moved by a fetch.

EXECUTE CHANNEL PROGRAM SUBROUTINE CHARTS WL AND WM

Objective: To reposition SYSRES if it has moved, and execute the requested channel program.

Method: If the entry point is EXCP, set a switch to wait after executing the channel program. If the logical unit determined by the Find-Library Routine is SYSRLB, execute the channel program.

If the logical unit is SYSRES, test to see if the SYSRES tape has been moved by a fetch. If SYSRES has moved, reposition the tape using the tapemark count and record count. Then execute the channel program. Wait if the switch has been set. Reset the switch and return to the calling routine.

The tapemark count and record mark count must be updated after each execution of this routine.

SOURCE STATEMENT LIBRARY SERVICE (SSERV)

SSERV displays on SYSLST or punches on SYSPCH or displays and punches books from the source statement library. SSERV is composed of one phase and is fetched when a // EXEC SSERV card is read by Job Control.

Figure 57 shows the I/O and program flow of SSERV.

INITIALIZE AND READ CONTROL CARD CHART XA

Objective: To read librarian control cards from SYSRDR and to analyze the operation code and operands of the control cards.

Entries:

1. From Job Control when a // EXEC SSERV card is read.
2. From the End-of-Book routine when a new control card is required.

Method: This routine determines if the source statement library is on SYSSLB or SYSRES. If the library is on SYSSLB, that private tape is opened. If the library is on SYSRES, the supervisor calls and the CCB are reset.

At CALLCS a control card is read and evaluated to determine if the operation code calls for printing, punching, or both. If a /* card is read, SSERV is complete and exit is to Job Control. If the operation is printing, control skips to the next routine, the Fill-Buffer routine.

If the operation includes punching, SYSPCH is opened and the operation field is scanned for the operand CMPRSD. This operand sets a switch to produce punched output in compressed form.

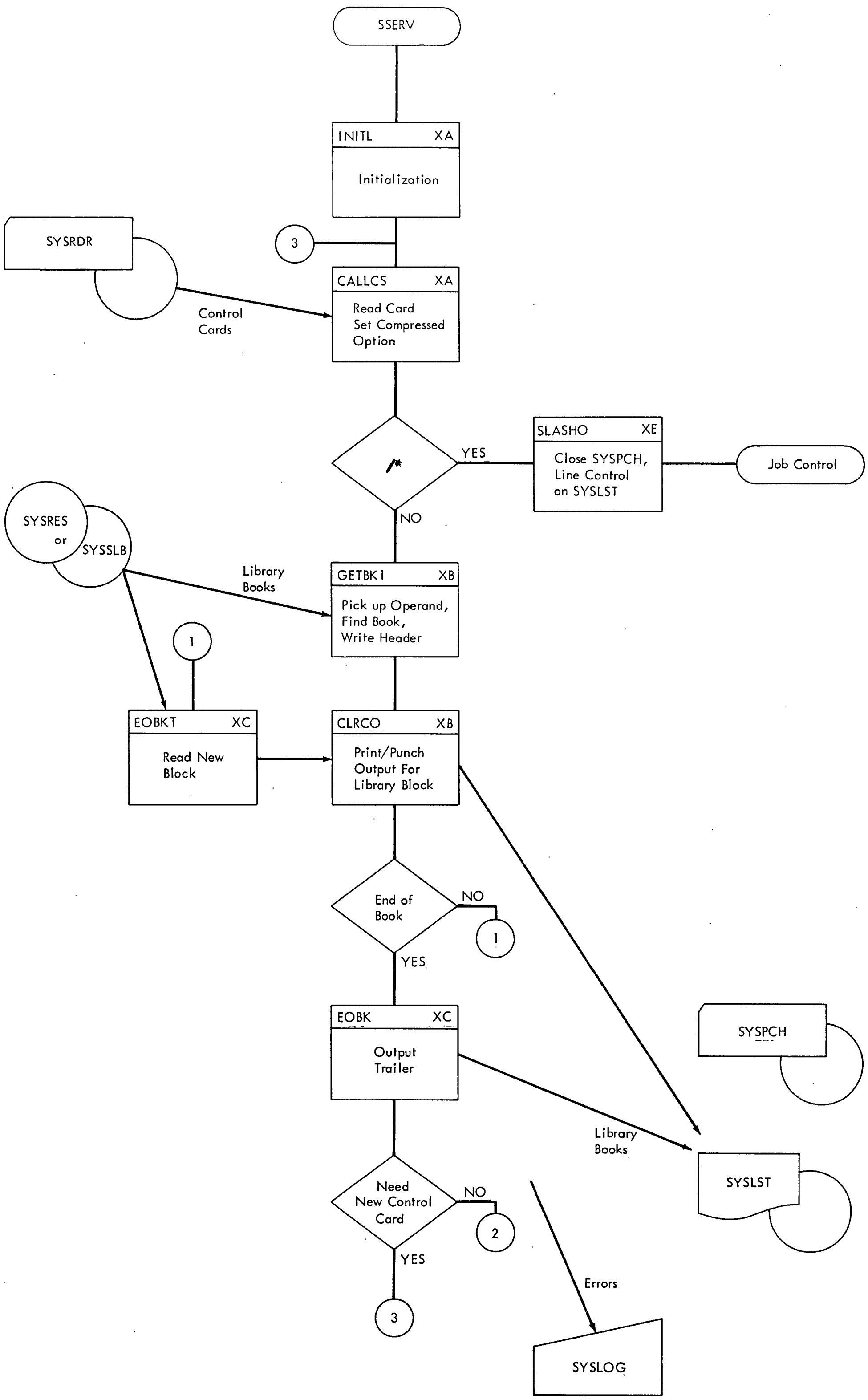


Figure 57. SSERV Program and I/O Flow

OUTPUT LIBRARY RECORDS CHART XB

Objective: To analyze an operand on the control card, to locate the book named, and to print or punch the library records.

Entries:
1. From the Read-Control-Card routine to process the first operand on a control card.
2. From the End-of-Book routine when end-of-book is not found and another library record must be processed.
3. From the End-of-Book routine to process subsequent operands on the control card.

Method: The first operand in the control card is extracted by the subroutine BLNENT. If present, the sublibrary prefix A. or C. is saved, and a test for the operand ALL is made. If ALL, FNDBK locates the first book in the sublibrary and BKNDOU writes the header.

If the operand is not ALL, it must be a bookname. FNDBK locates the book and BKNDOU writes the header.

At CLRCO the output buffers are filled and written out until the end of the input record is found. Records written on SYSPCH may be in compressed form as they are in the library record, or may be expanded to the original card format. Records written on SYSLST are always expanded.

When the end of the input record is encountered, control goes to the End-of-Book routine.

END OF BOOK CHART XC

Objective: To test for end-of-book and either to read in another record from the book or to complete processing for the book.

Entry: From the output library record routine when the end of a library record block is encountered.

Method: If the end of book test indicates more records remain to be processed in the book, another record is read from the library. Control returns to the Output-Library-Record routine.

If end-of-book has been reached, the last line on SYSLST and the last card on SYSPCH are written. A BKEND statement is also written. Control returns to:

- GETALL in the Output-Library-Record routine if the current operand is ALL and more books in the sublibrary remain to be processed.
- TSTQUA in the Output-Library-Record routine if more operands on the control card remain to be processed.
- CALLCS in the Read-Control-Card routine if a new control card is required.

OUTPUT SUBROUTINES CHART XD

Objective: To punch or display source statement library records.

CPRPCH (alternate entries CPPCH1 and CPPCH2) punches a compressed card image on SYSPCH. Columns 76-80 contain a card sequence number.

EXPOUT punches on SYSPCH and prints on SYSLST an expanded card image according to switch indicators for punch and display output. If compressed output is specified, this routine does not punch.

INPUT SUBROUTINES CHART XE

Objective: To read and analyze a control card.

RDRDR reads a control card. If the card is blank, it reads another. If the card is /*, control goes to SLASHO to display and punch the end-of-file indicator and to call in Job Control with an SVC 14.

BLNENT scans across the card for the first operand.

NBLENT locates an operand after the first by scanning the operand field for a comma or a blank. A blank delimiter indicates the last operand has been located.

FIND BOOK SUBROUTINE CHART XF

Objective: To locate a book in the source statement library.

Method: This routine determines the position of the source statement library file with respect to the desired book. The action taken depends on the position:

- Read tape to desired book if in the desired sublibrary and not past desired book.
- Forward-space the file to next tape mark if neither in nor beyond the desired sublibrary. Read tape to desired book.

* Backspace the file to tape mark if reading in the desired sublibrary but past the desired book. Forward space over the tape mark and read to desired book.
* Backspace the file two tape marks, and forward space over the last tape mark if tape was past desired sublibrary. Read to desired book.

A block count of each read is kept in case repositioning ever becomes necessary. If the source statement library is on SYSRES and the tape has been moved out of the library area by a FETCH, SYSRES must be repositioned back to its former position in the source statement library. If the tape was moved out of the source statement library area, it must be ahead of the library. The tape must be read forward to the desired book. Using the block count, it is repositioned to the block it was operating on when the FETCH occurred.

HEADER SUBROUTINE CHART XG

Objective: To punch and display the heading lines CATALS and BKEND.

A CATALS header is punched on SYSPCH if punching is required. A CATALS header is displayed on SYSLST if display and punching are required.

A BKEND header with the bookname is punched if punching is required. If compressed output is specified, a compressed output operand and a sequence-check operand are included in the header. A BKEND header with the bookname is displayed if display is required.

SPACE CONTROL SUBROUTINES CHART XH

Objective: To control spacing on SYSLST.

SPACE1, SPACE2, and SPACE3 skip 1, 2, 3 lines on SYSLST. In each case a check is made to determine if the skip requires starting a new page (line count equals zero).

EJECT skips to a new page on SYSLST. The line count is restored to the number of lines per page on SYSLST.

ERROR MESSAGE SUBROUTINE CHART XJ

Objective: To print error messages on SYSLOG and SYSLST.

Method: This routine writes the last control card on SYSLST. It gets the address and length of the error message to set up the CCW. There are three types of messages:

* Information messages
* Decision messages
* Action messages.

If it is an information type, the message is printed on SYSLST and the routine returns to the link register address.

If it is a decision type, the message is printed on SYSLST. If SYSLST and SYSLOG are different devices, the last control card read and the message are printed on SYSLOG. If SYSLOG is not a 1052, the job is canceled. If SYSLOG is assigned as a 1052, the user has an option to ignore the message or cancel the job.

If it is an action type, the message is printed as for a decision type but no control card is written on SYSLOG.

SUPERVISOR

Label	Chart
BORTN	CB
BSRRTN	CB
CALLCN	BC,BD,BF,BG,BH
CANCSW	AB
CHK1	BH,BE
CHNSCH	AA
CKCD	BG
CLRCCB	AA
CNCL	BC
COMPAR	BG
CORPUB	AC
CQDSPL	AA
CQUDSP	AA
CRJRTN	CD
CSSW1	AB
CSSW2	AB
CTLRTN	CC
CTROLL	CD
DISWHY	AC
DTCRTN	CC
DTCVRT	CA
ECHECK	CA
EOJ	BD
ERECEX	CD
ERETRY	CB
ERGRTN	CC
EXTINT	BH
FETCH	BG
FGTPUB	AA
FREDEV	AC
FROMTR	BA
FTCH1	BB
FTCH2	BB
FTCH3	BB
FWSTCL	CD
GETCHQ	AC
GETCOD	BA
GETSEN	AD
GIOADR	AB
ILSVC	BF
ILSVC2	BF
IMDCMD	AB
INITRG	AC
INTERR	AC
INTRTN	CB
IOPC	AD
ITEXIT	BE
JBOVER	AB
KEYRT	BH,BE
LCLRTY	CB
LDREGS	AC
LEXEQU	CB
LINTRQ	CB
LM2F	BB
LOAD4	BB
LTFETC	BG
LTRET	BD
LUNASG	AA
MVCOM	BB
NCLRET	BB,BC,BD,BE,BF
NONTRY	AA
NOQENT	AD
NOTFUL	AD
NOTLST	BG
NOTRES	AD
NXTDEV	AC
NXTPUB	AC
NXTQNT	AA
OCEXIT	BF
OVERTN	CB
PCEXIT	BE
PCHECK	BH
PTFETC	BG
PUNASG	AA
QUEERR	AD
RDBKWD	BG
RDBRTN	CC
RDCNT	CC
RDFOR	BG
RDRES	BG
RDRTN	CC
REGS	BE,BF
RESMVT	BD
REWND	BG
RSBUSY	BE,BF
RTNCHK	BE,BH
SELECT	AC
SETBSY	AB
SETCCW	BG
SETIME	BC
SKPF2	BB
SSELER	CC
STMODE	AB
STRTIO	AB
STXITC	BF
STXITP	BE
STXITT	BE
SVCINT	BA
SYSAND	BD
SYSOR	BD
TAPERR	CA
TCLRN	CD
TESTFP	CD
TIECRC	CD
TILOOP	CD
TOTRTN	BC
TOUSR	BC
TSTAPE	AB
TSTCDE	AD
TSTEOJ	AB
TSTERQ	AB
TSTHI	BA
TSTLDP	CD
TSTLRD	CC
TSTLST	CC
TSTNOS	CC
TSTNUQ	AA
TSTQEF	AC
TSTUEX	AC
UNCRTN	AD
UNTCHK	AD

URET	BC
WAIT	BC
WNDTP	BG
XRET	BE,BF,BH

TRANSIENT ROUTINES

Label	Chart
$$ANERRM	CE
$$ANERRN	CF
$$ANERRO	CF
$$ANERRP	CG
$$ANERRU	CH
$$ANERRV	CJ
$$BCHKPT	DA
$$BCNCL	DB
$$BDUMP	DC
$$BEOVRT	DK
$$BILSVC	DE
$$BJCOPT	DF
$$BMSGIN	DG
$$BPCHK	DH
$$BPDUMP	DC
$$BRSTRT	DJ

JOB CONTROL

Label	Chart
ACTION	EC
ALTTST	FA
ASGEND	EF
ASGNLS	FA
ASGNLU	EE
ASGNPR	EE
ASGNSY	EE
ASGNS3	EE
ASGNS4	EE
ASGTYP	EF
ASSGN	ED
BINCON	FB
BTLRTN	FB
BTOFRT	ER
BTONRT	ER
CANCEL	EG
CATAL	EP
CCLOUT	EG
CCLRTN	EG
CCPROC	EB
CHKLUN	EN
CHKNUM	FB
CHKTYP	EN
CHPR	FB
CHSYS	FB
CHSYU	EG
CHSYS0	FB
CHSYS1	FB
CHSYS2	FB
CKCVCU	FB
CKCVLU	FB
CKMT	EG
CLOSE	EG
CLRRTN	EK
CMNTPR	EB
CONCAT	FC
COPYLP	EC
COPYRT	EC
CRJBSQ	EL
DATE	EG
DEV	ED
DVCDN	EH
DVCDN1	EH
DVCDN3	EH
DVCDN5	EH
DVCDN6	EH
DVCUP	EH
EDTEST	EK
EFFASG	EF
ENDTST	FA
ENTRY	EC
EOFOFF	EL
EOJRTN	EJ
EXCEDT	EK
EXCPRG	FC
EXCUSR	EK
EXEC	EK
FETCHR	EK
GETJIB	FC
GETPUB	FC
GO	EP
GOCAT	EP
IGN	ED
IGNORE	EJ
IGNRTN	EJ
INCLUD	EC
INCPBP	EJ
JBCIN2	EA
JBINPR	EJ
JIBDCU	FD
JIBEND	FD
JIBLOP	FD
JOB	EL
JOBCTL	EA
LBLEXH	ES
LINK	EP
LINOUT	FG
LISTIO	EM
LOG	EN
LOGCHK	FD
LOGGER	FE
LOGOUT	FG
LOGPRT	FD
LPRGLP	EM
LSTDWN	EM
LSTPRG	EM
LSTSYS	EM
LSTUNA	EM
LSYSLP	EM
LUALP	EM
MASGN	EE
MOVERT	FD
MSGOUT	FD
MTC	EN
MTCUCV	EJ
MTEEND	EJ
MTERCV	EJ
MTTEST	EJ
MTZERO	EJ
NANJIB	FA
NDSCAN	FF
NMTLB	EN
NOALT	EF
NOASGN	FA
NOCATL	EJ
NOIPT	EJ

NOJBDC	EF
NOJIB	FA
NOJIBP	EE
NOLOG	EN
NOMORL	ES
NOTFSD	EF
NSCLGR	FG
NUMPBL	FB
NXTBIT	ER
NXTOPT	EP
OPASG	EF
OPTION	EP
OPTLOG	EP
OPTNLG	EP
OUTPUT	FA
PAUSE	EQ
PGASG	EF
PGASG1	EF
PGASG2	EF
PGATYP	EF
PHASE	EC
PRPSY0	FE
PUBSRH	EM
QLOGER	FE
RDSTMT	FE
READLG	FE
READRD	FE
REISUE	FC
RESET	EQ
RESFCH	EK
RNOJIB	FF
RSJBPT	FF
RSTASG	FF
RSTCOM	EL
RSTNTR	EE
RSTPRG	FF
RSTRT	EQ
RSTRTN	FF
RSTSYS	FF
SAVPUB	EE
SCANR1	FF
SCANR2	FF
SCANR3	FF
SCNRL2	FF
SET	ER
SIMEND	EL
SIMRET	EL
SIZCHK	FF
SKPUNX	FC
SLINCT	ER
SRCHP2	FB
SSYSDT	ER
STDTST	FA
STIGNO	FA
STMTIN	EB
STPLOG	FG
STPLST	FG
STUASG	FA
SYSLGR	FG
SYSPAR	EM
SYSPTR	FG
TAPEXC	EN
TIMLOG	FG
TIMSTP	FG
TMPCHK	ED
TOPND1	ES
TPCOML	EN
TPLAB	ES
TPLEND	ES
TSTALT	ED
TSTNLB	EN
TSTTMP	ED
UA	ED
UPSI	ER
UPSIRT	ER
UPSICH	ER
VOL	ES
YESALT	EE
YESAL2	EE
YESAL4	EE

INITIAL PROGRAM LOAD

Label	Chart
$$A$IPL1	GA
$$A$IPL2	GA
$IPLRT2	GC
ADDRTN	GF
CDSCH	GE
CHKCOM	GC
CLEAR	GA
COMDOK	GC
CONTIN	GA
DBLSCN	GF
DELLOP	GG
DELRTN	GG
ENADR	GF
ENDRD	GA
ENTER	GA
EXTRTN	GA
GO	GA
IOHLD	GD
IPLEND	GC
KEYCHK	GF
LOGRED	GD
LOGSTR	GD
LOGWRK	GB
LUBRTN	GB
MSGRTN	GD
NUMCVT	GF
OFFINT	GC
ONINT	GC
OPFIN+4	GC,GH
OPFIN	GC,GE,GF,GG
OPRTN	GE
ORIGIN	GC
PBFFIN	GC
PUBCLC	GC
PUBDEQ	GG
PUBMKE	GB,GF
PUBMK1	GF
RDRPT	GA
READER	GA
READGO	GC
READRT	GD
RESWRK	GB
REWND	GA
SCNEND	GG
SCNLOP	GG
SETLOG	GA
SETRTN	GH
SKPMVT	GB
TEBCLC	GC
TEBLOP	GG

LINKAGE EDITOR
Pass 1, Coreload 1

Label	Chart
ACTGO	JN
ACTRTN	JN
ACTSCN	JN
ALNKCD	JP
ALNKPR	JP
ALNKSC	JP
ALNKVL	JP
ALNKVR	JP
ATOINC	JK
BLKLOP	JJ
CDGO	JC
CHKRTN	JF
CHKRT1	JF
CNTCHK	JA
CTLCHK	JH
CTLFCH	JD
CTLGO	JH
CTLRTN	JH
ENDRPS	JF
ENDSCN	JJ
EOFCHK	JE
ERR04E	JL
ERR15E	JL
ERR31E	JM
ERR33E	JL
ERR35E	JM
ERR36E	JN
ERR37E	JN
ERR44	JS
ESDCHK	JM
FDSRTN	JJ
FINCK	JG
FNDLOP	JK
FSRRPS	JF
GETCD	JC
GETGO	JC
GETRCD	JE
GETYES	JC
INCRTN	JK
INIT1	JB
INIT2	JB
INIT4	JB
INTCRE	JB
INTFIL	JA
INTIO	JN
INTOPN	JA
IORTN	JR
IORT1	JR
IPTSWT	JQ
LABCK1	JG
LABCK2	JG
LIBERR	JM
LIBEST	JM
LIBRTN	JM
LODORG	JD
LOGCHK	JA
LSETB	JS
LSTCHK	JA
LTCDAD	JS
LTCDNO	JS
LTCDRF	JS
LTESID	JS
MAPGO	JN
MODCHK	JK
MODNFD	JM
MODNF1	JM
NMELOP	JL
NOPHCK	JC
NSTDND	JQ
NSTMAN	JQ
NSTUPD	JQ
NXTSZ	JB
POSRTN	JJ
RETSTR	JJ
RPSLOP	JF
RPSRTN	JF
RSTNST	JQ
SCHCHK	JH
SCHDET	JK
SCHDT1	JK
SCNLOP	JJ
SCNRTN	JL
SETHDR	JN
SETTAP	JN
SKPRD	JE
SMDDIF	JL
SMDSME	JL
SPECIO	JR
STPLOP	JJ
TNTFCH	JD
TNTLOP	JC
TRYLE	JB

LINKAGE EDITOR
Pass 1, Coreload 2

Label	Chart
ACSLTH	KM
BLKRLD	KL
BLKTXT	KJ
CNCALK	KA
EISDPC	KB
EISFXR	KM
ELBCER	KB
ELBDSD	KC
ELBELR	KD
ELBER	KD
ELBGSD	KC
ELBINT	KE
ELBLD	KG
ELBLDR	KH
ELBNAS	KH
ELBNCD	KE
ELBNLR	KG
ELBSD	KC
ENDPRC	KM
ENDRTN	KM
ENDSBM	KM
ENLD	KA
ENOXFR	KM
EPHSCD	KF
EPHSCN	KF
EPHULD	KF
ERR15E	KN
ERR40	KA
ERR41	KE
ERR42	KA
ERR43	KC,KG,KH
ERR45	KB
ERR46	KB,KC
ERR47	KE,KM

ERR50	KJ
ERR55	KK
ERR58	KM
ERR70	KJ,KK,KM
ERR70A	KJ
ESCNCD	KF
ESDGO	KA
ESDNXT	KA
ESDRET	KA
ESDRTN	KA
ESDSBM	KB
ESLBCD	KB
EUPDLT	KE
EUPDOK	KE
EUPDXT	KE
EUPTRY	KE
MVRLD	KL
MVTST	KJ
OPTRLD	KL
OPTXT	KJ
PRTRTN	KN
REPLOP	KN
REPRTN	KN
RLCONS	KK
RLDRTN	KK
RLRET	KK
RLSTP	KK
RLSW1	KK
STDCHK	KM
TSTRTN	KJ

LINKAGE EDITOR
Pass 1, Coreload 3

Label	Chart
ABSCHK	LA
CHQUAL	LD
COLCMN	LH
COMNVR	LH
CTDCST	LH
CTDOVR	LH
CTDPHS	LH
CTDSCN	LH
CTPHST	LH
DECCHK	LA
ENTMVE	LH
ENTRTN	LH
ERROAE	LA
ERRO5E	LB
ERR10E	LA
ERR11E	LA
ERR12E	LB
ERR13E	LA
ERR14E	LB
ERR15E	LH
ERR20	LC
ERR21	LC
ERR22	LC
ERR24	LD
ERR81	LG
ERR82	LH
ER26D0	LG
ILLPHS	LB
ISDISP	LD
ISROOT	LD
NEWPHS	LE
NOAUTO	LB
NOT1ST	LC
NTROOT	LC
NUMCHK	LA
NUMSTR	LA
PHSFIN	LC
PHSPRO	LC
PHSRTN	LA
PHXADD	LG
QALCHK	LB
SCAGAN	LD
SCNCMN	LH
SCSYM1	LC
STRXFR	LH
TRFRAD	LG
WRTHDR	LD

LINKAGE EDITOR
Pass 2

Label	Chart
ABORT	MA,MG
A1	MB
A11	MC
A3	MB
A4	MC
B1	MC
B2	MC
B3	MC
B4	MC
B44	MC
B5	MC
B6	MC
B7	MC
CLOSE	MD
CMPDCT	MA
CMPDLP	MA
CMPDVR	MA
COMCHK	ME
CSCAN	ME
DECSON	MA
DUPLAB	MF
ENDBW1	MC
ENDBW2	MC
ENDPH2	MD
ERAD	MB
ERR085	MA
EXECWR	MA
EXTNLP	MF
EXTNPR	MF
EXTSCN	MF
GET	MB
LDRGO	MF
LDRSCN	MF
MAPCST	ME
MAPHAS	ME
MAPHNM	ME
MOVE	MC
MVXFR	MD
OVFLOW	MG
OVRLSW	MF
PHADMD	ME
PHSTOR	ME
PRBLNK	MG
PREXEC	MG
PREXTN	MF
PRINT	MG
PRLCNT	MG

PRLIST	MG
PRTMAP	ME
SCNCMN	ME
SPLIT	MC
START	MB
START1	MA
SW5	MC
TERSXY	MF
TISTPT	MA
TXT	MB
TXT1	MB
UNRSPC	MF
XFR	MD

LINKAGE EDITOR
Pass 3, Coreload 1

Label	Chart
BUILDHDR	NF
COMPUT	ND
COMPUTE	NB
ENDPH	NN
ENDTST	NM
ERRESD	NH
ESDTST	NH
FNDIDX	ND,NG,NK
GETFIRST	NF
GETRF	NJ
HDRPRT	NA
INITIAL	NA
INITAL	NB
LNECNT	NE
MOVPH	ND
MOVRFAO	NK
MOVWRK	ND,NN
NEWRCD	NG
NEXTRP	NG
NOTLR	NG
OPEN	NA
PERR	NK
PESDOK	NK
PHMOVE	ND
PHRCD	ND
POSPPR	NA
PPOINT	NK
PRT	NA
PRTHDR	NE
PRTIT	NA
RAO	NL
RLDFMT	NG
RLDRD	NC
RPOINT	NG
SAMETP	NN
SAVNEW	NL
SETCDE	NA
SETEOP	ND
SKIPRT	NA
STOR3	NE
TMWORK	NN

LINKAGE EDITOR
Pass 3, Coreload 2

Label	Chart
ADDCOM	PE
COMPNT	PC
CONMOVE	PC
CONTST	PC
DOBBLK	PG
ENDPHA	PH
EXTREAD	PF
FULLBK	PB
GETINFO	PA
GETRLD	PA
HDRWRT	PB
INSERT	PE
MOVHDR	PB
NEWPHA	PB
NEWRP	PD
NOTIN	PD
PULLOUT	PE
RDRLD	PA
RDTXT	PB
REPLACE	PE
RIGRLD	PD
RLDTST	PD
SETBYT	PB
START	PA
SUBSTI	PE
SUBTRACT	PE
TESTCOPY	PA
TESTSUB	PE
TXTWRT	PG
UPDATE	PG

LINKAGE EDITOR
Pass 4

Label	Chart
ABORT	QA
ABTERR	QA
CANCEL	QA
CNVCNT	QA
ENDCOPY	QA
ERRTST	QA
FETCH	QA
RDONE	QA
RLD	QA
START	QA
TXTREP	QA
WRITET	QA

LIBRARIAN
MAINT - Common

Label	Chart
AB10	SC
AB11	SC
AB15	SC
ACATAL	SA
ADELET	SA
AEND	SB
ANALEN	SL
ANEWVO	SC
ANSWER	SF
BCANCL	SE
BLOPER	SB
BSFRTN	SJ
BSFR1	SJ
BSFT1	SJ
BSRINI	SG
BSRRTN	SJ

BUFCOM	SG
CALLRL	SM
CALLSL	SN
CANCEL	SE
CLBRCH	SP
CLRRCT	SJ
COPOPR	SC
CPYASL	SK
CPYECL	SL
CPYERL	SN
CPYRL	SK
DMSG	SF
DONTRD	SB
EMAINS	SA
ENDCL	SM
ENTMAI	SA
EOFINC	SK
EOJ	SE
ERRINV	SA
ERRRTN	SF
EXCP	SH
EXCP1	SH
EXTOPR	SC
FNSHLD	SB
FRSTCH	SD
FSFRTN	SJ
FSRRTN	SJ
FSTCRD	SC
IMSG	SF
INITA1	SD
LBPOST	SJ
LGCARD	SF
LOADS	SN
LOADS1	SN
LOOPXY	SK
LOOPXZ	SK
MNCOPY	SK
MODIF	SF
MODIFY	SK
NEWRD	SA
NOCOMP	SK
NOFNDL	SG
NOFOND	SD
NOPRLB	SM
NOP1	SF
NOP2	SF
NOREAD	SB
NXTOPR	SD
OPERCL	SB
OPEREN	SB
OPERRL	SB
OPERSL	SB
OPN002	SG
OPN2SW	SG
OPRCL	SL
OPREN	SP
OPRERR	SC
OPRRL	SM
OPRSL	SN
OUTORD	SA
PRPHSE	SM
PRVRLB	SM
PRVRL1	SM
PRVSLB	SN
PRVSL1	SN
PUTLST	SA
PVEROR	SM
RCOVER	SJ
RDAFM	SG
RDIPT	SA
RETLST	SF
REWBCK	SJ
RLBRCH	SP
RLSLTP	SN,SP
RSTPOS	SJ
SCANFS	SB
SCANR1	SD
SCANR2	SD
SIXTHC	SB
SKIPIT	SF
SKIPTO	SG
SKIPTT	SG
SLBRCH	SP
SWIRL	SC
SWISL	SC
TPIPT	SA
TSTCAT	SM
TSTEOV	SG
TSTLCH	SA
TSTMV	SH
TXTTST	SF
WRTEOV	SE
WTSW	SH

MAINTC - Core Image Library

Label	Chart
ALLDUN	SR
BGMNT2	SS
BGNMNT	SR
BRNGDN	SR
BYPROG	SS
CATIPL	SQ
CATSPV	SQ
COPIPL	SQ
COPONE	SQ
COPRES	SS
COPYBG	SQ
DELCHK	SS
DELERR	SS
FRSTRU	SR
LEMSG	SQ
MVNAME	SS
NDCILR	SR
NDINIT	SQ
NEWIPL	SQ
RDNUHD	SR
RDOLHD	SR
TRNOFF	SQ
TSTDEL	SS
TSTFRS	SQ
TSTIPL	SQ
TSTSPV	SQ
WINDUP	SR
WRAPUP	SR
WRIPL2	SQ

MAINTR - Relocatable Library

Label	Chart
ACATA1	TA
ADELE1	TA
ADELRS	TA
AINIT	TA
AINITC	TA
AINITF	TA

AINIT1	TA
AINIT2	TA
AINIT3	TA
AINIT4	TA
CATAL	TB
CATCMP	TB
CATEN	TB
CATHP	TB
CATTP	TB
DELET	TC
DELTCM	TC
DELTCP	TC
DELTSK	TC
DEPALL	TC
DEPCMP	TC
DEPCPY	TC
DEPSKP	TC
NOPRG	TC
OUTSEQ	TB
RCAT	TB
RCATA	TB
RCATAL	TB
RCEAP	TB
RCEICD	TB
RCEND	TB
RCESD	TD
RCESDC	TD
RCESDR	TD
RCESDU	TD
RCESMC	TD
RCESWR	TD
RCOCRD	TB
RCOPY	TG
RCOPYL	TG
RCOPYT	TG
RCOPYX	TG
RCPS2	TF
RCPS2R	TF
RCRLD	TE
RCRLDB	TE
RCRLDC	TE
RCRLDE	TE
RCRLDL	TE
RCRLDM	TE
RCRLDN	TE
RCRLDR	TE
RCRLDU	TE
RCTXT	TF
RCTXTB	TF
RCTXTC	TF
RCTXTN	TF
RCTXTR	TF
RCTXTS	TF
RCTXTU	TF
RCWSUR	TD
RDELER	TC
RLEND	TG
RLENDB	TG
RLENDE	TG
RSKIP	TG
RSKIPL	TG
RSKPCT	TG
SKPTMK	TG

MAINTS - Source Statement Library

Label	Chart
ADRSTT	UJ
ADSW	UH
ALLTHT	UD
BKCPRS	UF
BKNDCK	UF
BKNDPR	UG
BKNMT	UE
BKOT1	UC
BKSW	UC
BKWAIT	UC
BK2TST	UH
BNDERR	UG
CALCCB	UJ
CATALS	UA
CATT2	UJ
CAT21	UH
CAT22	UJ
CDCTPR	UG
CDNDT1	UF
CDVSW	UD
CLINOU	UE
CNTERR	UG
CPYS	UJ
CTLCHT	UC
DCSW	UE
DELETS	UB
DELOR1	UB,UE
DELOUT	UB
DELQ	UB
DELSW	UH
DINIRT	UB
DIOSW	UD
DIRDEL	UE
DIRSCN	UE
DIRSW	UD
DIRW	UD
DOUTT	UE
DSBLT	UE
DSCNCN	UE
DSWSET	UB
EOBLKT	UE
EOINT	UE
ERRSEQ	UD
FILOU	UF
FINDIR	UH
FINSHS	UG
FLGBLN	UF
FLGINC	UF
FRESET	UD
GETBKN	UA
GET1CE	UC
IBKNM	UE
INCCON	UE
INCINA	UE
INCOUA	UE
INITS	UJ
IOFIN	UD
LASLID	UA
LCDPRC	UG
LCDPR1	UG
LCDSW1	UF
LCDTST	UF
LCDTSW	UF
LDCDND	UF
LDDEL	UH
LNGCMS	UH
MACCK	UF
MERGE2	UH
MLOOP	UH

MVBCAT	UA
MVBKN1	UA
MVSTMT	UD
NMCK	UA
NMSEQT	UA
NOBADR	UF
OFULLB	UF
OPRERI	UA
OPRERT	UA,UB
OPRRTN	UG
OUTOR2	UA
PAS2SW	UD
PRERSW	UC
P2HIBT	UH
RDDHD	UJ
RDDULB	UJ
RDSAGN	UJ
REALLB	UJ
RECASW	UC
RSETSW	UA
RSTSKP	UD
RSTSWS	UG
SBLEQT	UA
SBLMSG	UA
SBLPRT	UD
SLB2T	UH
SCNCD	UF
SEQNPR	UG
SEQSW	UD
SETDEL	UB
SPRTSW	UD
STPSC1	UH
SWBOA	UD
SWOARA	UC
WAITBI	UD
WAITBO	UD
WAITD1	UC
WAITDO	UC
WAITS1	UC
WAITS2	UC
WAITS3	UC
WAITS4	UC
WNDFM	UG
WTODEL	UB
XITSET	UE

DSERV - Directory Service

Label	Chart
ANSWER	VC
ANUTHR	VH
BGNDSP	VG
BGNLST	VG
BSFRTN	VF
BSFR1	VF
BSFT1	VF
BSRRTN	VF
BYPAGN	VG
CANCEL	VC
CLRRCT	VF
CNVDEC	VG
CORDIR	VG
CORPHS	VG
DIRTYP	VA
DMSG	VC
DONPRT	VC
EDITBK	VJ
EDTMOD	VH
EMAINS	VA
ENTMAI	VA
ERRINV	VA
ERRRTN	VC
EXCP	VE
EXCP1	VE
FINCOR	VG
FINISD	VJ
FINREL	VH
FNDL2	VD
FRSTCH	VB
FSFINS	VG
FSFRTN	VG
FSRRES	VG,VH
FSRRTN	VF
IMSG	VC
INIFND	VD
INITSHL	VB
IOREW	VD
LBPOST	VF
LGCARD	VC
LSTMOD	VH
MODPHS	VH
NDSPLY	VJ
NEWRD	VA
NOFNDL	VD
NOFOND	VB
NOP1	VC
NOP2	VC
NOREAD	VA
NOTBLK	VA
NTONTP	VJ
NUPAGE	VJ
NXTOPR	VB
OPNTST	VD
OPRERR	VA
PAGEND	VJ
PRTHDR	VG
PSEUDO	VG
RCOVER	VF
RDAFM	VD
RDIREC	VJ
RELDIR	VH
RETLST	VC
REWBCK	VF
RSTPOS	VF
SCANR1	VB
SCANR2	VB
SKIPTO	VD
SOURCE	VJ
TRNALL	VA
TRNON1	VA
TRNON2	VA
TRNON4	VA
TSTEOV	VD
TSTLCH	VA
TSTMV	VE
TXTTST	VC
UPFMCT	VG
WTSW	VE

RSERV - Relocatable Library Service

Label	Chart
AINITS	WA
AINITT	WA
ANSWER	WJ

BSFRTN	WM
BSFR1	WM
BSFT1	WM
BSRAGN	WD
BSRRTN	WM
CANCEL	WA
CLRRCT	WM
CMPDIR	WD
CMPMBR	WD
CMP2	WD
CMP2&4	WD
CNVORG	WE
CPLSOP	WB
CPSLSH	WA
DMSG	WJ
DRASFN	WD
EDCPGP	WF
EDPCAI	WF
EDPCBA	WF
EDPCIR	WF
EDPCLA	WF
EDPCLI	WF
EDPCLT	WF
EDPCNA	WF
EDPCRT	WF
EDPCSC	WF
EDPCSI	WF
ENDRTN	WA
ENDRT1	WA
EOMTST	WE
ERRRTN	WJ
ESDPCH	WF
ESDPIA	WF
ESDPRT	WF
ESDPTL	WF
EXCP	WL
EXCP1	WL
EXTK3	WE
EXTRCT	WB
EXTRT1	WB
FNDL2	WK
FSFRTN	WM
FSRRTN	WM
GETFIA	WC
GETFIL	WC
GETFID	WC
GETFR1	WC
GETFR2	WC
GETFSC	WC
GETFSI	WC
GETFSN	WC
GETFSR	WC
GETFTS	WC
GETFWM	WC
IFDOT	WB
ILCCRD	WA
ILOPRD	WB
IMSG	WJ
INIFND	WK
IOREW	WK
LBPOST	WM
LGCARD	WJ
MODIF	WJ
MOVEFD	WB
NOFNDL	WK
NOLIB	WA
NOP1	WJ
NOP2	WJ
OPNTST	WK
PCHSWT	WA
PPSWT	WA
PRTHCM	WE
PRTHDR-4	WD
PRTHDR	WE
PRTHGS	WE
PRTHUD	WE
PRTHUT	WE
PRTNME	WD
PRTSWT	WA
PRT1	WE
RCOVER	WM
RDAFM	WK
RDPCAI	WG
RDPCBB	WG
RDPCCT	WG
RDPCS8	WG
RDPCTS	WG
RDRD2	WE
REINT1	WF
REPPL	WH
REPPRT	WH
RETLST	WJ
REWBCK	WM
RLDPBP	WG
RLDPCH	WG
RLDPCO	WG
RLDPDR	WG
RLDPHE	WG
RLDPRR	WG
RLDPRT	WG
RLDPSW	WG
RLDPUD	WG
RLDPXA	WG
RSTPOS	WM
SKIPIT	WJ
SKIPTO	WK
START	WA
TSTDAL	WD
TSTEOV	WK
TSTMV	WL
TSTPCH	WE
TSTPRT	WE
TXPCIM	WH
TXPCSC	WH
TXTPCH	WH
TXTPIA	WH
TXTPRT	WH
TXTPSA	WH
TXTTST	WJ
WTSW	WL

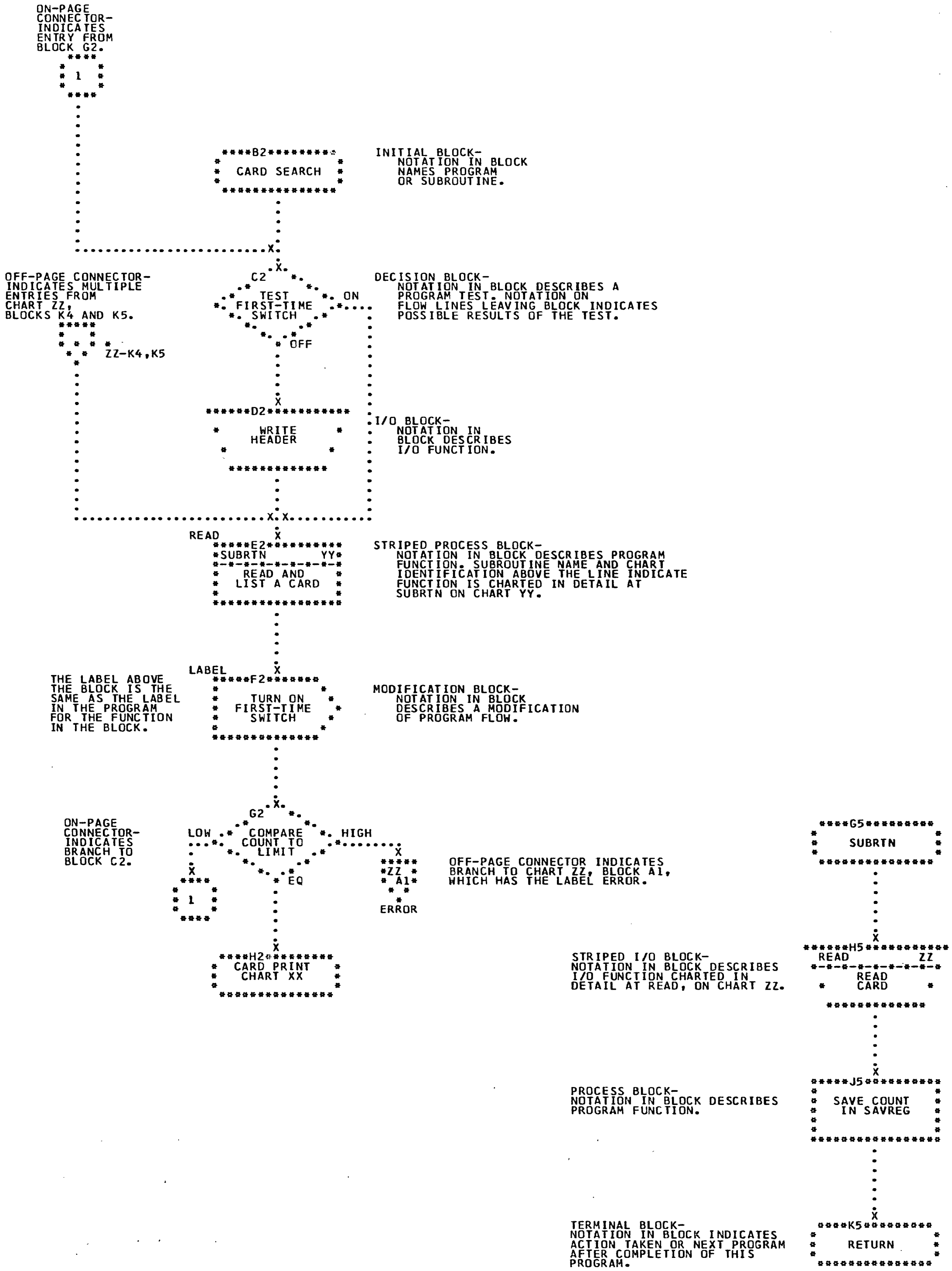
SSERV - Source Statement Library Service

Label	Chart
ALLSW	XC
ALLTST	XB
ANSWER	XJ
BKENDO	XC
BKNDOU	XG
BKNDO1	XG
BLNENT	XE
CALLCS	XA
CDTST	XG
CHNGSB	XB
CLRCO	XB
CMPRST	XC

CPPCH1	XD	NBLENT	XE
CPPCH2	XD	NOP1	XJ
CPRPCH	XD	NOP2	XJ
CPRSCN	XA	PCHEM	XD
CPRSW	XB	PCHESW	XD
CSERR1	XA	PRTNF	XF
CSERR2	XE	PRTTST	XD
CSERR3	XB	RDINI	XF
DMSG	XJ	RDRDR	XE
EJECT	XH	RESCN	XA
EOBK	XC	RES1	XA
EOBKT	XC	RETLST	XJ
EOSYST	XF	REWLIB	XA
ERRRTN	XJ	RSTPRT	XD
EXP	XB	SBLTST	XF
EXPCD	XB	SCNNBL	XE
EXPOUT	XD	SETBIT	XF
EXPTIN	XB	SETOP	XF
FNDBK	XF	SKIPBO	XC
FWDINI	XF	SKIPIT	XJ
GETALL	XB	SLASHO	XE
GETBK1	XB	SLASHX	XE
GETOUT	XE	SLASO1	XE
GOTOSB	XF	SPACE1	XH
IMSG	XJ	SPACE2	XH
INIT1	XA	SPACE3	XH
LCNTST	XH	SVCLIB	XF
LGCARD	XJ	TPPOST	XF
LINCT1	XH	TSTCM1	XG
MODIF	XJ	TSTQUA	XB
MVBKF	XF	TXTTST	XJ
MVBKN	XB	WAITF	XF
MVBKN1	XB	WRGLTP	XA

APPENDIX B. FLOWCHART ABBREVIATIONS

A/O	Assembled Origin
ADCON	Address Constant
ADDR	Address
ALT	Alternate
ASGN	Assign
ASGNMT	Assignment
ASSM	Assembled
BBC	Block Byte Count
BKSP	Backspace
BLK	Block
C/D	Control Dictionary
CCB	Channel Command Block
CCW	Channel Command Word
CHAN	Channel
CHANQ	Channel Queue
CHAR	Character
CHKPT	Checkpoint
CL	Core Image Library
CM	Common
CMD	Command
CNT	Count
CNTL	Control
COMM	Communications
COMREG	Communications Region
CSW	Channel Status Word
CURR	Current
CUU	Channel and Unit Address
DEV	Device
DIR	Directory
EOF	End of File
EOS	End of Supervisor
ER	External Reference
ESD	External Symbol Dictionary
ESID	ESD Identification
EXCP	Execute Channel Program
FAVP	First Available JIB Pointer
FLD	Field
FOCL	First on Channel List
FWD	Forward
FWSP	Forward Space
HDR	Header
HEX	Hexadecimal
I/O	Input/Output
ID	Identification
IGN	Ignore
INFO	Information
JIB	Job Information Block
L/E	Linkage Editor
LD	Label Definition
LIB	Library
LR	Label Reference
LUB	Logical Unit Block
MAX	Maximum
MIN	Minimum
MOD	Module
MPX	Multiplex
MSG	Message
NUM	Number
OP	Operator, Operation
OPND	Operand
PC	Private Code
PIOCS	Physical IOCS
POS	Position
PROB	Problem
PROG	Program, Programmer
PSW	Program Status Word
PTR	Pointer
PUB	Physical Unit Block
R/F	Relocation Factor
R/O	Relocated Origin
REC	Record
REG	Register
REP	Replace
RL	Relocatable Library
RLD	Relocation Dictionary
RSTRT	Restart
SD	Section Definition
SIO	Start I/O
SL	Source Statement Library
SPEC	Specification
STD	Standard
STMNT	Statement
SVC	Supervisor Call
SYS	System
SW	Switch
TCH	Test Channel
TEB	Tape Error Block
TEMP	Temporary
TM	Tapemark
TXT	Text
UA	Unassign
VOL	Volume
WTM	Write Tapemark


ON-PAGE CONNECTOR-INDICATES ENTRY FROM BLOCK G2.
1
B2
CARD SEARCH
INITIAL BLOCK-NOTATION IN BLOCK NAMES PROGRAM OR SUBROUTINE.
OFF-PAGE CONNECTOR-INDICATES MULTIPLE ENTRIES FROM CHART ZZ, BLOCKS K4 AND K5.
ZZ-K4,K5
C2
TEST FIRST-TIME SWITCH
ON
OFF
DECISION BLOCK-NOTATION IN BLOCK DESCRIBES A PROGRAM TEST. NOTATION ON FLOW LINES LEAVING BLOCK INDICATES POSSIBLE RESULTS OF THE TEST.
D2
WRITE HEADER
I/O BLOCK-NOTATION IN BLOCK DESCRIBES I/O FUNCTION.
READ
E2
SUBRTN YY
READ AND LIST A CARD
STRIPED PROCESS BLOCK-NOTATION IN BLOCK DESCRIBES PROGRAM FUNCTION. SUBROUTINE NAME AND CHART IDENTIFICATION ABOVE THE LINE INDICATE FUNCTION IS CHARTED IN DETAIL AT SUBRTN ON CHART YY.
THE LABEL ABOVE THE BLOCK IS THE SAME AS THE LABEL IN THE PROGRAM FOR THE FUNCTION IN THE BLOCK.
LABEL
F2
TURN ON FIRST-TIME SWITCH
MODIFICATION BLOCK-NOTATION IN BLOCK DESCRIBES A MODIFICATION OF PROGRAM FLOW.
ON-PAGE CONNECTOR-INDICATES BRANCH TO BLOCK C2.
G2
LOW
COMPARE COUNT TO LIMIT
HIGH
EQ
1
ZZ A1
ERROR
OFF-PAGE CONNECTOR INDICATES BRANCH TO CHART ZZ, BLOCK A1, WHICH HAS THE LABEL ERROR.
H2
CARD PRINT CHART XX
G5
SUBRTN
STRIPED I/O BLOCK-NOTATION IN BLOCK DESCRIBES I/O FUNCTION CHARTED IN DETAIL AT READ, ON CHART ZZ.
H5
READ ZZ
READ CARD
PROCESS BLOCK-NOTATION IN BLOCK DESCRIBES PROGRAM FUNCTION.
J5
SAVE COUNT IN SAVREG
TERMINAL BLOCK-NOTATION IN BLOCK INDICATES ACTION TAKEN OR NEXT PROGRAM AFTER COMPLETION OF THIS PROGRAM.
K5
RETURN

To perform PIOCS, the user builds a Channel Command Word (CCW) or chain of CCW's defining the channel program, and issues the macros CCB and EXCP (and possibly WAIT).

The CCW provides the I/O command and the data address. The CCW has the following format:

Bits 0-7: command code
Bits 8-31: data address
Bits 32-36: flags
Bits 37-39: zeros
Bits 40-47: unused
Bits 48-63: count

The CCB macro generates a 16-byte field called the Command Control Block (CCB) in which the problem program specifies to the PIOCS routines:

- the CCW address
- the symbolic unit address
- user options.

PIOCS routines use the CCB to return information to the problem program and to save bits from the Channel Status Word (CSW) status bytes when the I/O is completed. The CCB has the following format.

	Count	Transmission Information	CSW ¤ Status Bytes	Symbolic Unit Address	Not Used Must Contain X'00'	CCW Address	Reserved for Physical IOCS	CCW Address in CSW
Bytes	0 1	2 3	4 5	6 7	8	9 11	12	14 15
Used For	Residual Count	Transmitting Information between Physical IOCS and Problem Program	(see Byte 4 / Byte 5)	Hexadecimal Representation of SYSnnn SYSRDR = 0000 SYSIPT = 0001 SYSPCH = 0002 SYSLST = 0003 SYSLOG = 0004 SYSRES = 0006 SYSSLB = 0007 SYSRLB = 0008 SYS000 = 0100 SYS001 = 0101 etc.	Not Used	Address of CCW Associated with this CCB	Must Contain X'00'	Address of CCW in the CSW stored at Channel End. It is 8 Greater than last CCW.

Byte 4 BIT	DESIGNATION	Byte 5 BIT	DESIGNATION
32	Attention	40	Program-controlled interruption
33	Status modifier		
34	Control unit end	41	Incorrect length
35	Busy	42	Program check
36	Channel end	43	Protection check
37	Device end	44	Channel data check
38	Unit check	45	Channel control check
39	Unit exception	46	Interface control check
		47	Chaining check

Byte 2

Bits	0	1	2	3	4	5	6	7
	Traffic Bit (Wait)	End-of-File	Unrecoverable I/O Error	Acceptable Unrecoverable I/O Error	Unused	Post at Device End	Accept tape read data check or 2540 punch equipment check.	User Error Routine
Set On By	PIOCS	PIOCS	PIOCS	Pr. Pr.		Pr. Pr.	PIOCS	Pr. Pr.

Byte 3

Bits	0	1	2	3	4	5	6	7
	Unused	Unused	Unused	2540 Punch Equipment or Tape Read Error	Questionable Condition	Unused	Carriage Channel 9	Unused
Set On By				PIOCS	PIOCS		PIOCS	

PIOCS = Physical IOCS
Pr. Pr. = Problem Program

¤ Bytes 4 and 5 contain the status bytes of the Channel Status Word (Bits 32-47).
If byte 2, bit 5 is on and device end results as a separate interrupt, device end status will OR'd in.

* Indicates /* or /& statement encountered on SYSRDR or SYSIPT.
Byte 4, bit X'01' (unit exception) is also on.

The EXCP macro puts a CCB address in register 1 and issues a supervisor call of 0. The WAIT macro tests the traffic bit in the CCB and, if it is not on, issues a supervisor call of 7; control returns to the user when the traffic bit is tested and found on.

PIOCS routines are entered through an I/O interrupt or a supervisor call of 0. The old and new Program Status Words (PSW's) hold the information necessary to process the interrupts. The PSW has the following format:

Bits 0-7: system mask
Bits 8-11: protection key
Bits 12-15: AMWP mask
Bits 16-31: interruption code
Bits 32-33: instruction length code
Bits 34-35: condition code
Bits 36-39: program mask
Bits 40-63: instruction address

When a request for I/O can be processed, the Actual I/O routine places the CCW address from the CCB into the CSW and the Channel Address Word (CAW) in location 72. The CAW has the following format:

Bits 0-3: key
Bits 4-7: zeros
Bits 8-31: CCW address

The actual I/O routine issues I/O instructions (e.g., SIO) directed to a device specified by a channel address and device address. The addresses are taken from the PUB assigned to the symbolic unit addressed by the CCB.

The actual I/O routine tests the condition codes for each I/O instruction. These condition codes are defined as follows.

Condition code: Start I/O
0: I/O operation initiated
1: CSW stored
2: Channel busy
3: Not operational

Condition code: Test I/O
0: Available
1: CSW stored
2: Channel busy
3: Not operational

Condition code: Test Channel
0: Channel available
1: Interruption pending
2: Operating in burst mode
3: Not operational

Condition code: Halt I/O
0: Channel not working
1: CSW stored
2: Burst operation terminated
3: Not operational

The CSW in location 64 is tested to determine the results of the I/O operation. The CSW has the following format.

Bits 0-3: key
Bits 4-7: zeros
Bits 8-31: CCW address
Bits 32-47: status bytes
Bits 48-63: count

Bits 8-31 of the CSW are saved in the CCB.

FORMAT OF THE ESD CARD	
1	Multiple punch (12-2-9). Identifies this as a loader card.
2-4	ESD -- External Symbol Dictionary card.
11-12	Number of bytes of information contained in this card.
15-16	External symbol identification number (ESID) of the first SD, PC, CM, or ER on this card. Relates the SD, PC, CM, or ER to a particular control section.
17-72	Variable information. 8 positions - Name 1 position - Type code to indicate SD, PC, LD, CM, or ER 3 positions - Assembled origin 1 position - Blank 3 positions - Length, if an SD-type, CM-type, or a PC-type. If an LD-type, this field contains the external symbol identification number (ESID) of the SD containing the label.
73-80	May be used by the programmer for identification.

FORMAT OF THE TXT CARD	
1	Multiple punch (12-2-9). Identifies this as a loader card.
2-4	TXT -- Text card.
6-8	Assembled origin (address of first byte to be loaded from this card).
11-12	Number of bytes of text to be loaded.
15-16	External symbol identification number (ESID) of the control section (SD or PC) containing the text.
17-72	Up to 56 bytes of text -- data or instructions to be loaded.
73-80	May be used for program identification.

FORMAT OF THE RLD CARD	
1	Multiple punch (12-2-9). Identifies this as a loader card.
2-4	RLD -- Relocation Dictionary card.
11-12	Number of bytes of information contained in the card.
17-72	Variable information (multiple items). 1. Two positions - R-pointer (relocation identifier) to the relocation factor of the contents of the load constant. 2. Two positions - P-pointer (position identifier) to the relocation factor of the control sections in which the load constant occurs. 3. One position - flag indicating type of constant. 4. Three positions - assembled origin of load constant.
73-80	May be used for program identification.

FORMAT OF THE END CARD	
1	Multiple punch (12-2-9). Identifies this as a loader card.
2-4	END
6-8	Assembled origin of the label supplied to the Assembler in the END card (optional).
15-16	ESID number of the control section to which this END card refers (only if 6-8 present).
17-22	Symbolic label supplied to the Assembler if this label was not defined within the assembly.
29-32	Control section length (if not specified in last SD or PC).
73-80	Not used.

FORMAT OF THE REP CARD	
1	Multiple punch (12-2-9). Identifies this as a loader card.
2-4	REP -- Replace Text card.
5-6	Blank
7-12	Assembled address of the first byte to be replaced (hexadecimal). Must be right justified with leading zeros if needed to fill the field.
13	Blank
14-16	External symbol identification number (ESID) of the control section (SD) containing the text (hexadecimal). Must be right justified with leading zeros if needed to fill the field.
17-70	From 1 to 11 four-digit hexadecimal fields separated by commas, each replacing two bytes. A blank indicates the end of information in this card.
71-72	Blank
73-80	May be used for program identification.

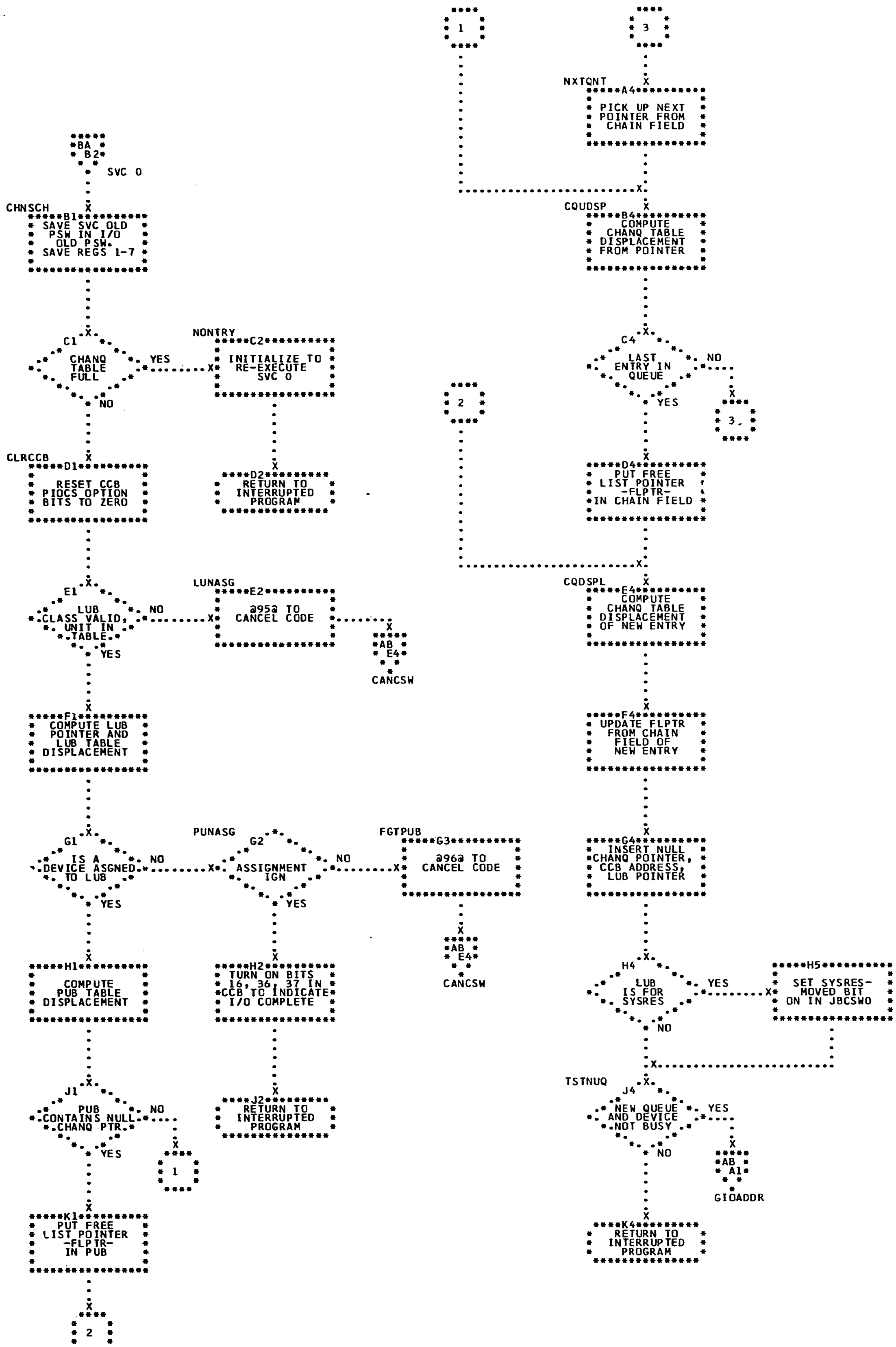


Chart AA. Channel Scheduler

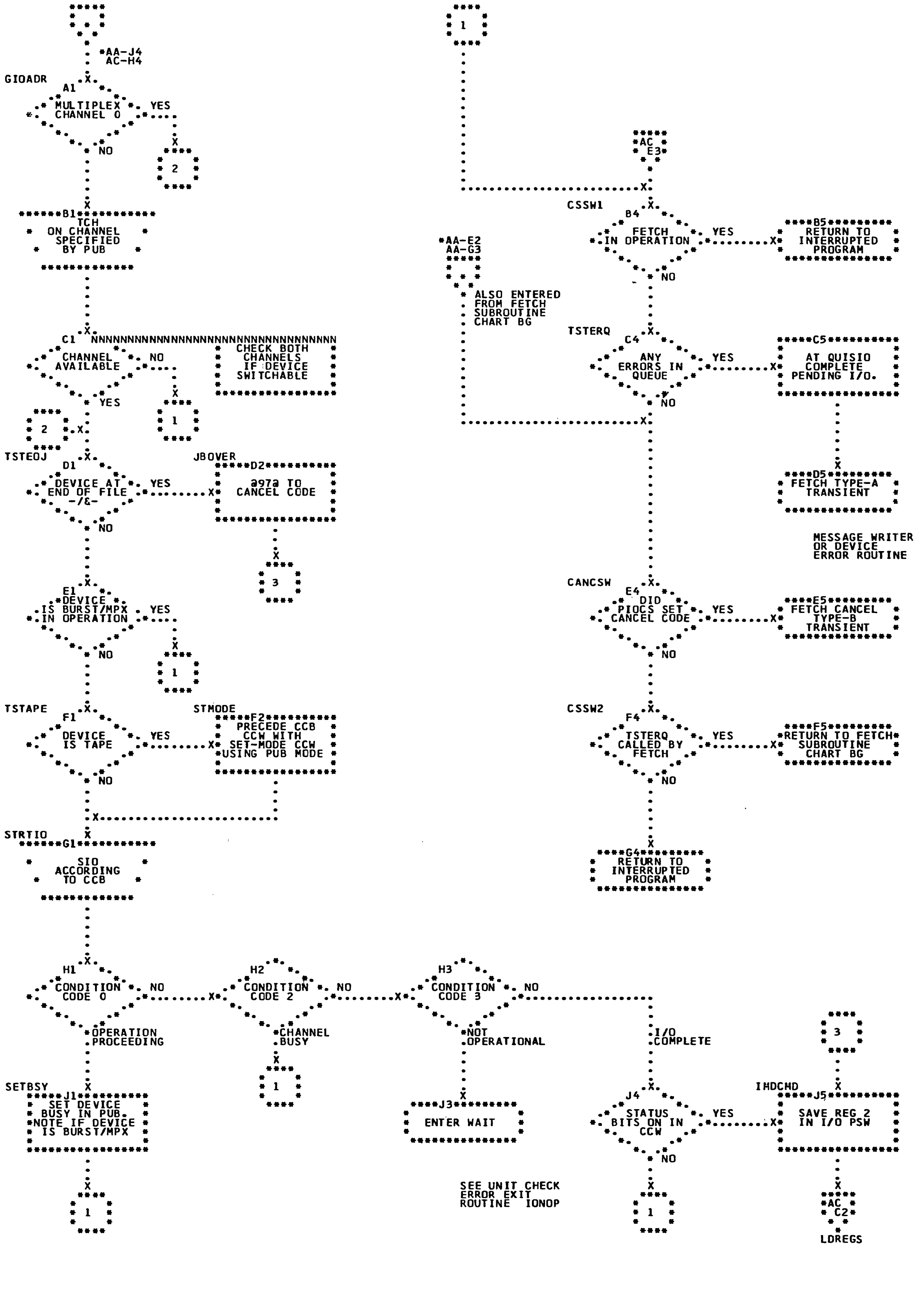


Chart AB. Actual I/O

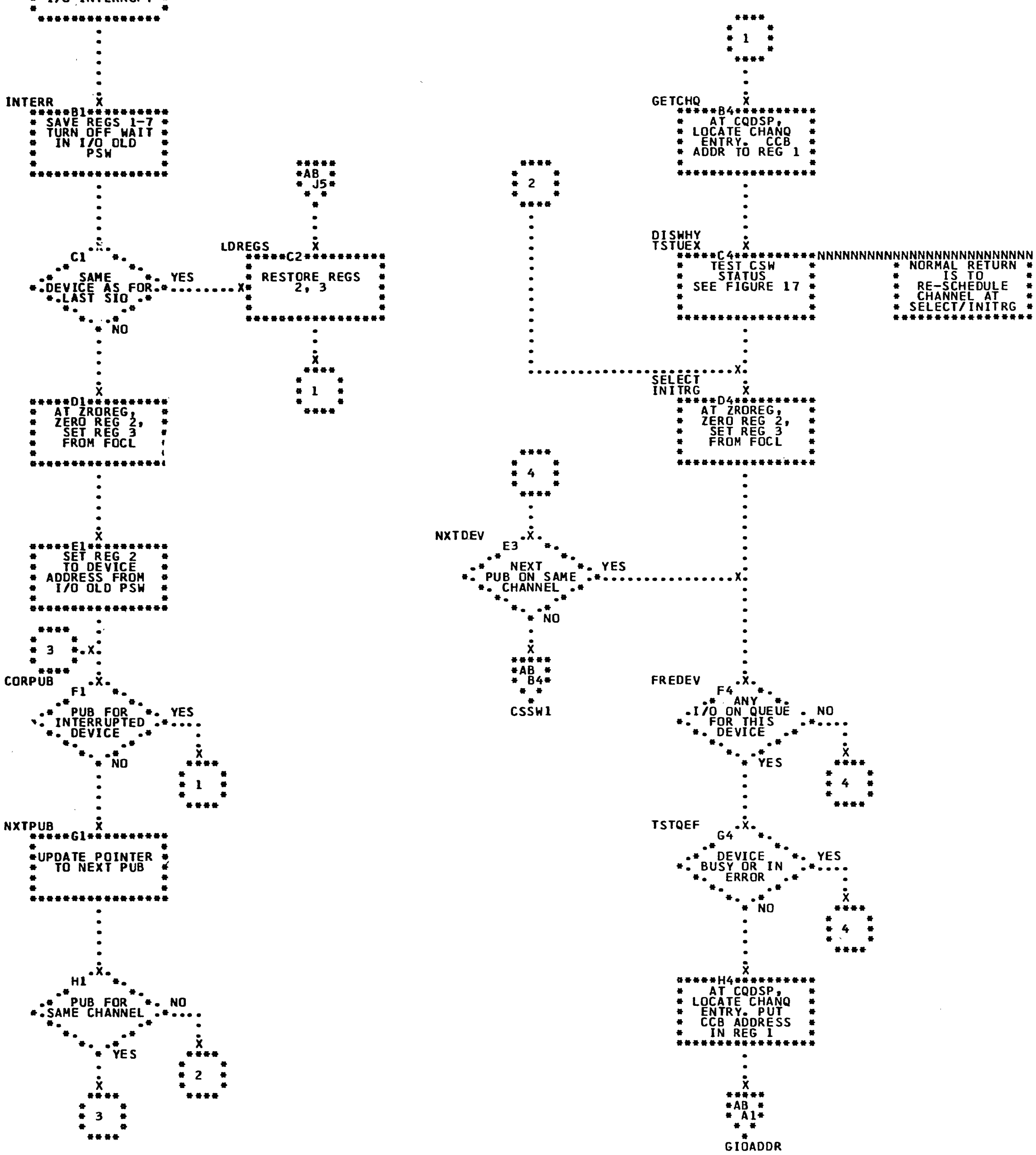


Chart AC. I/O Interrupt

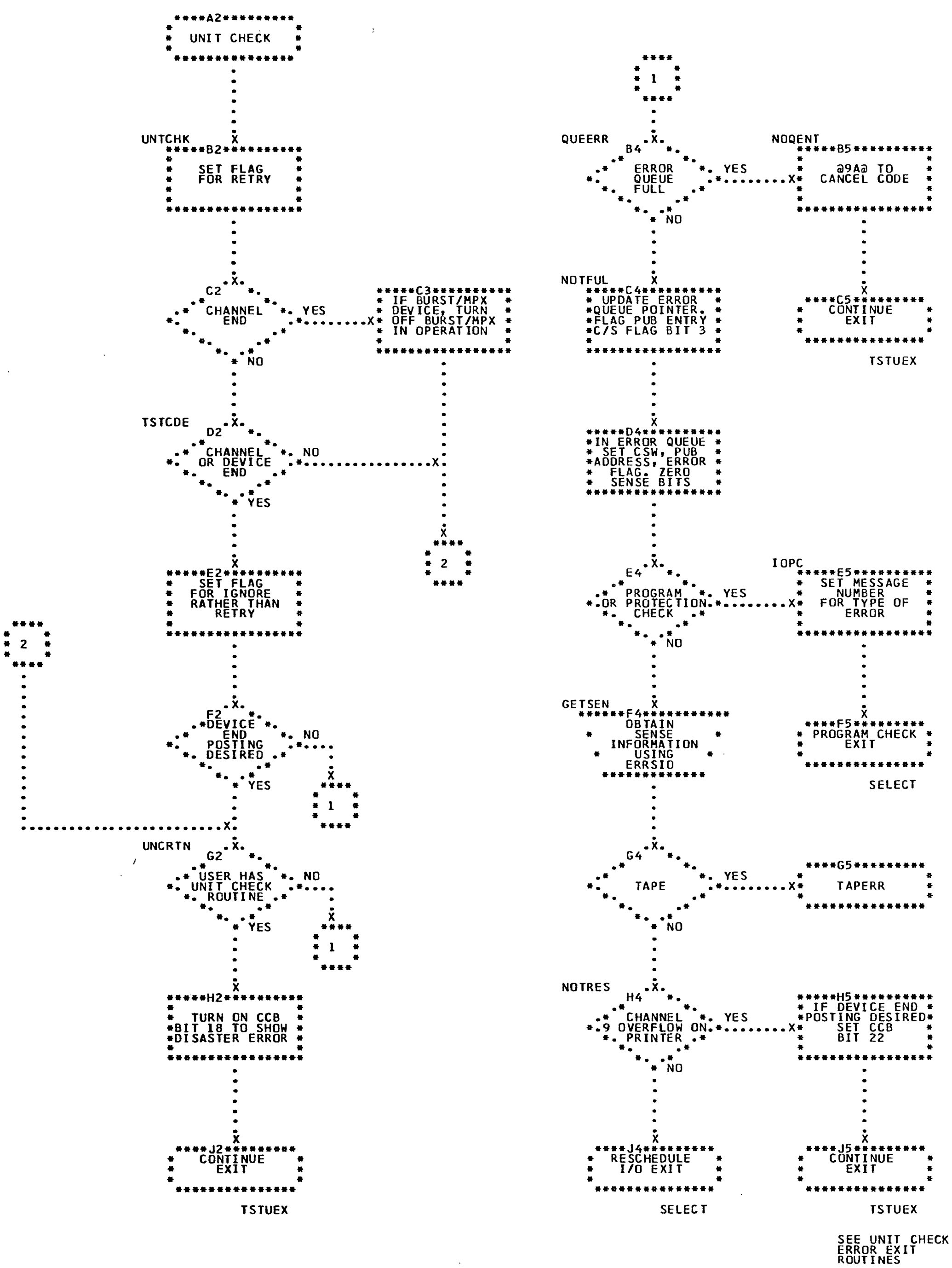
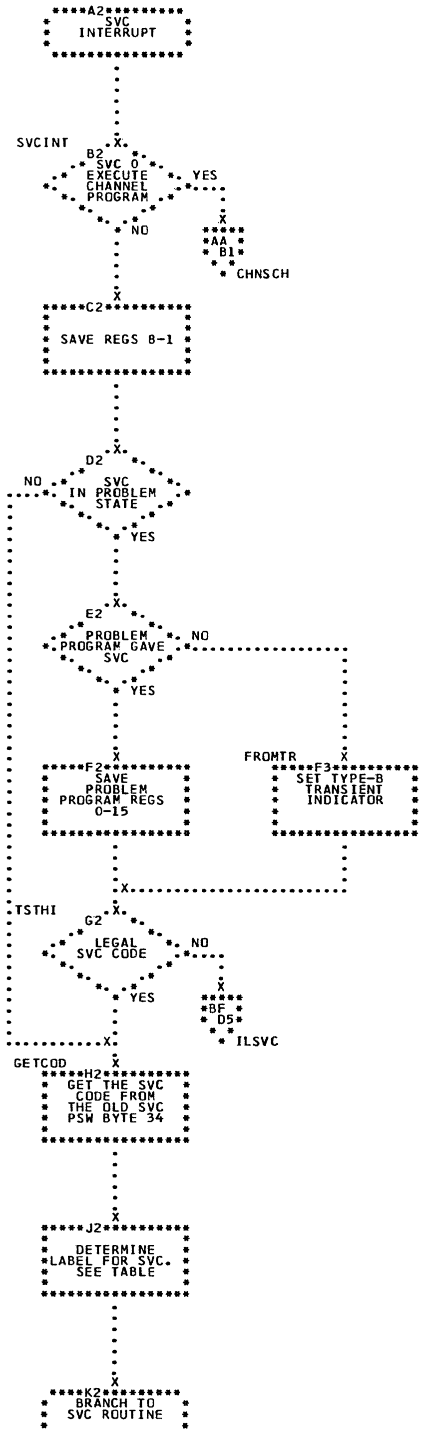


Chart AD. Unit Check



ROUTINE LABEL	SVC	TABLE
FTCH1	1	SVC1
FTCH2	2	SVC2
FTCH3	3	SVC3
LOAD4	4	SVC4
MVCOM	5	SVC5
CNCL	6	SVC6
WAIT	7	SVC7
TOUSR	8	SVC8
TOTRN	9	SVC9
SETIME	10	SVC10
LTRET	11	SVC11
SYSAND	12	SVC12
SYSOR	13	SVC13
EOJ	14	SVC14
RESMVT	15	SVC15
STXITP	16	SVC16
PCEXIT	17	SVC17
STXITT	18	SVC18
ITEXIT	19	SVC19
STXITC	20	SVC20
OCEXIT	21	SVC21

Chart BA. Supervisor Call Interrupt

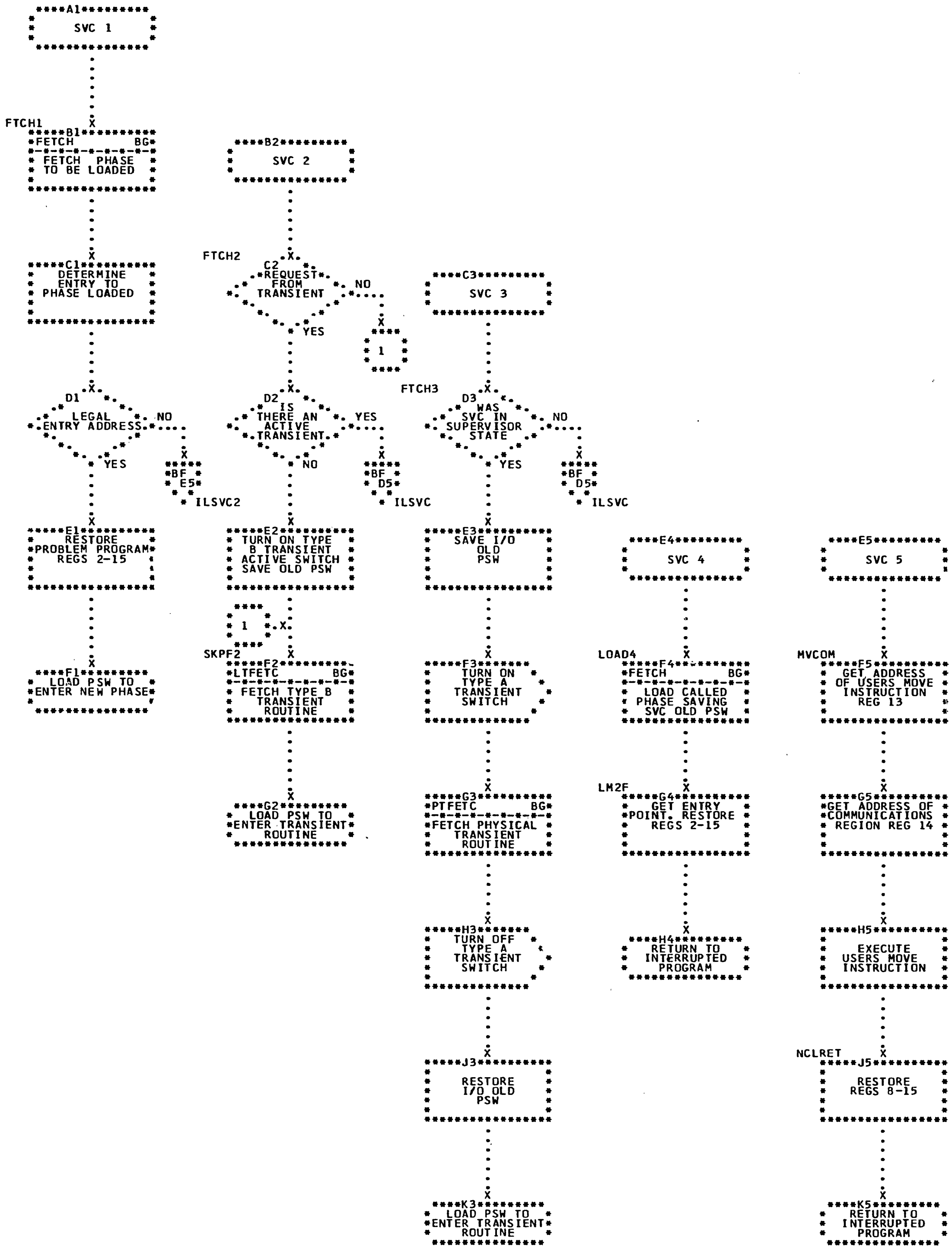


Chart BB. SVC 1 - SVC 5

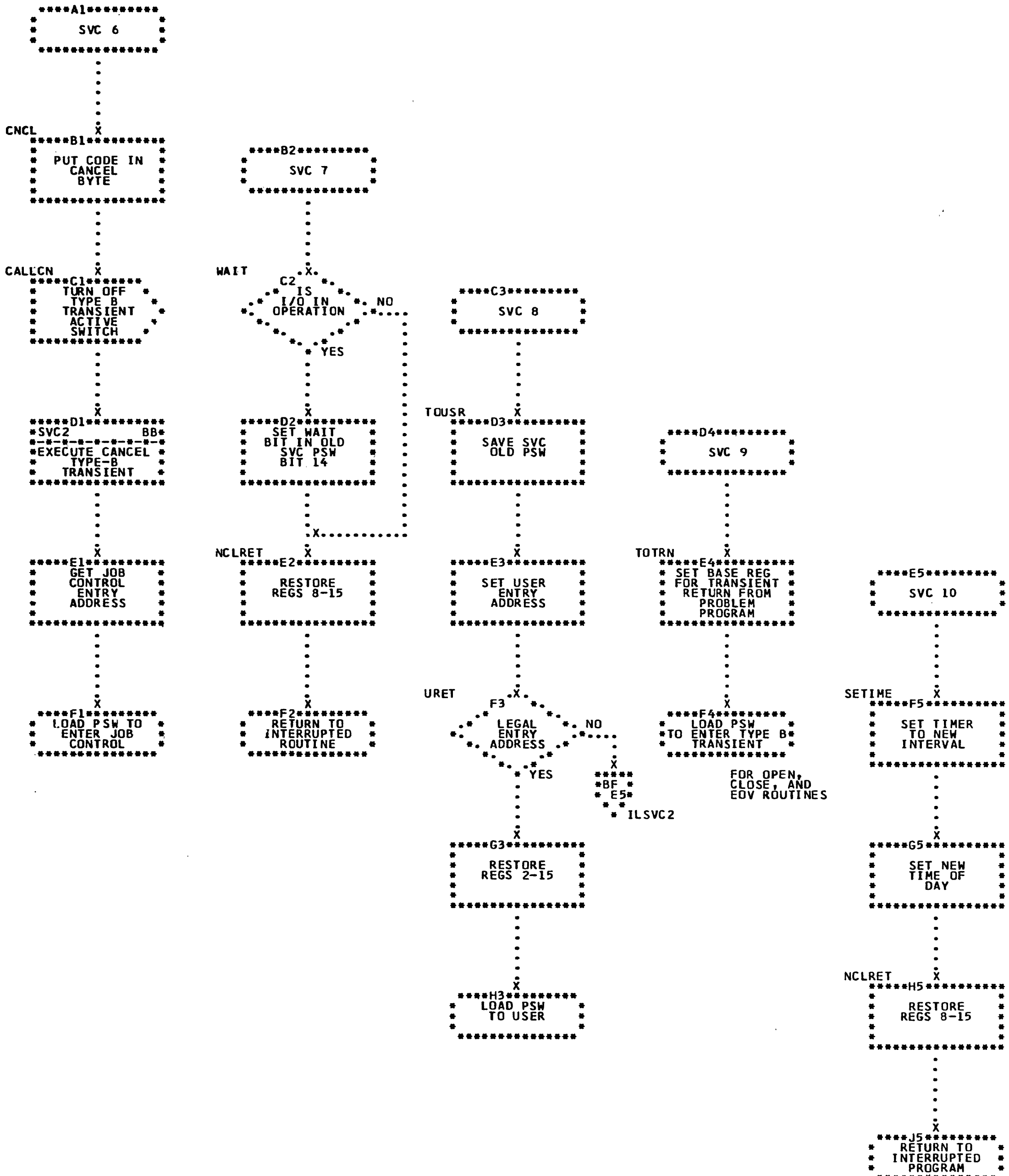


Chart BC. SVC 6 - SVC 10

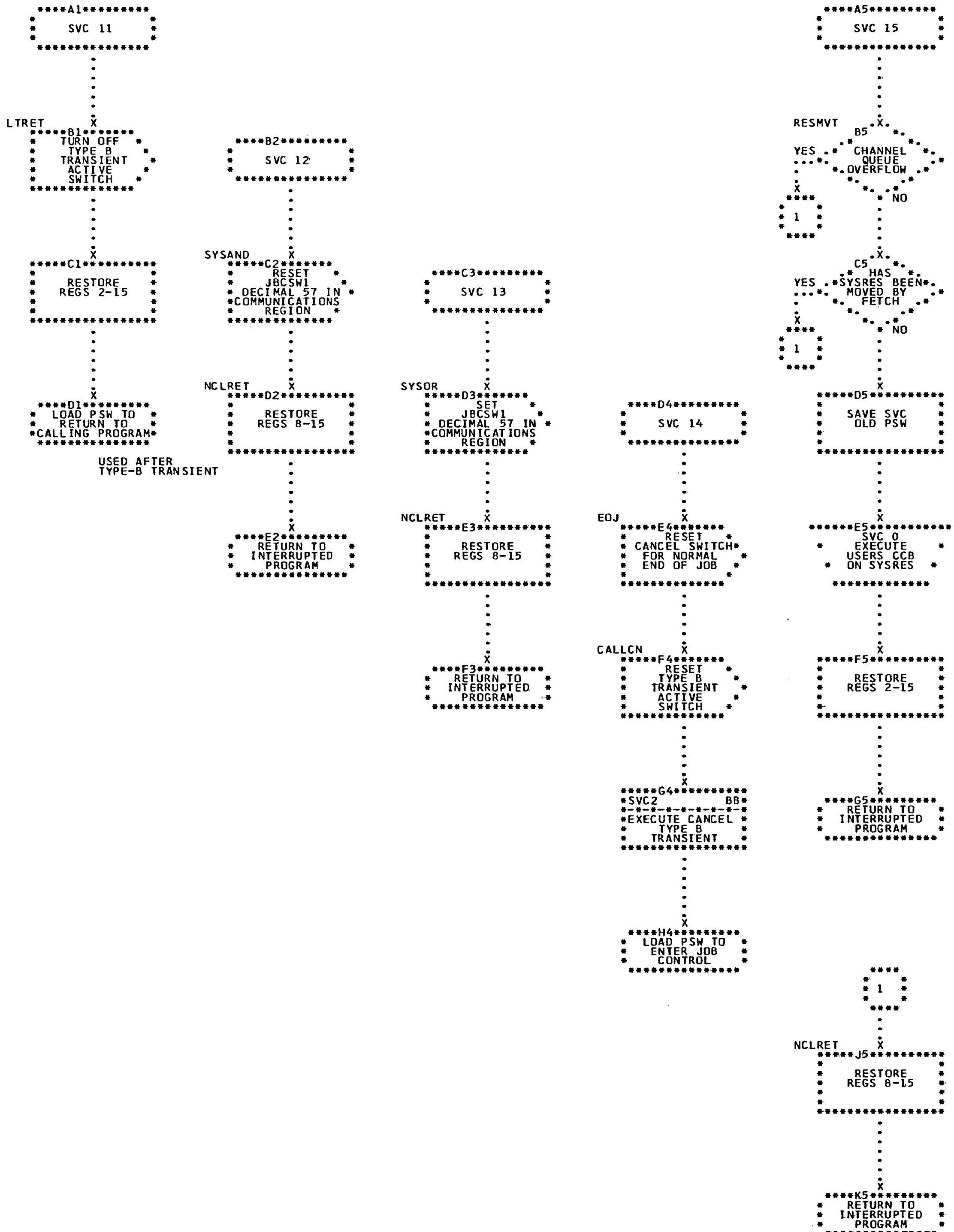


Chart BD. SVC 11 - SVC 15

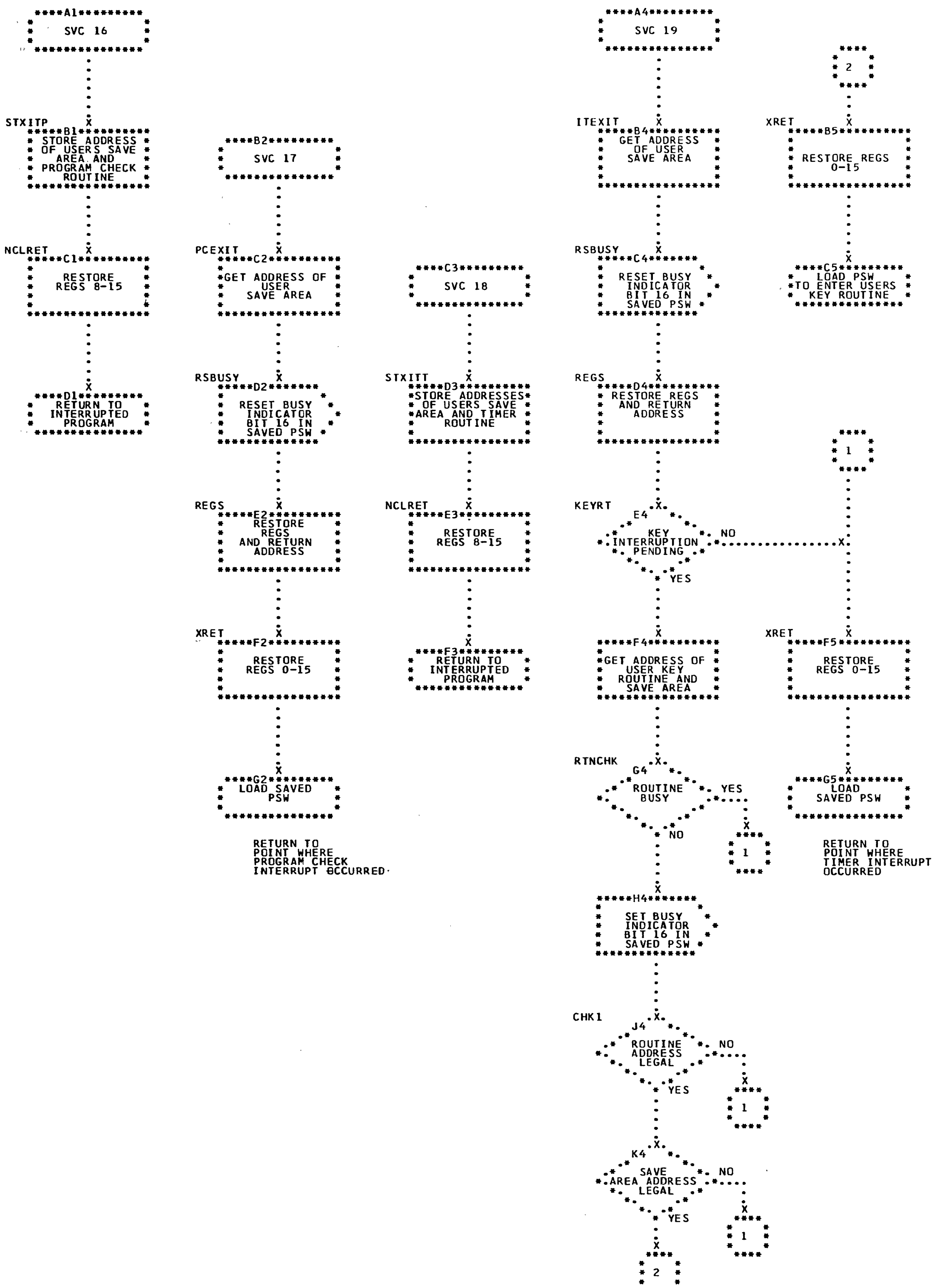


Chart BE. SVC 16 - SVC 19

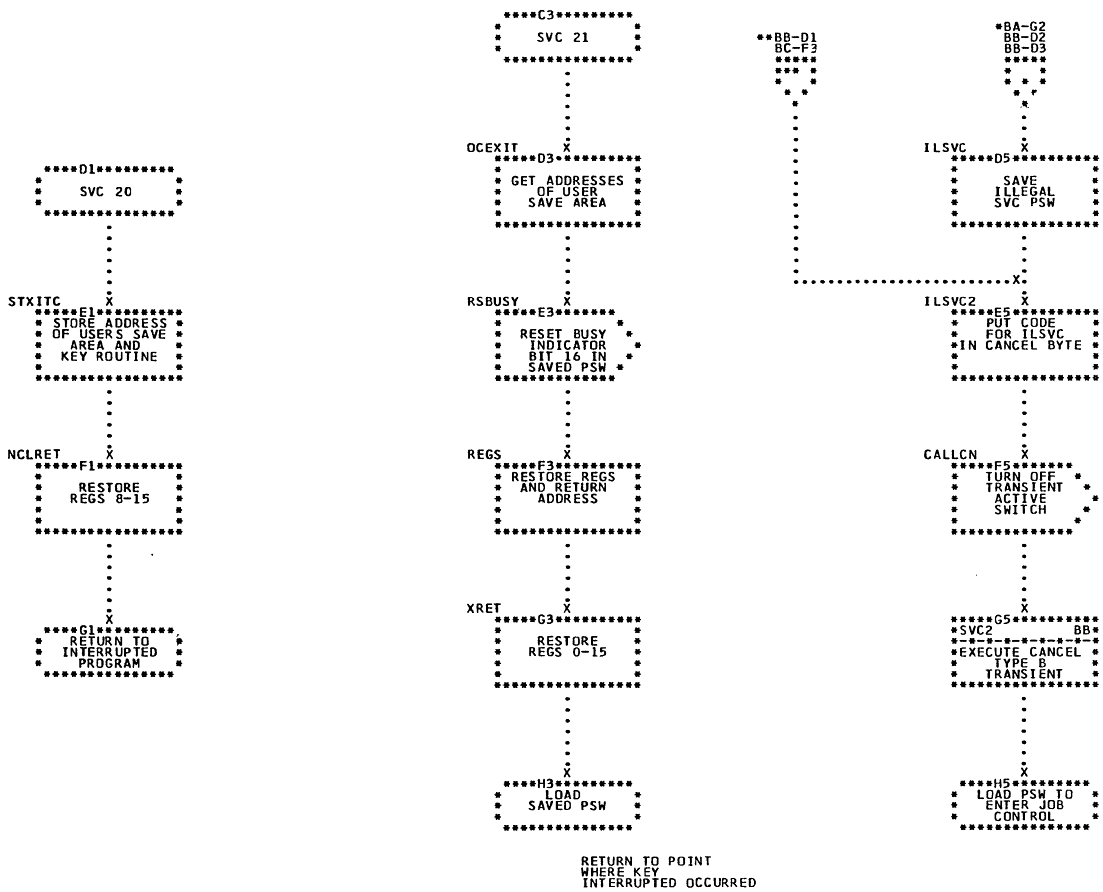


Chart BF. SVC 20 - SVC 21; Illegal SVC

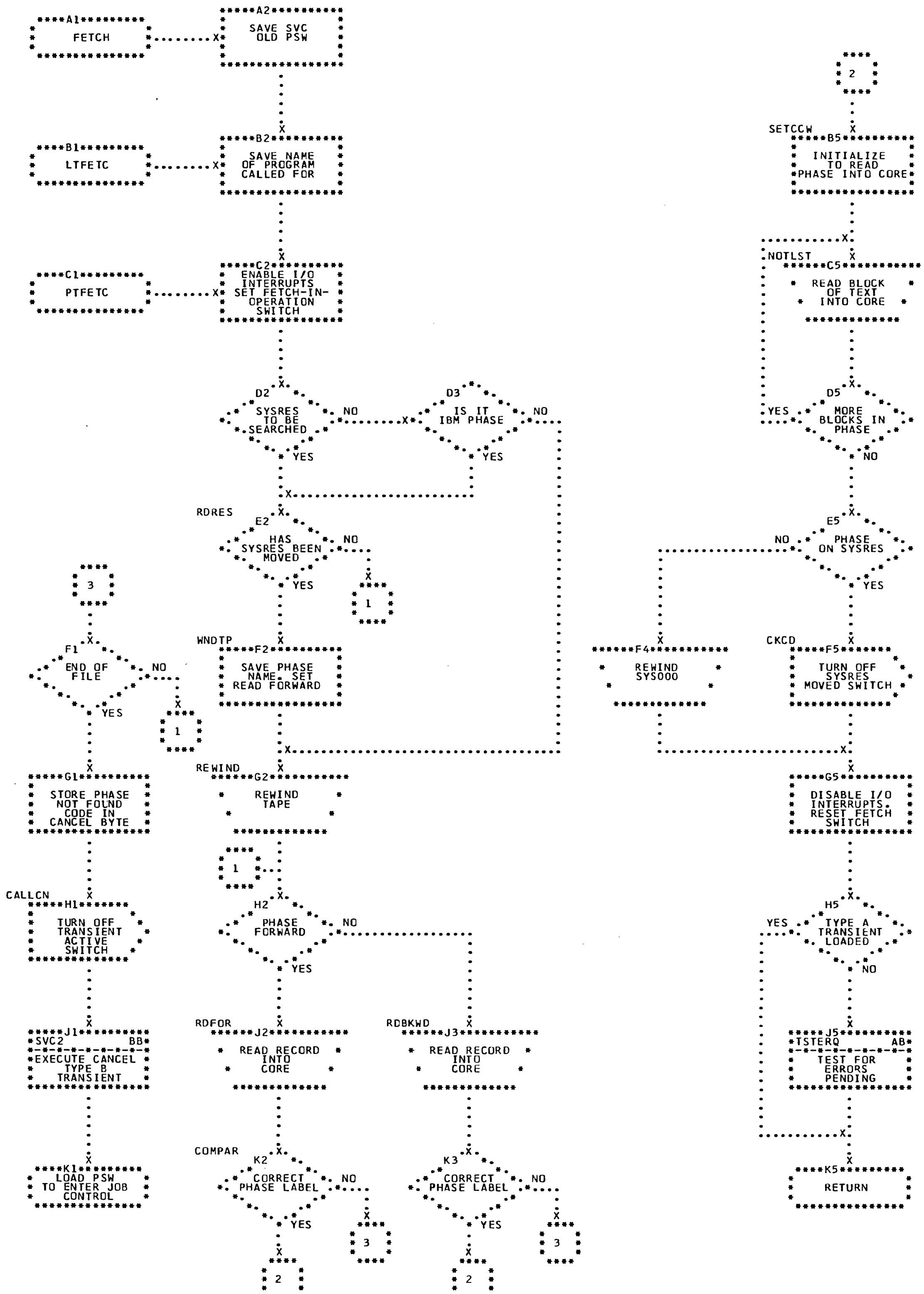


Chart BG. Fetch Subroutine

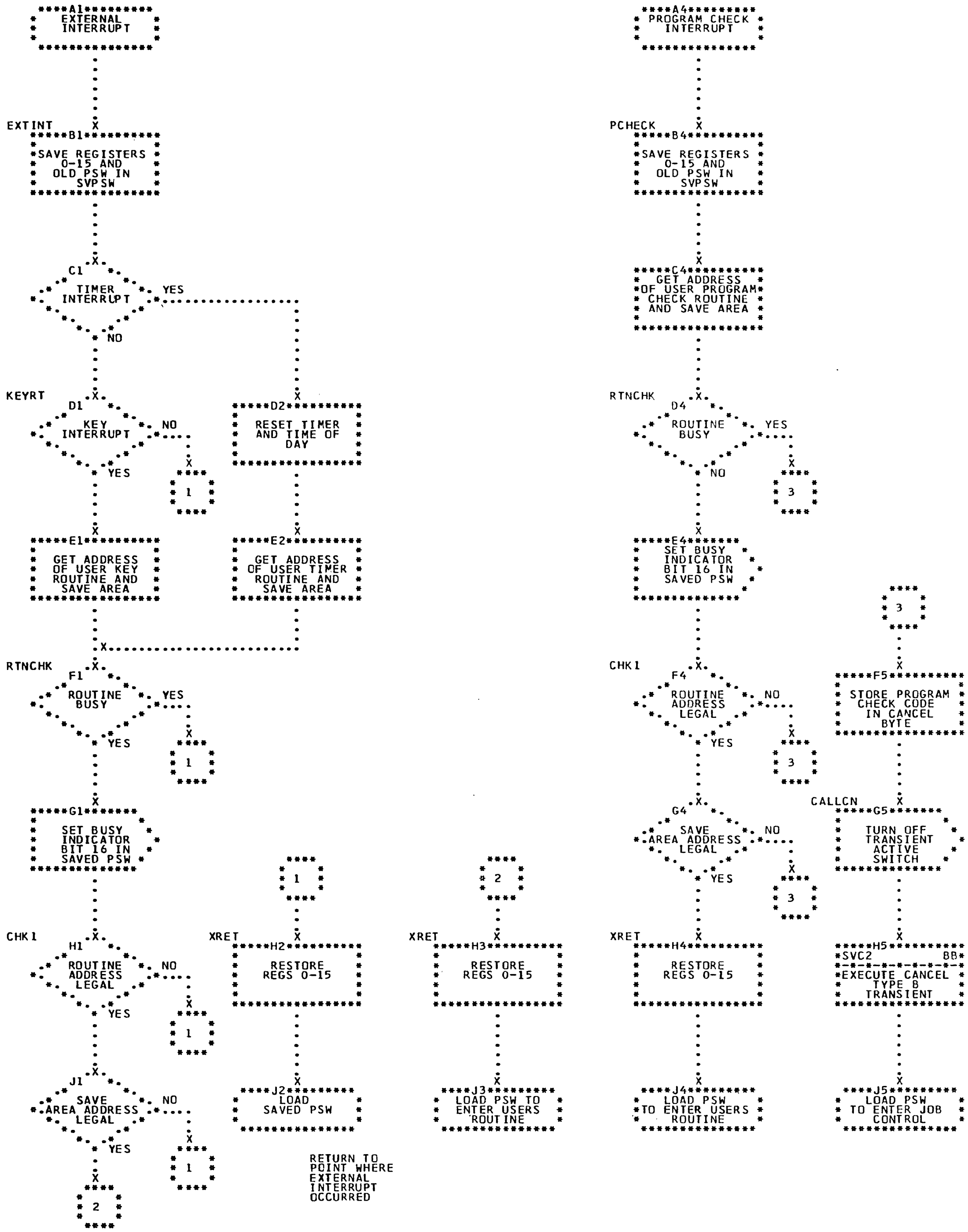


Chart BH. External Interrupt; Program Check Interrupt

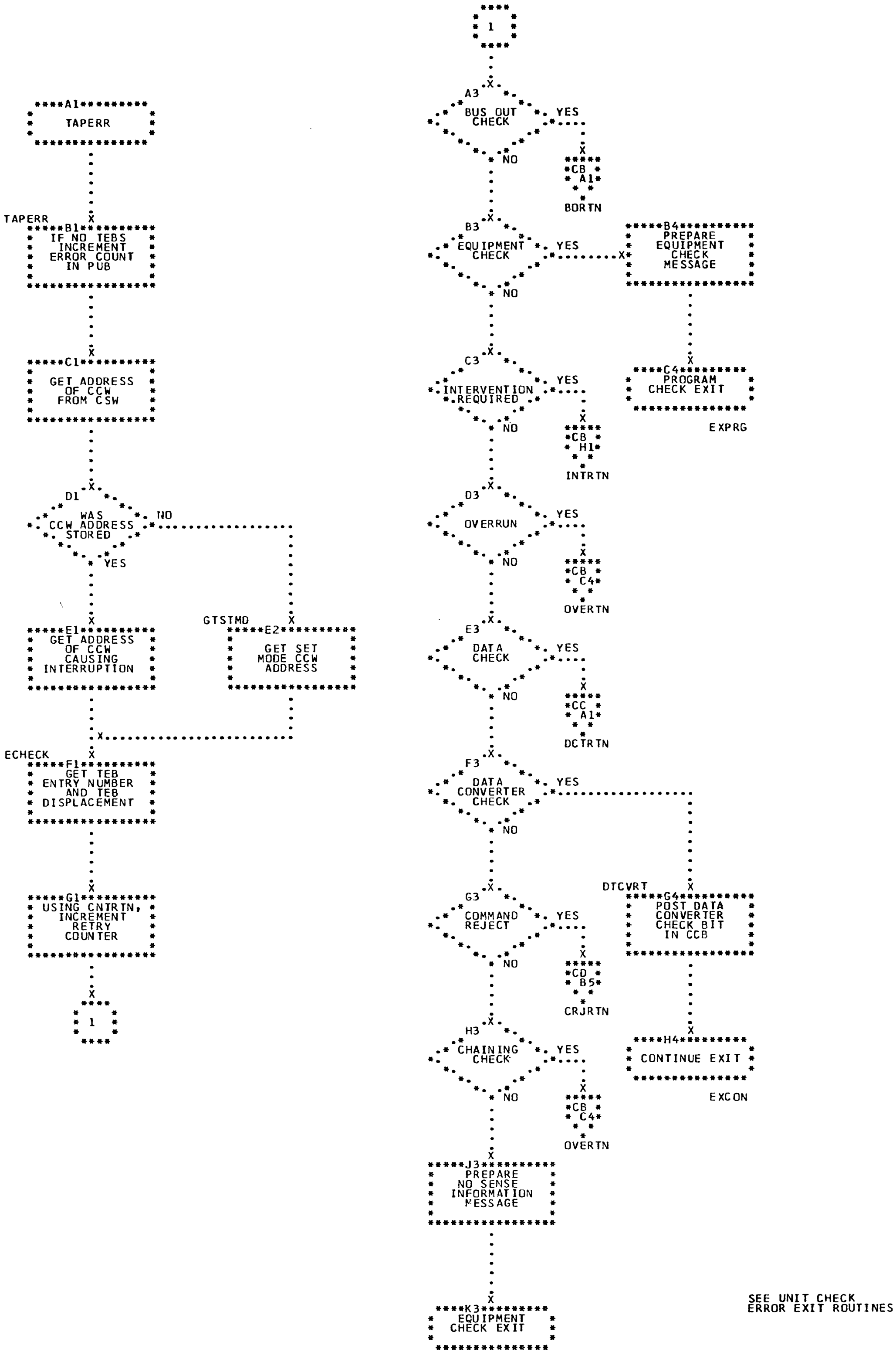


Chart CA. Tape Error Recovery: Equipment Check, Data Converter Check

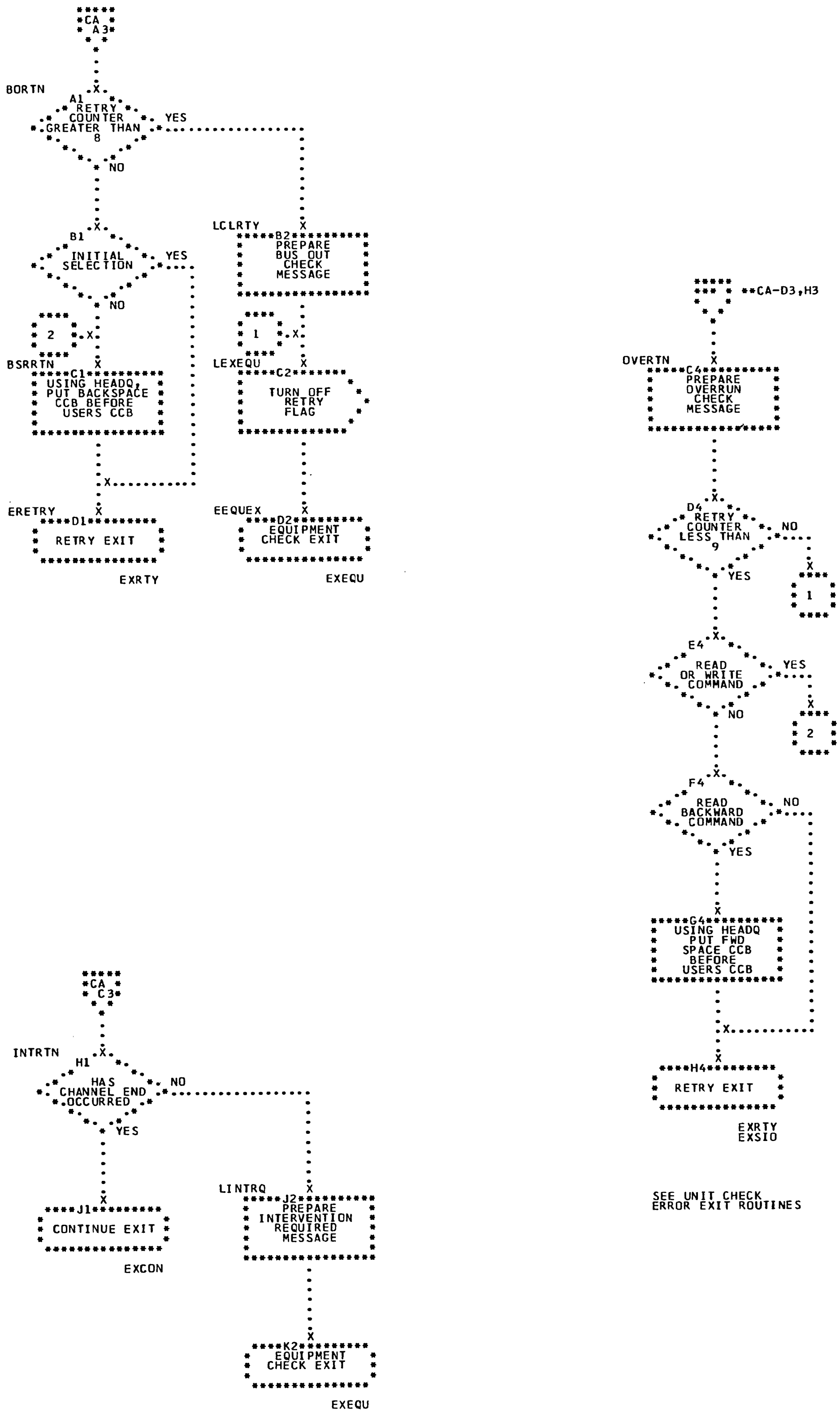


Chart CB. Tape Error: Bus Out Check, Intervention Required, Overrun, Chaining Check

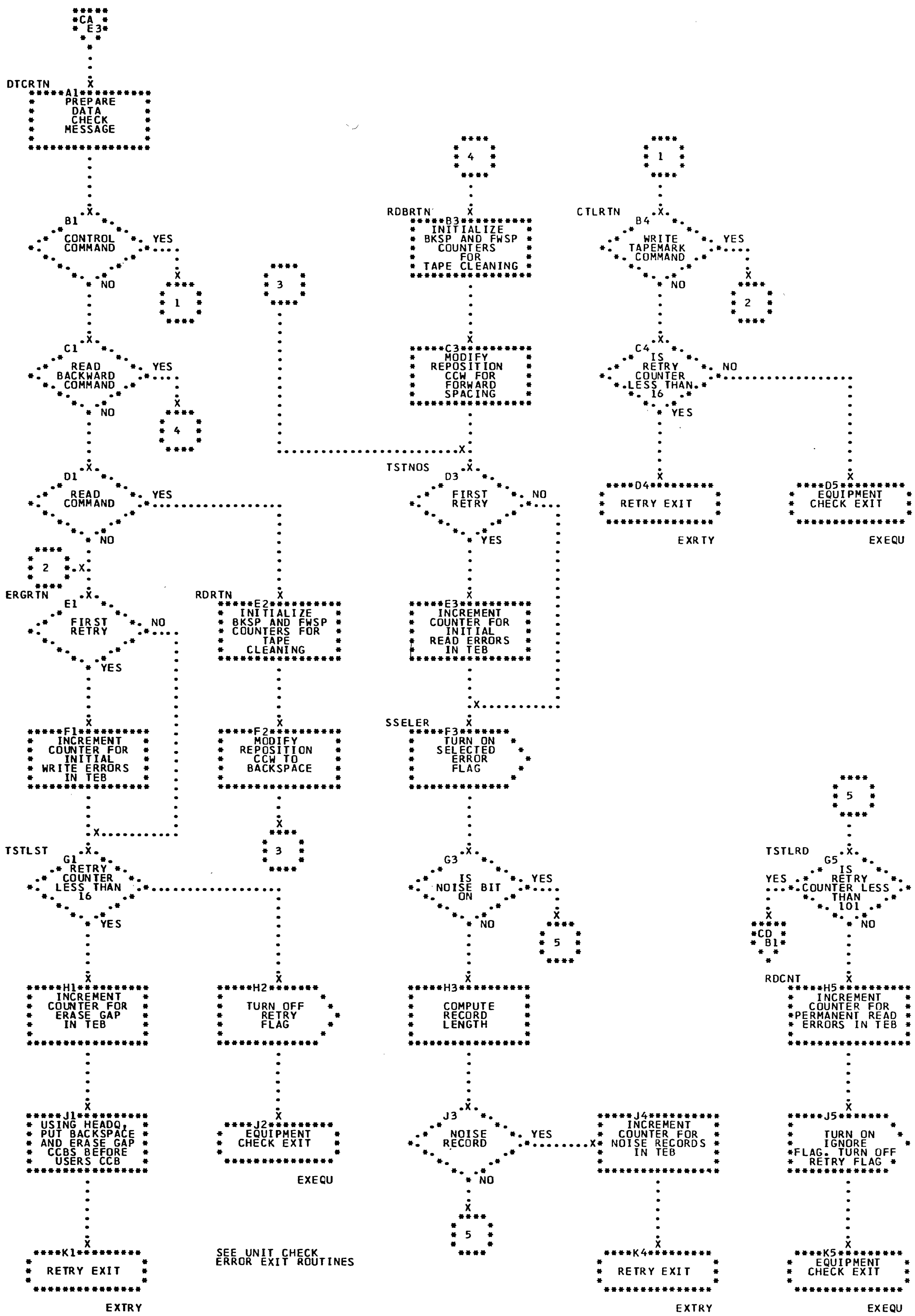


Chart CC. Tape Error: Data Check

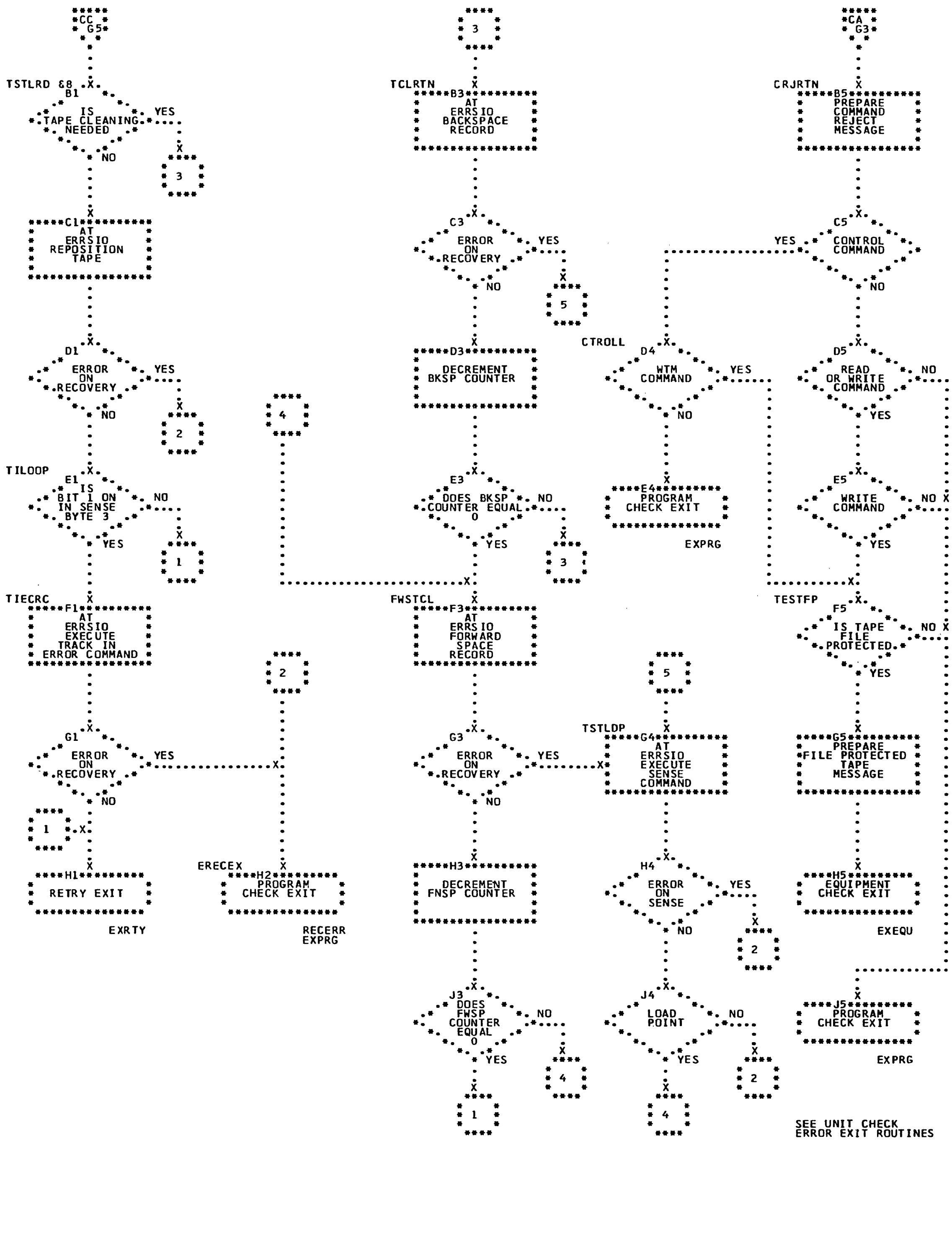


Chart CD. Tape Error: Command Reject

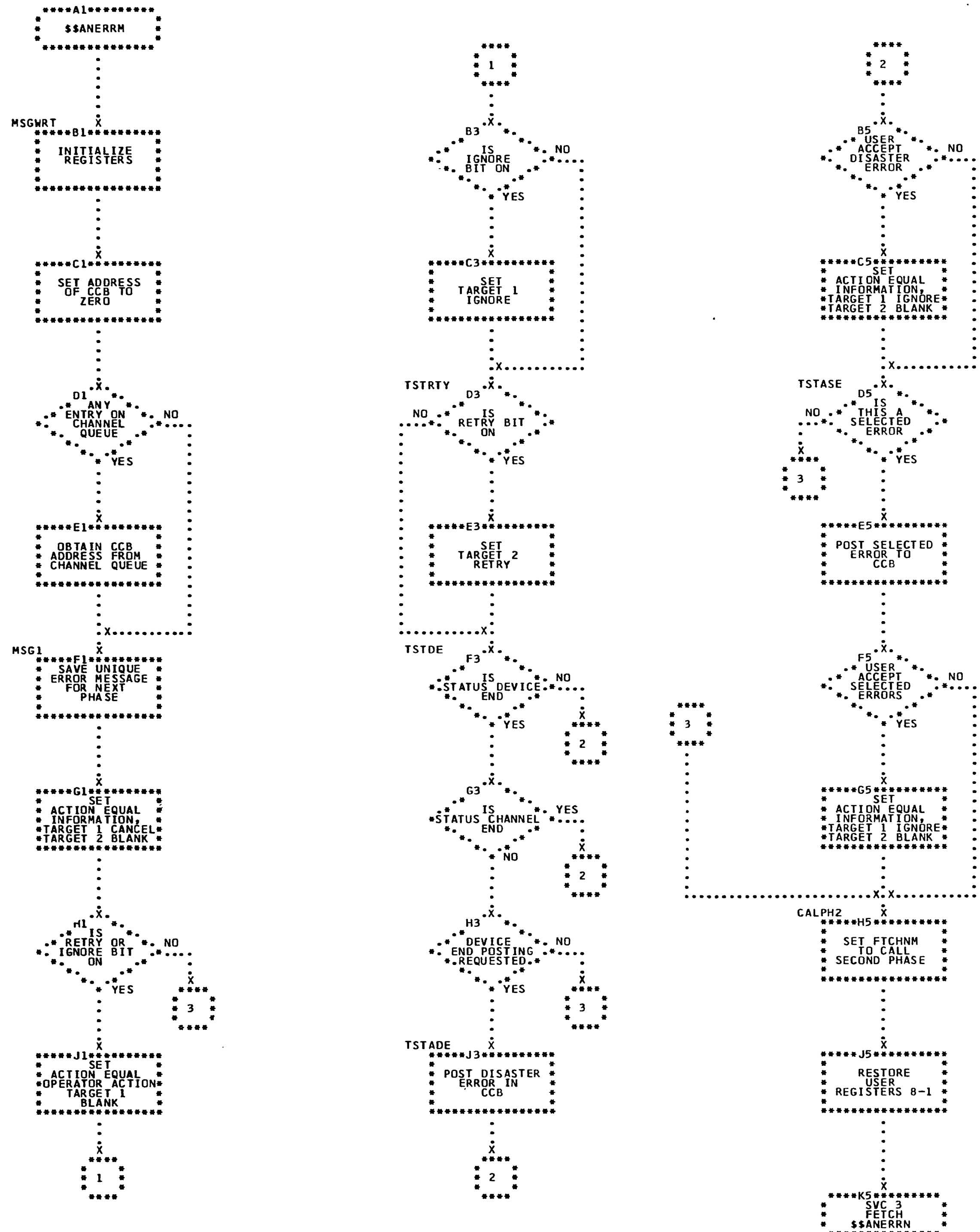


Chart CE. Message Writer: $$ANERRM

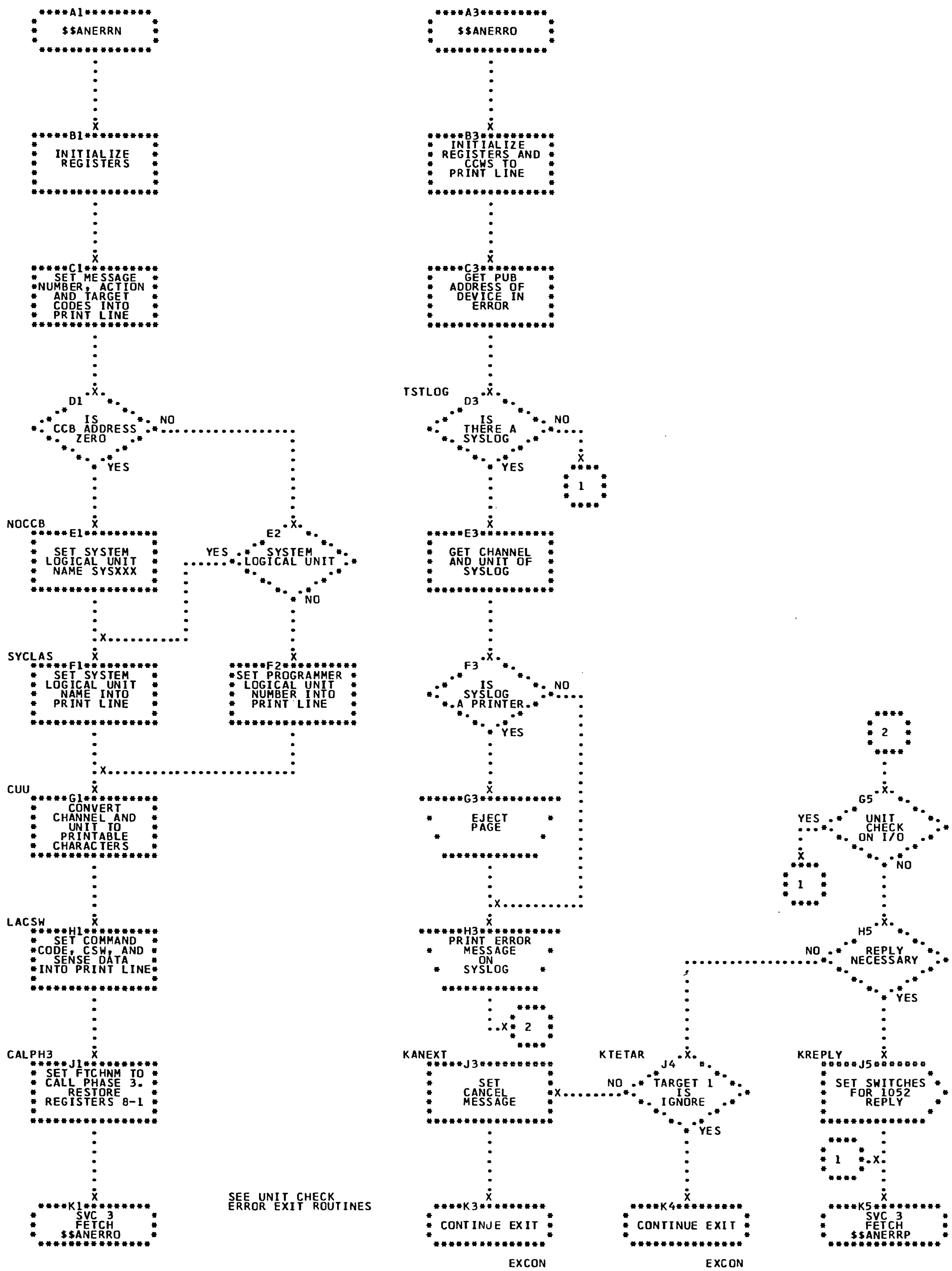


Chart CF. Message Writer: $$ANERRN, $$ANERRO

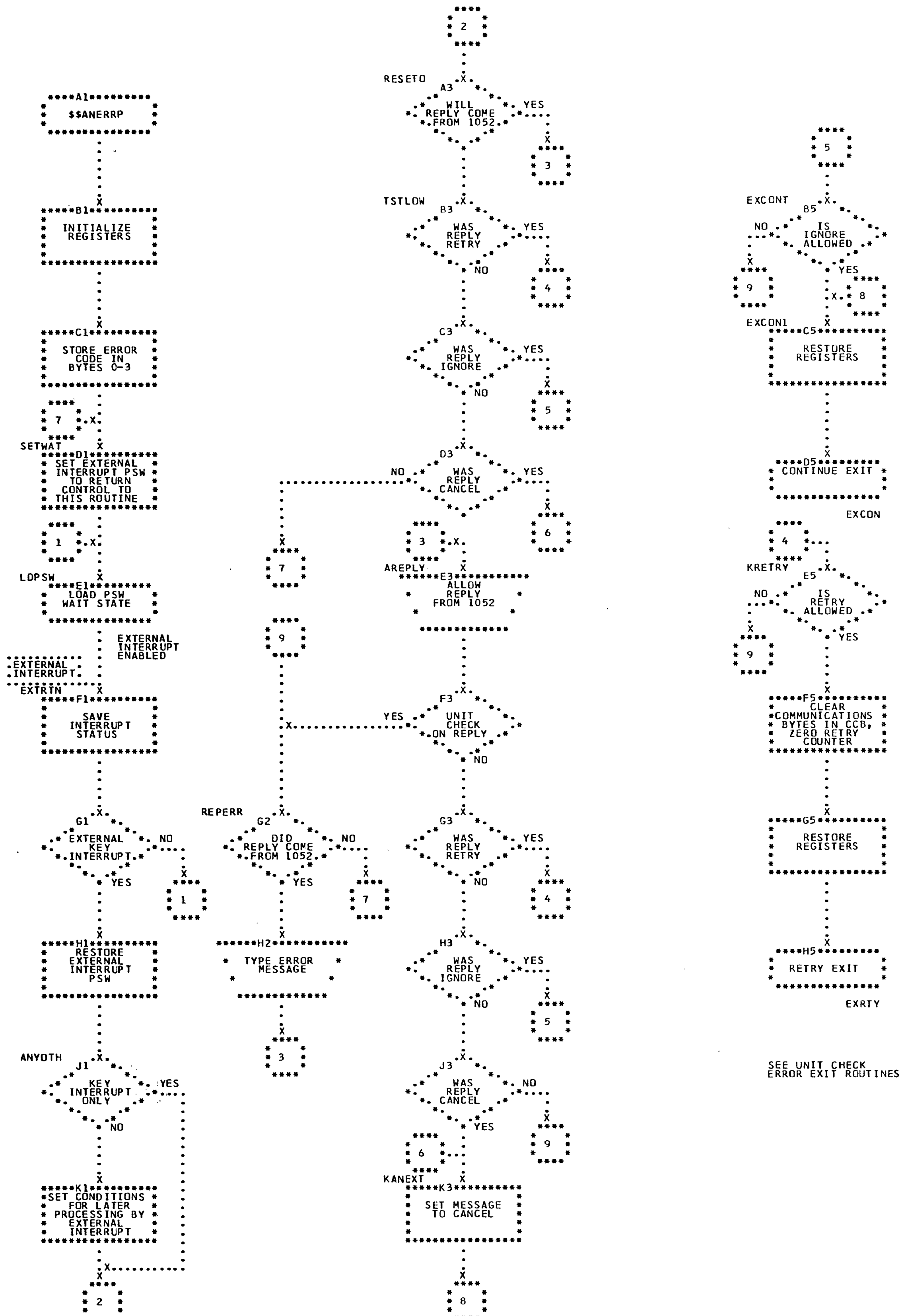


Chart CG. Message Writer: $$ANERRP

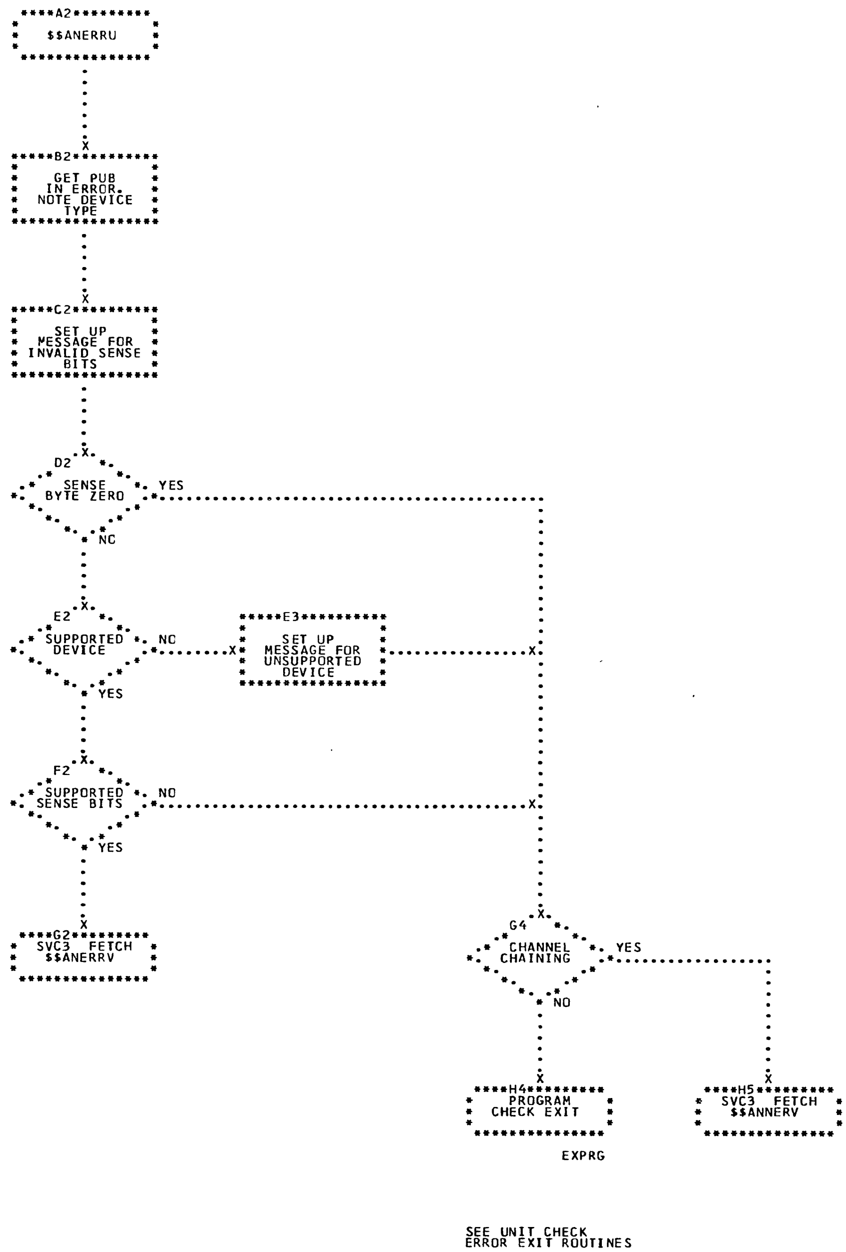


Chart CH. Device Error Recovery: $$ANERRU

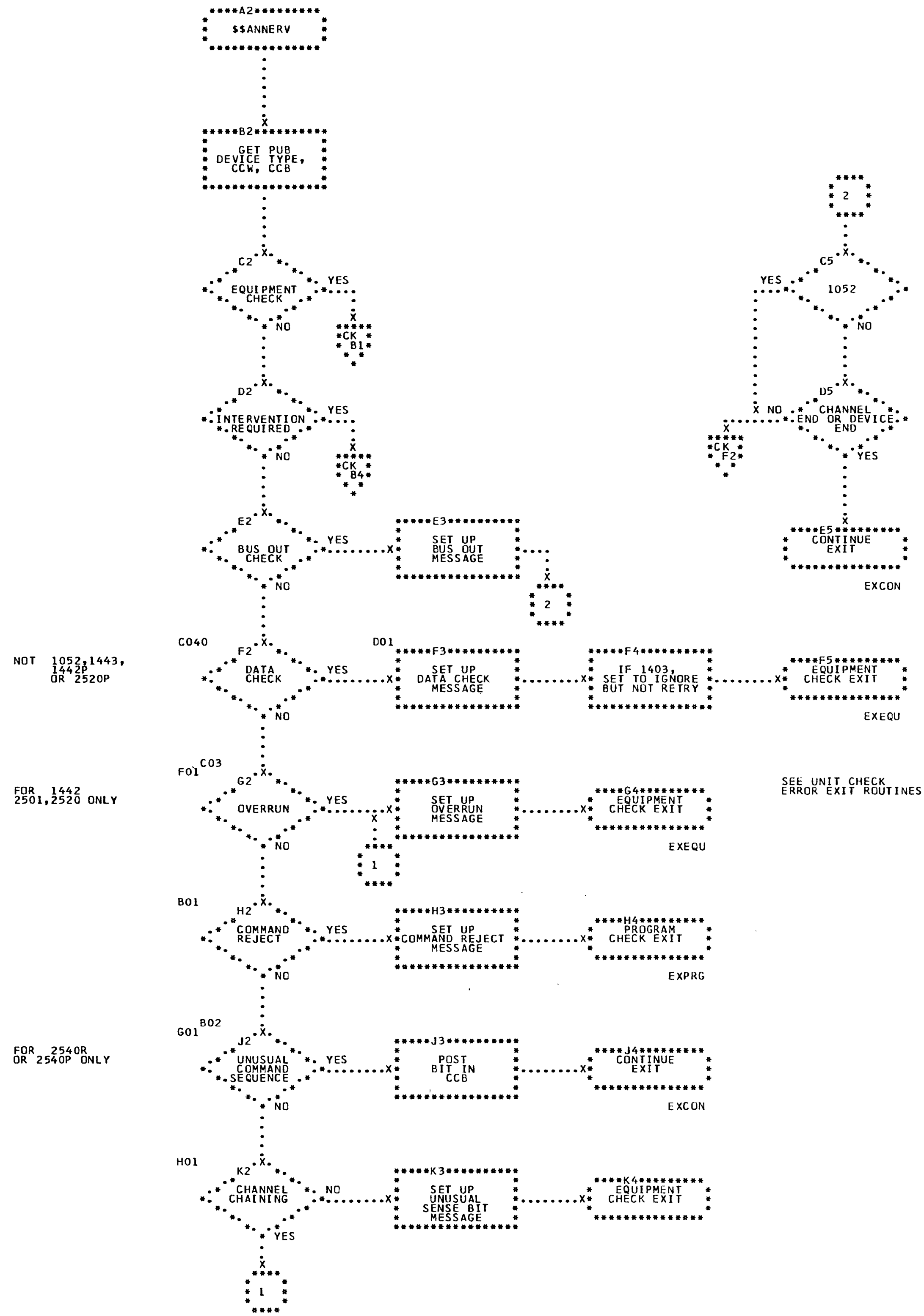


Chart CJ. Device Error: $$ANERRV (Part 1 of 2)

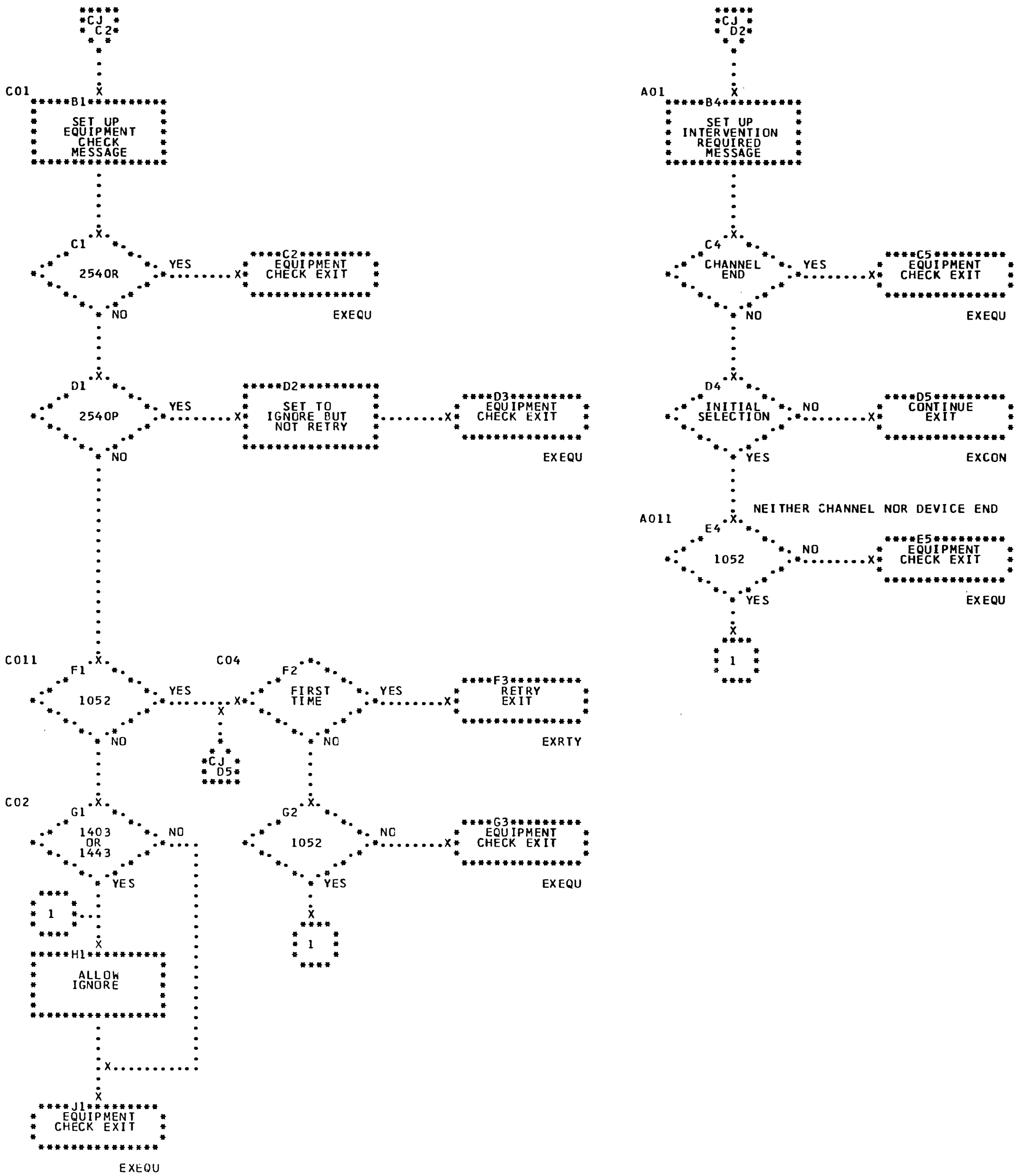


Chart CK. Device Error: $$ANERRV (Part 2 of 2)

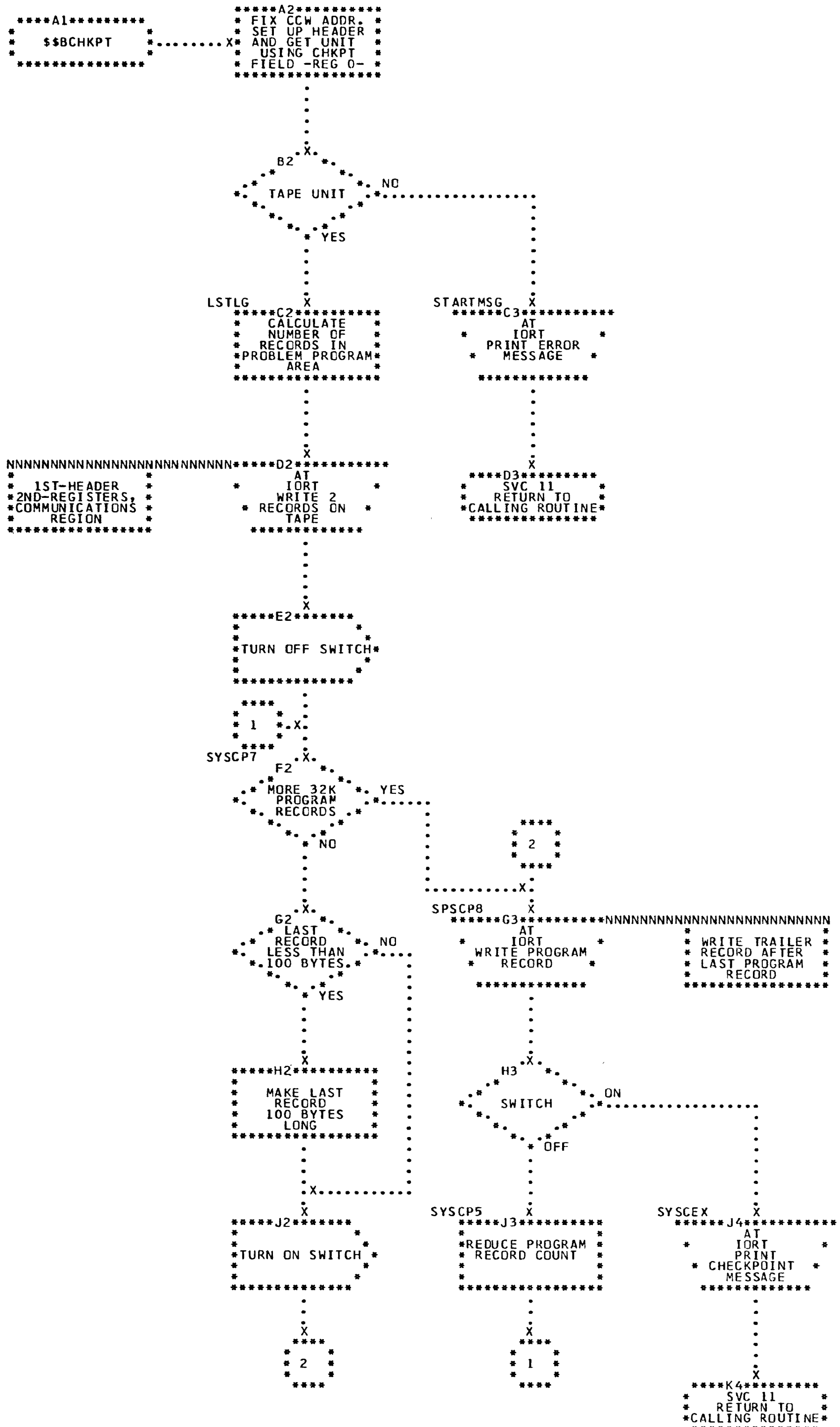


Chart DA. Checkpoint: $$BCHKPT

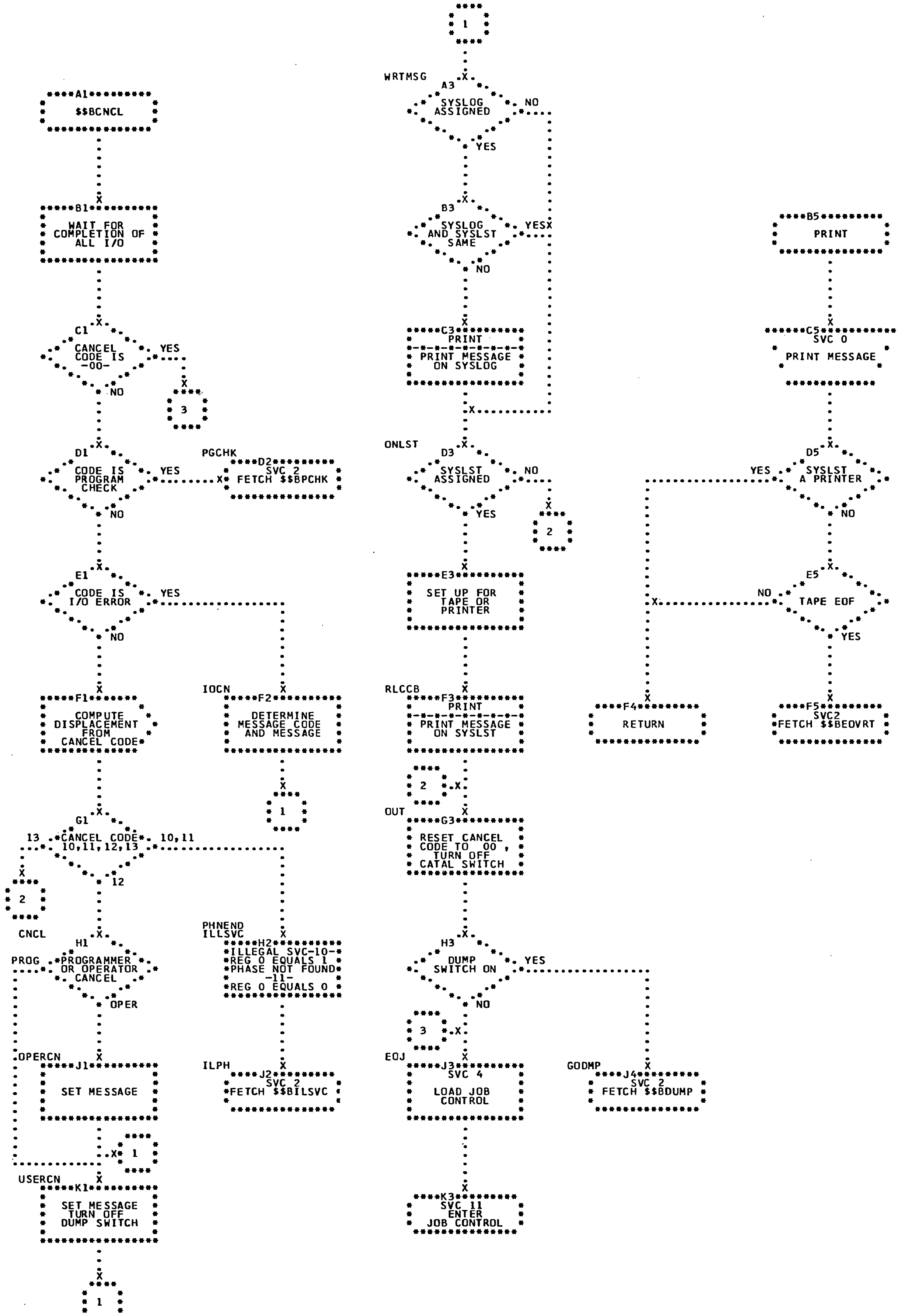


Chart DB. Cancel: $$BCNCL

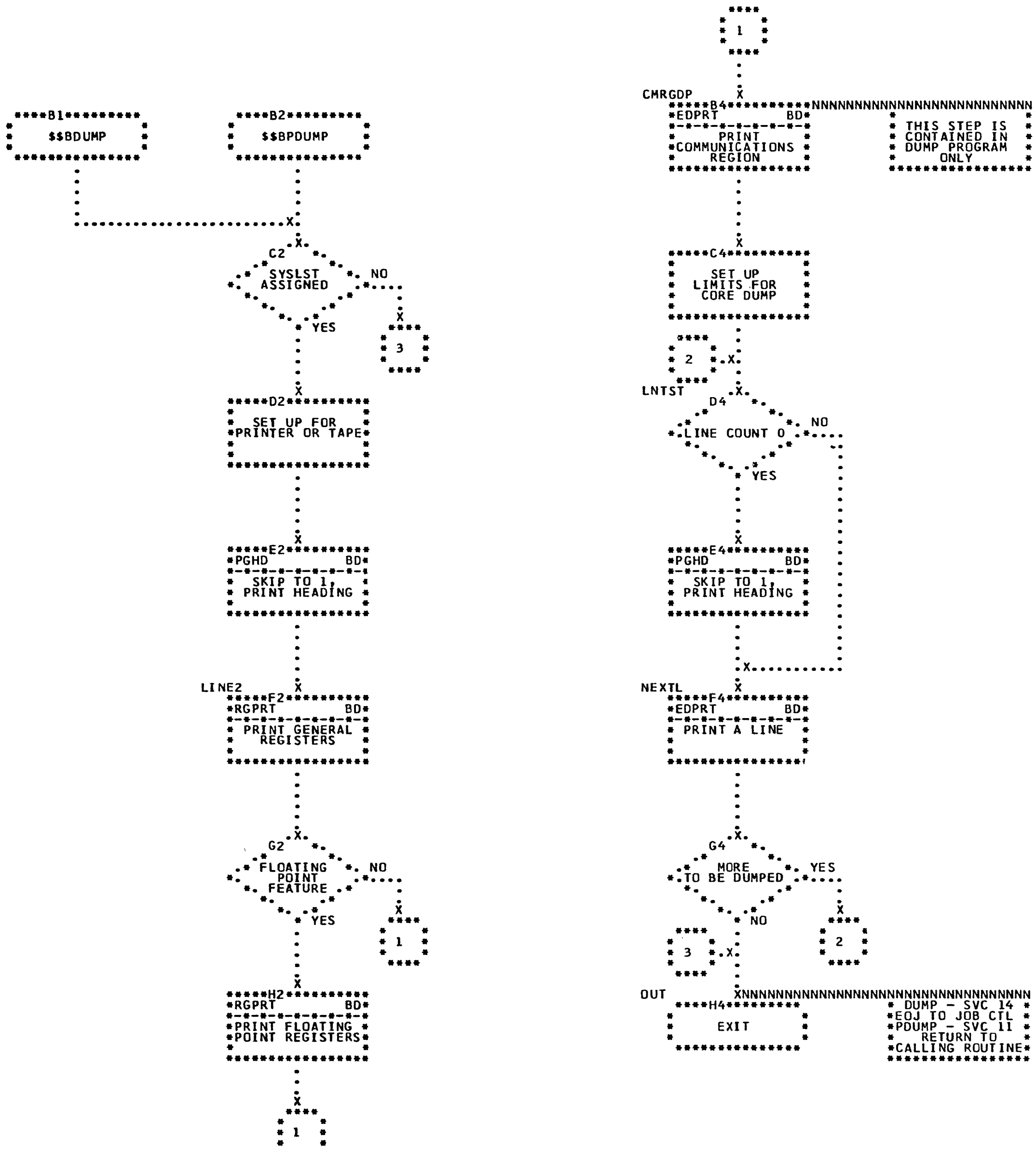


Chart DC. Dump: $$BDUMP, $$BPDUMP

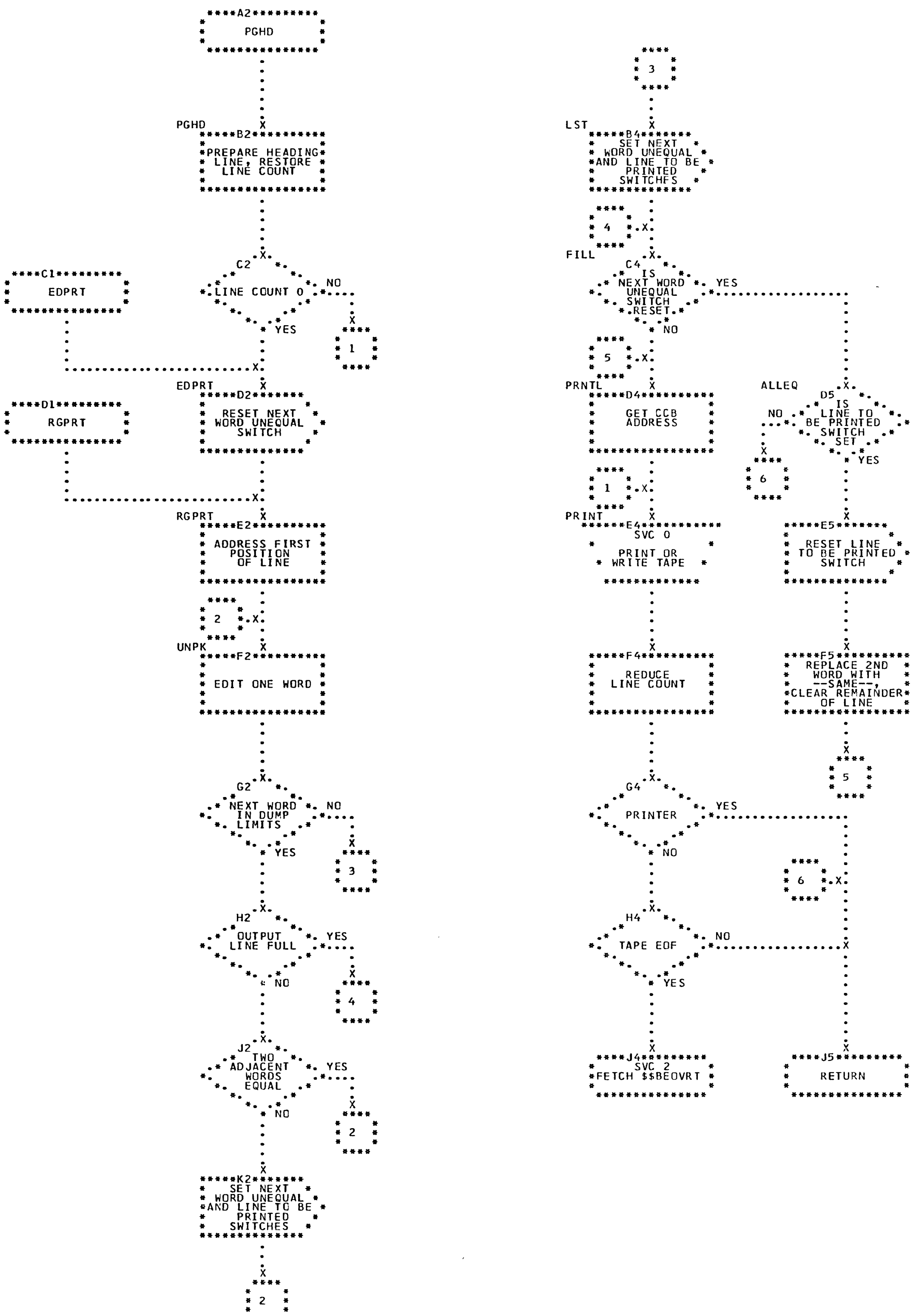


Chart DD. Dump Subroutines

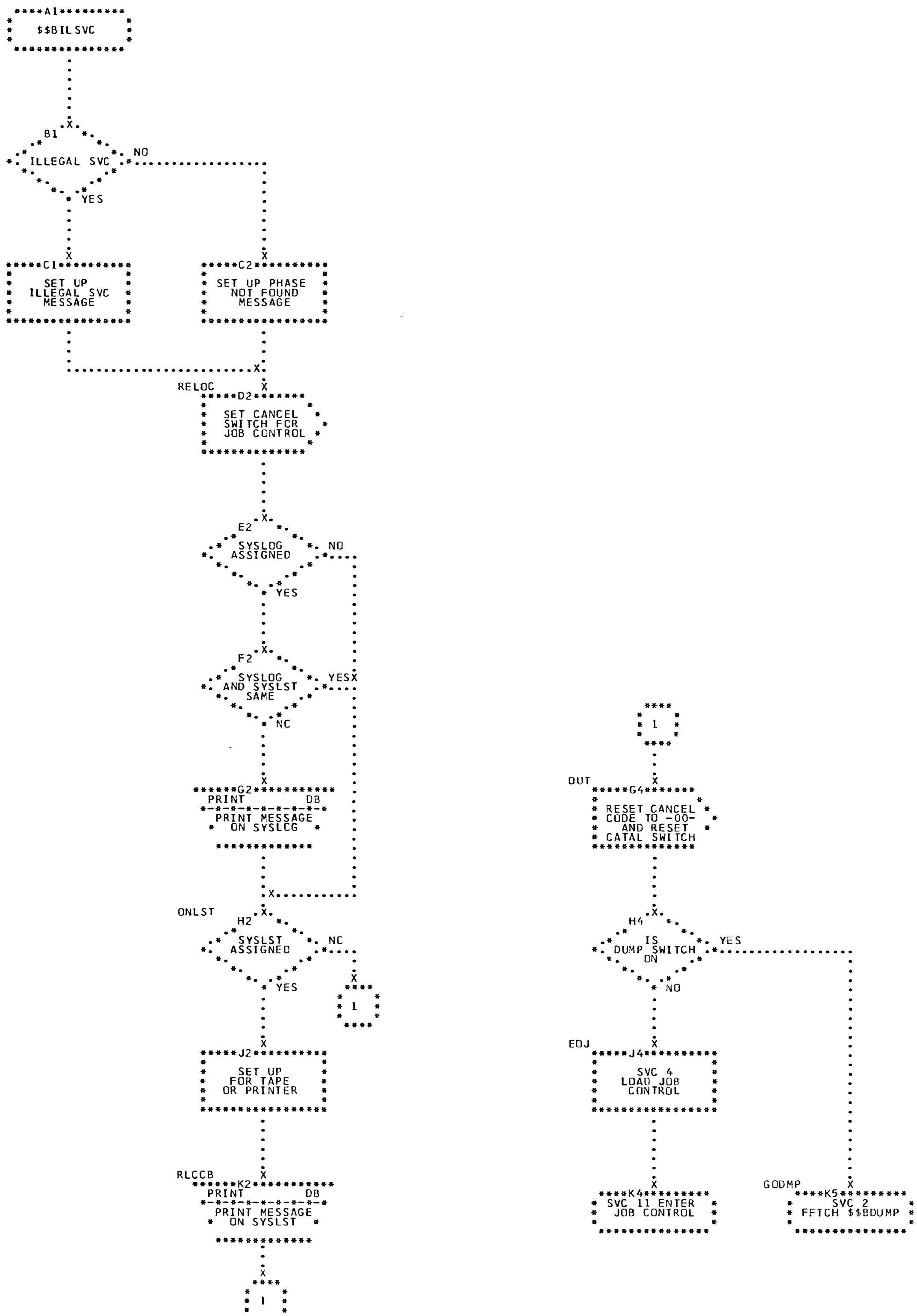


Chart DE. Illegal SVC: $$BILSVC

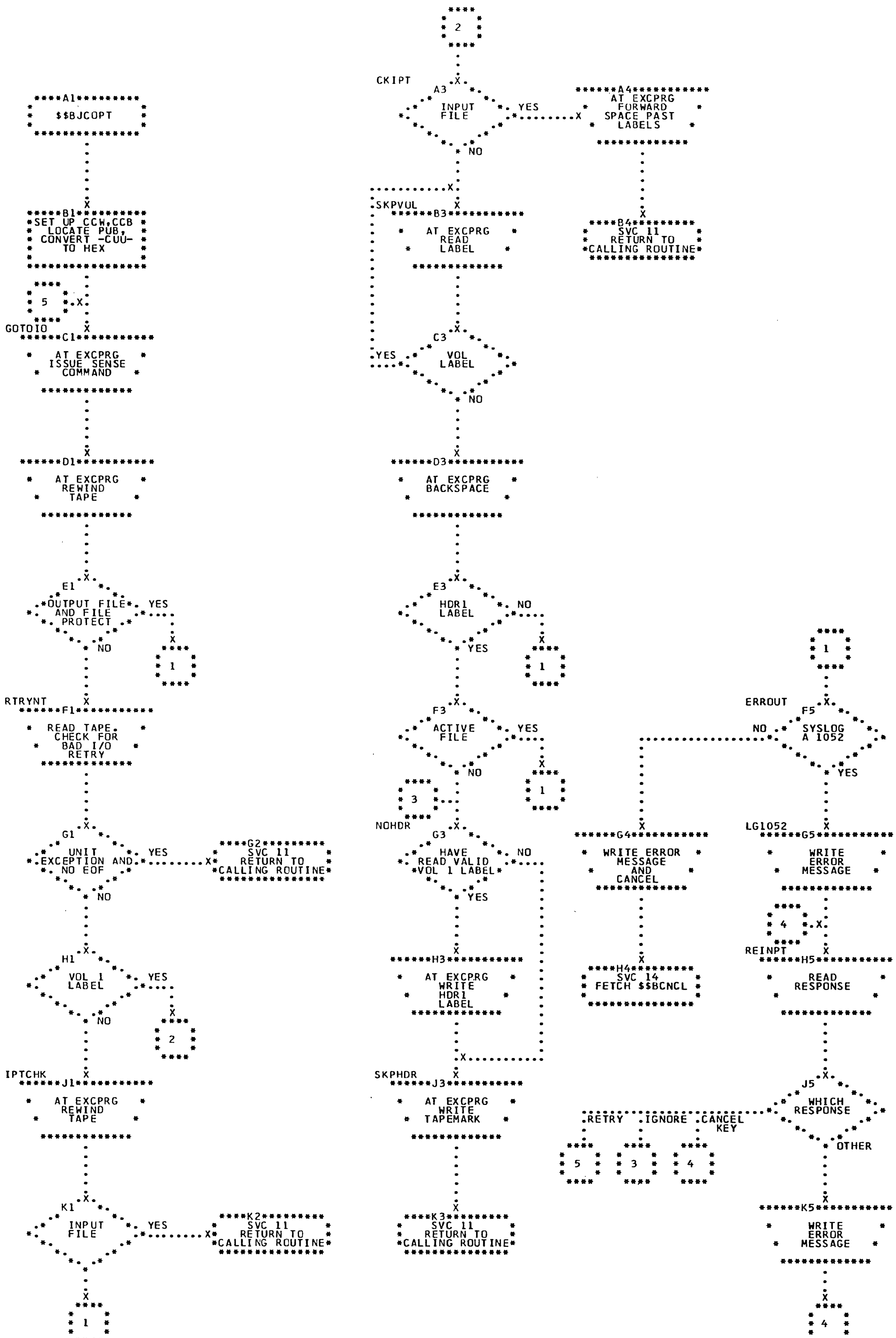


Chart DF. Job Control Open: $$BJCOPT

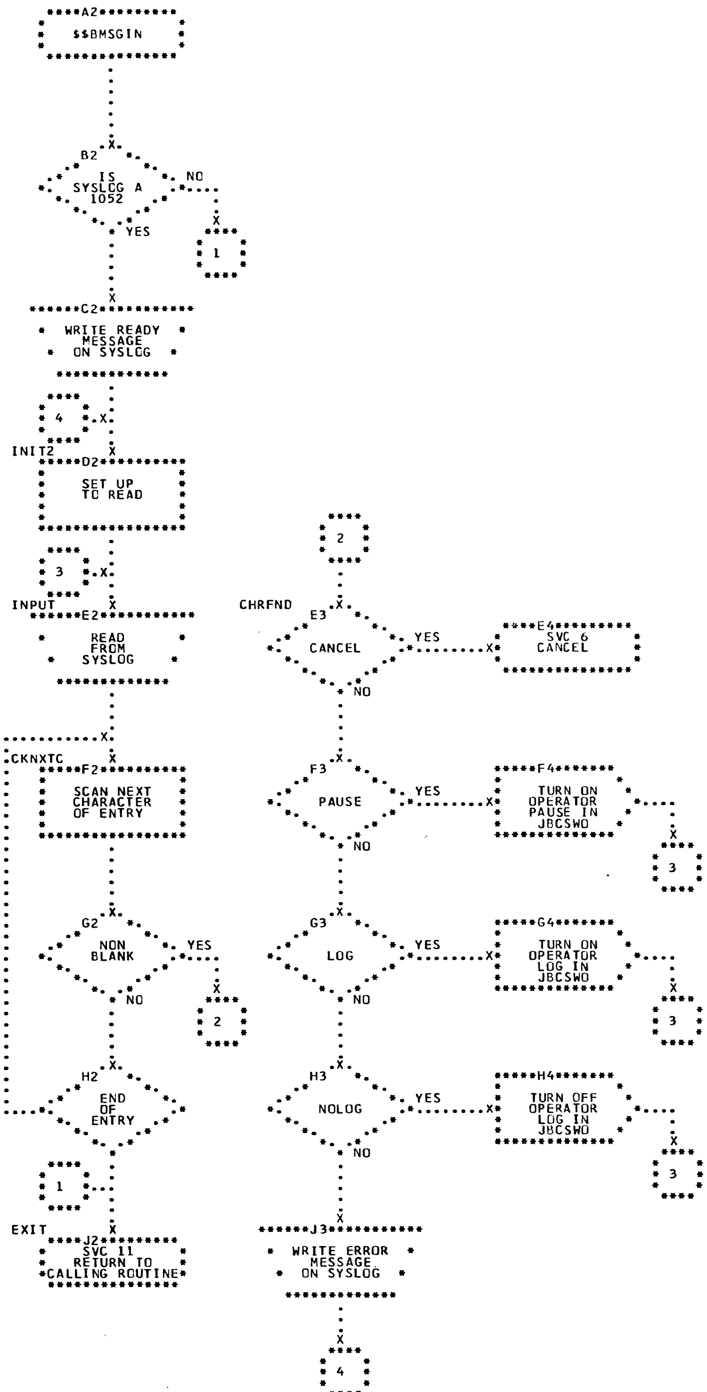


Chart DG. Message Input: $$BMSGIN

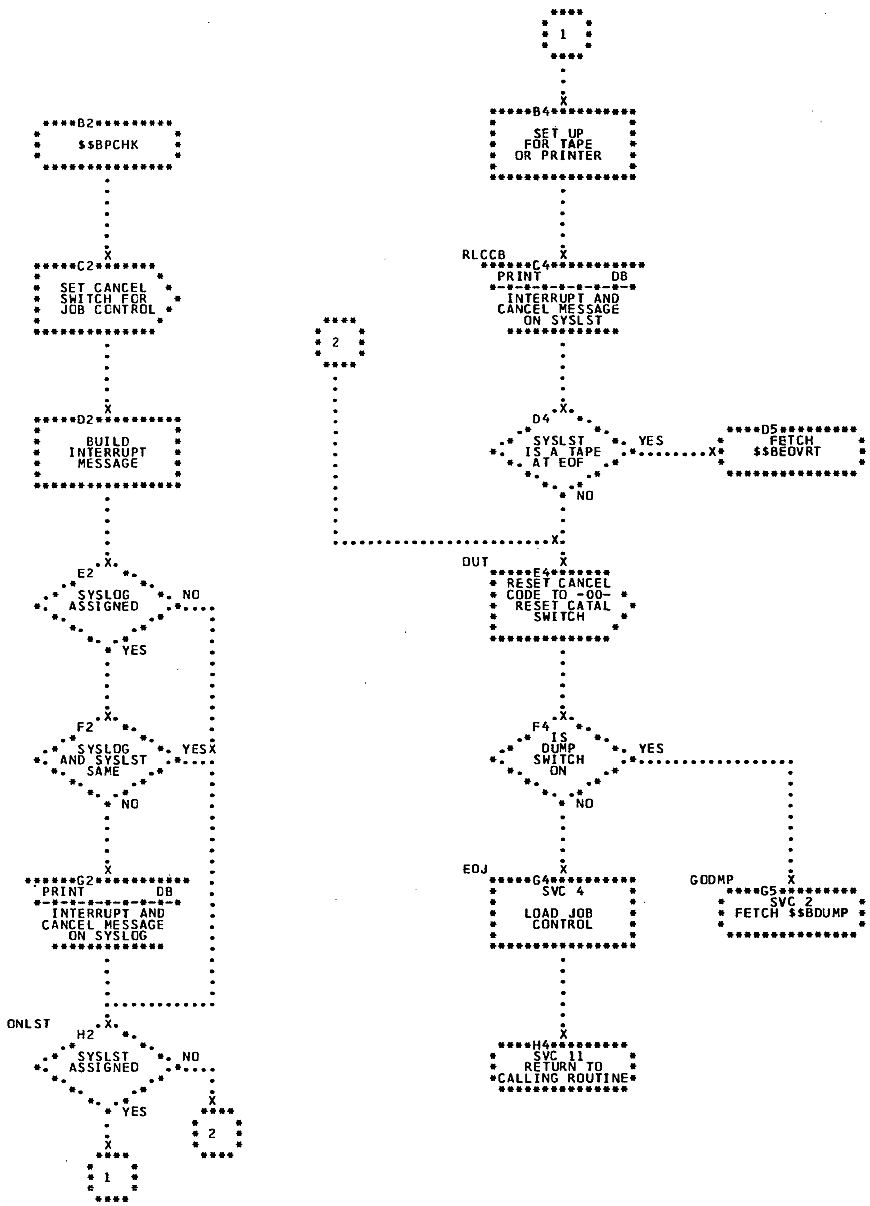


Chart DH. Program Check: $$BPCHK

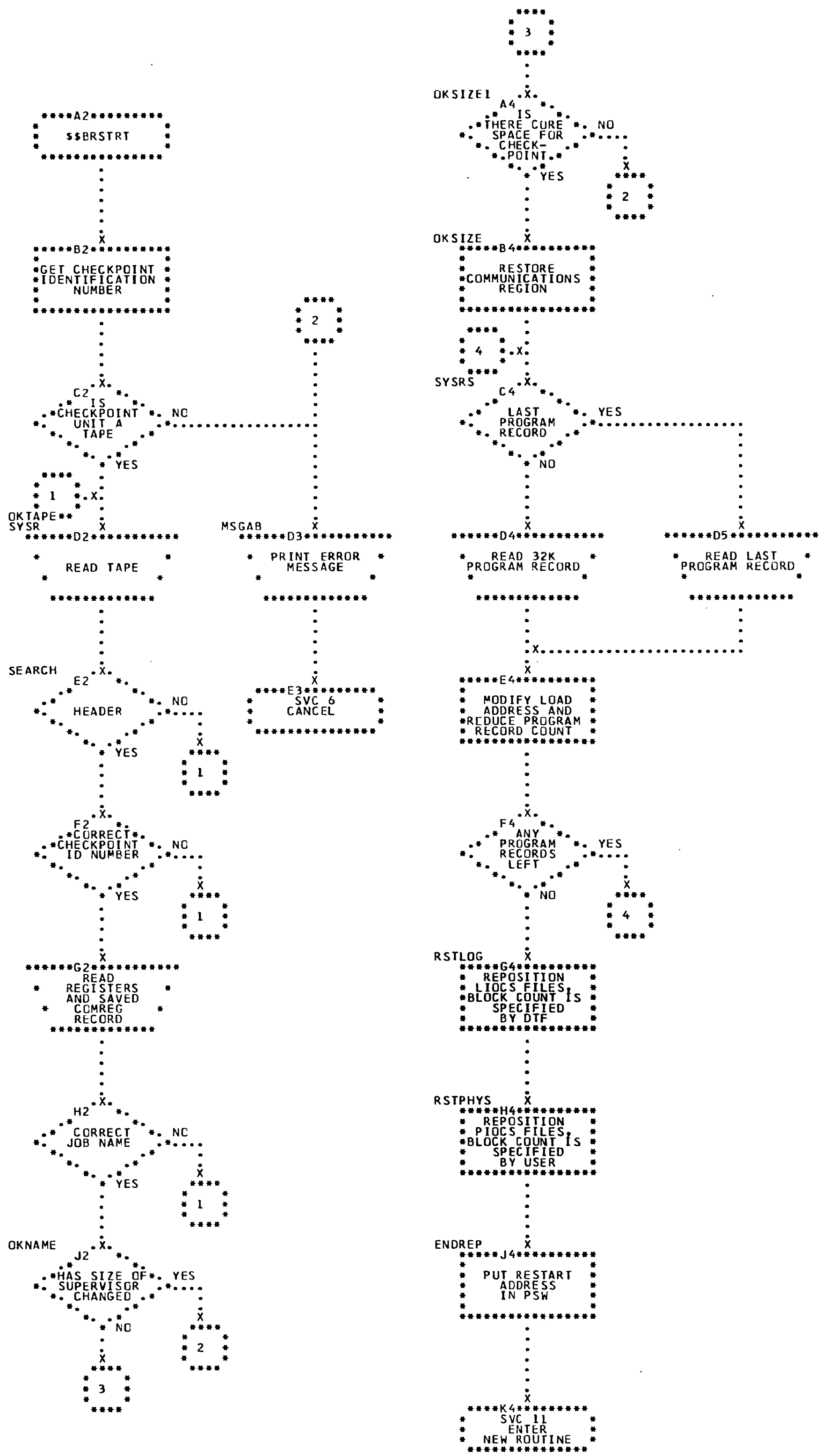


Chart DJ. Restart: $$BRSTRT

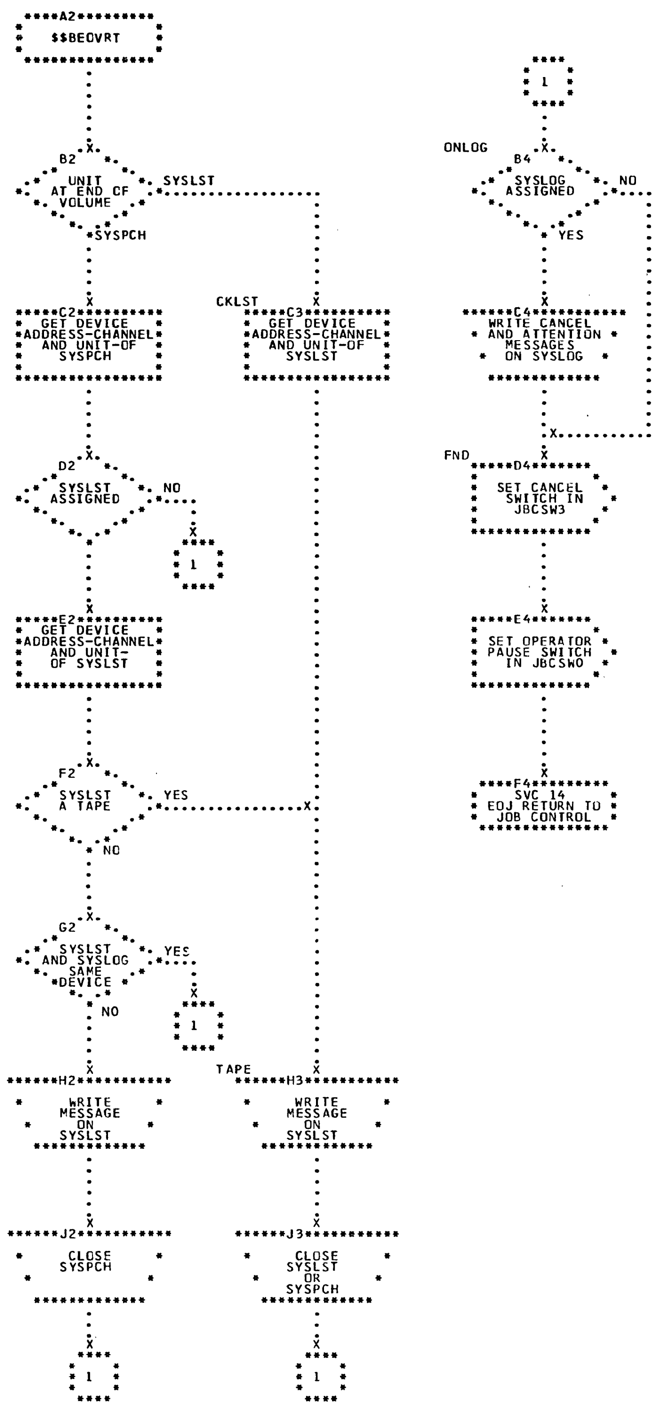


Chart DK. End of Volume: $$BEOVRT

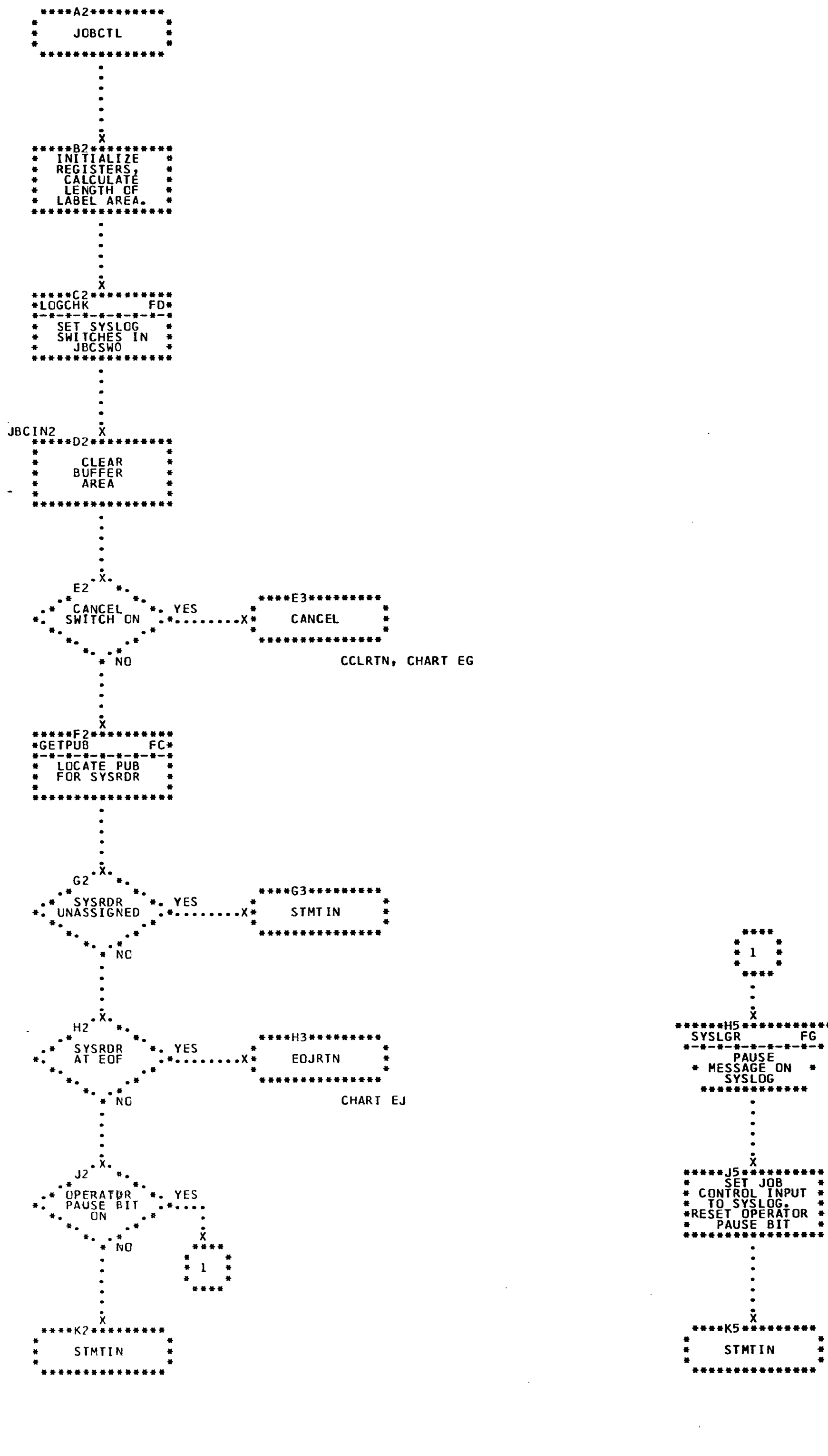


Chart EA. Initialization

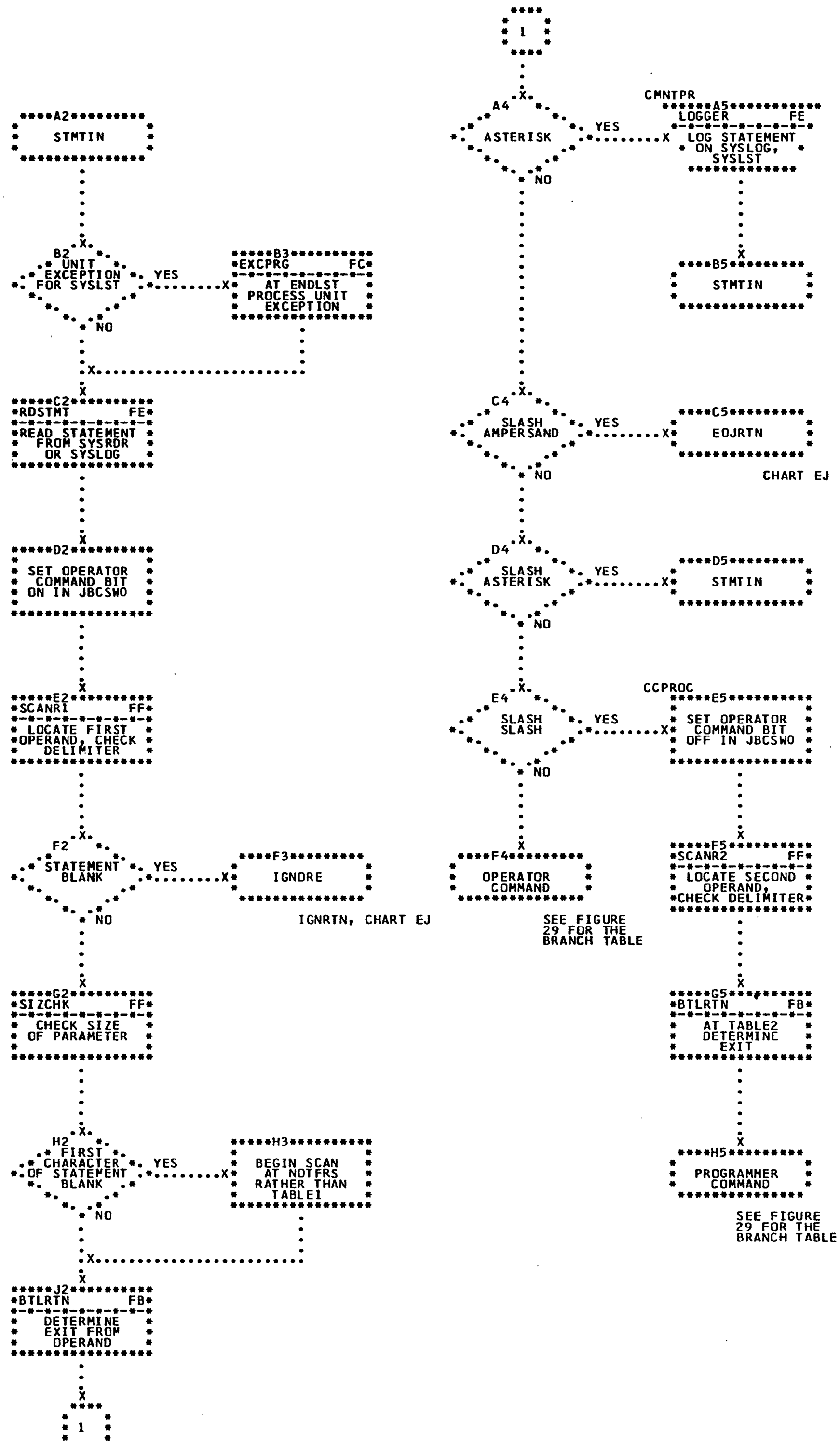


Chart EB. Input

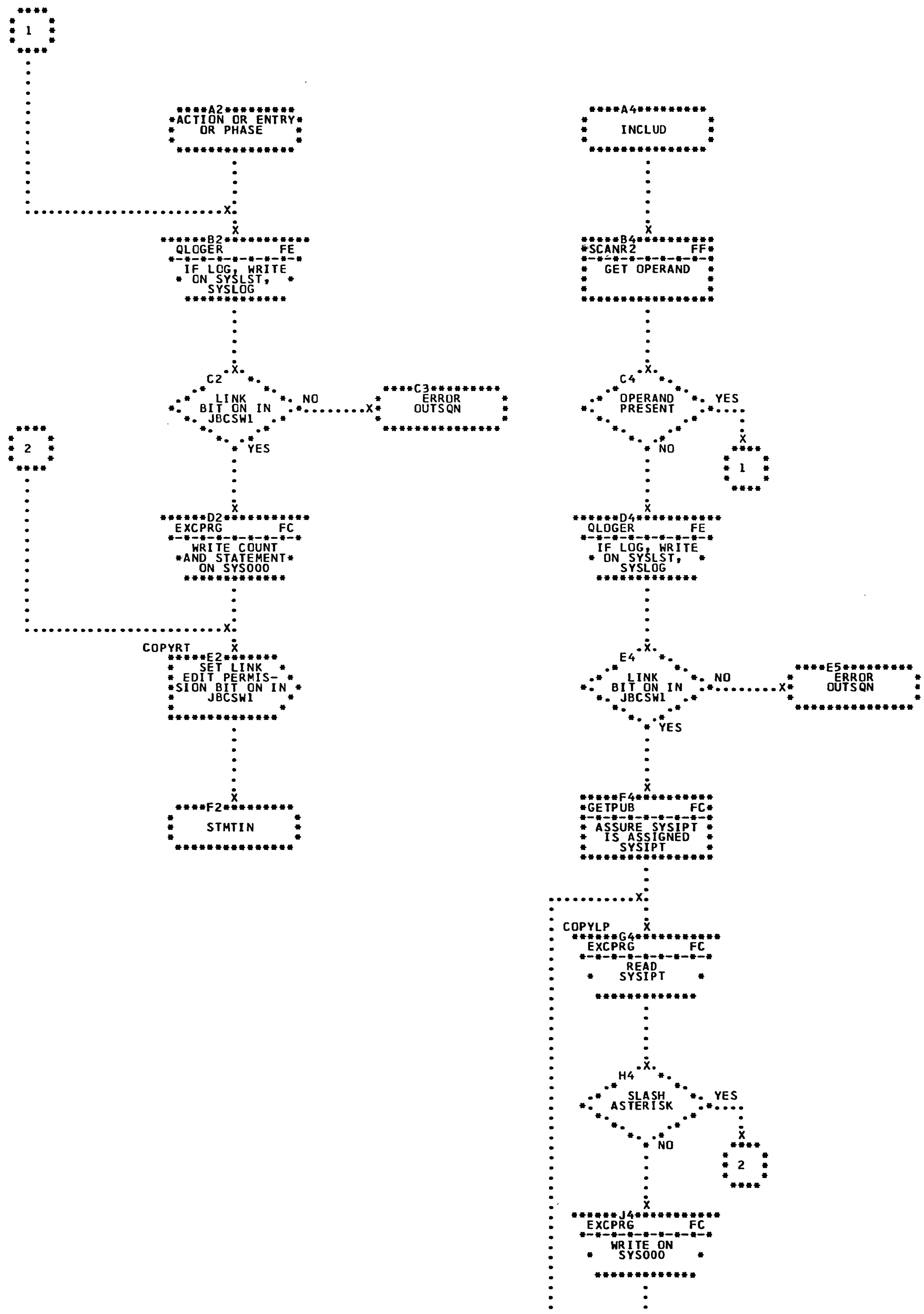


Chart EC. ACTION, ENTRY, PHASE, INCLUD

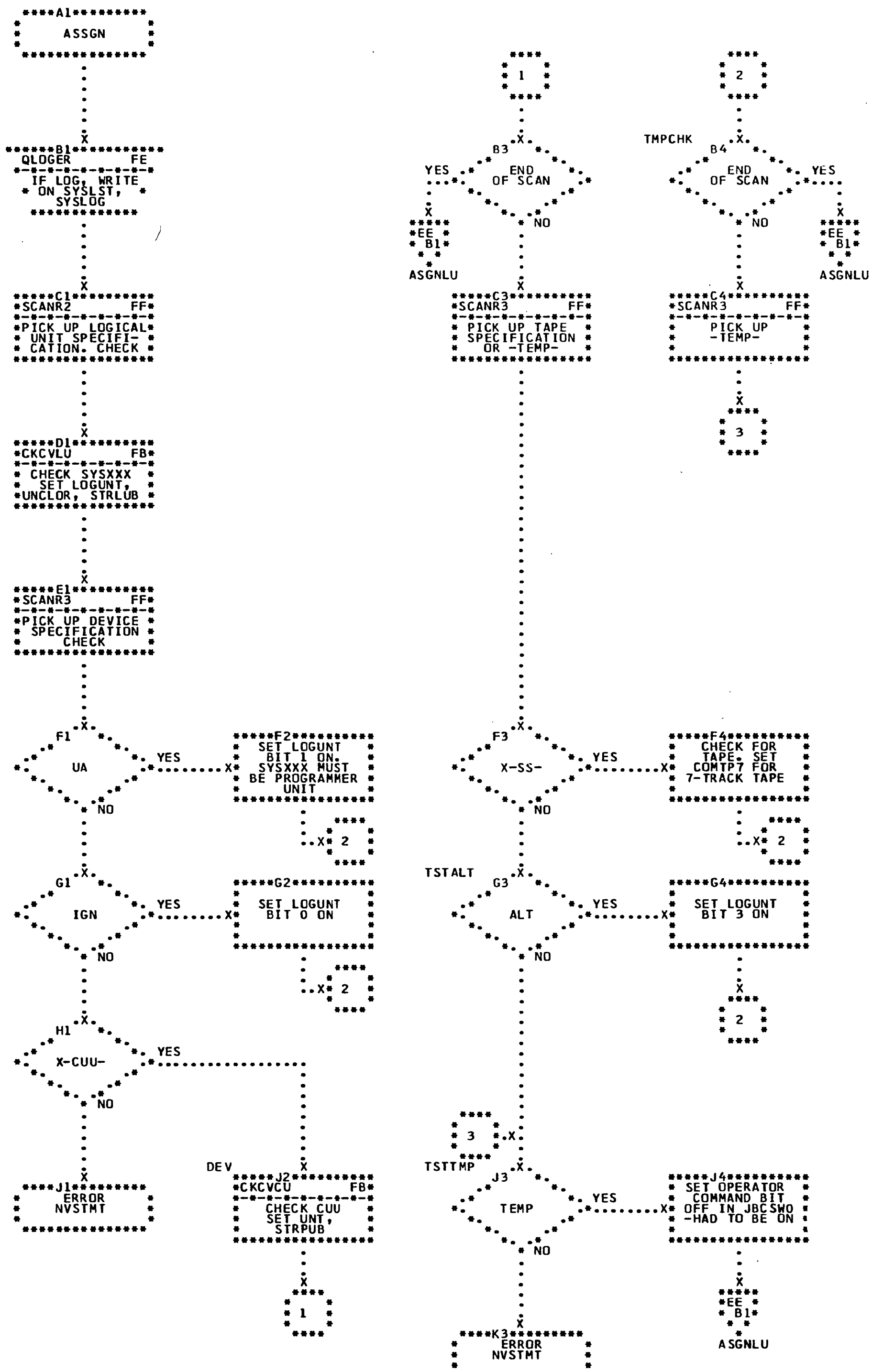


Chart ED. ASSGN: Analyze Operands

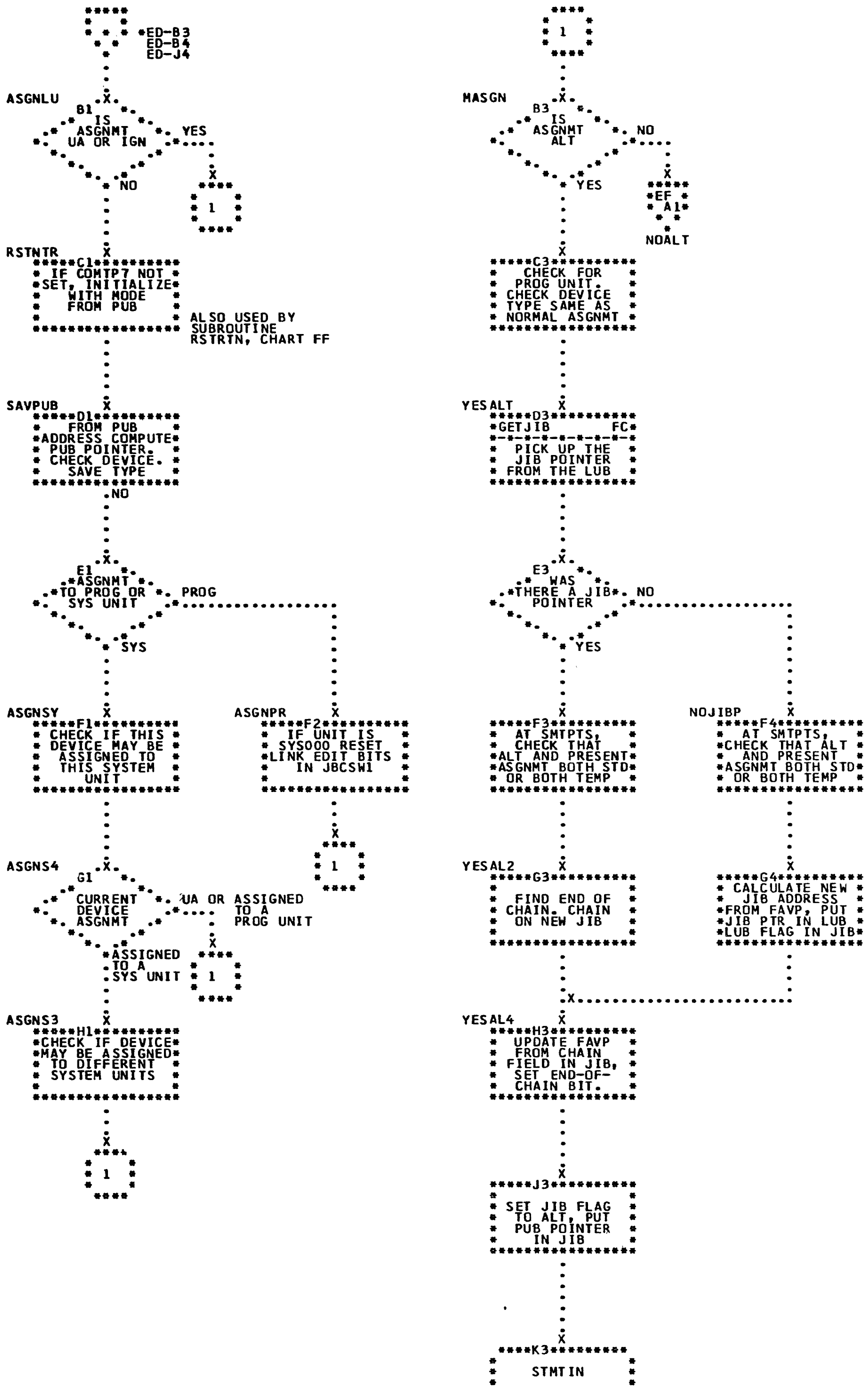


Chart EE. ASSGN: Check Device Type: Make Alternate Assignment

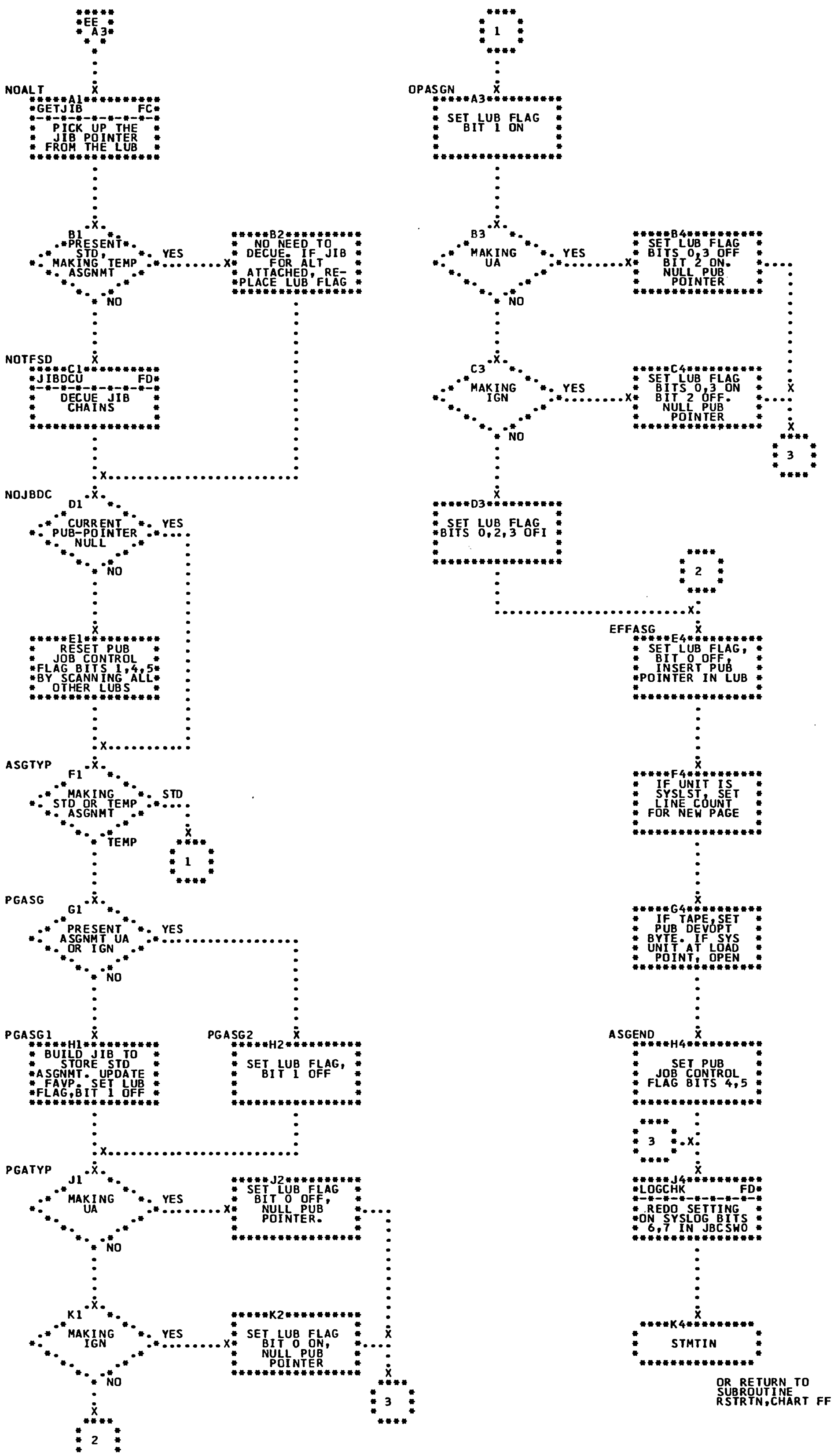


Chart EF. ASSGN: Make Normal Assignment

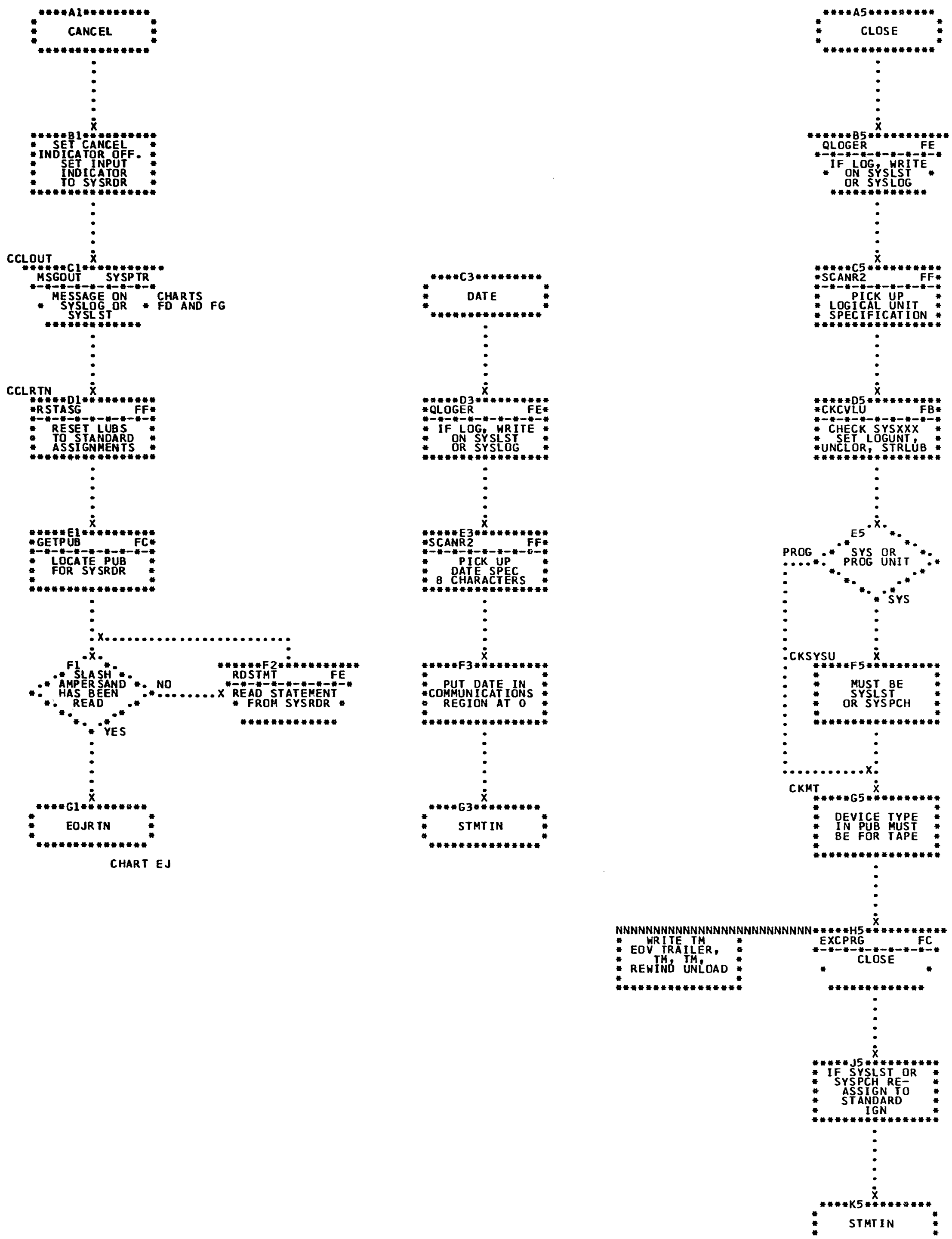


Chart EG. CANCEL, CLOSE, DATE

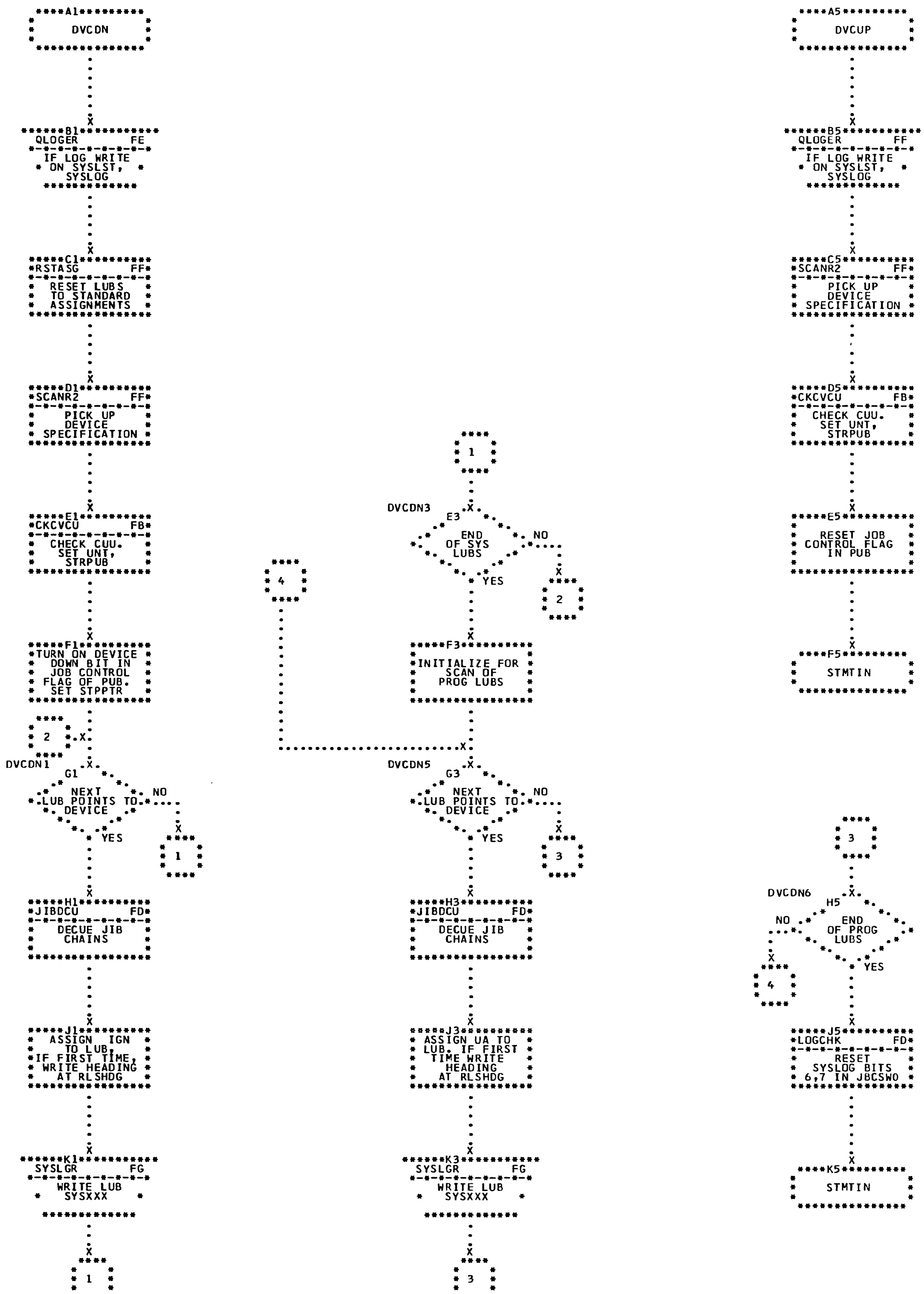


Chart EH. DVCDN, DVCUP

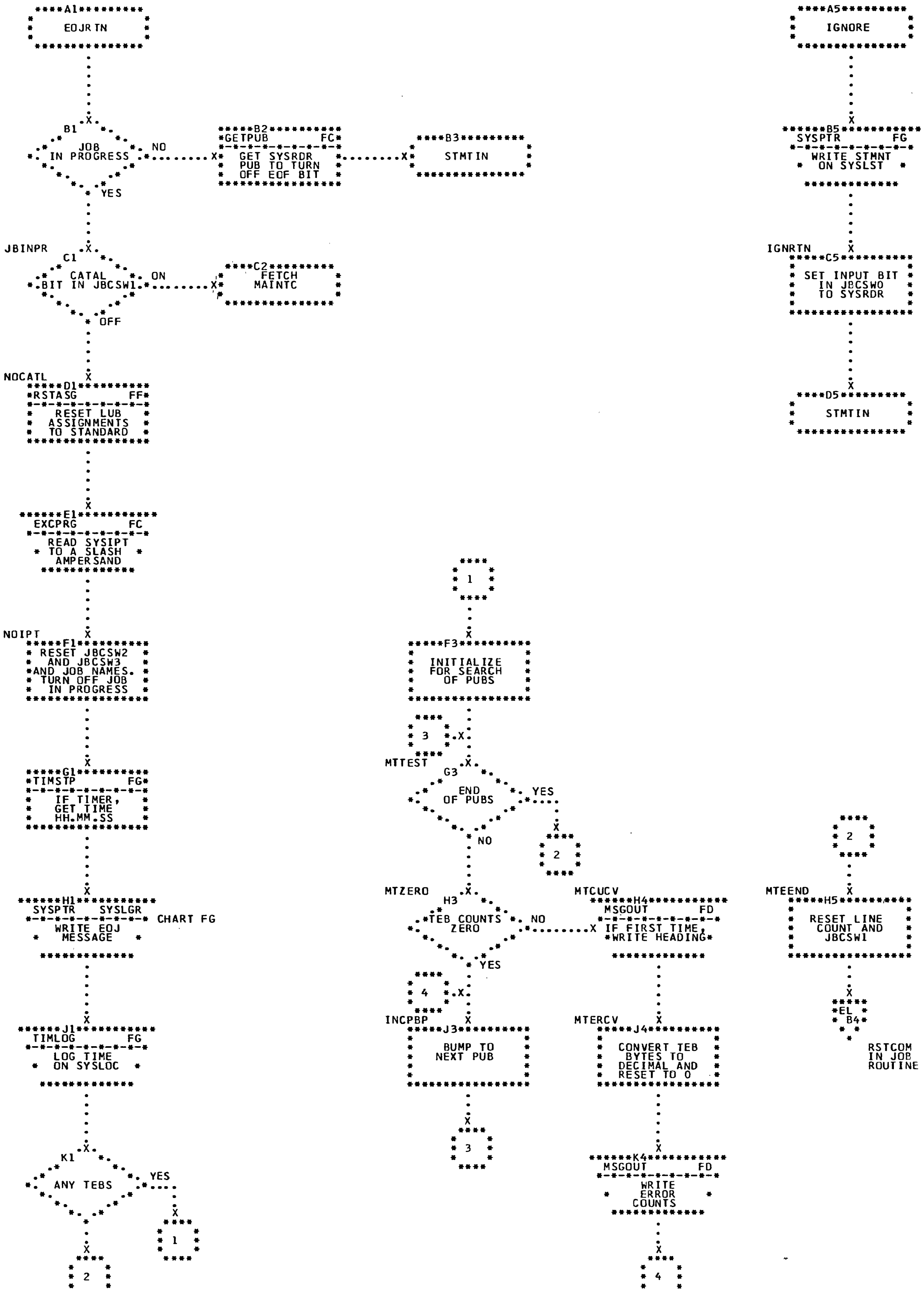


Chart EJ. End of Job

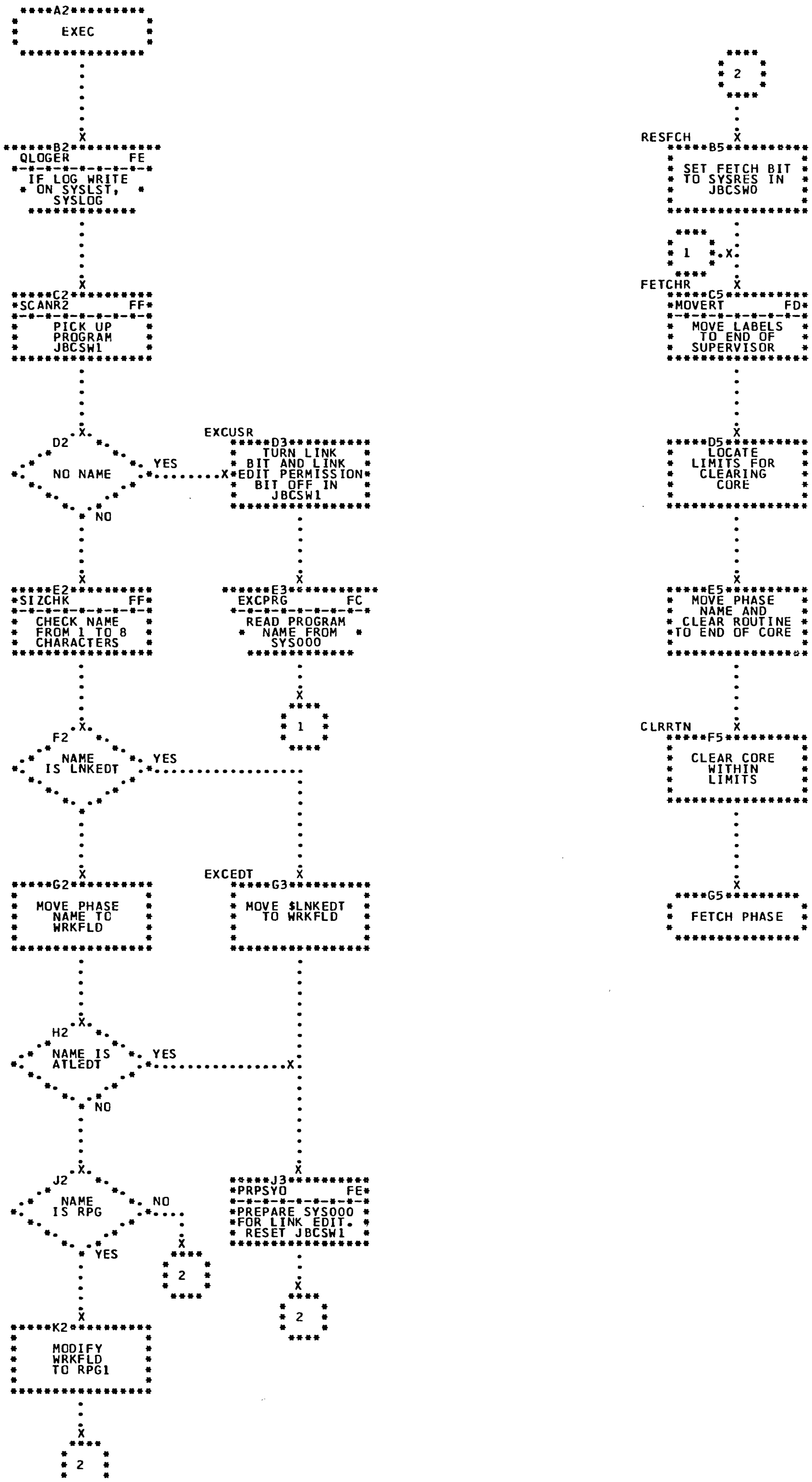


Chart EK. EXEC; Fetch

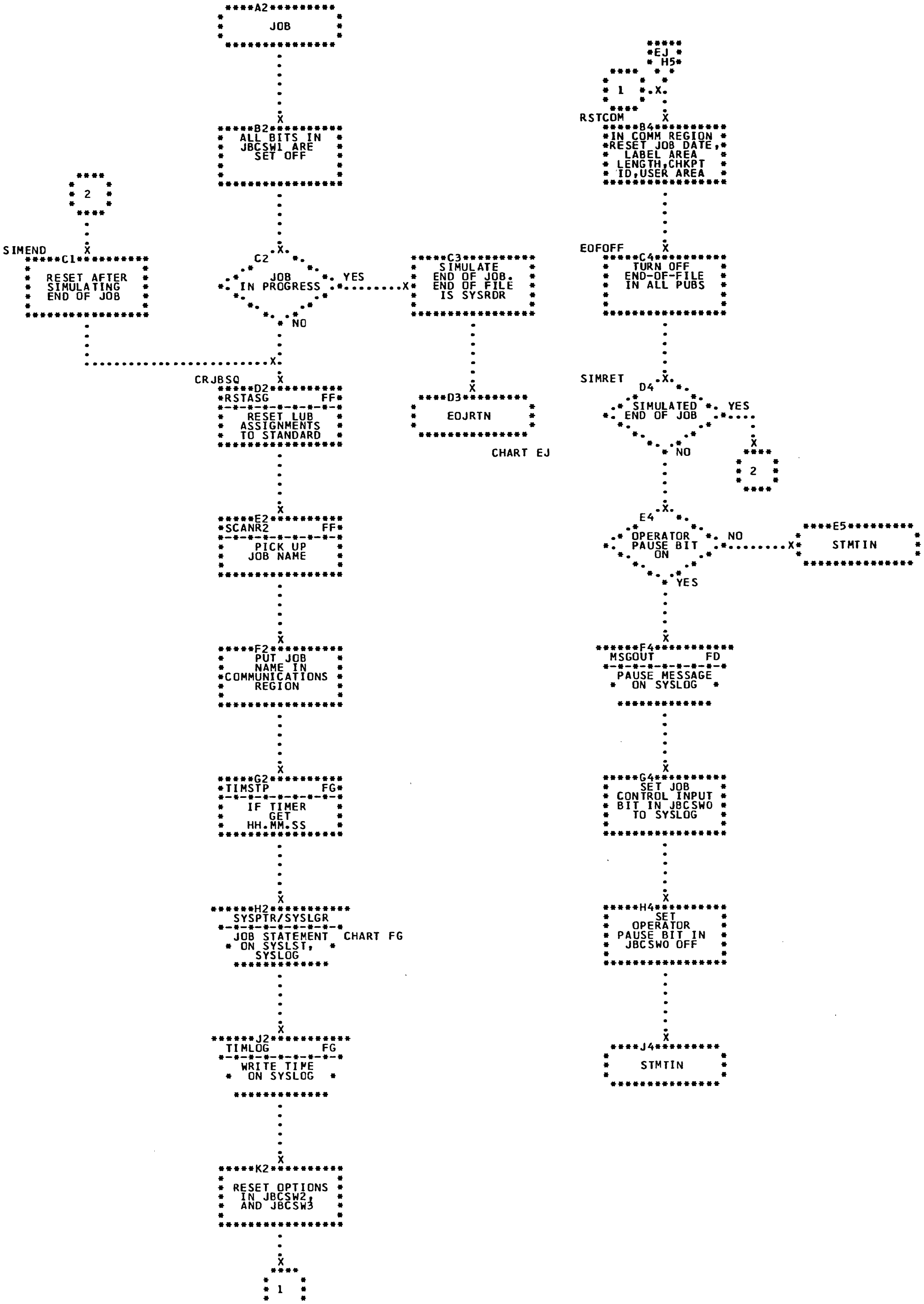

A2
JOB
B2
ALL BITS IN JBCSW1 ARE SET OFF
2
SIMEND
C1
RESET AFTER SIMULATING END OF JOB
C2
JOB IN PROGRESS
YES
NO
C3
SIMULATE END OF JOB. END OF FILE IS SYSRDR
CRJBSQ
D2
RSTASG FF
RESET LUB ASSIGNMENTS TO STANDARD
D3
EOJRTN
CHART EJ
E2
SCANR2 FF
PICK UP JOB NAME
F2
PUT JOB NAME IN COMMUNICATIONS REGION
G2
TIMSTP FG
IF TIMER GET HH.MM.SS
H2
SYSPTR/SYSLGR
JOB STATEMENT ON SYSLST, SYSLOG
CHART FG
J2
TIMLOG FG
WRITE TIME ON SYSLOG
K2
RESET OPTIONS IN JBCSW2, AND JBCSW3
1
EJ
H5
1
RSTCOM
B4
IN COMM REGION RESET JOB DATE, LABEL AREA LENGTH, CHKPT ID, USER AREA
EOFOFF
C4
TURN OFF END-OF-FILE IN ALL PUBS
SIMRET
D4
SIMULATED END OF JOB
YES
2
NO
E4
OPERATOR PAUSE BIT ON
NO
E5
STMTIN
YES
F4
MSGOUT FD
PAUSE MESSAGE ON SYSLOG
G4
SET JOB CONTROL INPUT BIT IN JBCSW0 TO SYSLOG
H4
SET OPERATOR PAUSE BIT IN JBCSW0 OFF
J4
STMTIN

Chart EL. JOB

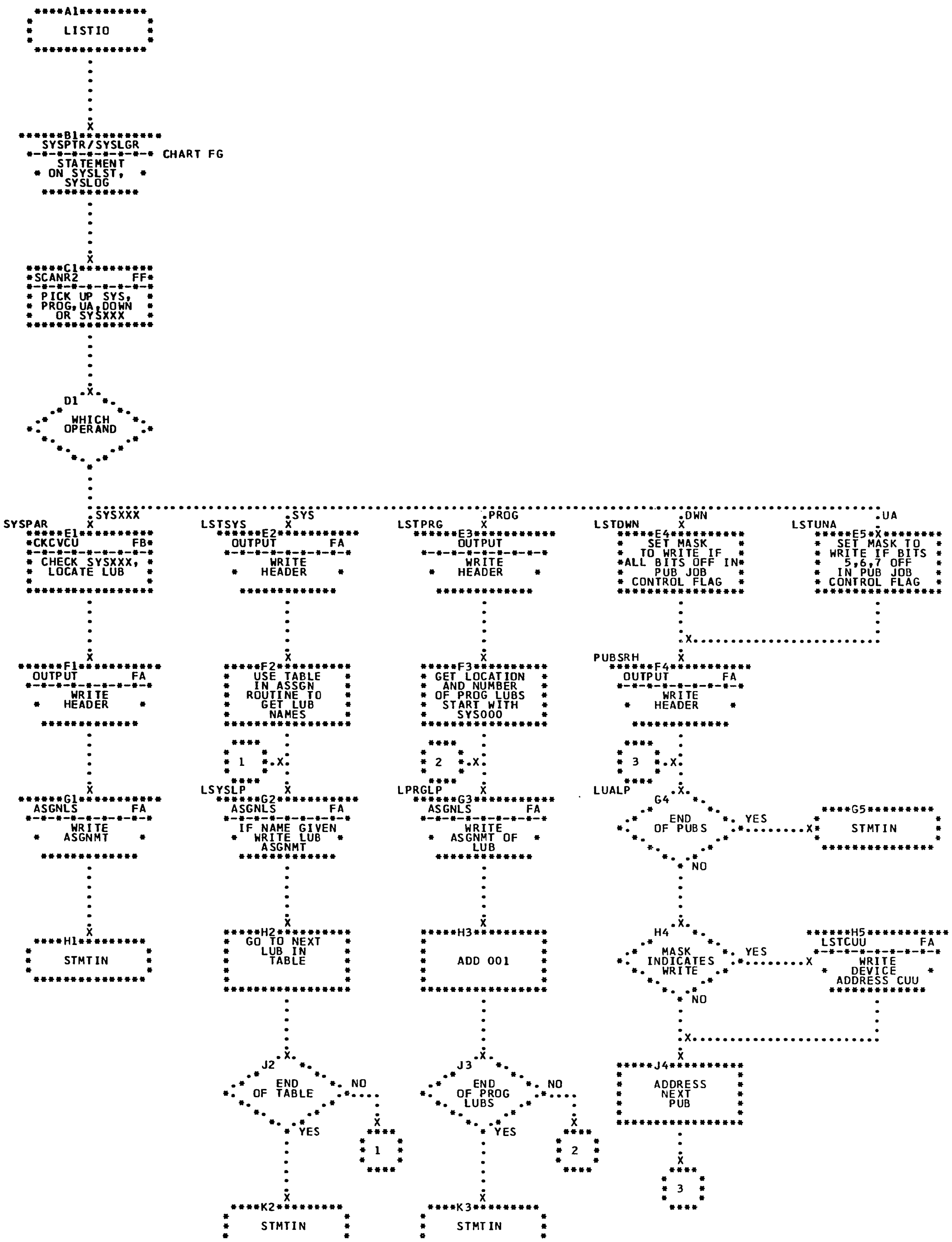


Chart EM. LISTIO

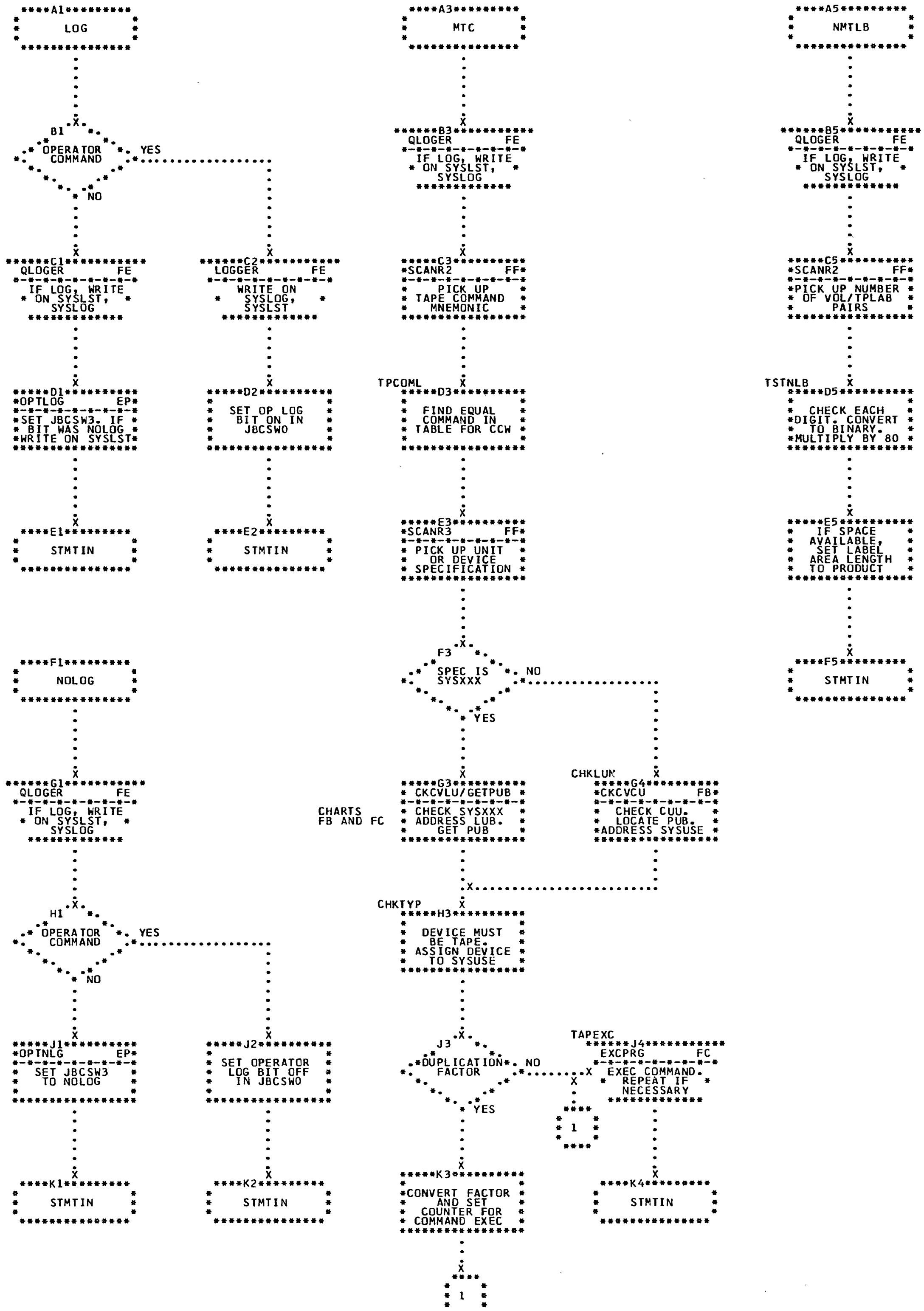


Chart EN. LOG, MTC, NMTLB, NOLOG

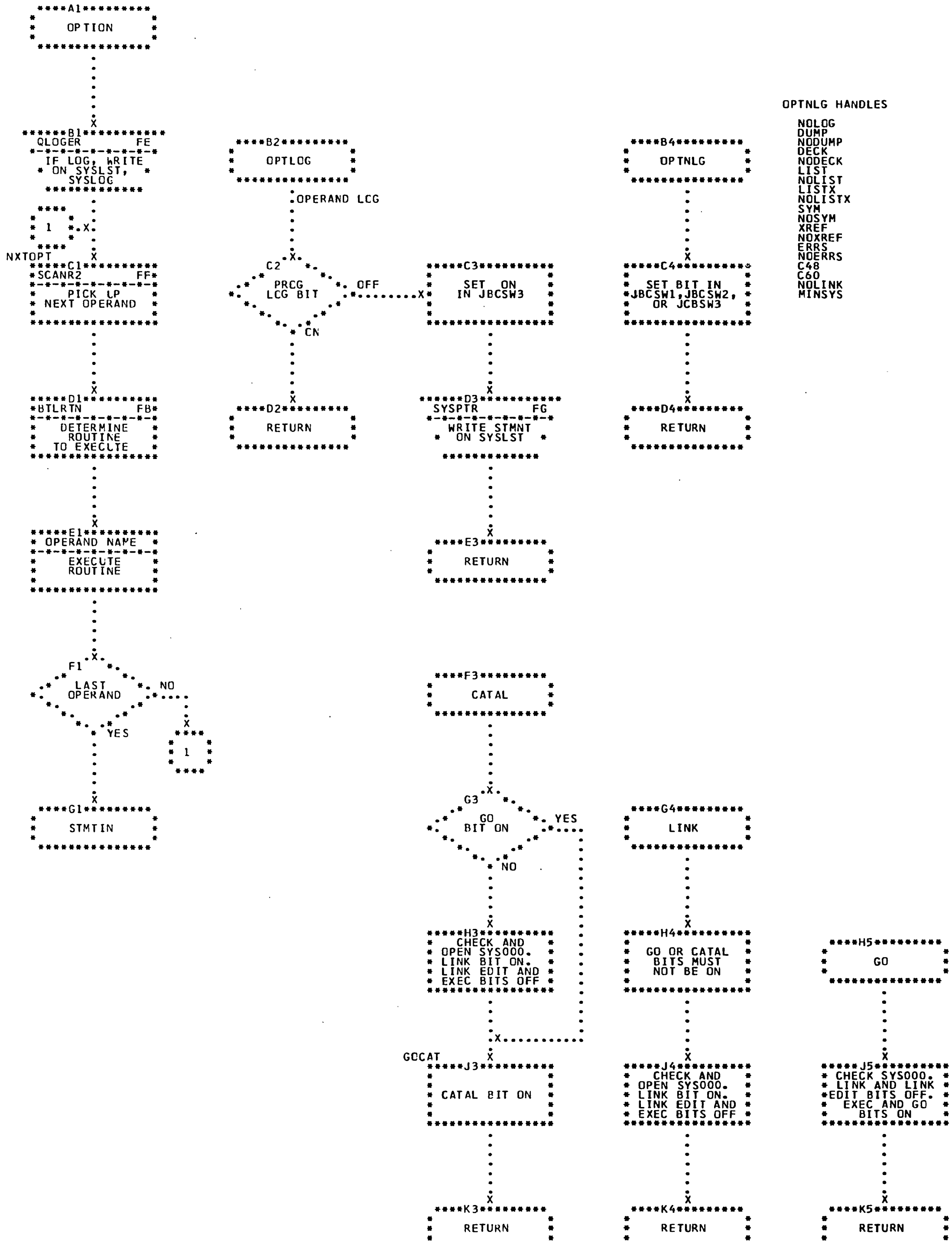


Chart EP. OPTION

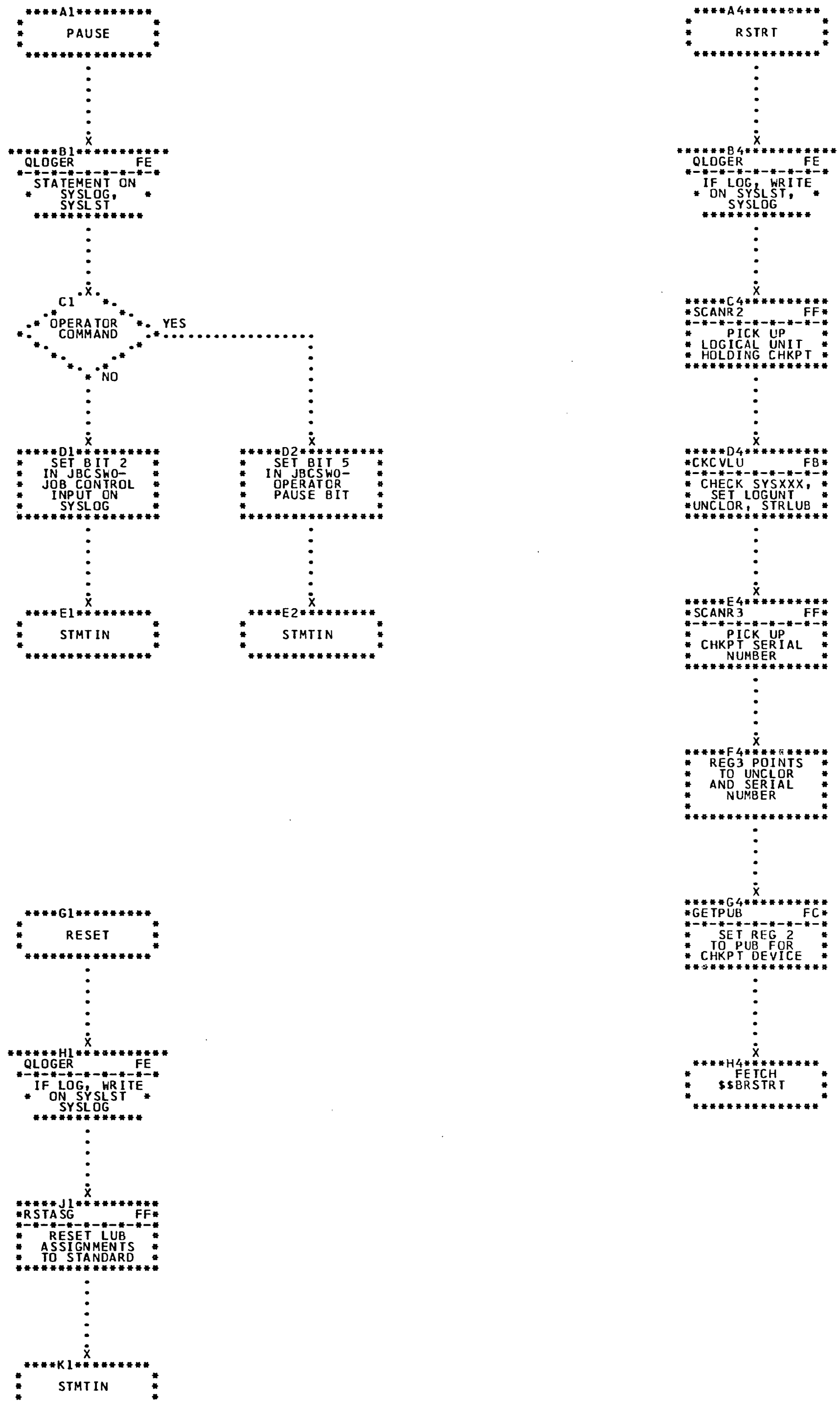


Chart EQ. PAUSE, RESET, RSTRT

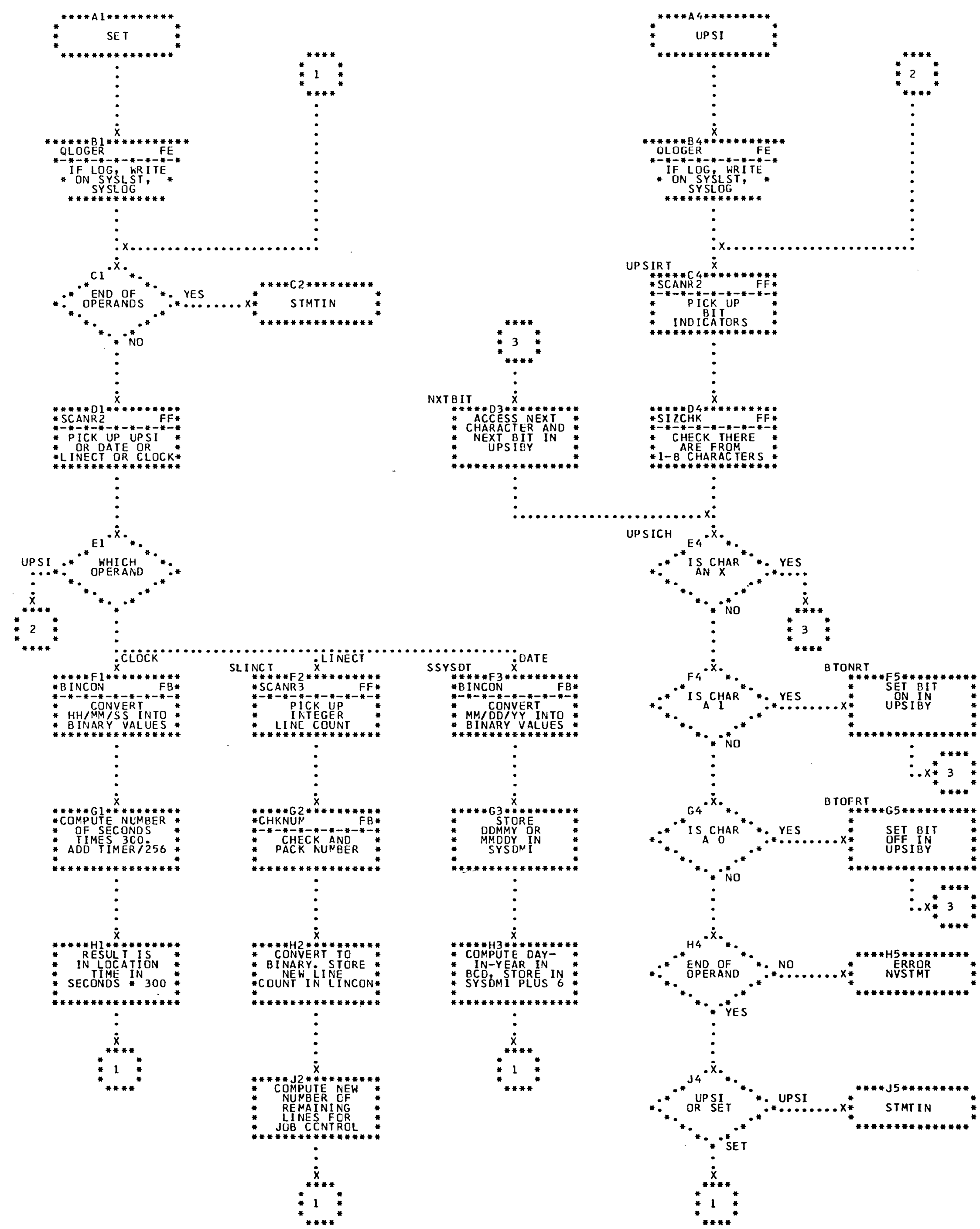


Chart ER. SET, UPSI

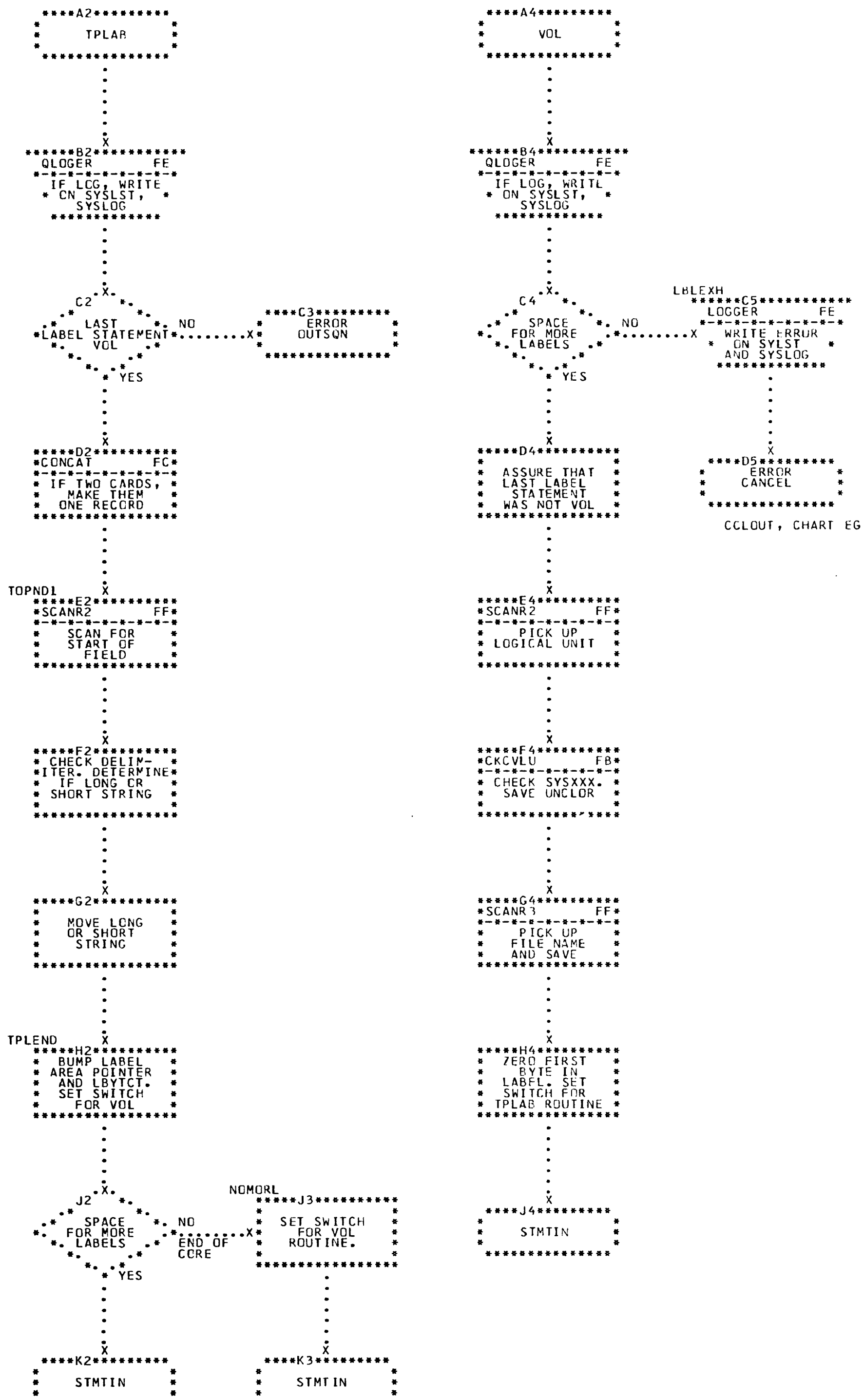


Chart ES. TPLAB, VOL

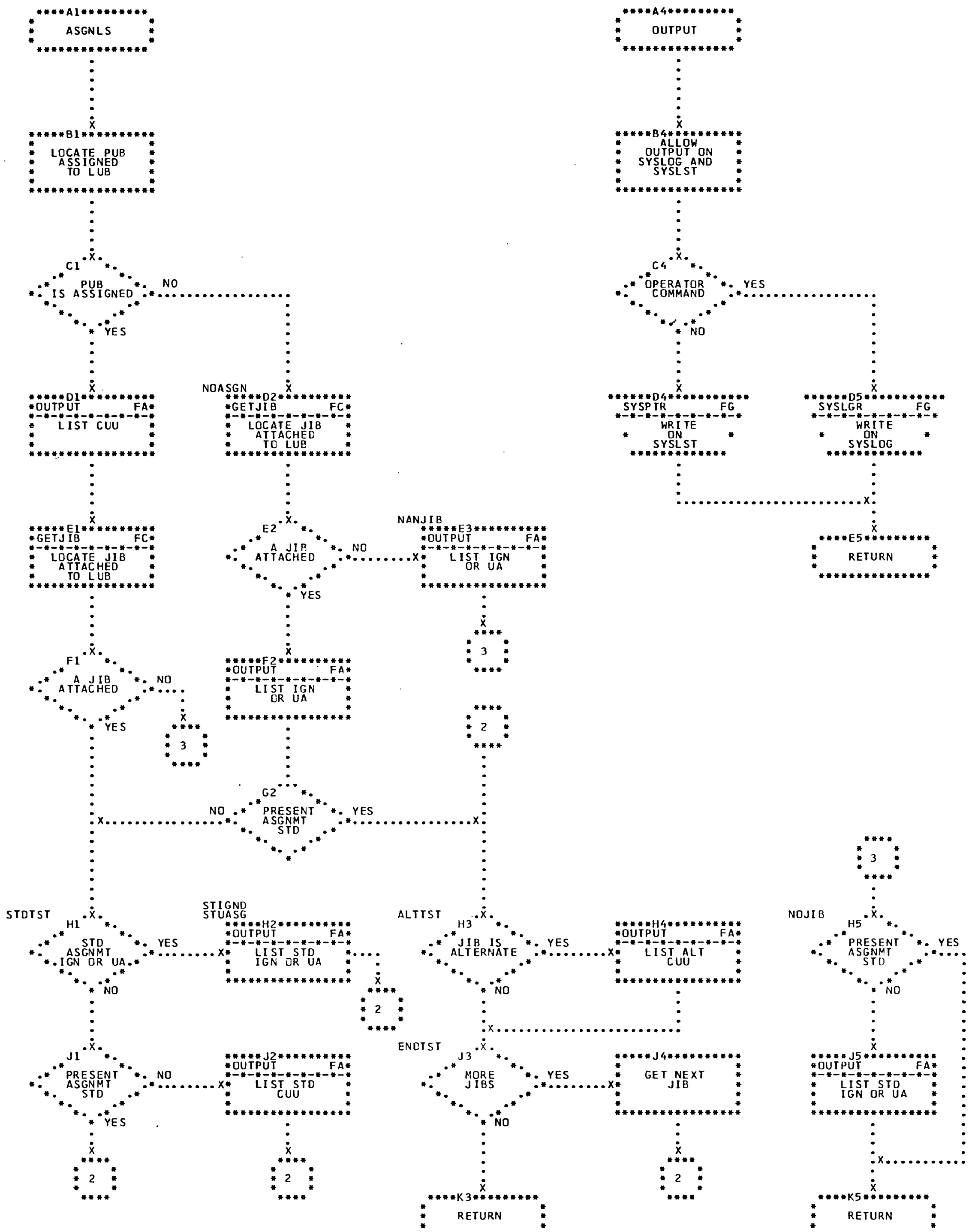


Chart FA. ASGNLS, OUTPUT Subroutines

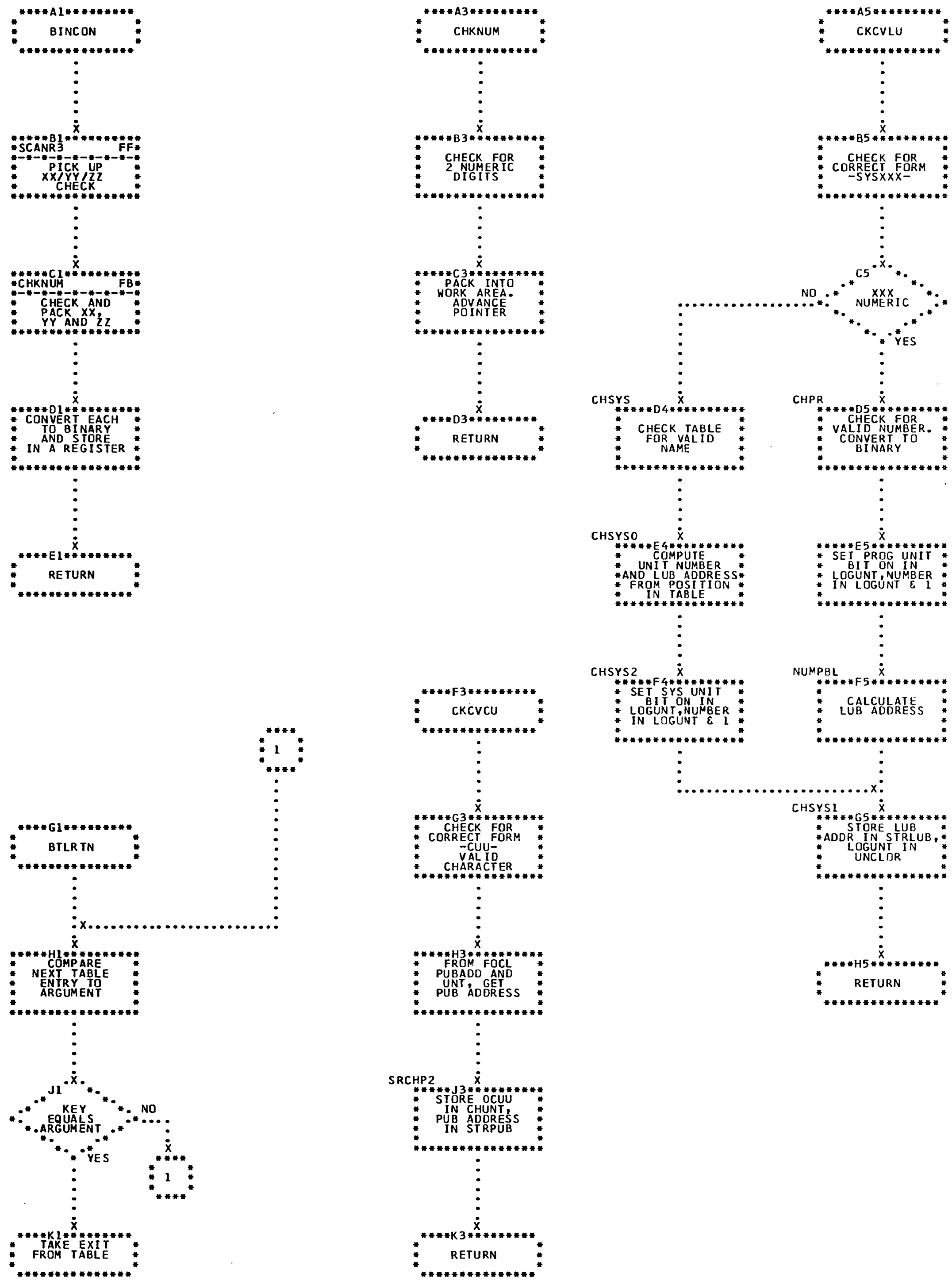


Chart FB. BINCON, BTLRTN, CHKNUM, CKCVCU, CKCVLU Subroutines

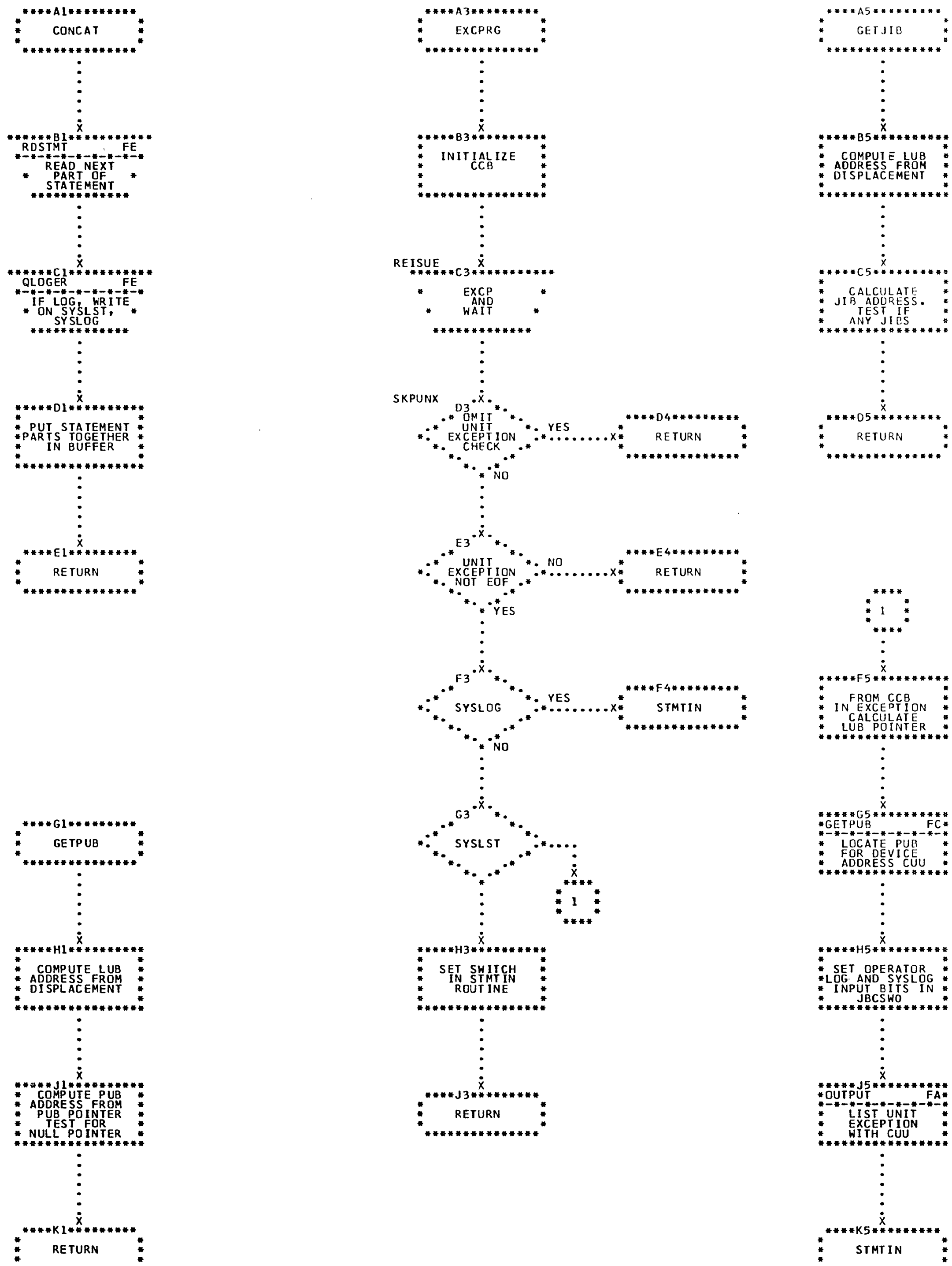


Chart FC. CONCAT, EXCPRG, GETJIB, GETPUB Subroutines

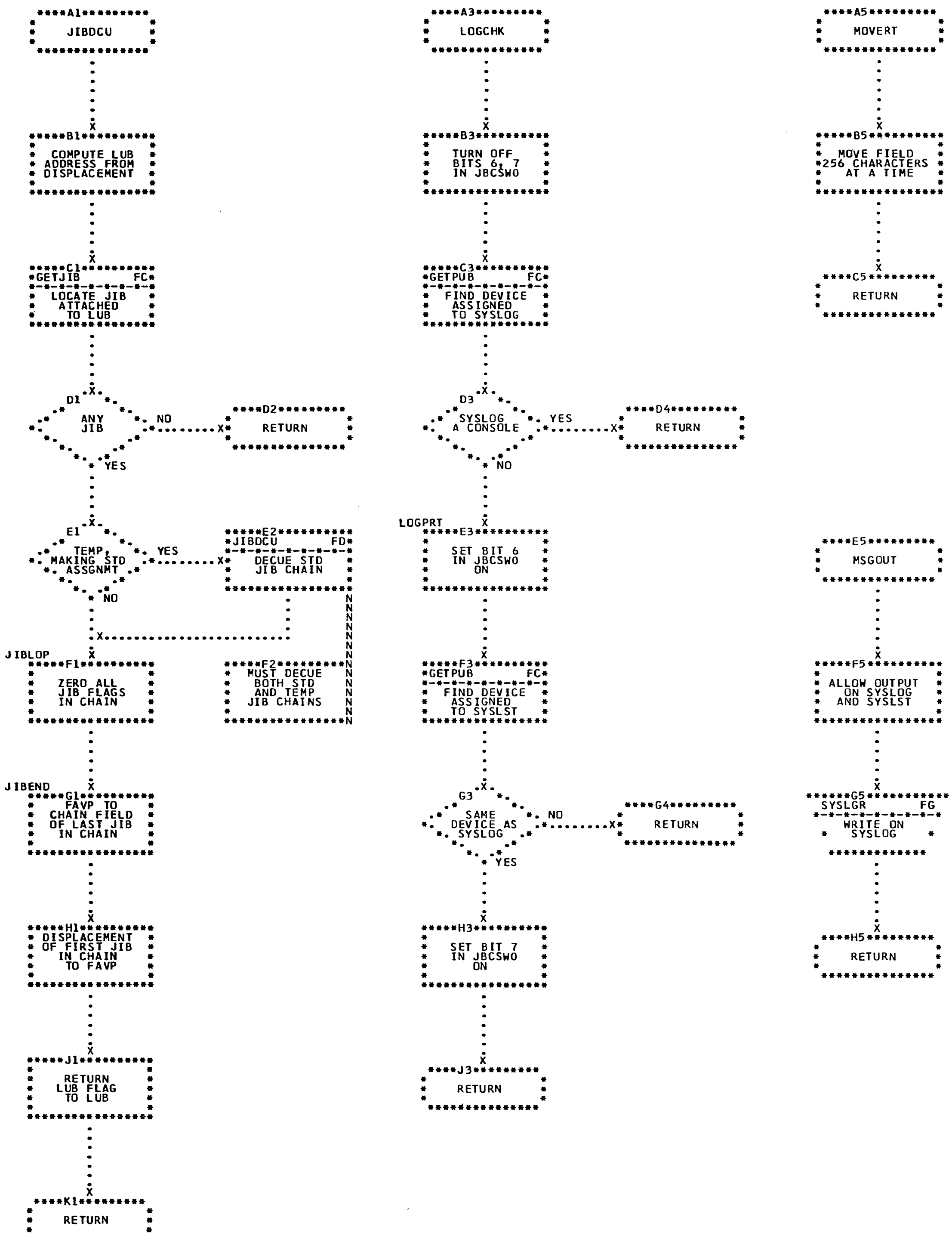


Chart FD. JIBDCU, LOGCHK, MOVERT, MSGOUT Subroutines

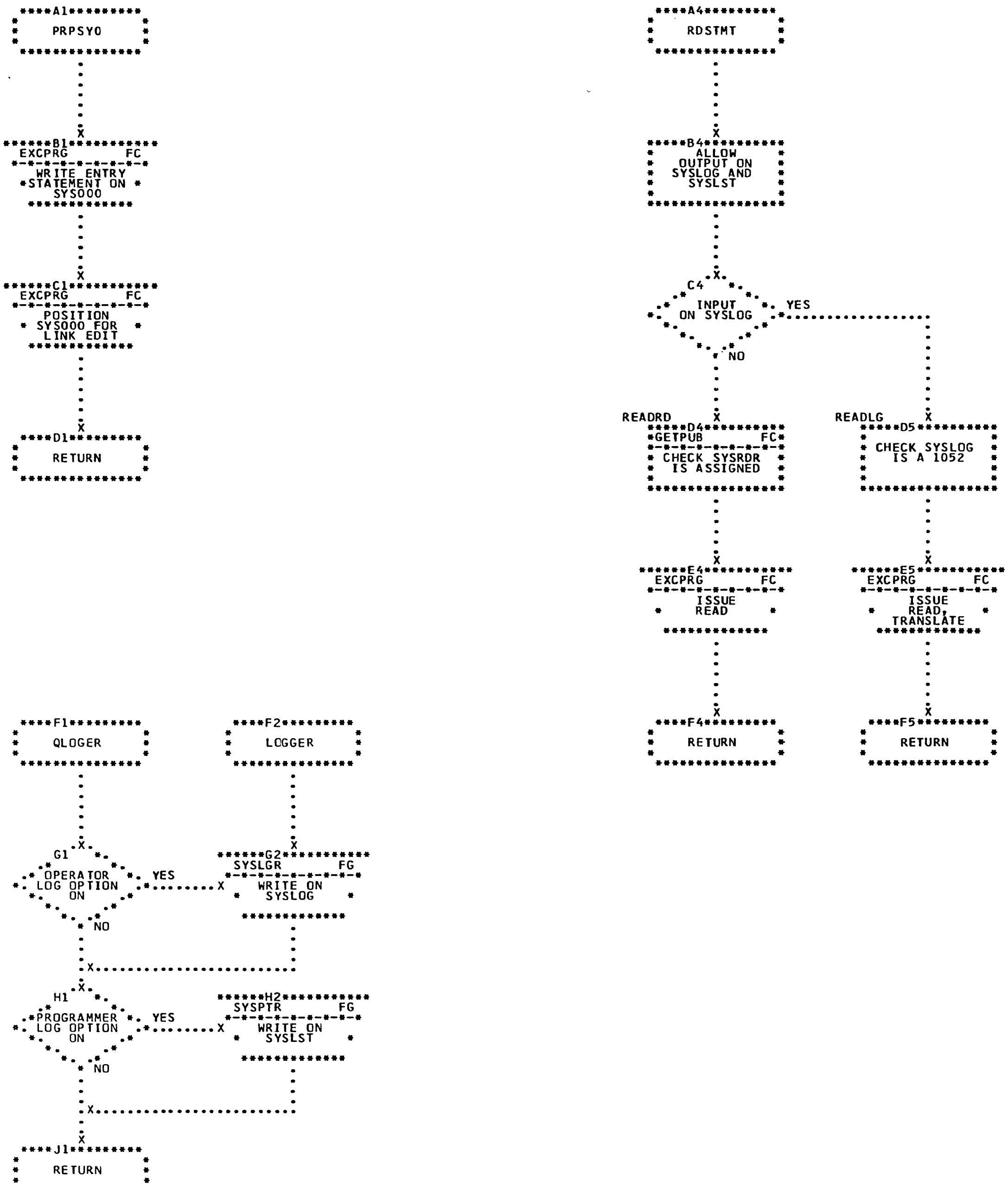


Chart FE. PRPSY0, QLOGER, LOGGER, RDSTMT Subroutines

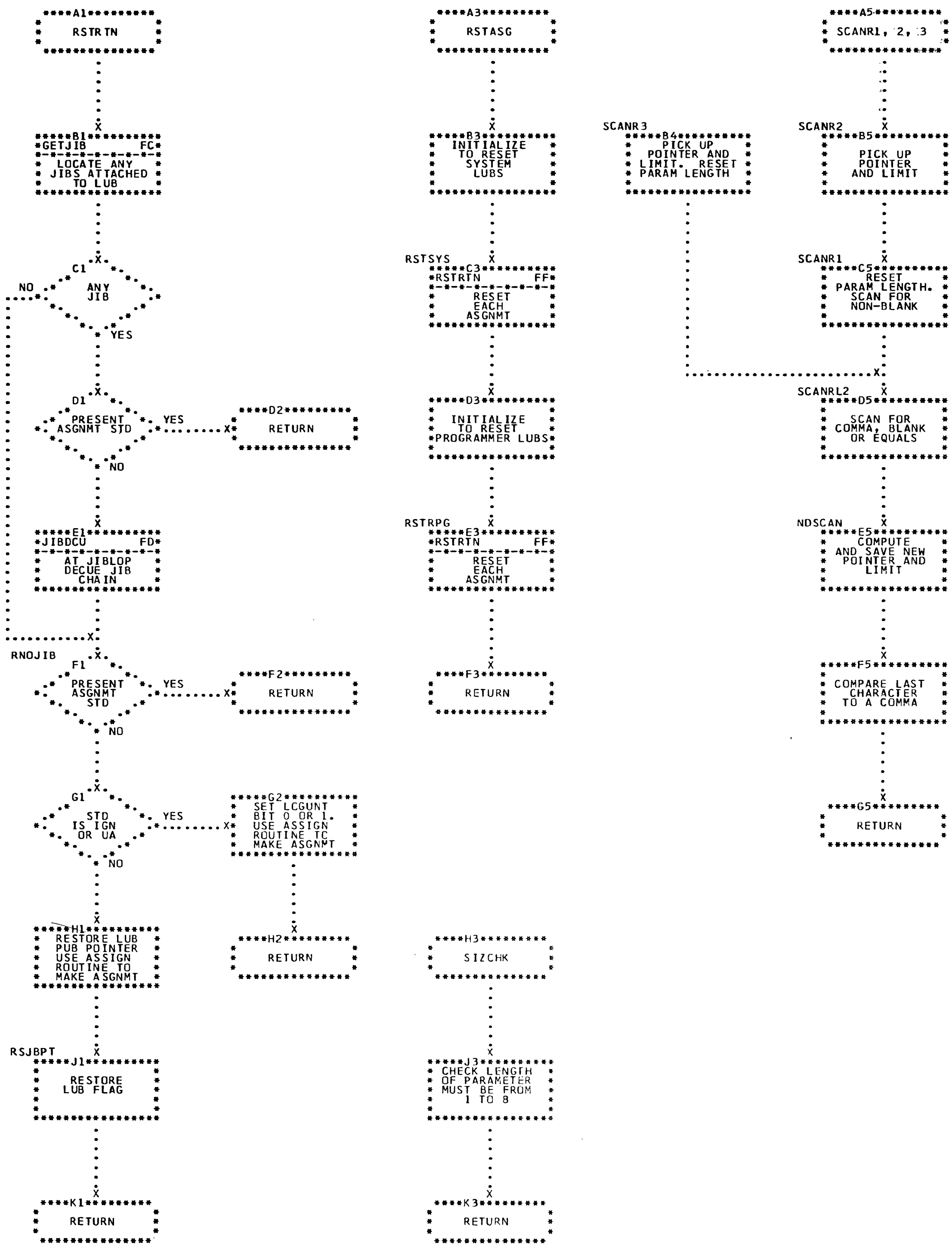


Chart FF. RSTRTN, RSTASG, SIZCHK, SCAN Subroutines

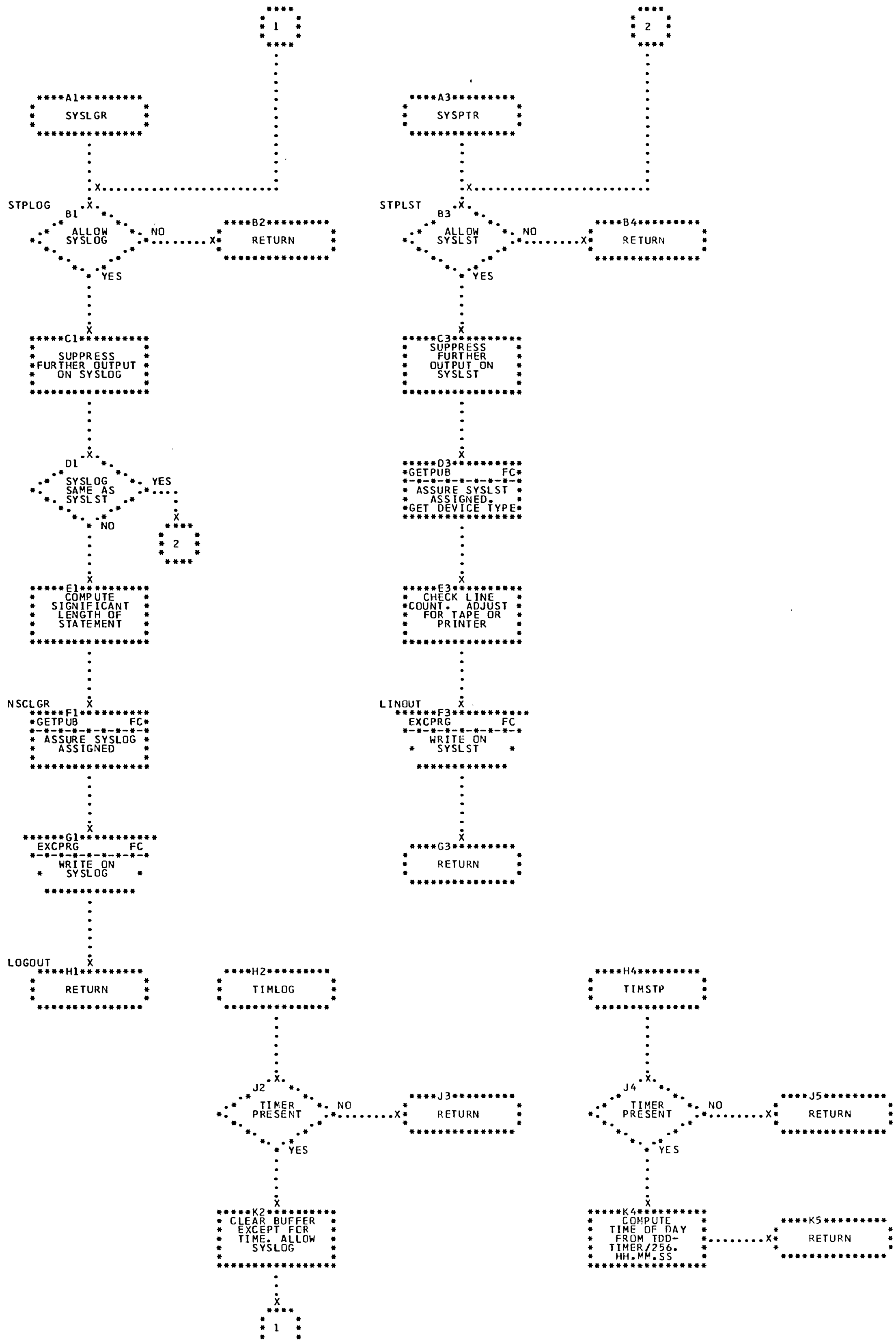


Chart FG. SYSLGR, SYSPTR, TIMLOG, TIMSTP Subroutines

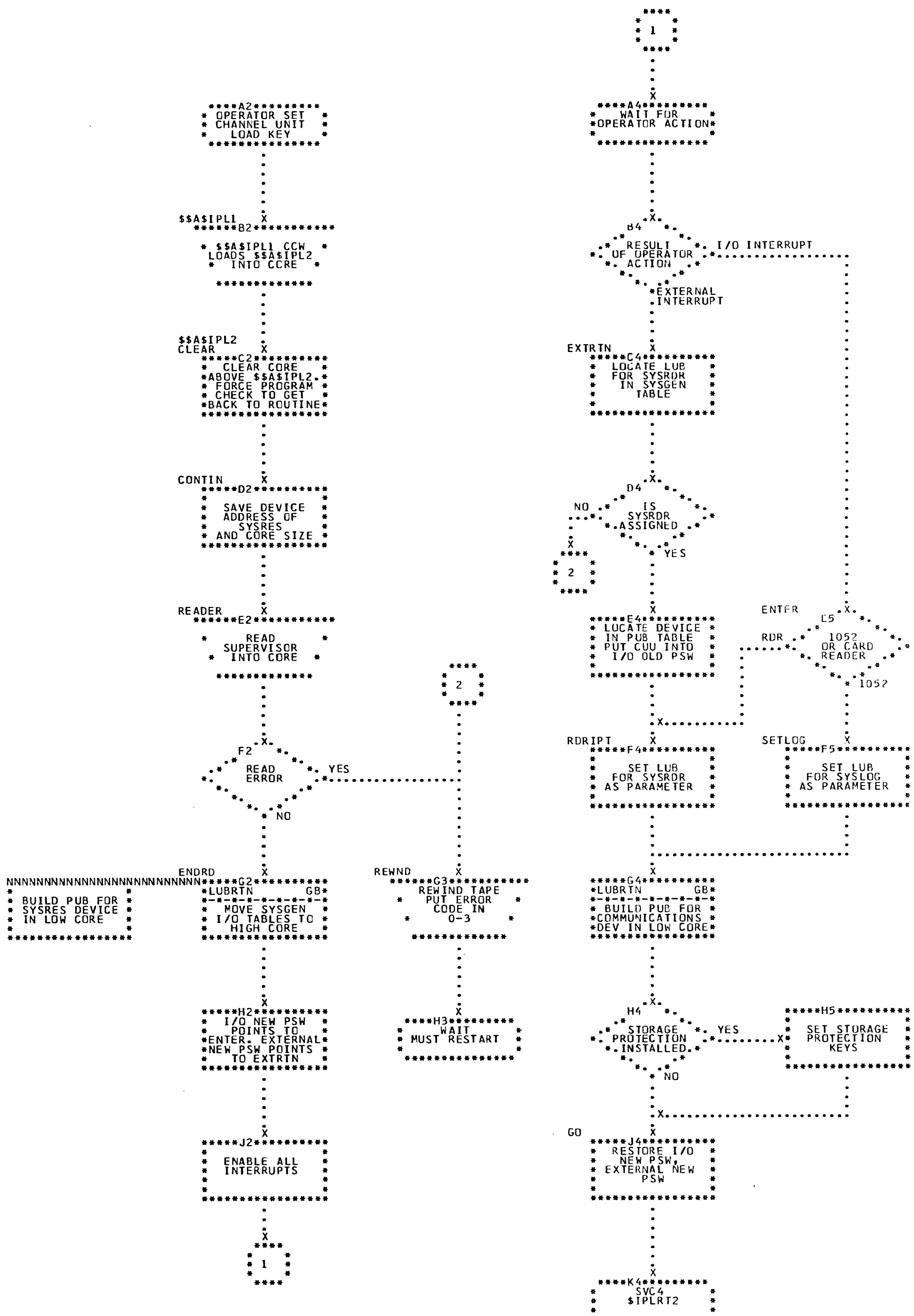


Chart GA. Bootstrap and System Load

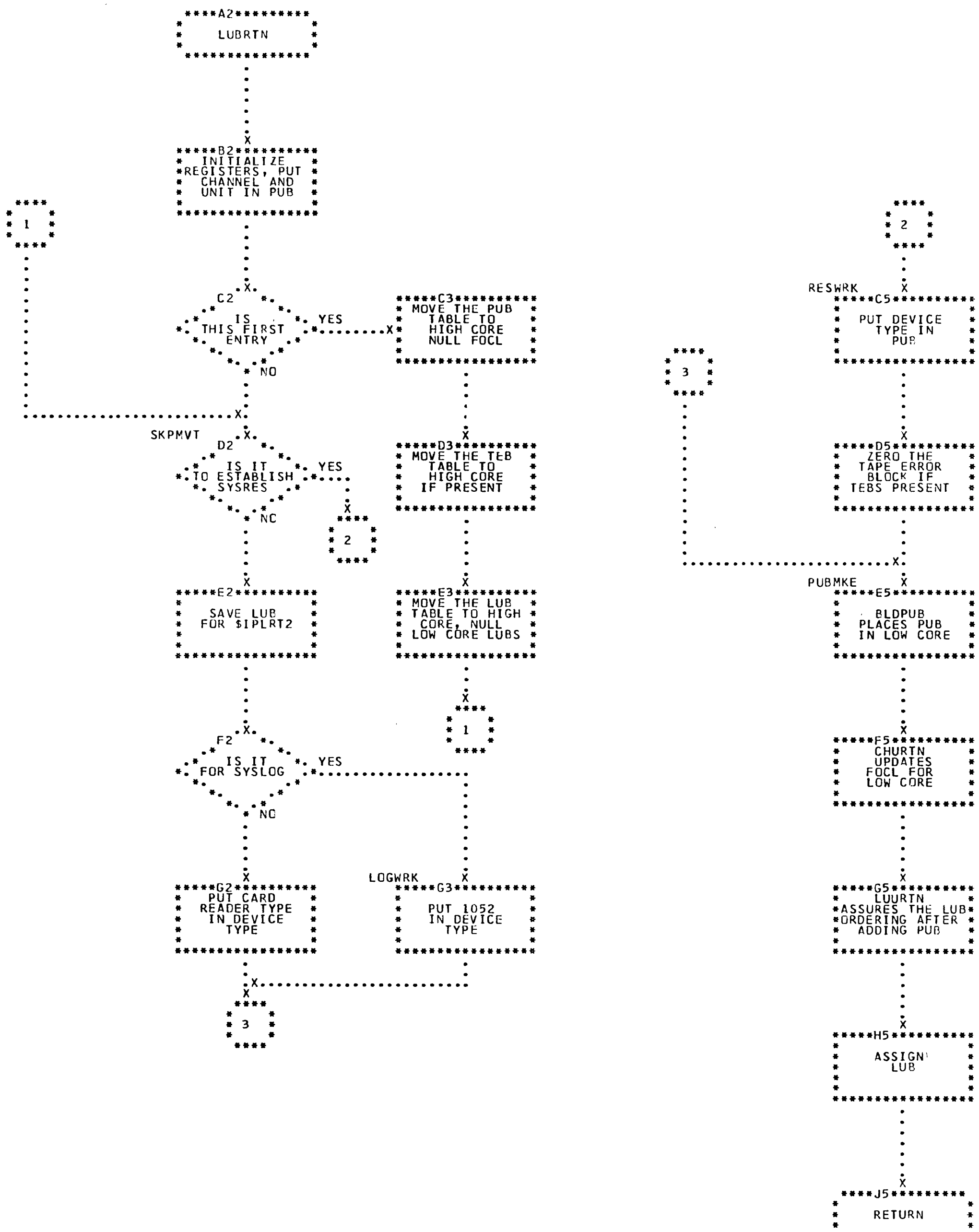


Chart GB. Build PUB Subroutine

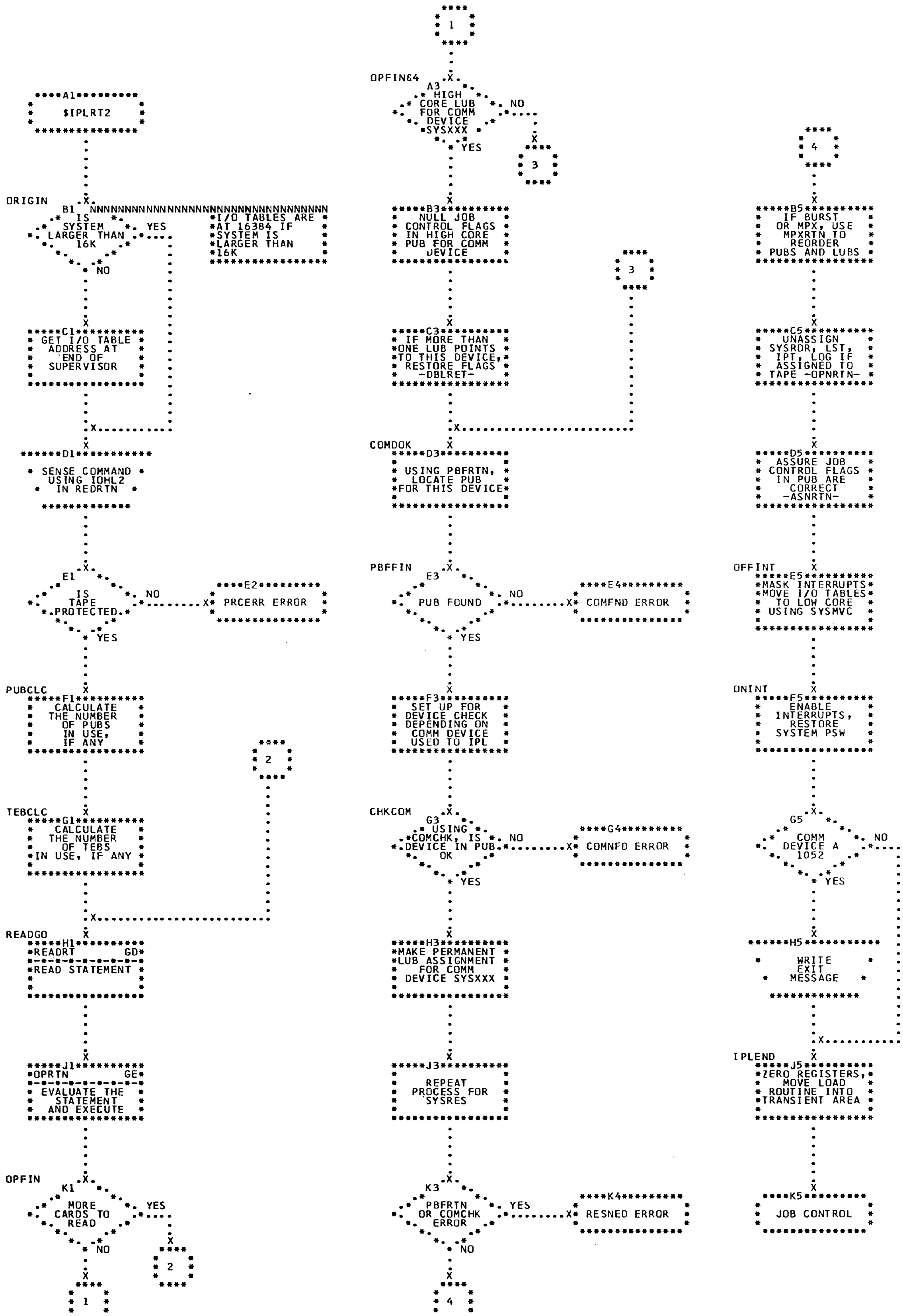


Chart GC. System Initialization

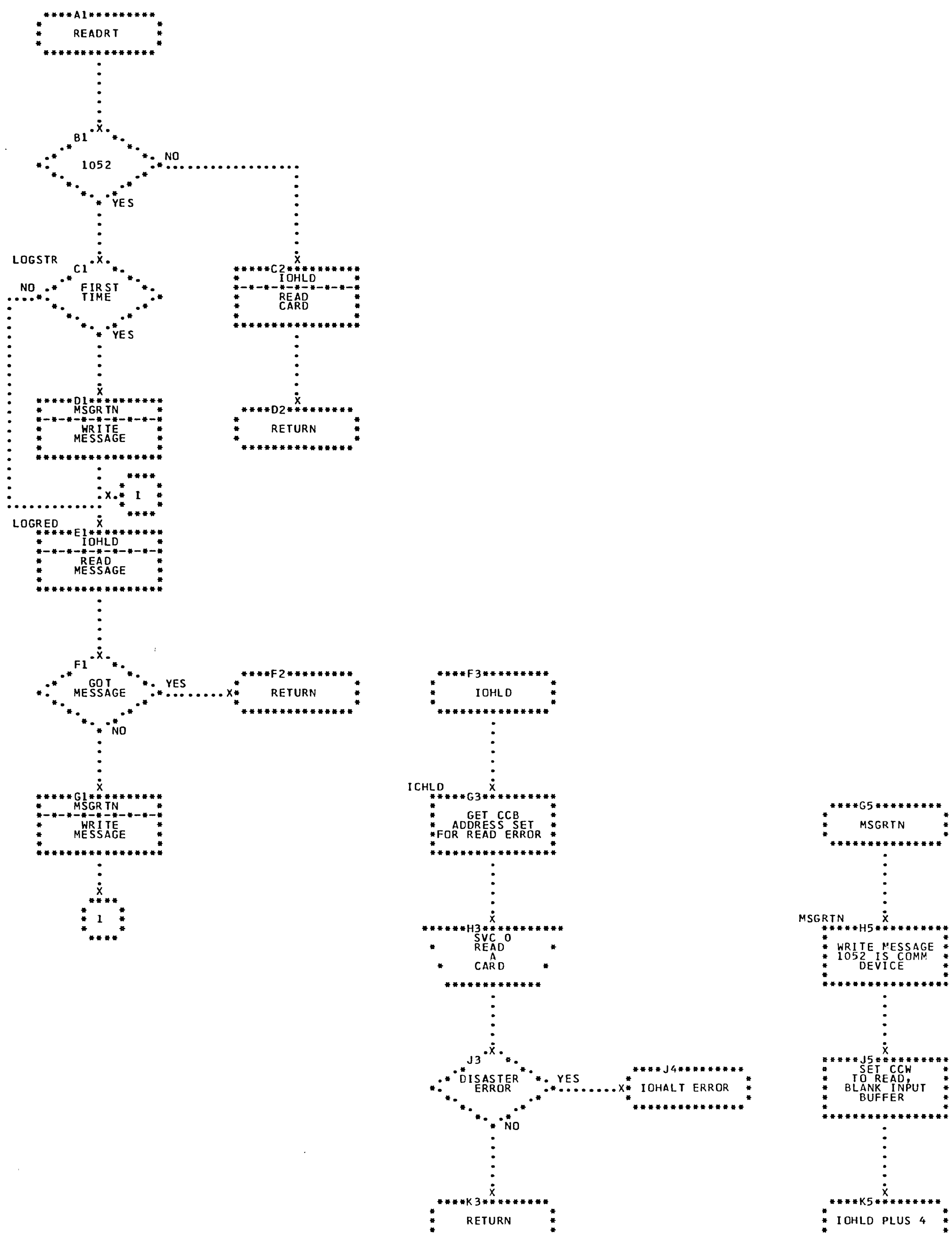


Chart GD. Read Subroutine

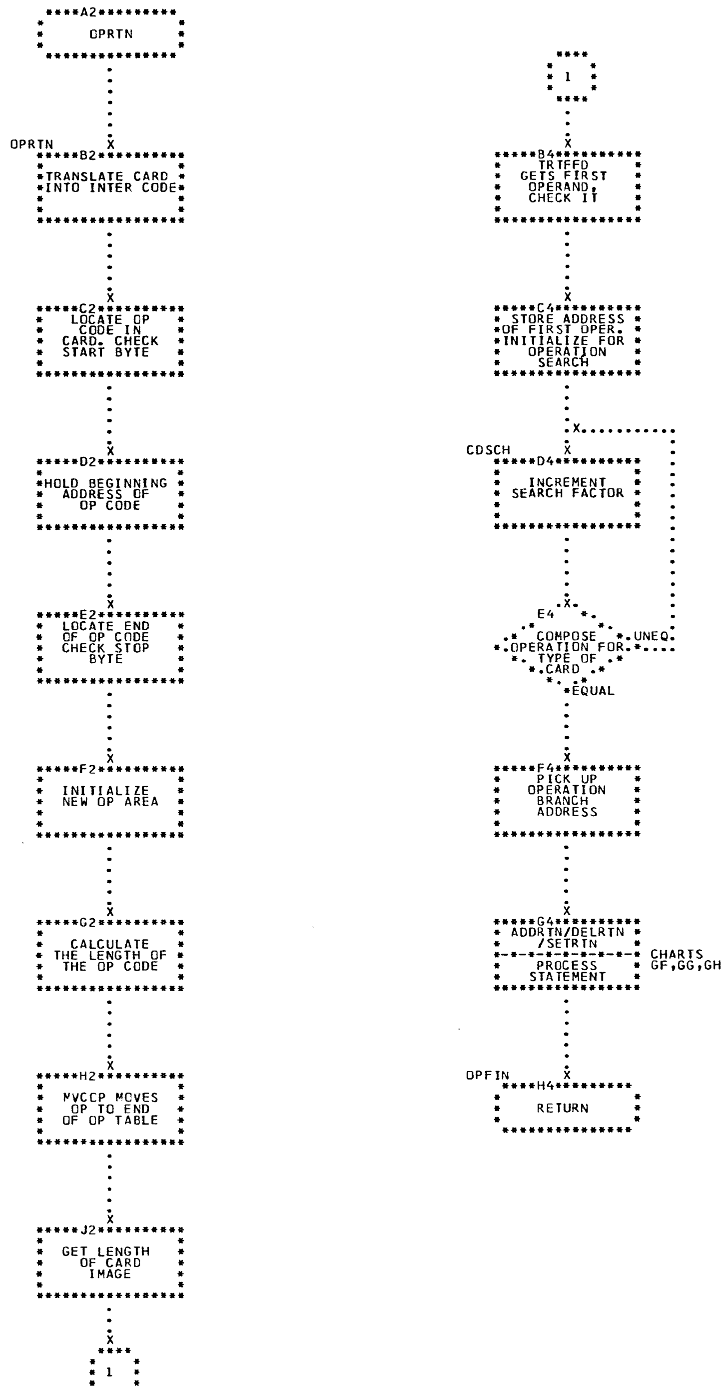


Chart GE. Scan Subroutine

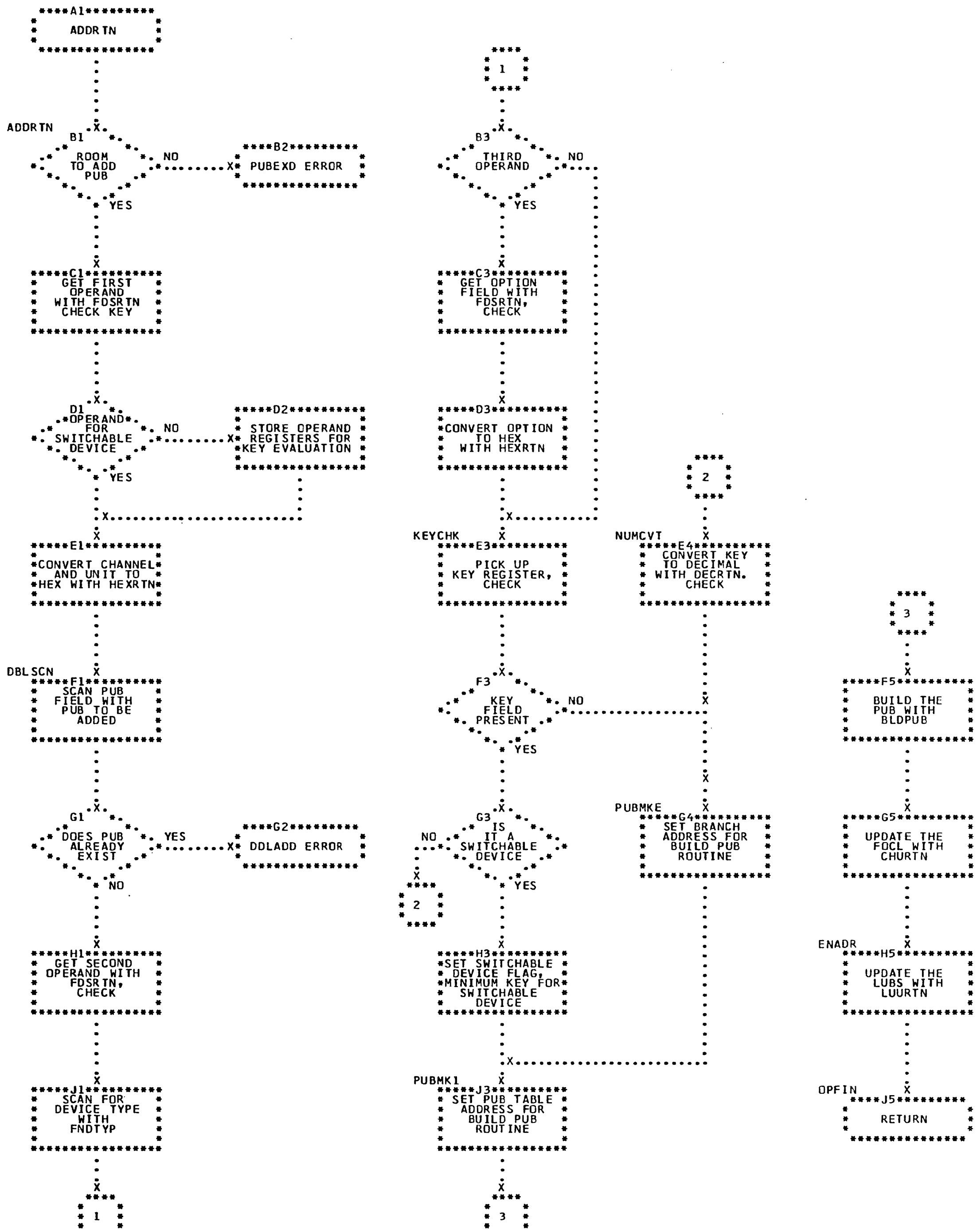


Chart GF. ADD Statement Subroutine

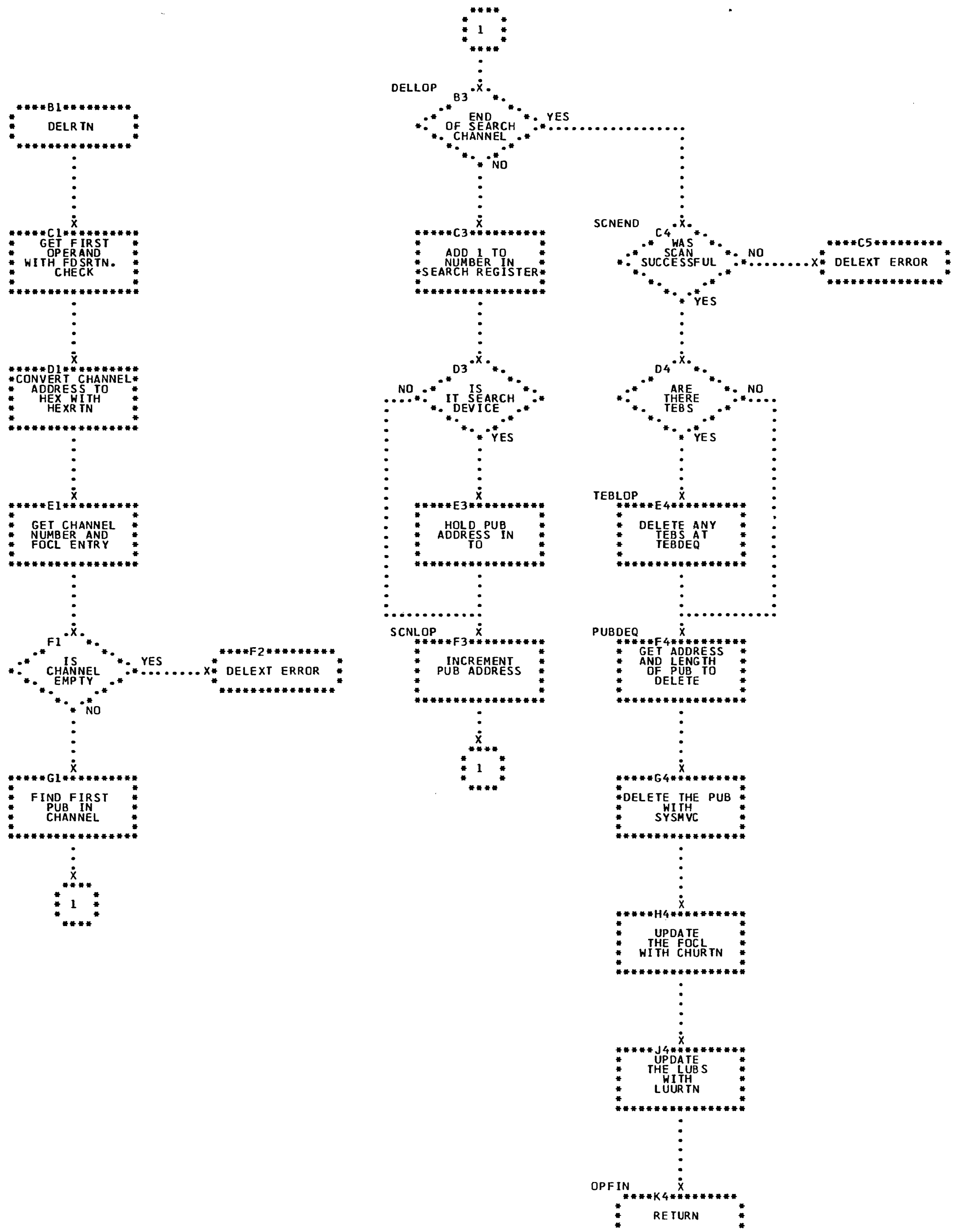


Chart GG. DEL Statement Subroutine

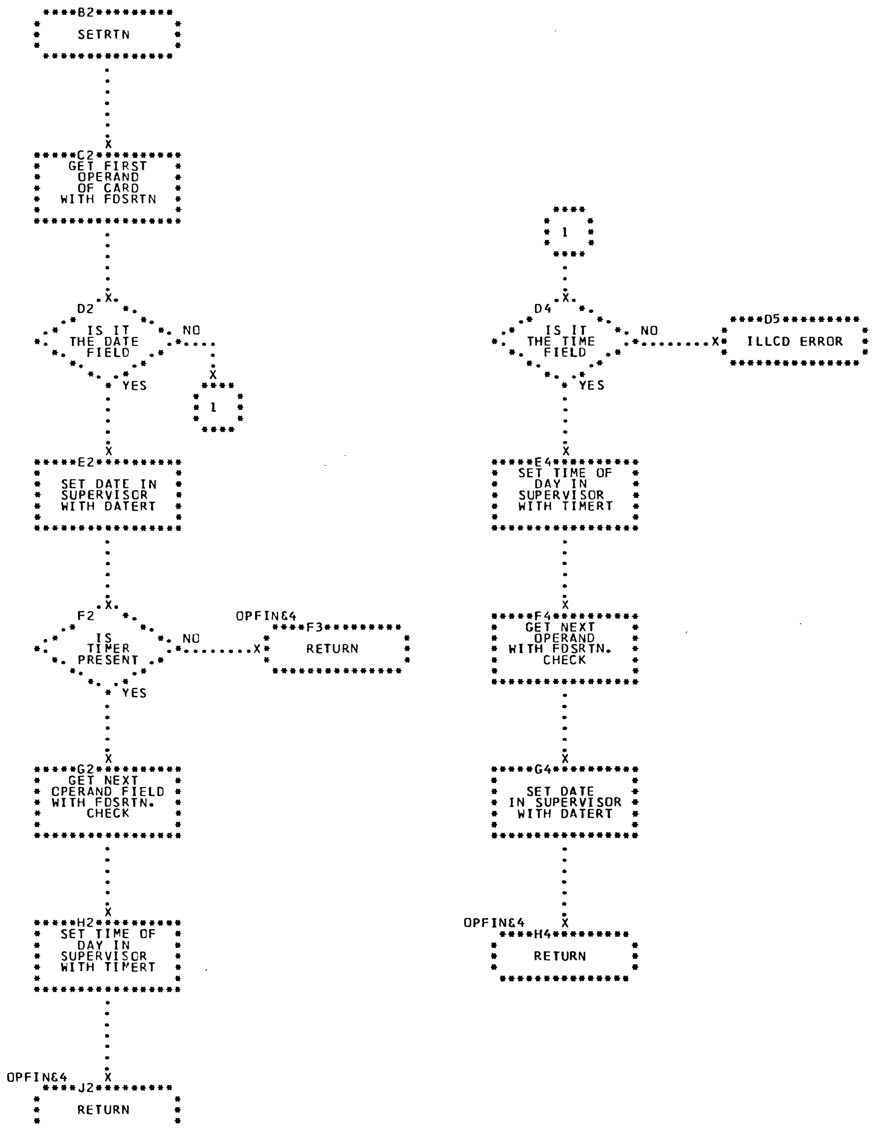


Chart GH. SET Statement Subroutine

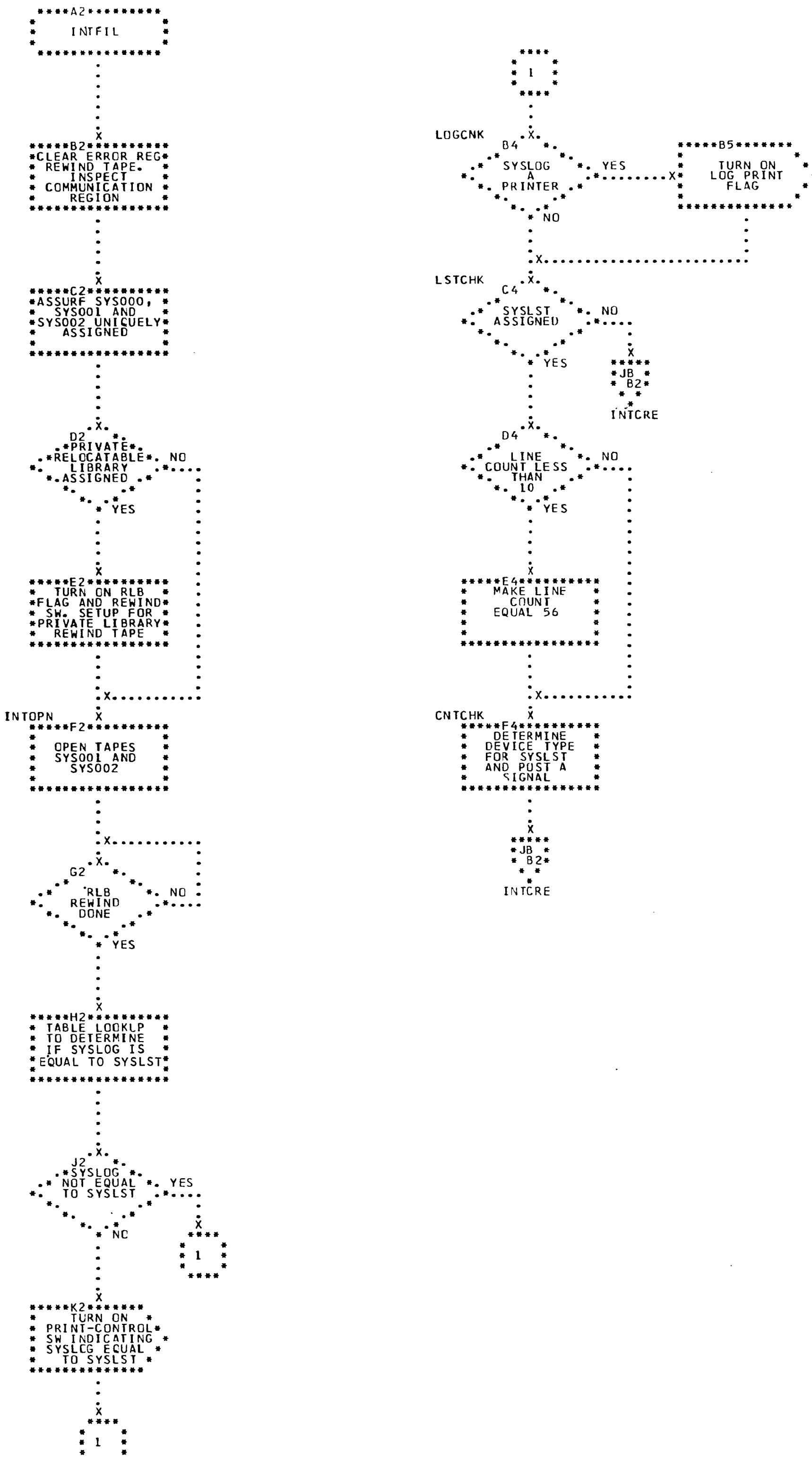


Chart JA. I/O Initialization

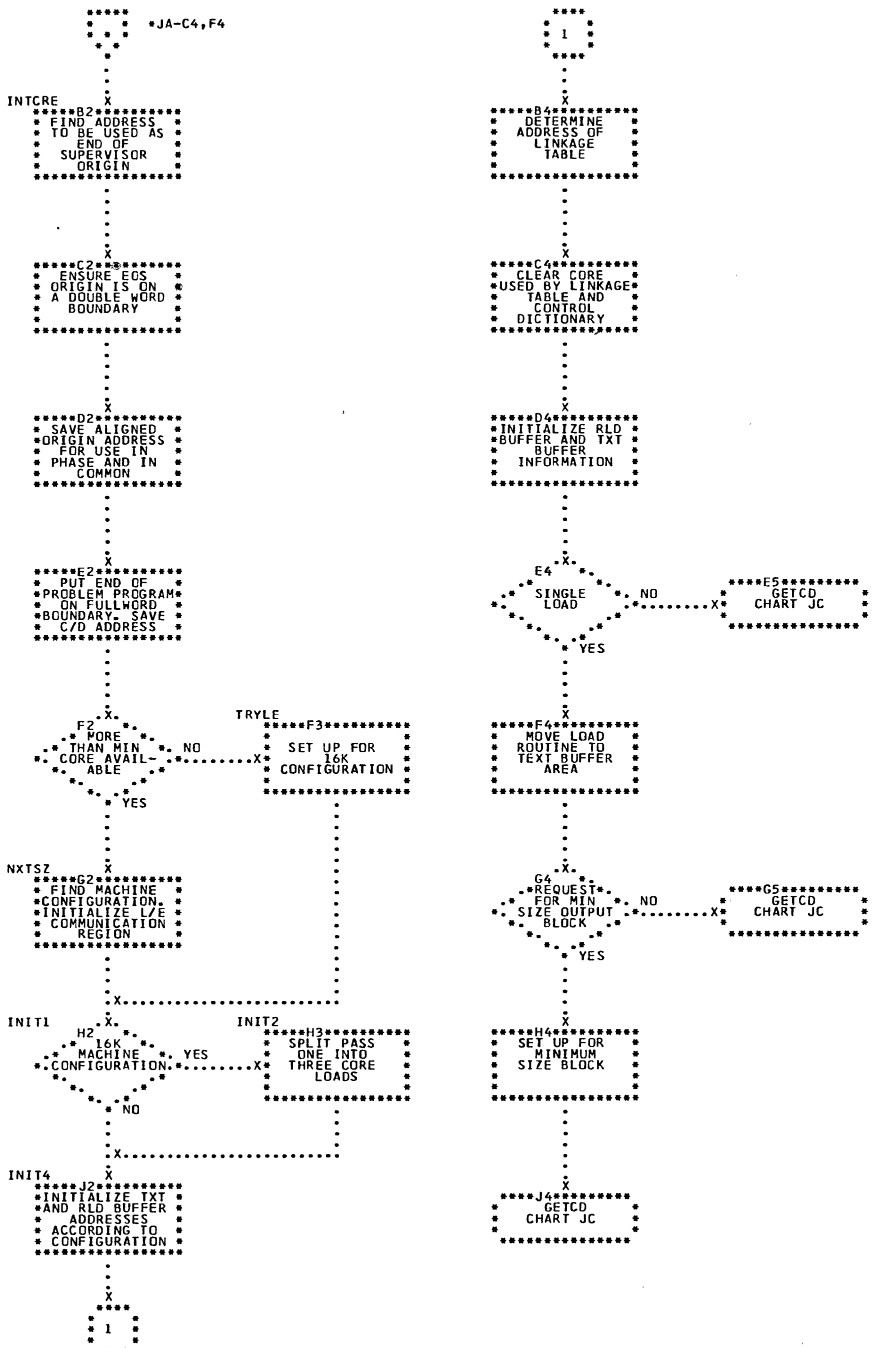


Chart JB. Storage Initialization

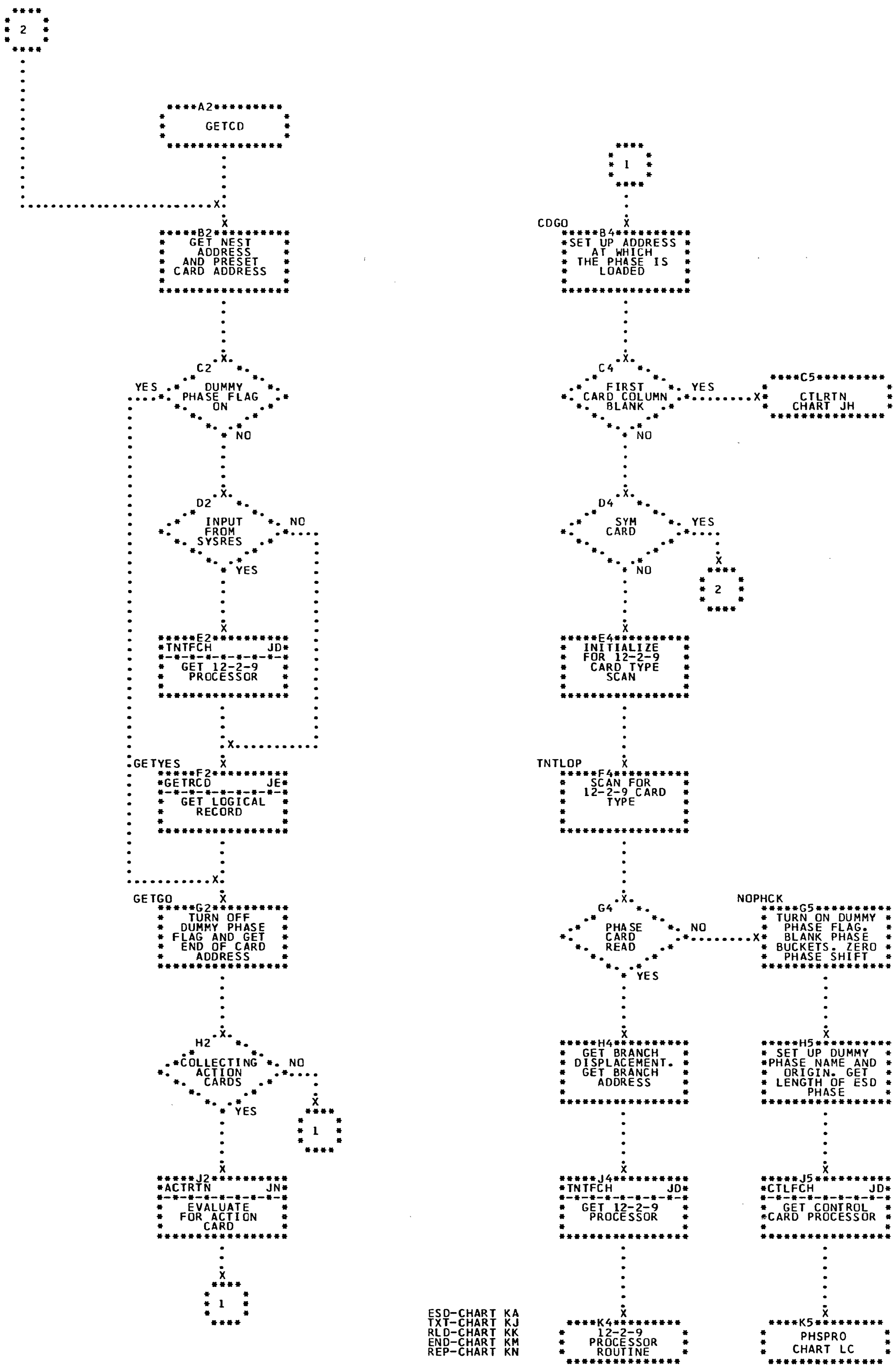


Chart JC. Get Card Processor

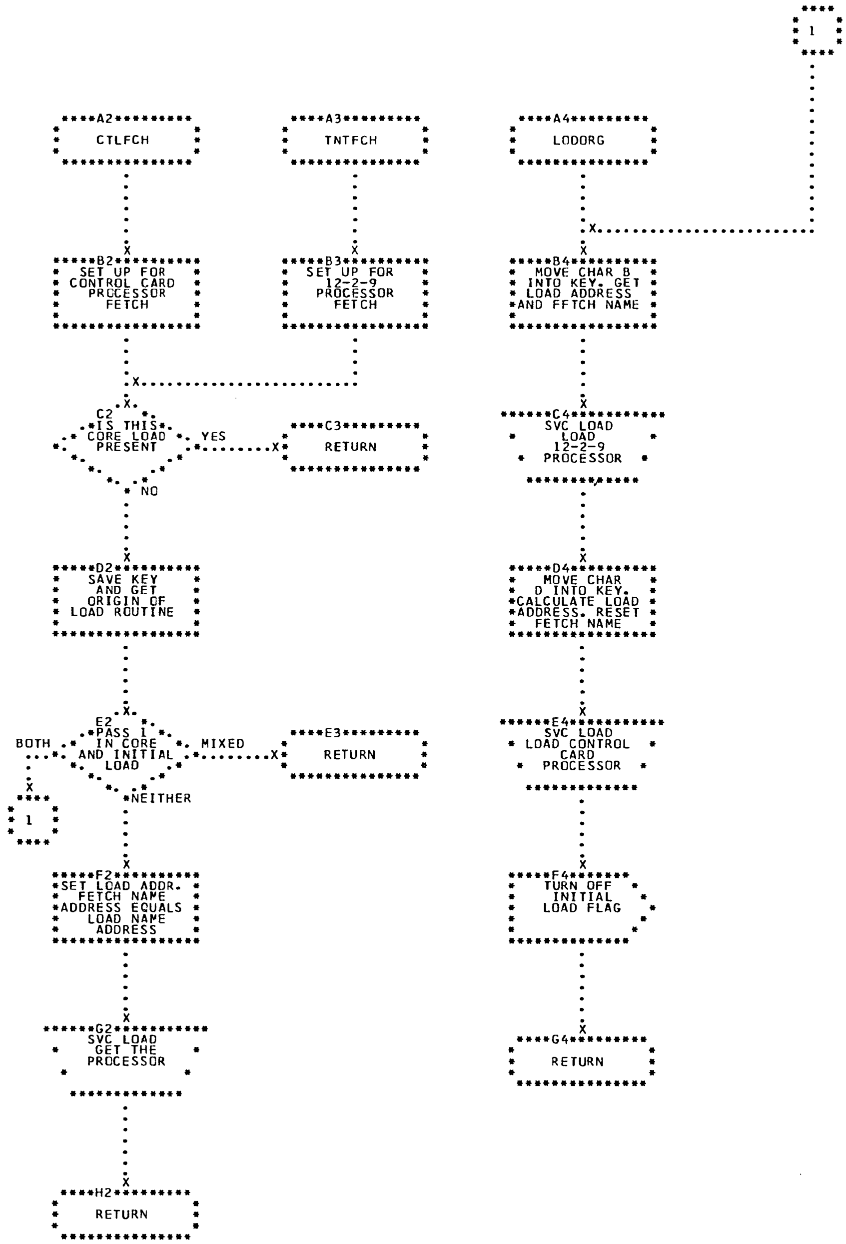


Chart JD. Fetch Subroutines

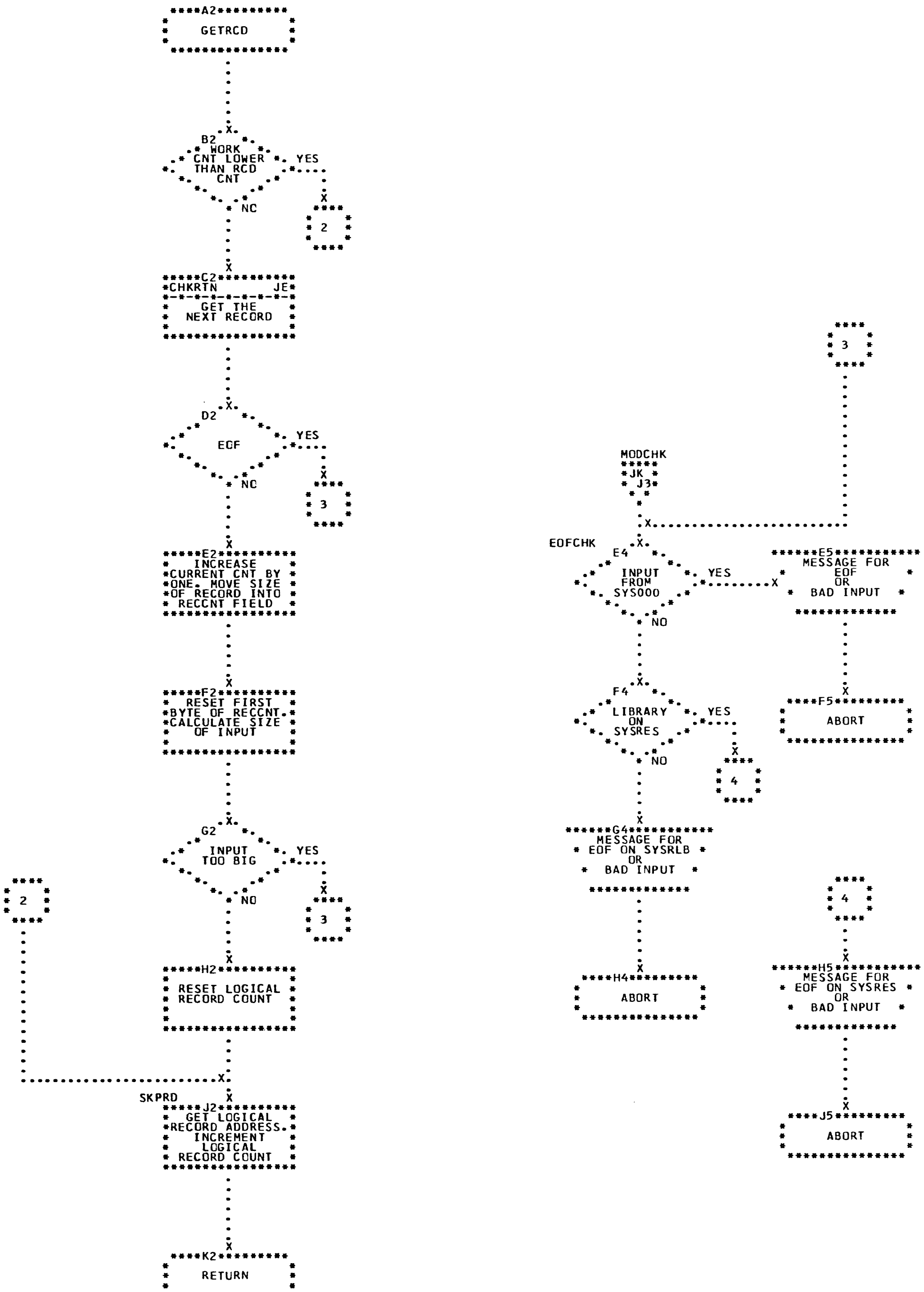


Chart JE. Input Subroutine

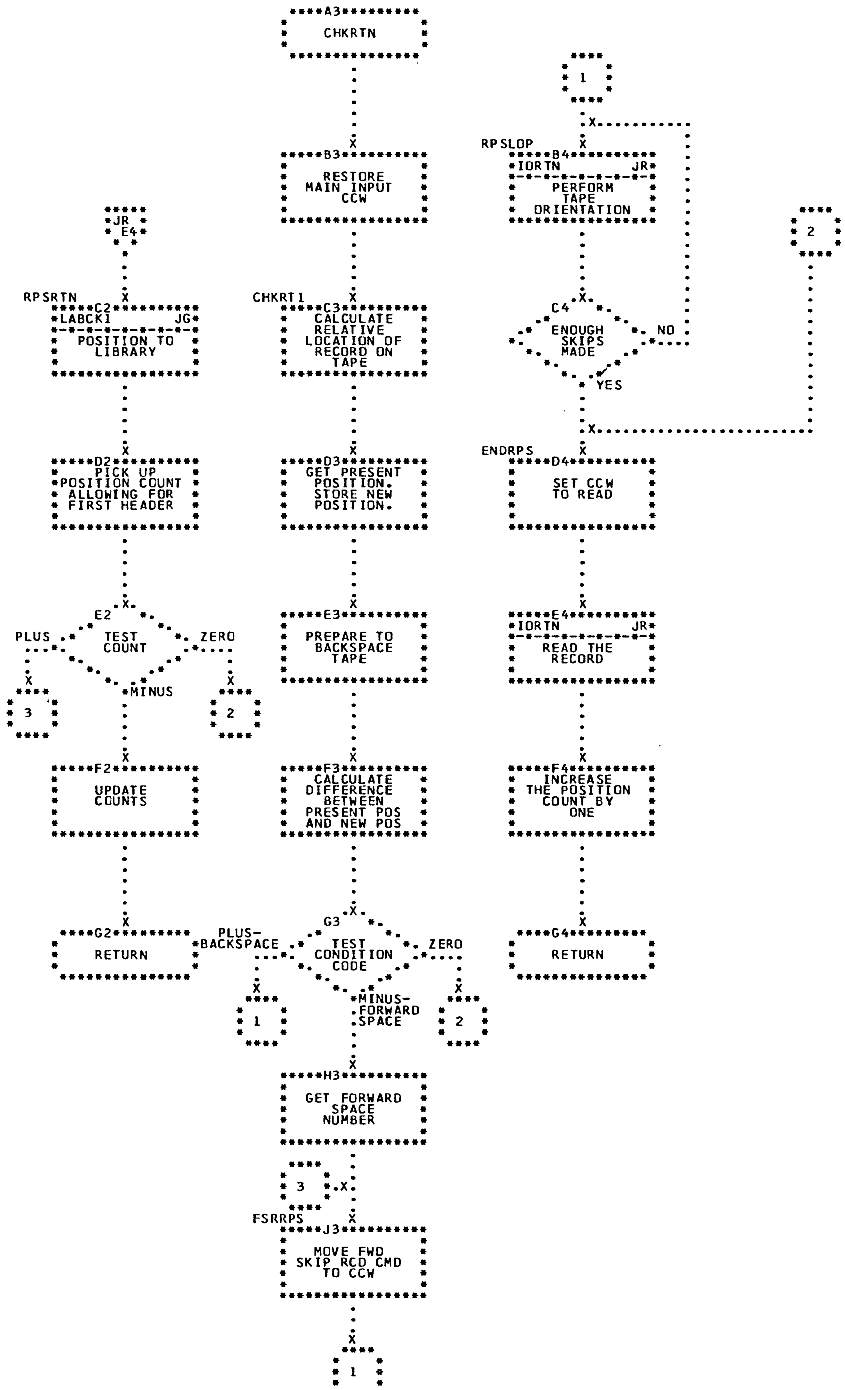


Chart JF. Get Record Subroutine

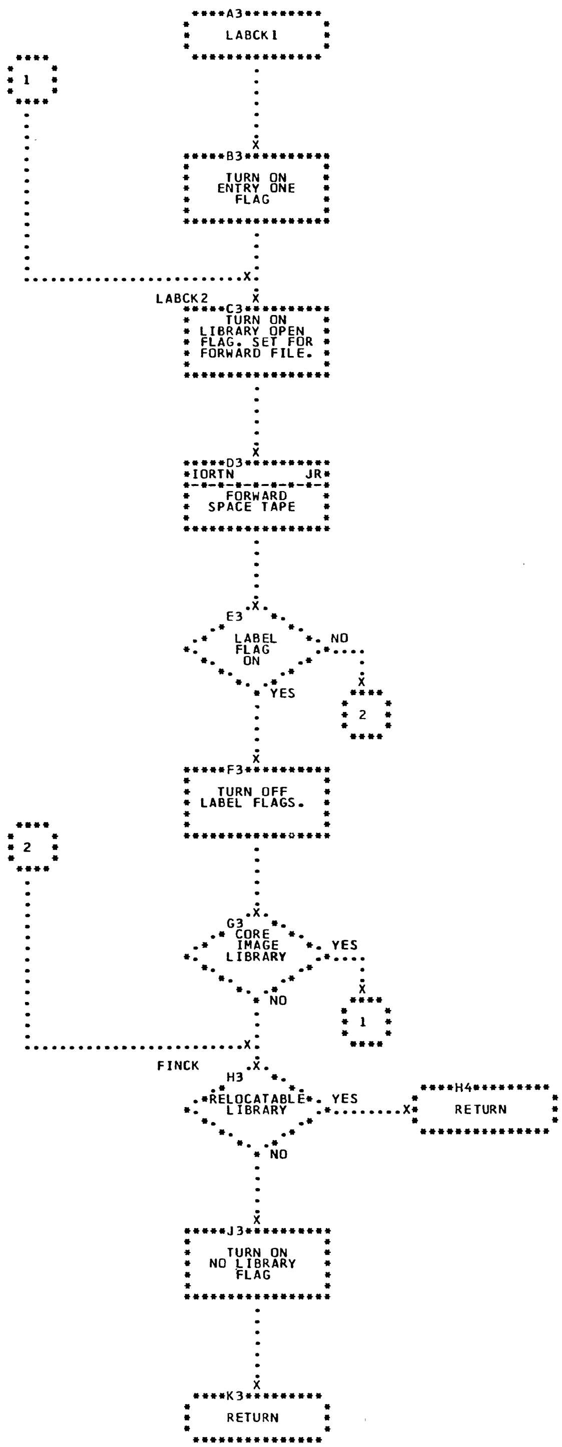


Chart JG. Relocatable Library Label Check

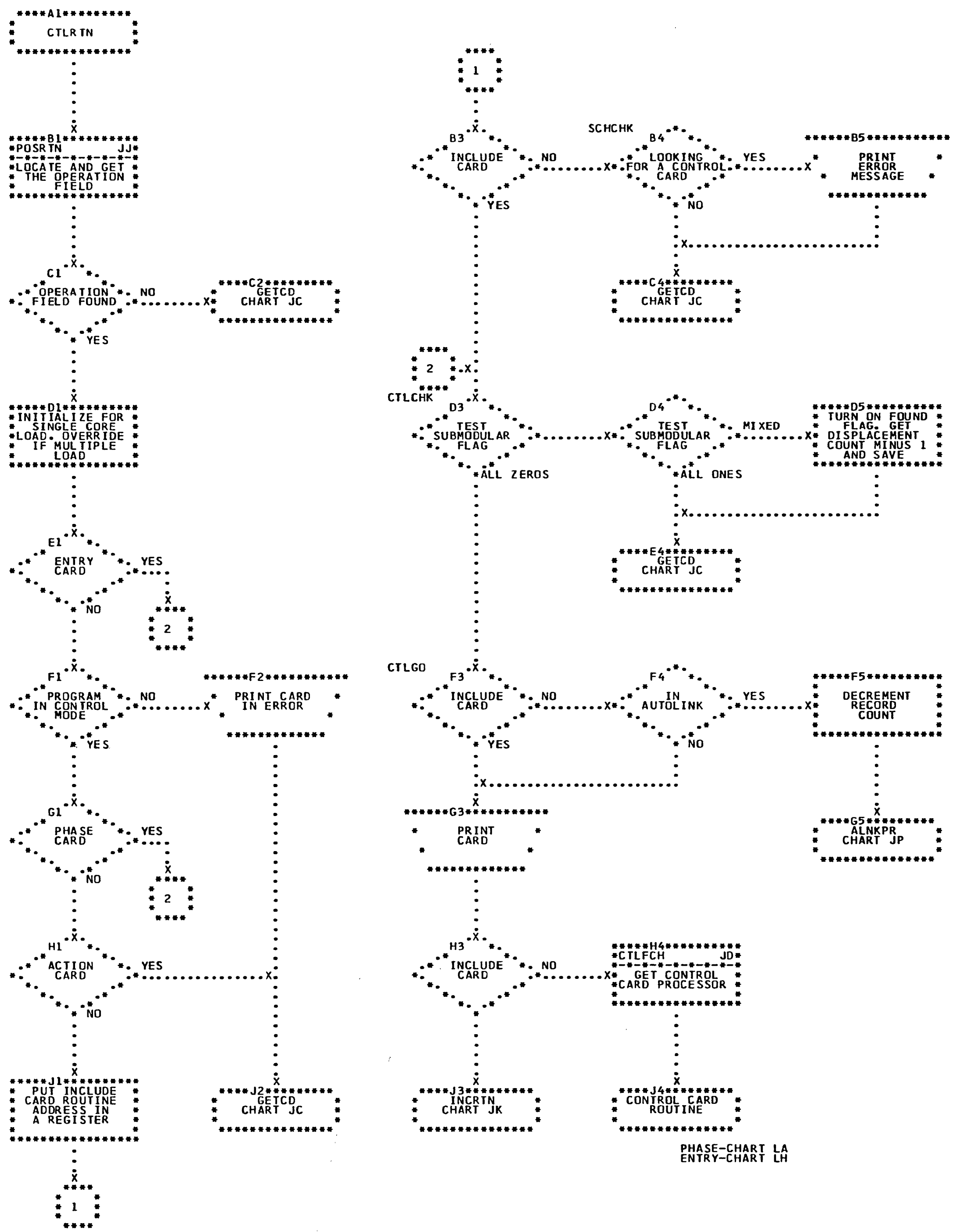


Chart JH. Identify Control Card

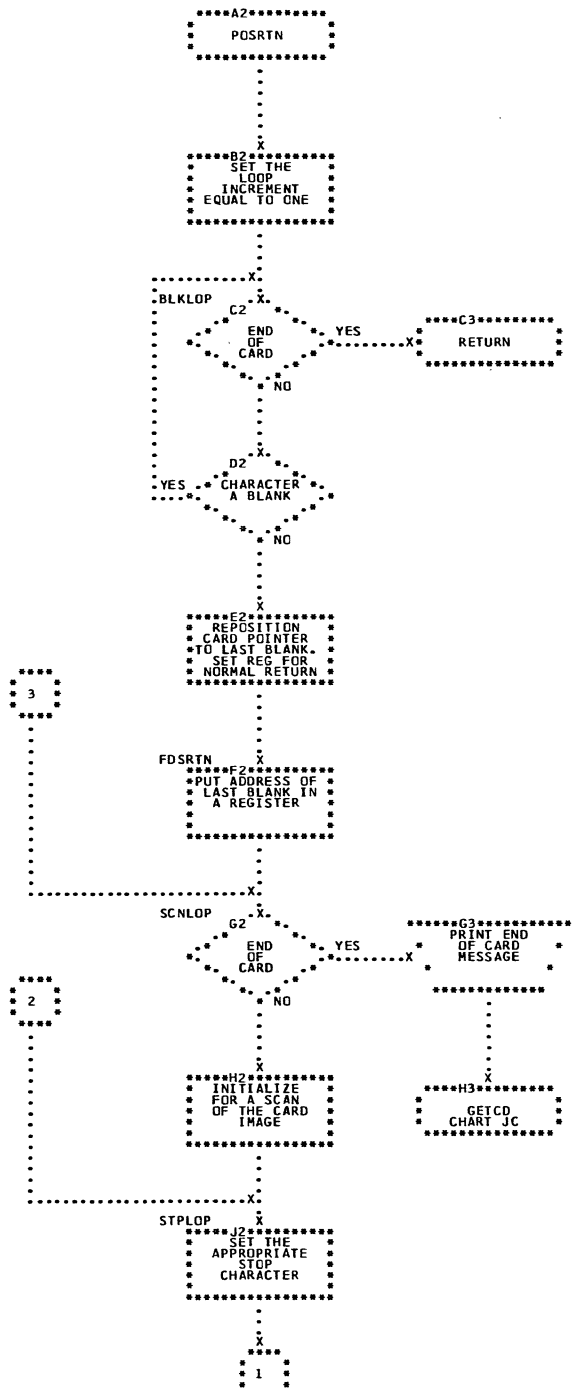


Chart JJ. Position to Operand Subroutine

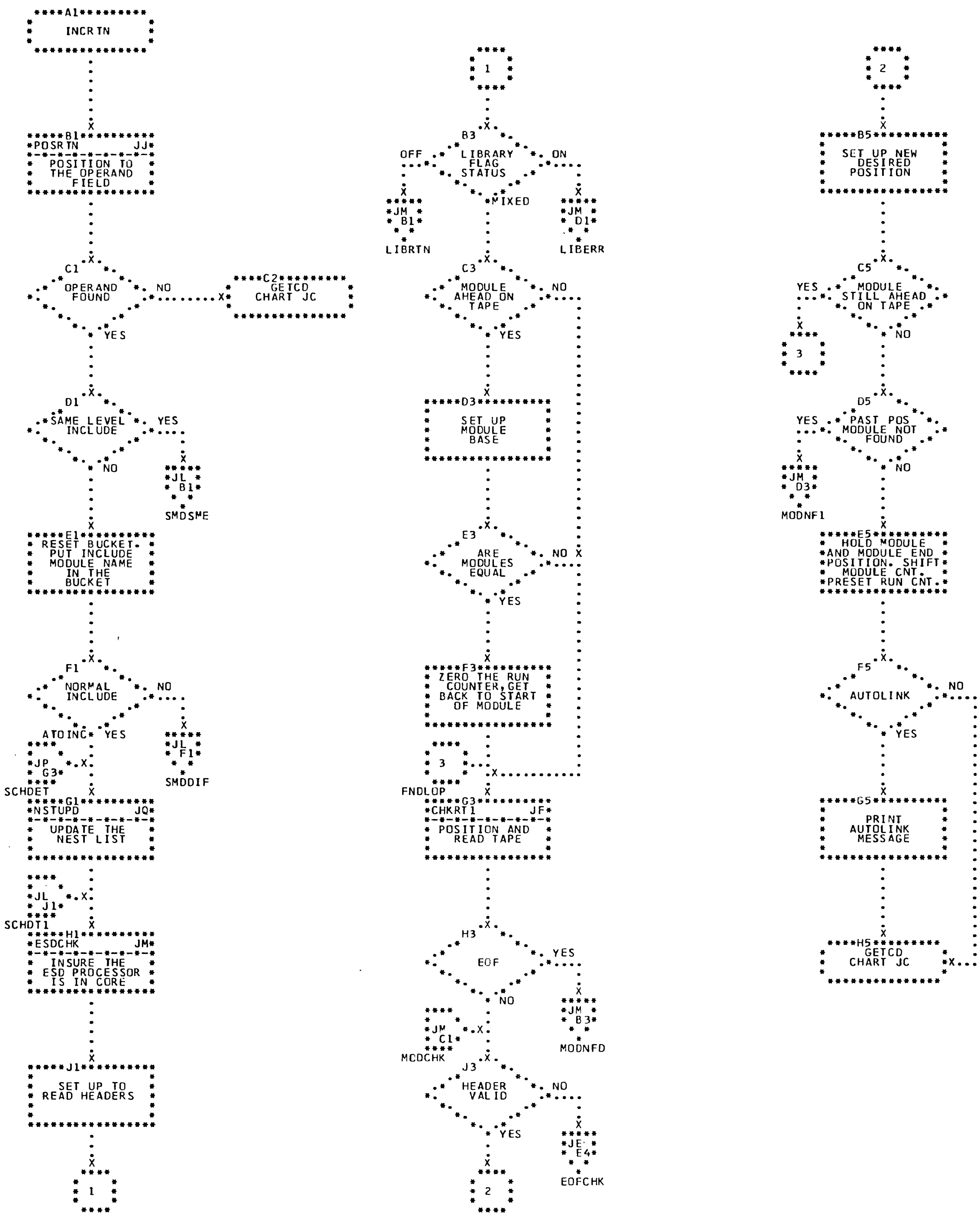


Chart JK. INCLUD (Part 1 of 2)

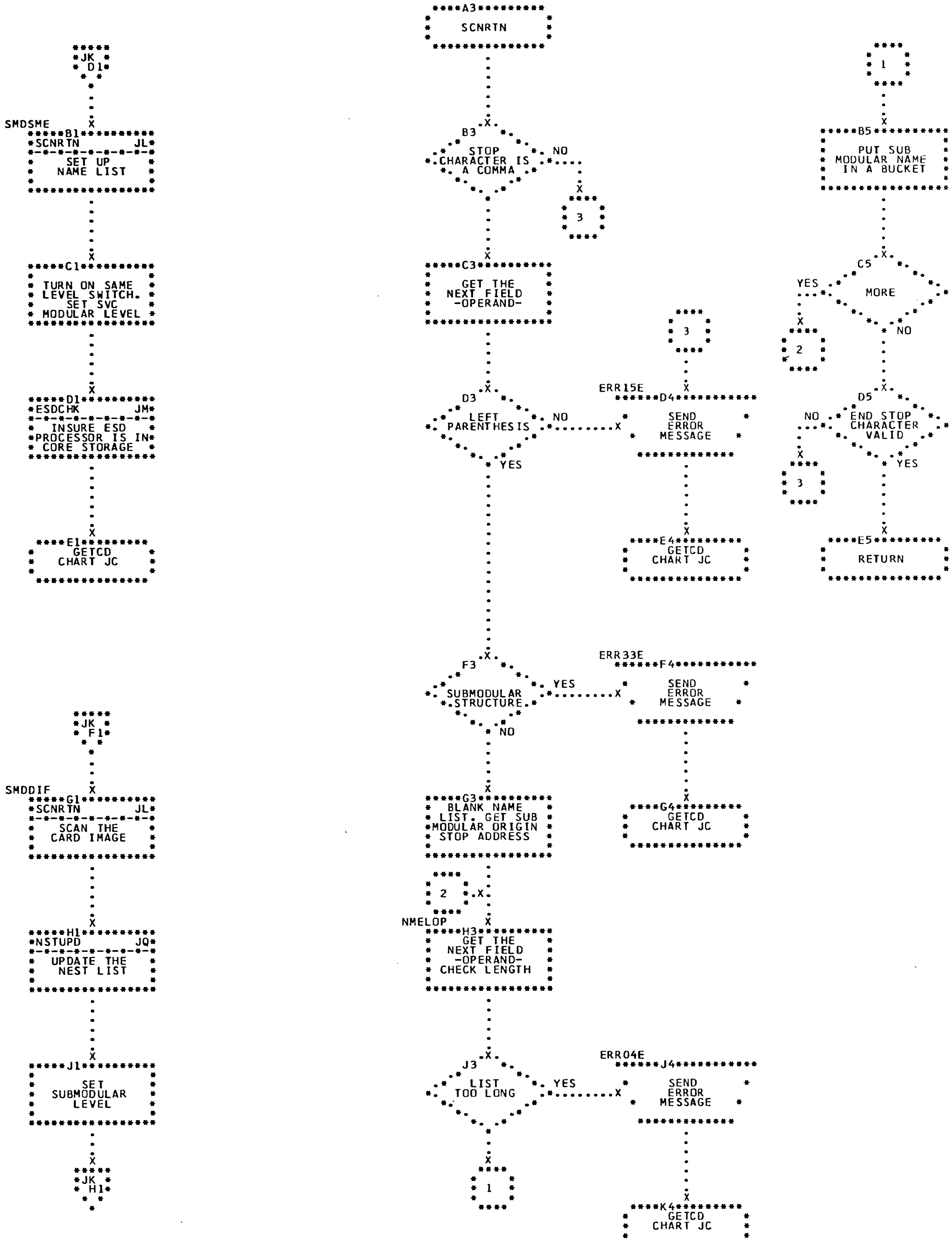


Chart JL. INCLUD (Part 2 of 2)

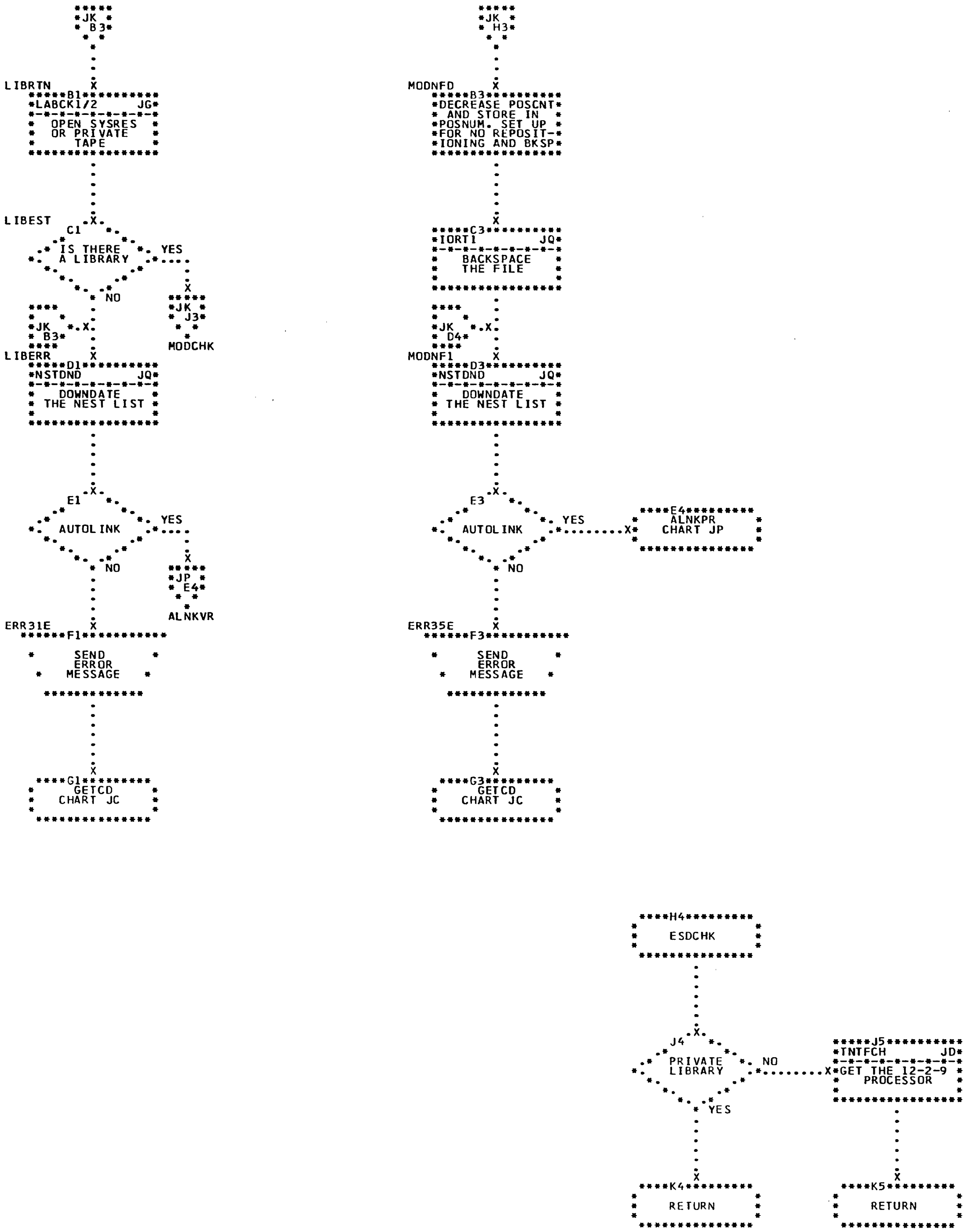


Chart JM. Scan Card Subroutine

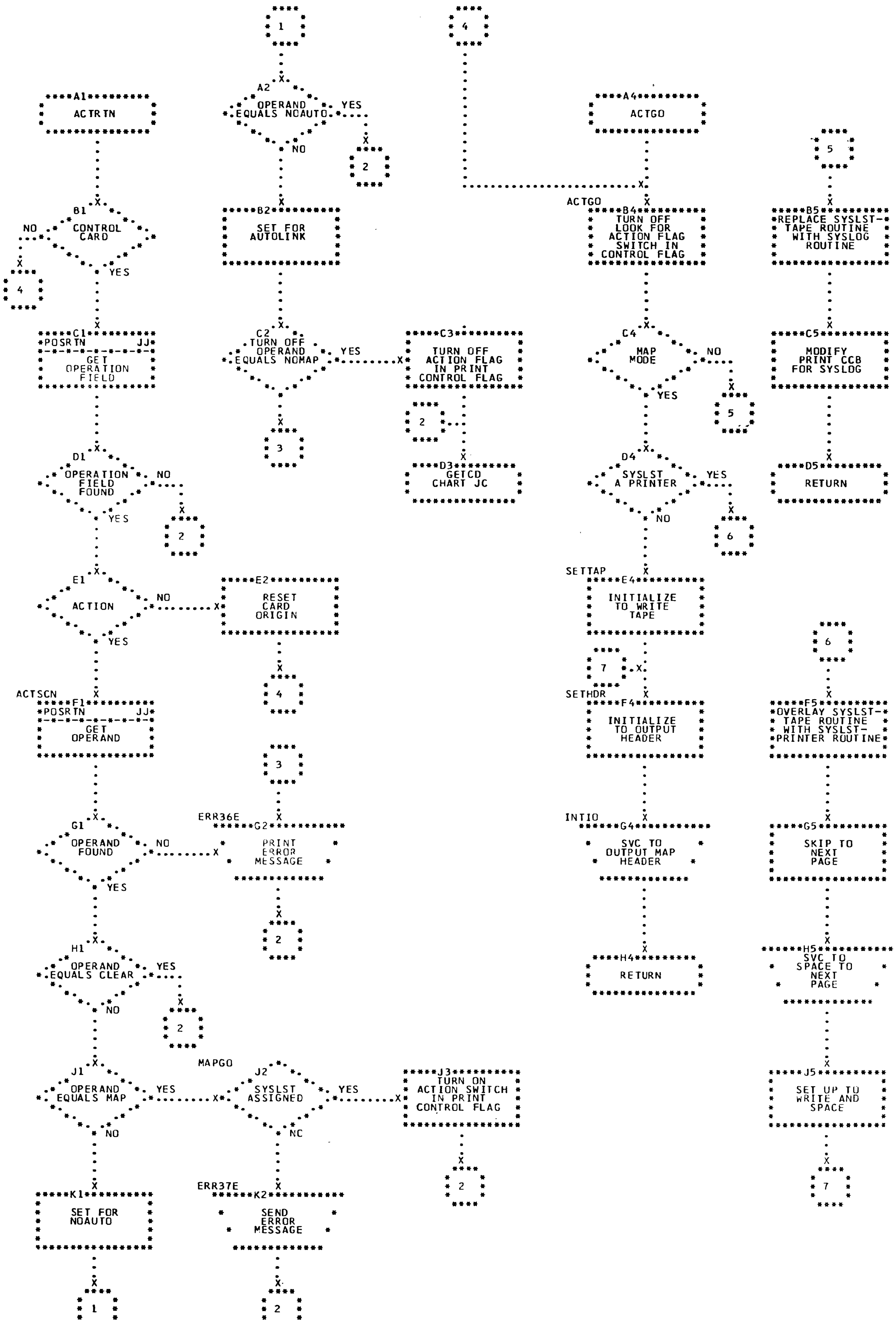


Chart JN. ACTION

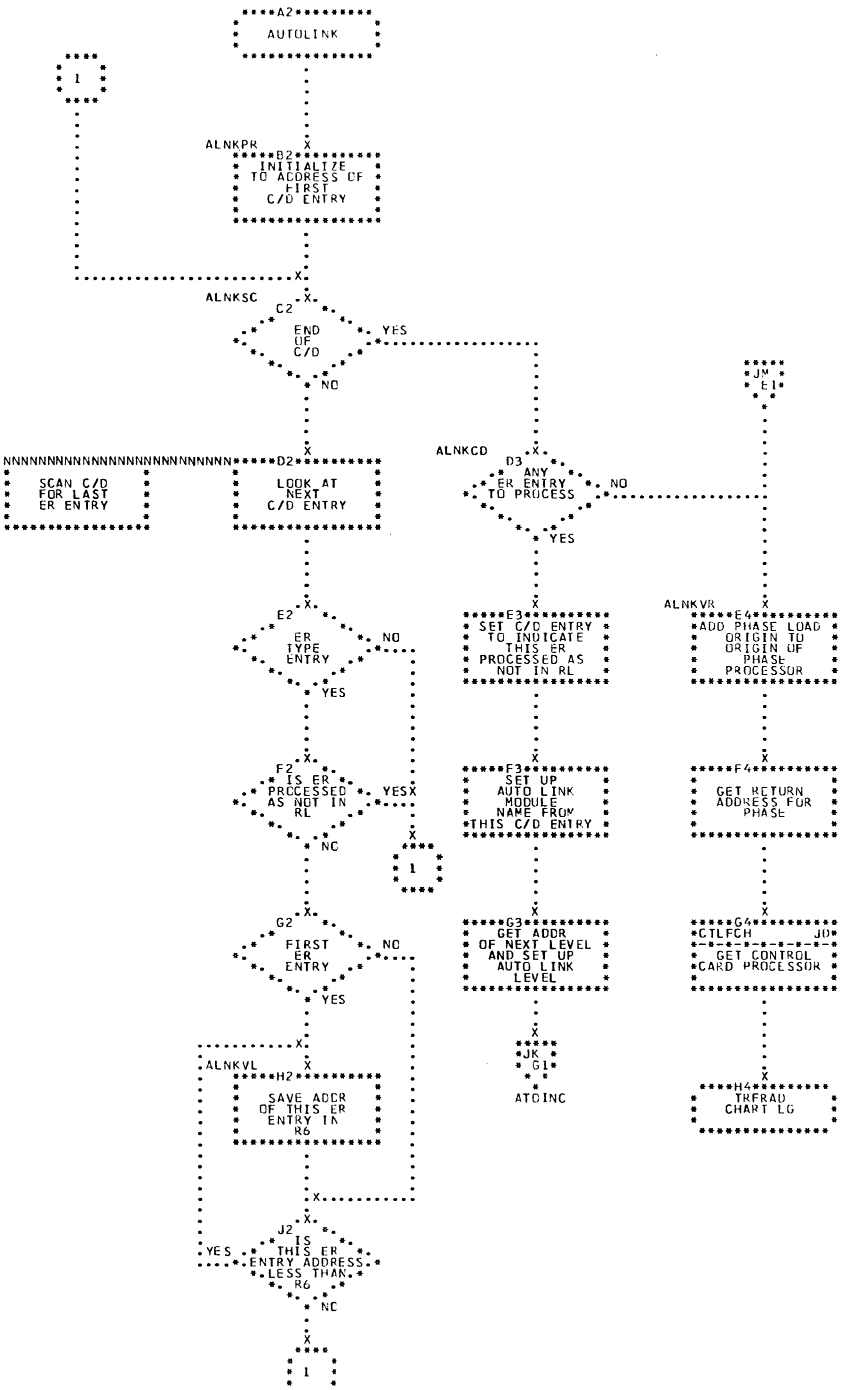


Chart JP. AUTOLINK

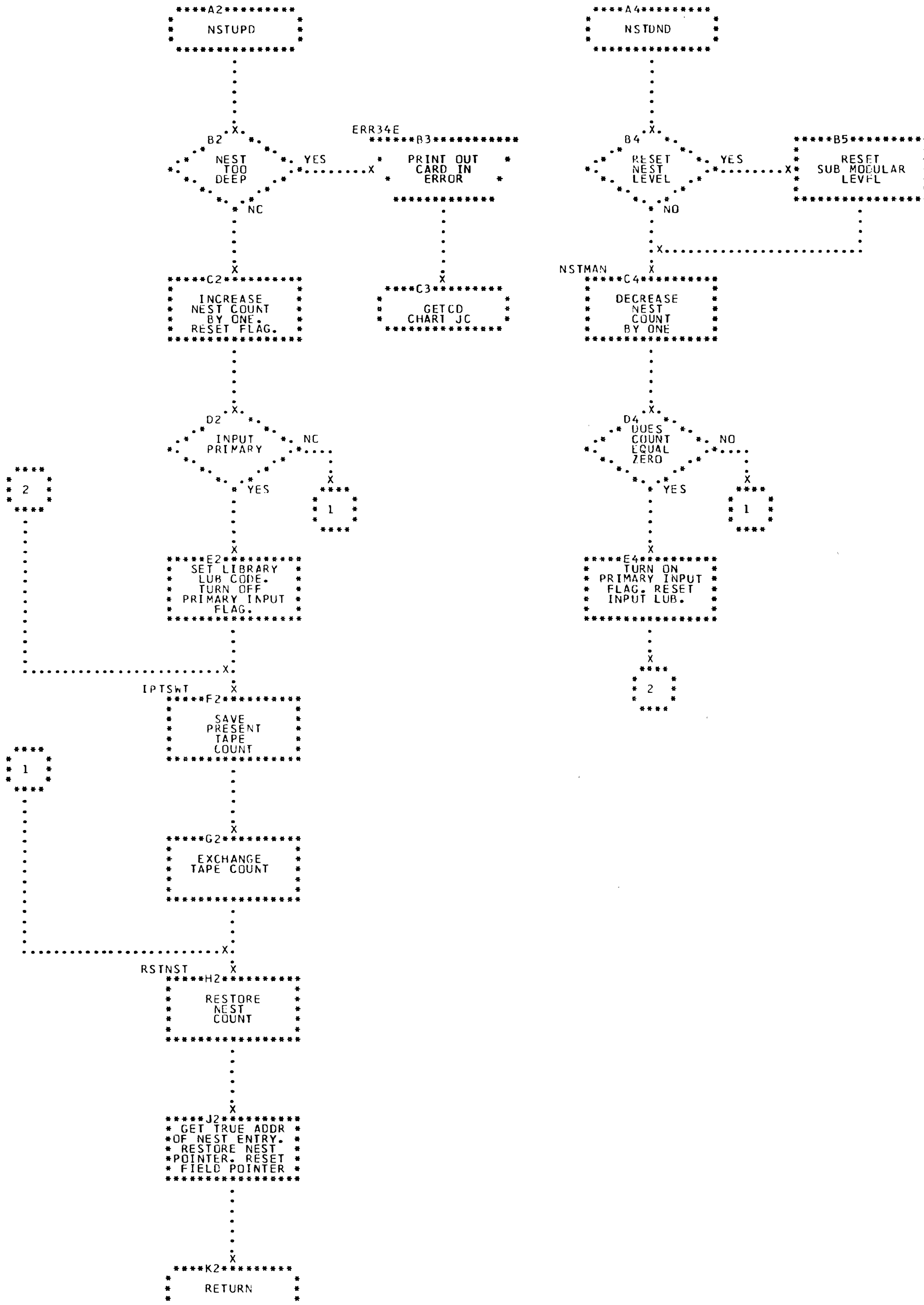


Chart JQ. Nesting Subroutines

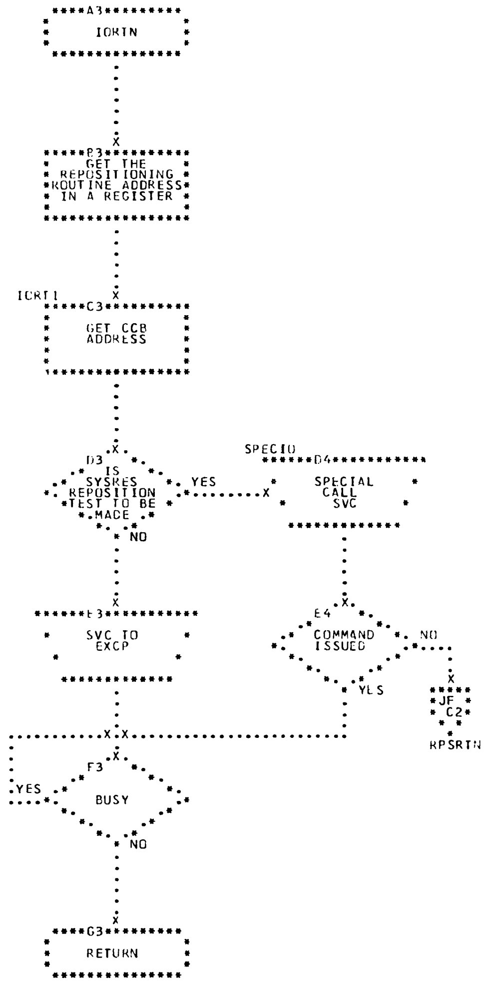


Chart JR. I/O Subroutine

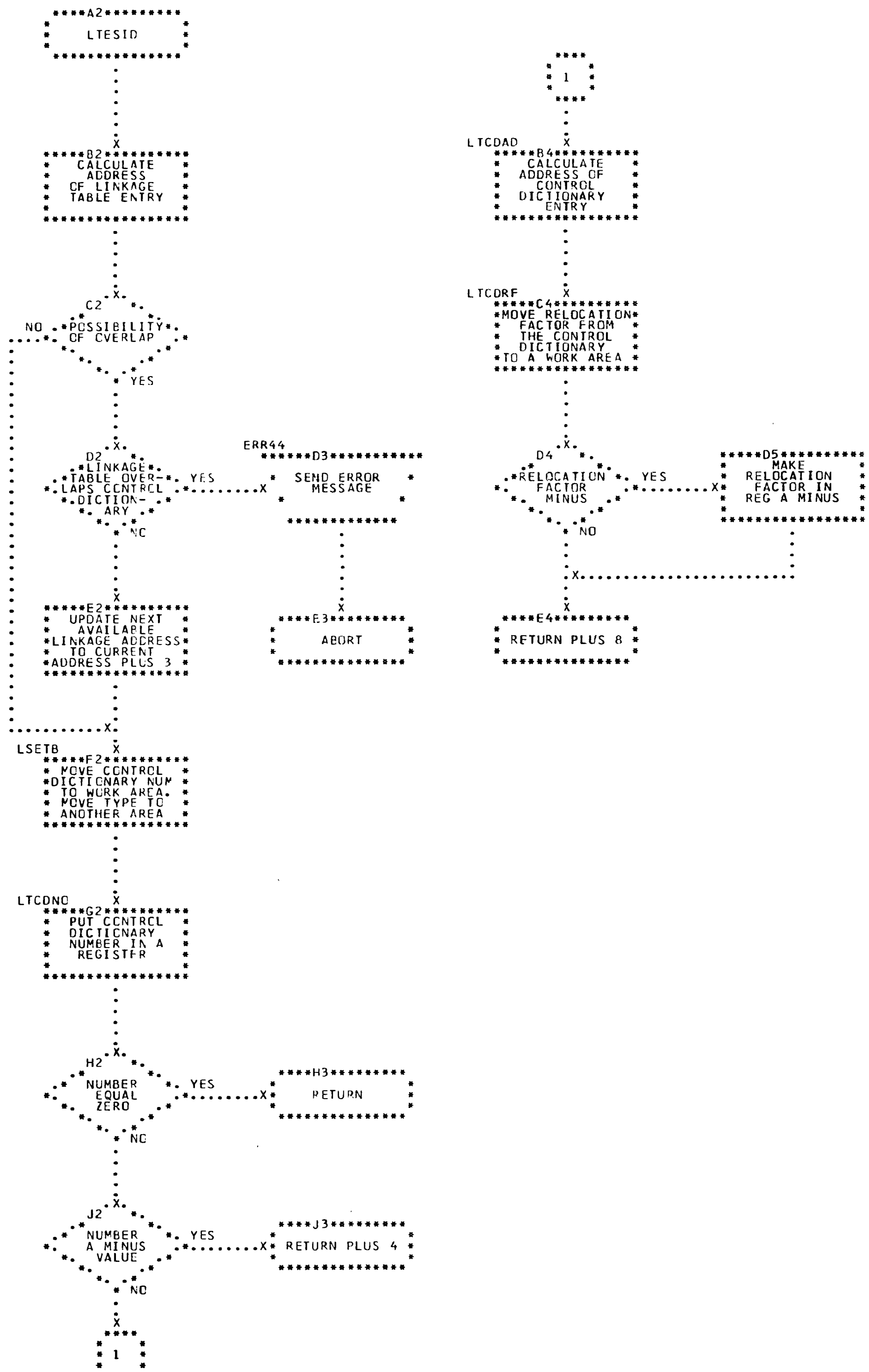


Chart JS. ESID Number to Control Dictionary Subroutine

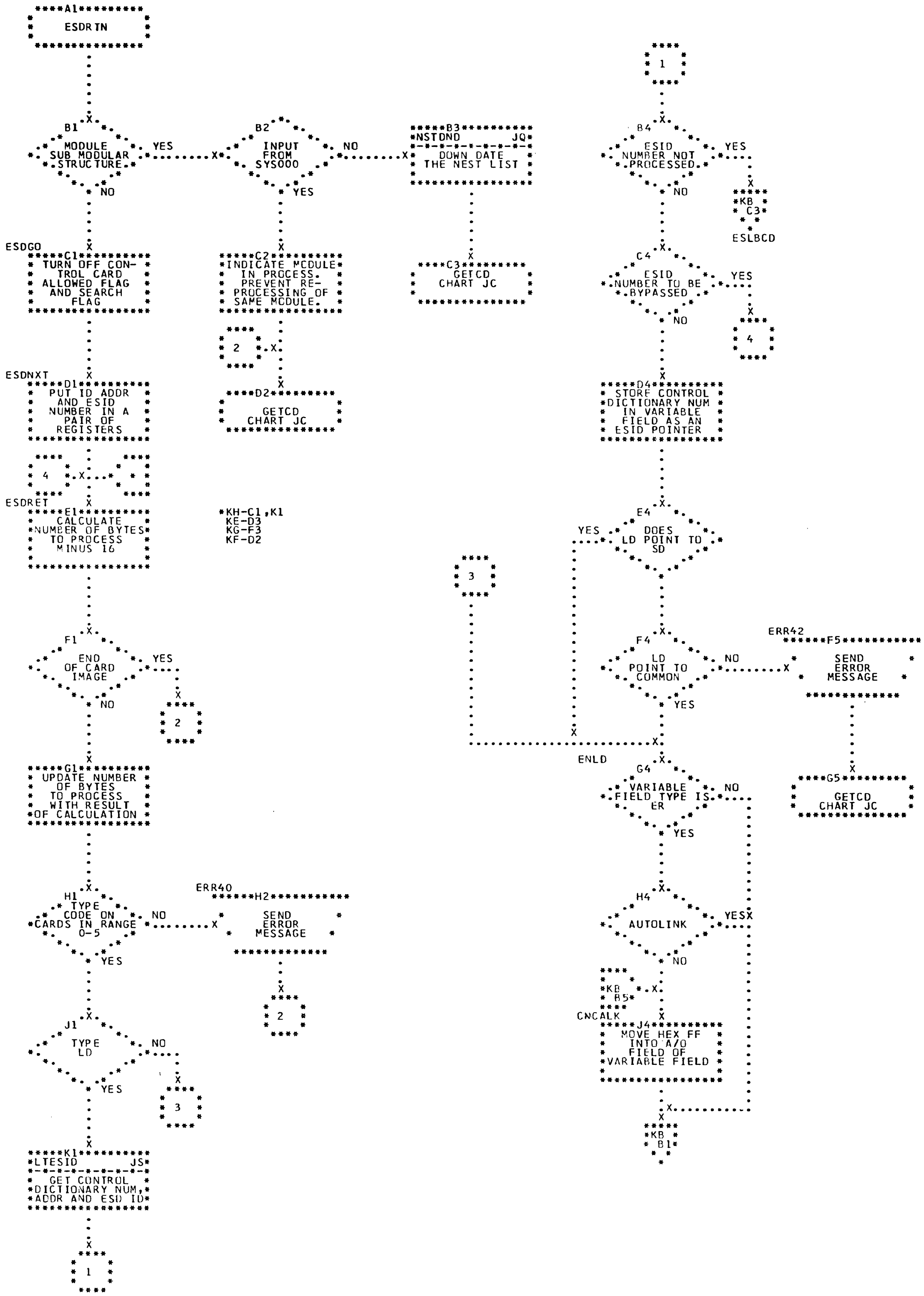


Chart KA. ESD (Part 1 of 8)

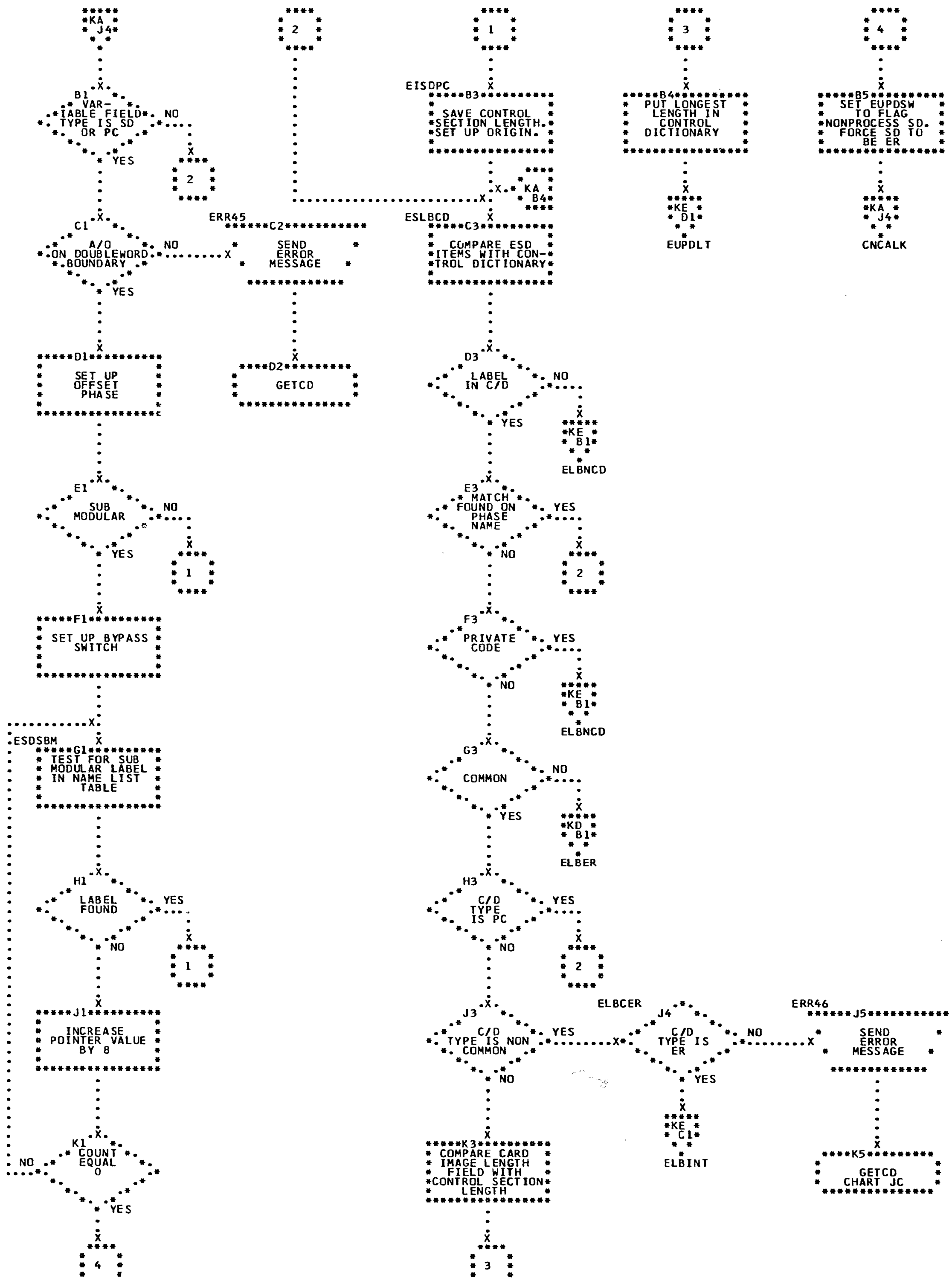


Chart KB. ESD (Part 2 of 8)

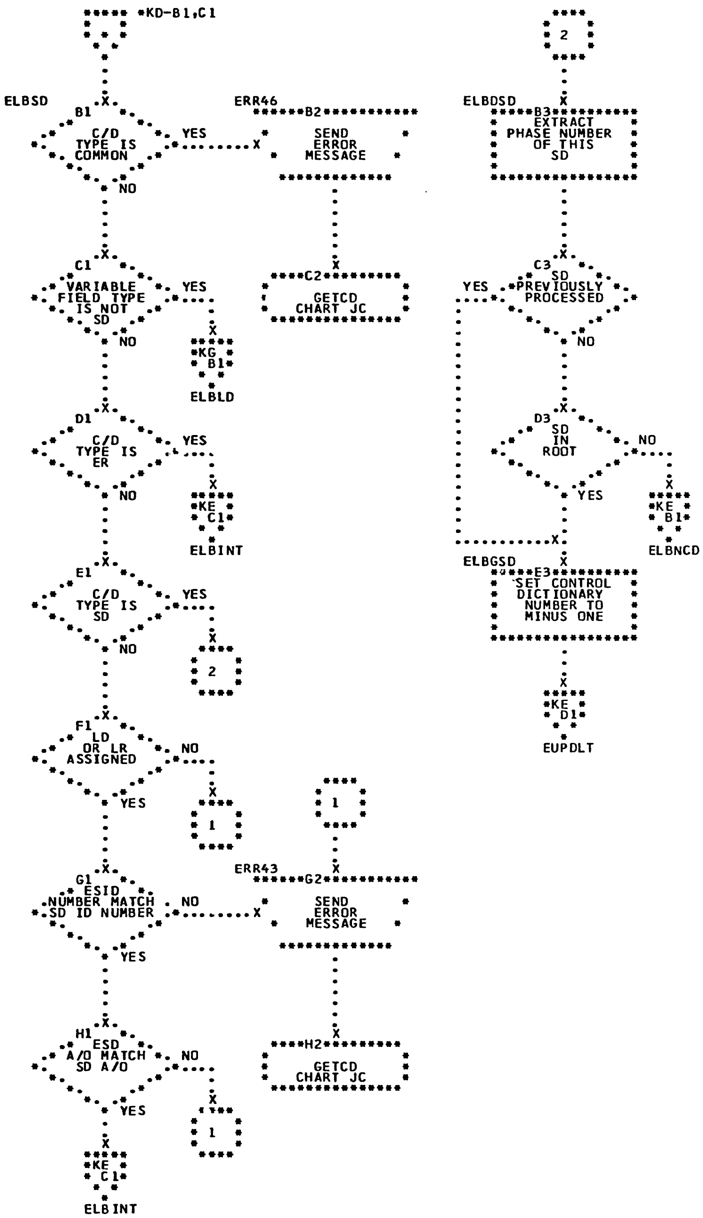


Chart KC. ESD (Part 3 of 8)

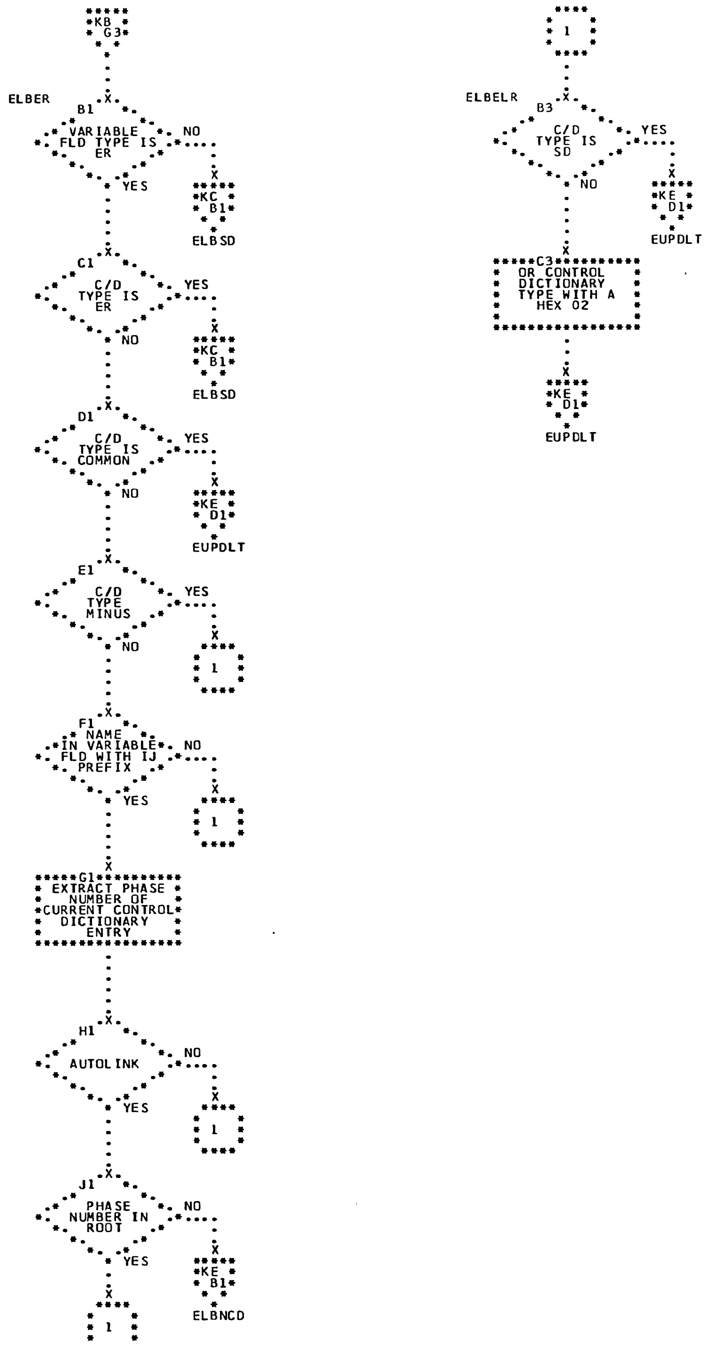


Chart KD. ESD (Part 4 of 8)

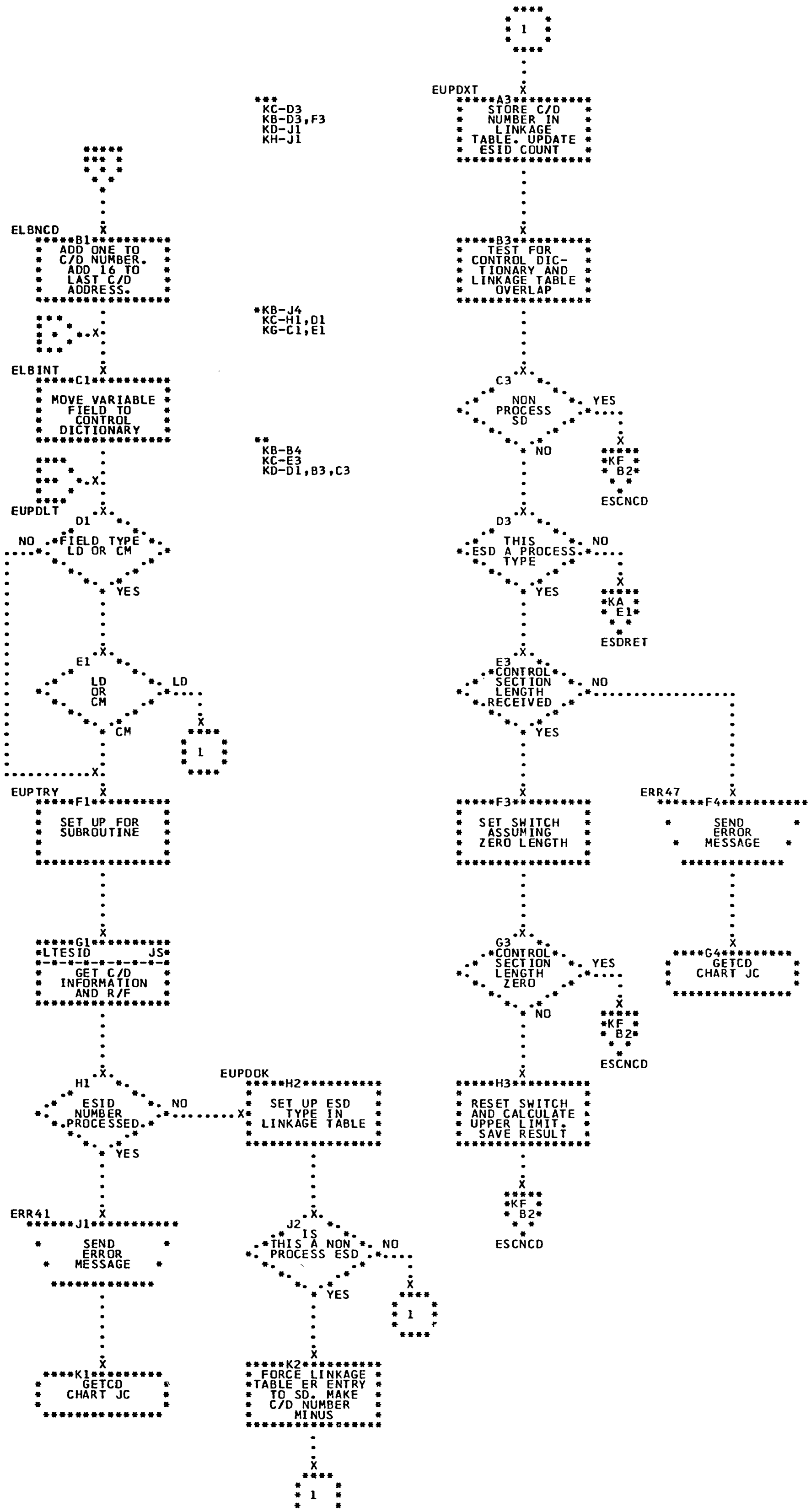


Chart KE. ESD (Part 5 of 8)

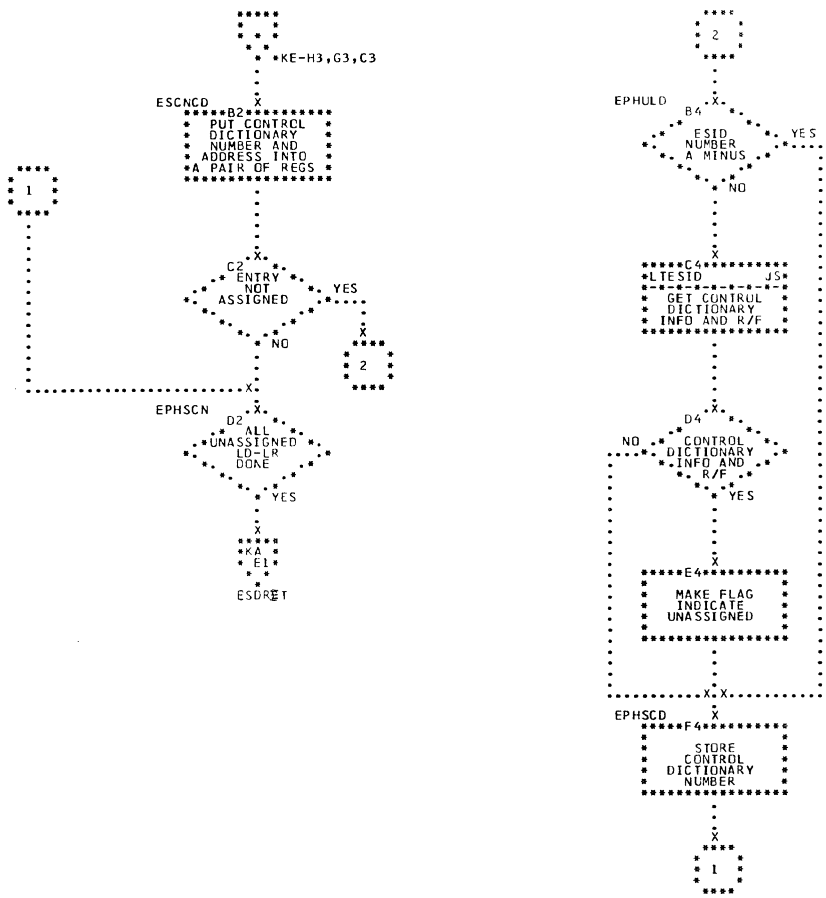


Chart KF. ESD (Part 6 of 8)

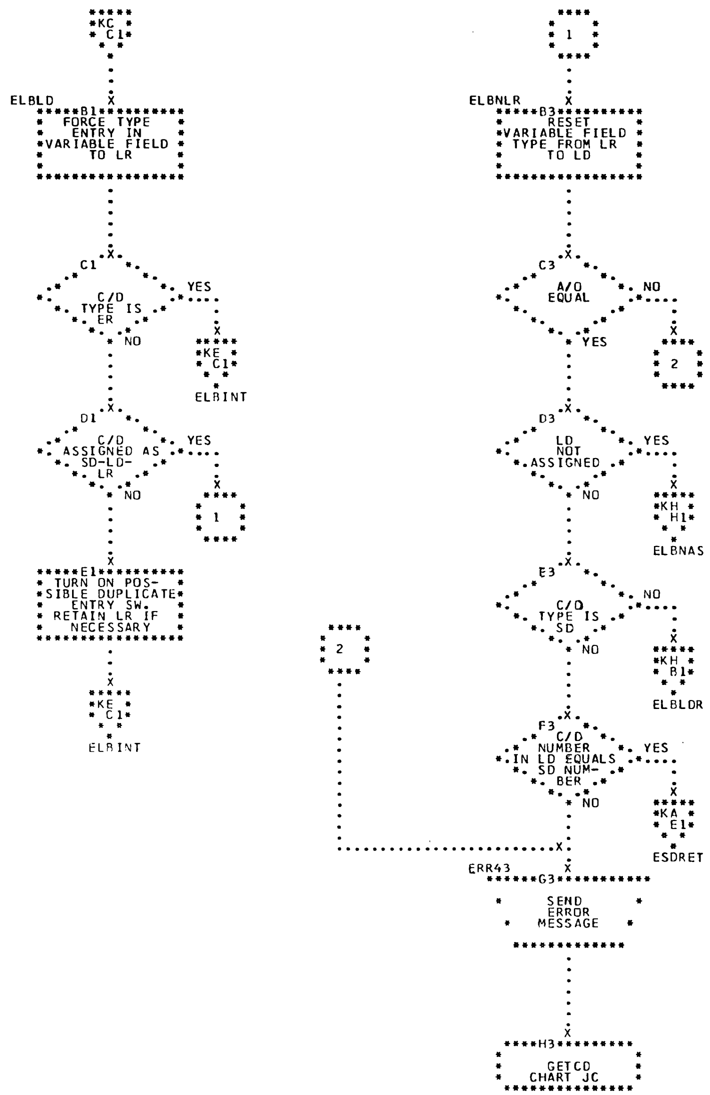


Chart KG. ESD (Part 7 of 8)

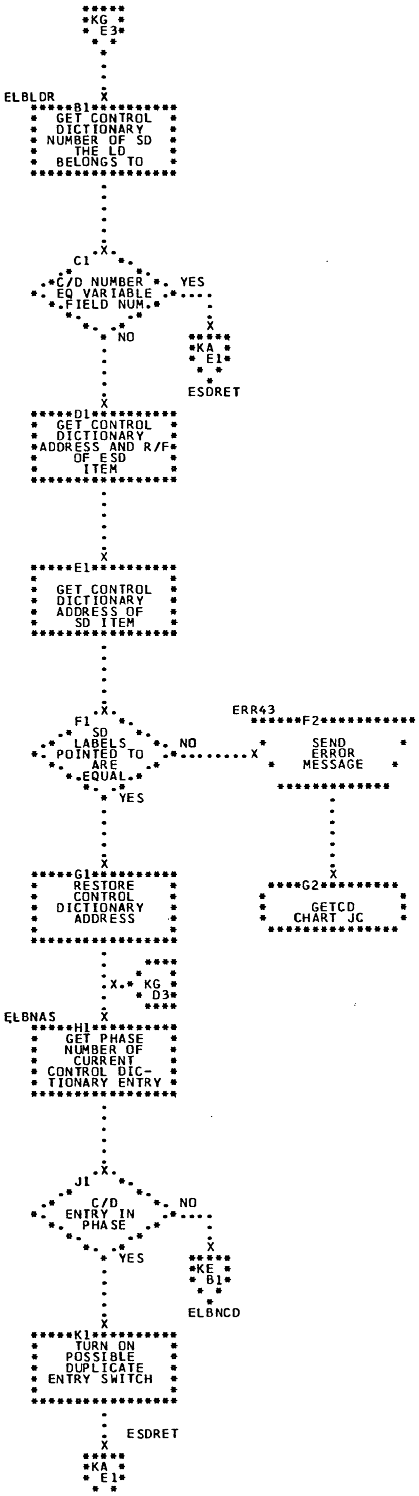


Chart KH. ESD (Part 8 of 8)

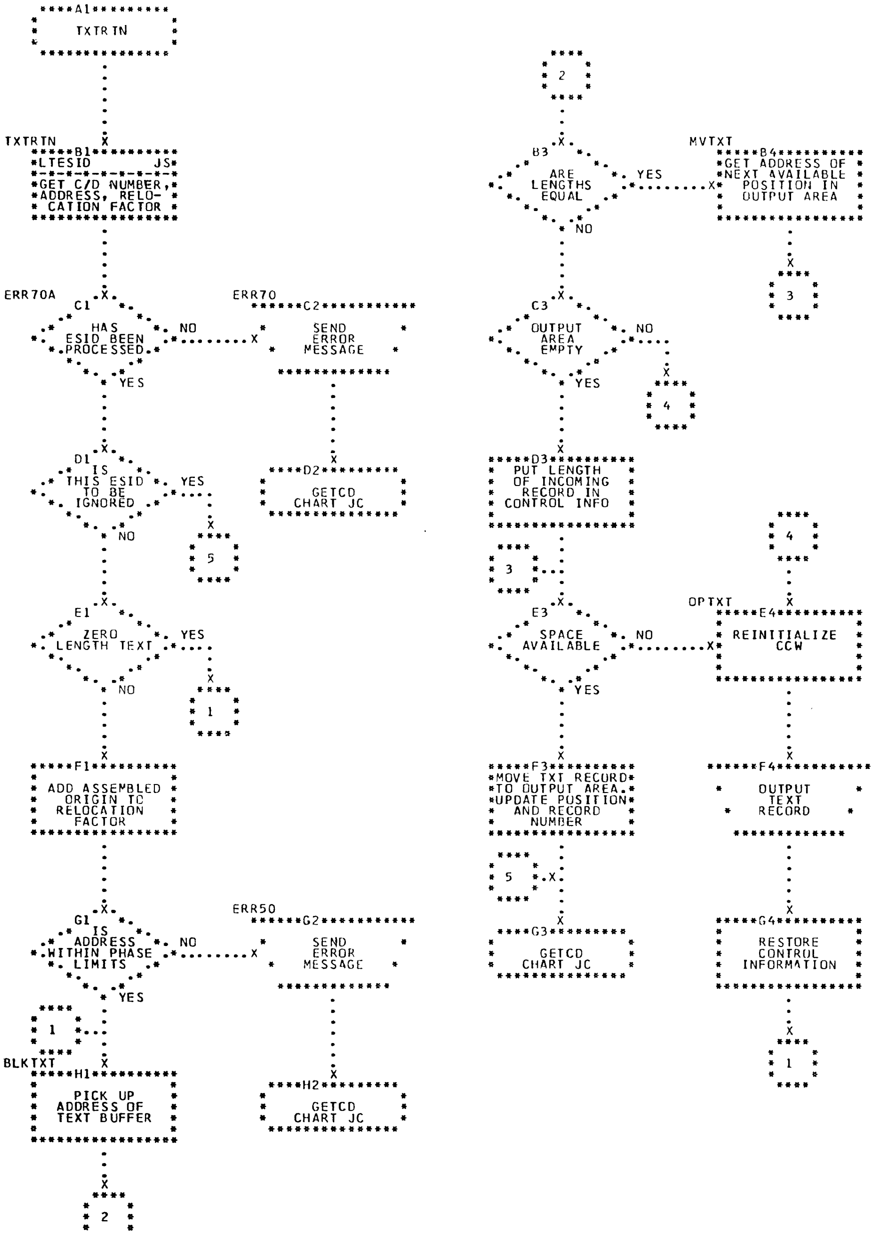

A1
TXTRTN
TXTRTN B1
LTESID JS
GET C/D NUMBER, ADDRESS, RELOCATION FACTOR
ERR70A C1
HAS ESID BEEN PROCESSED
NO
YES
ERR70 C2
SEND ERROR MESSAGE
D1
IS THIS ESID TO BE IGNORED
YES
NO
5
D2
GETCD CHART JC
E1
ZERO LENGTH TEXT
YES
NO
1
F1
ADD ASSEMBLED ORIGIN TO RELOCATION FACTOR
G1
IS ADDRESS WITHIN PHASE LIMITS
NO
YES
ERR50 G2
SEND ERROR MESSAGE
1
BLKTXT H1
PICK UP ADDRESS OF TEXT BUFFER
H2
GETCD CHART JC
2
2
B3
ARE LENGTHS EQUAL
YES
NO
MVTXT B4
GET ADDRESS OF NEXT AVAILABLE POSITION IN OUTPUT AREA
3
C3
OUTPUT AREA EMPTY
NO
YES
4
D3
PUT LENGTH OF INCOMING RECORD IN CONTROL INFO
3
4
E3
SPACE AVAILABLE
NO
YES
OPTXT E4
REINITIALIZE CCW
F3
MOVE TXT RECORD TO OUTPUT AREA. UPDATE POSITION AND RECORD NUMBER
F4
OUTPUT TEXT RECORD
5
G3
GETCD CHART JC
G4
RESTORE CONTROL INFORMATION
1

Chart KJ. TXT

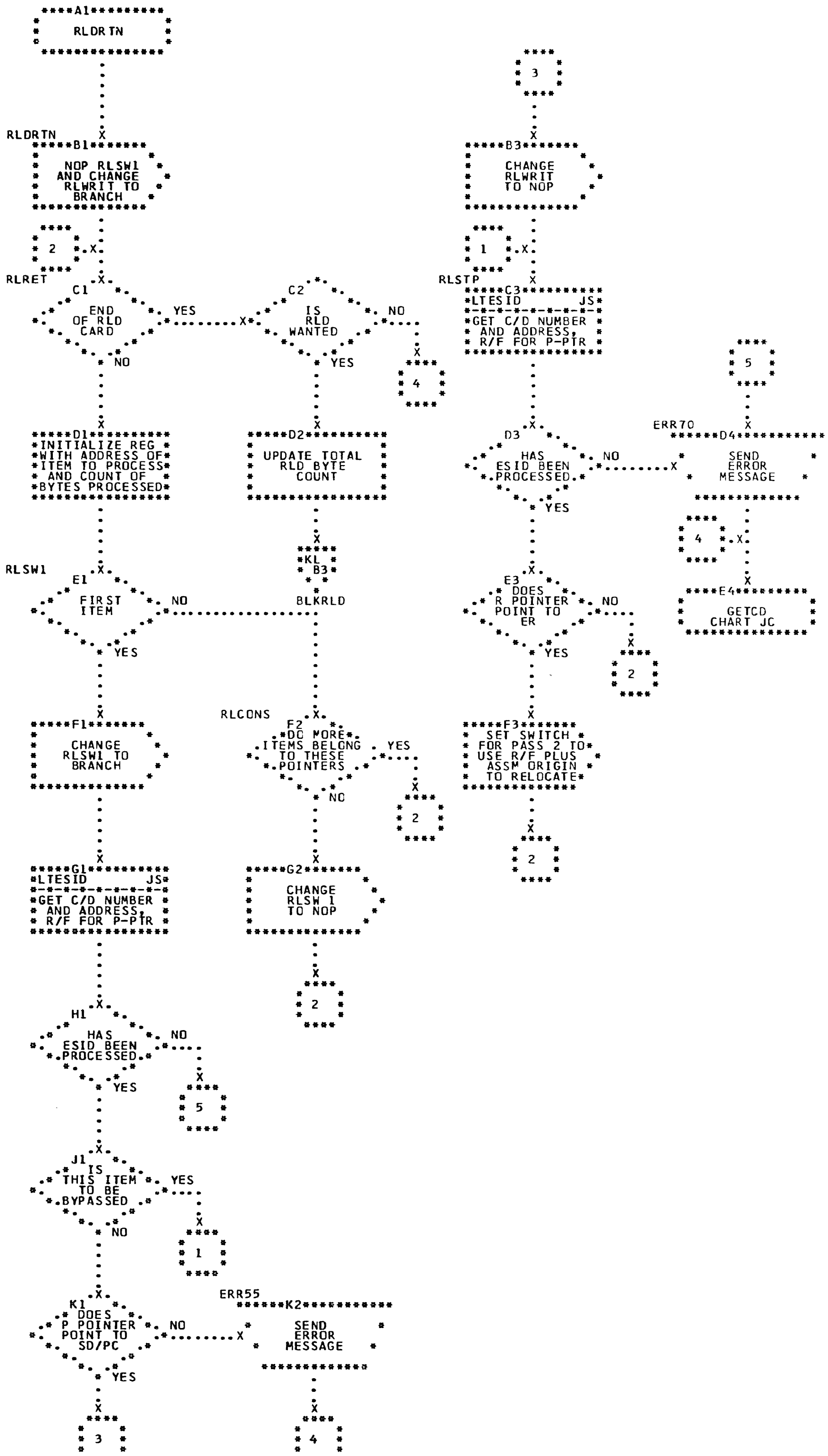


Chart KK. RLD (Part 1 of 2)

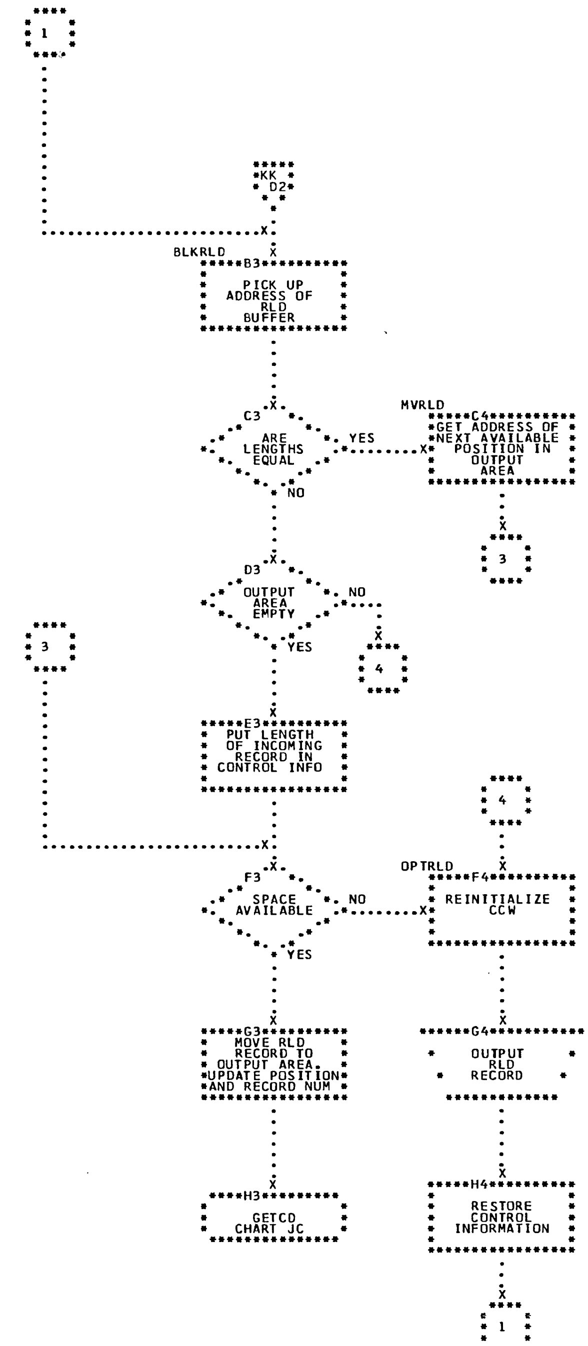


Chart KL. RLD (Part 2 of 2)

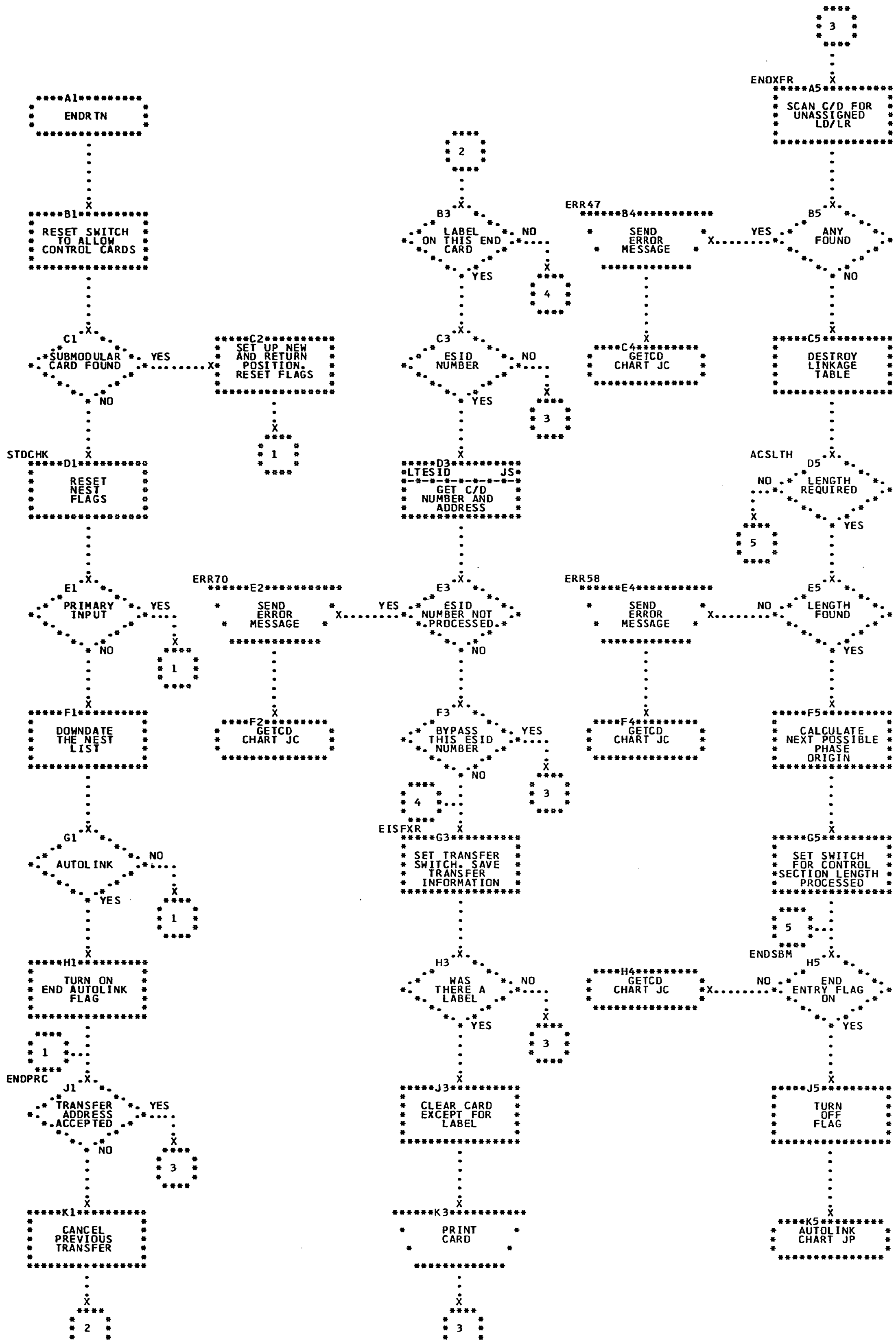

ENDRTN
RESET SWITCH TO ALLOW CONTROL CARDS
SUBMODULAR CARD FOUND
SET UP NEW AND RETURN POSITION. RESET FLAGS
STDCHK
RESET NEST FLAGS
PRIMARY INPUT
DOWNDATE THE NEST LIST
AUTOLINK
TURN ON END AUTOLINK FLAG
ENDPRC
TRANSFER ADDRESS ACCEPTED
CANCEL PREVIOUS TRANSFER
LABEL ON THIS END CARD
ESID NUMBER
LTESID JS
GET C/D NUMBER AND ADDRESS
ERR70
SEND ERROR MESSAGE
GETCD CHART JC
ESID NUMBER NOT PROCESSED
BYPASS THIS ESID NUMBER
EISFXR
SET TRANSFER SWITCH. SAVE TRANSFER INFORMATION
WAS THERE A LABEL
CLEAR CARD EXCEPT FOR LABEL
PRINT CARD
ERR47
ERR58
ENDXFR
SCAN C/D FOR UNASSIGNED LD/LR
ANY FOUND
DESTROY LINKAGE TABLE
ACSLTH
LENGTH REQUIRED
LENGTH FOUND
CALCULATE NEXT POSSIBLE PHASE ORIGIN
SET SWITCH FOR CONTROL SECTION LENGTH PROCESSED
ENDSBM
END ENTRY FLAG ON
TURN OFF FLAG
AUTOLINK CHART JP

Chart KM. END

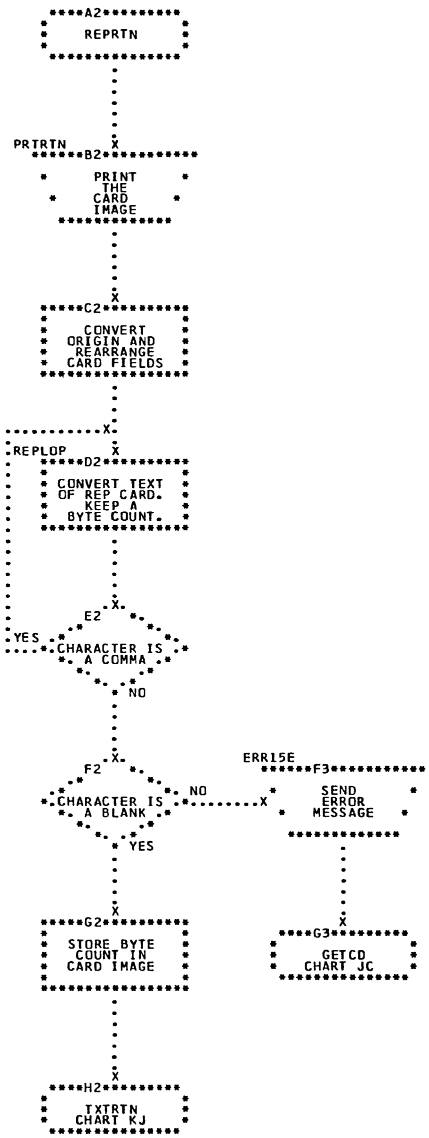

A2
REPRTN
PRTRTN
B2
PRINT THE CARD IMAGE
C2
CONVERT ORIGIN AND REARRANGE CARD FIELDS
REPLOP
D2
CONVERT TEXT OF REP CARD. KEEP A BYTE COUNT.
E2
CHARACTER IS A COMMA
YES
NO
F2
CHARACTER IS A BLANK
NO
YES
ERR15E
F3
SEND ERROR MESSAGE
G2
STORE BYTE COUNT IN CARD IMAGE
G3
GETCD CHART JC
H2
TXTRTN CHART KJ

Chart KN. REP

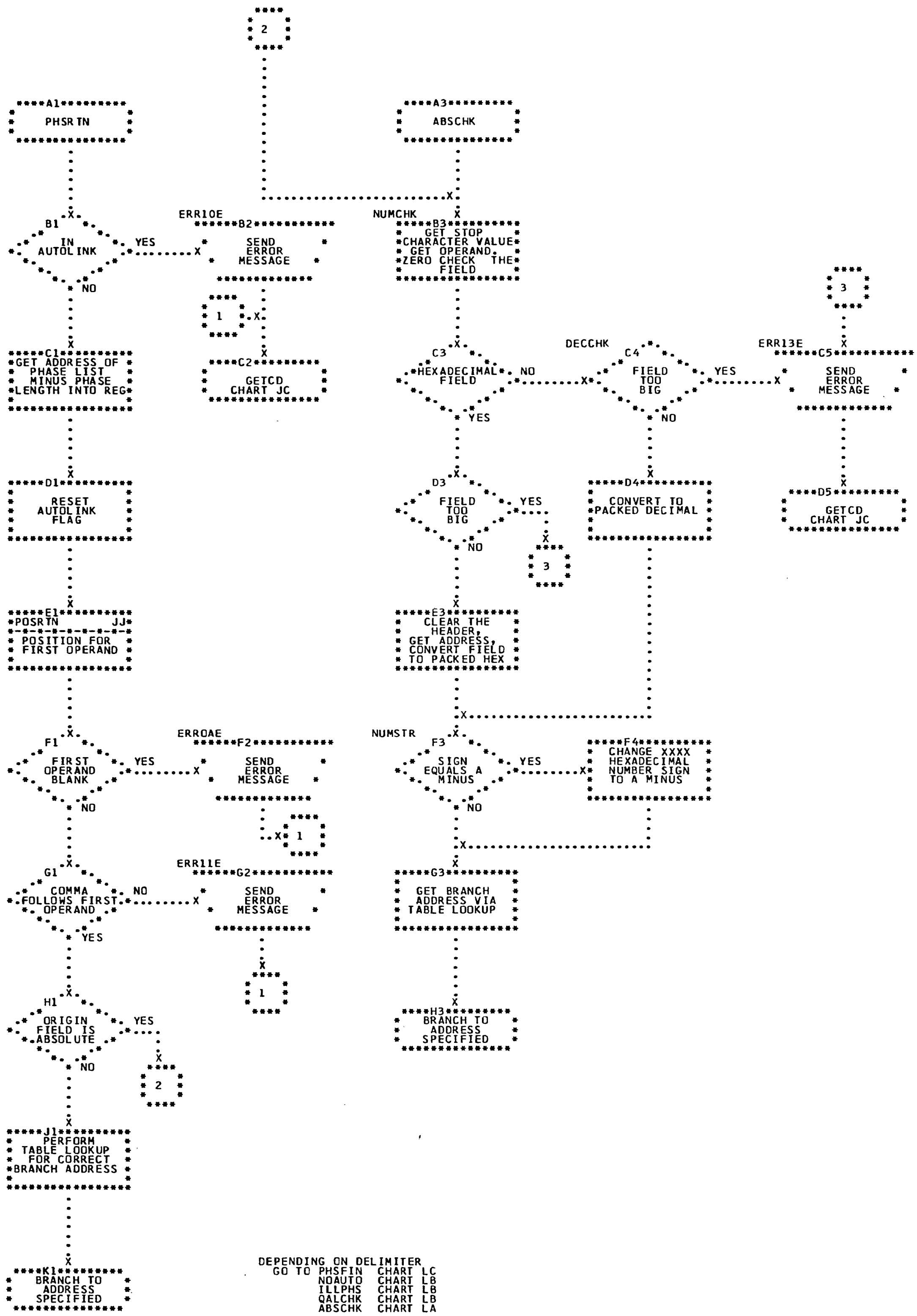


Chart LA. PHASE (Part 1 of 5)

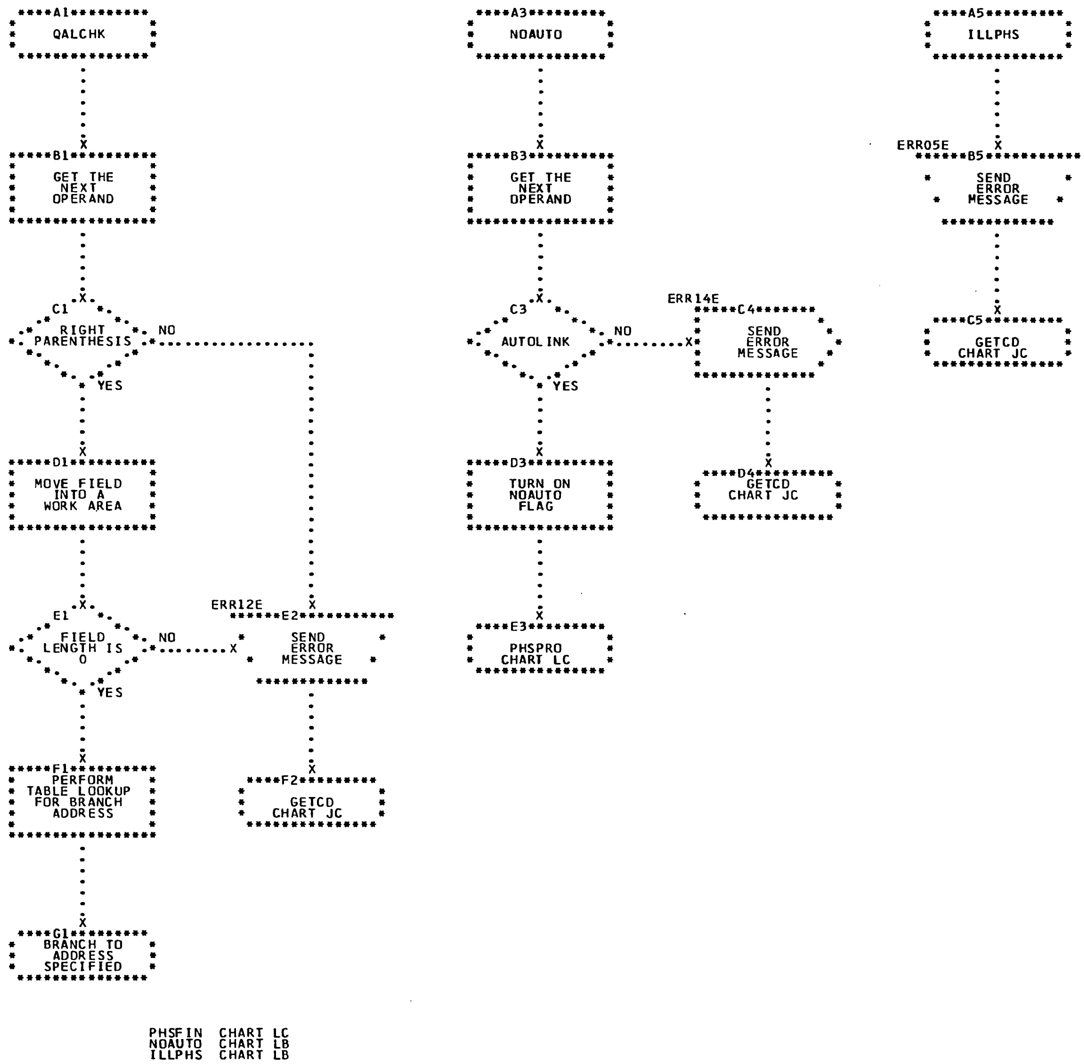


Chart LB. PHASE (Part 2 of 5)

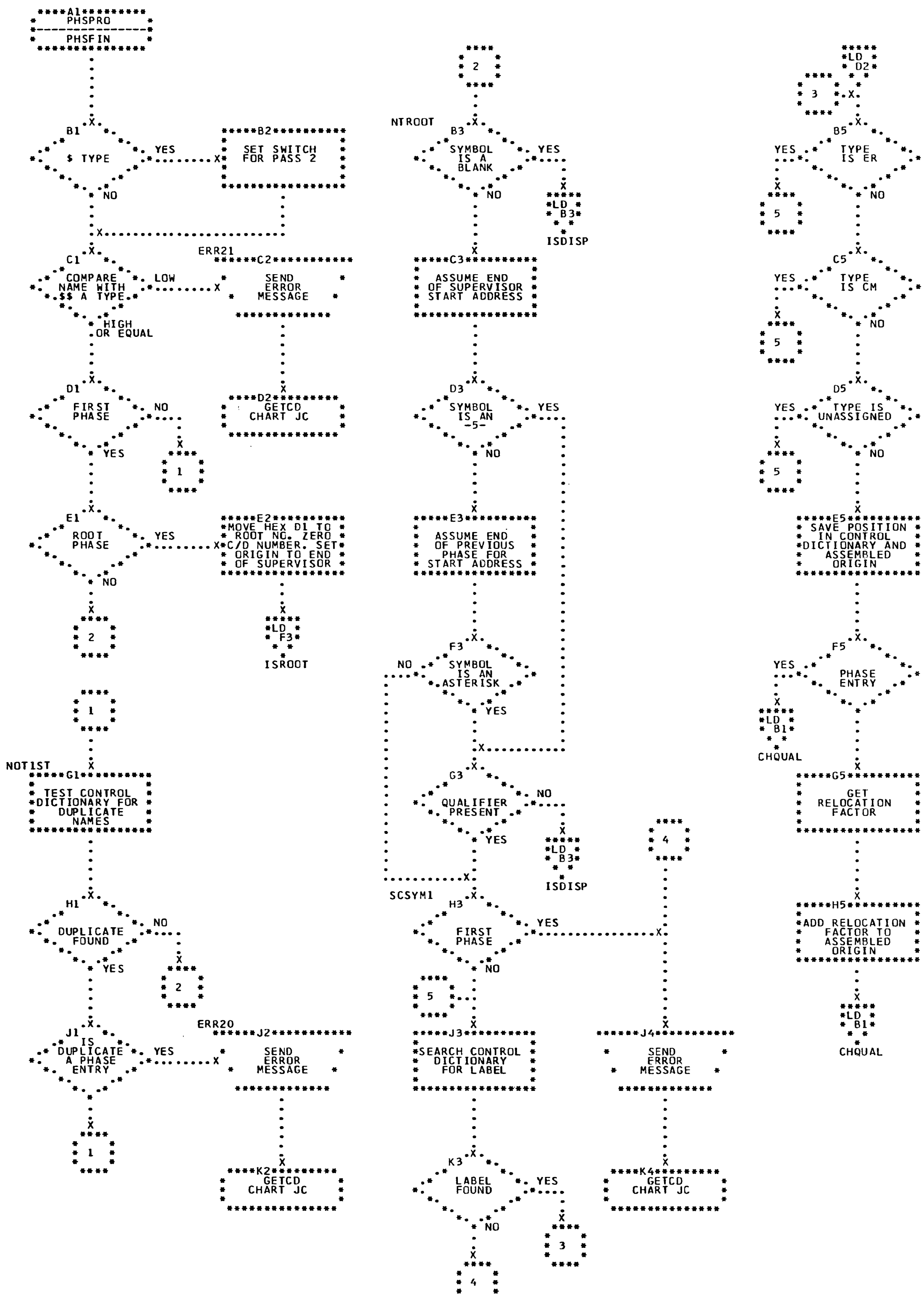


Chart LC. PHASE (Part 3 of 5)

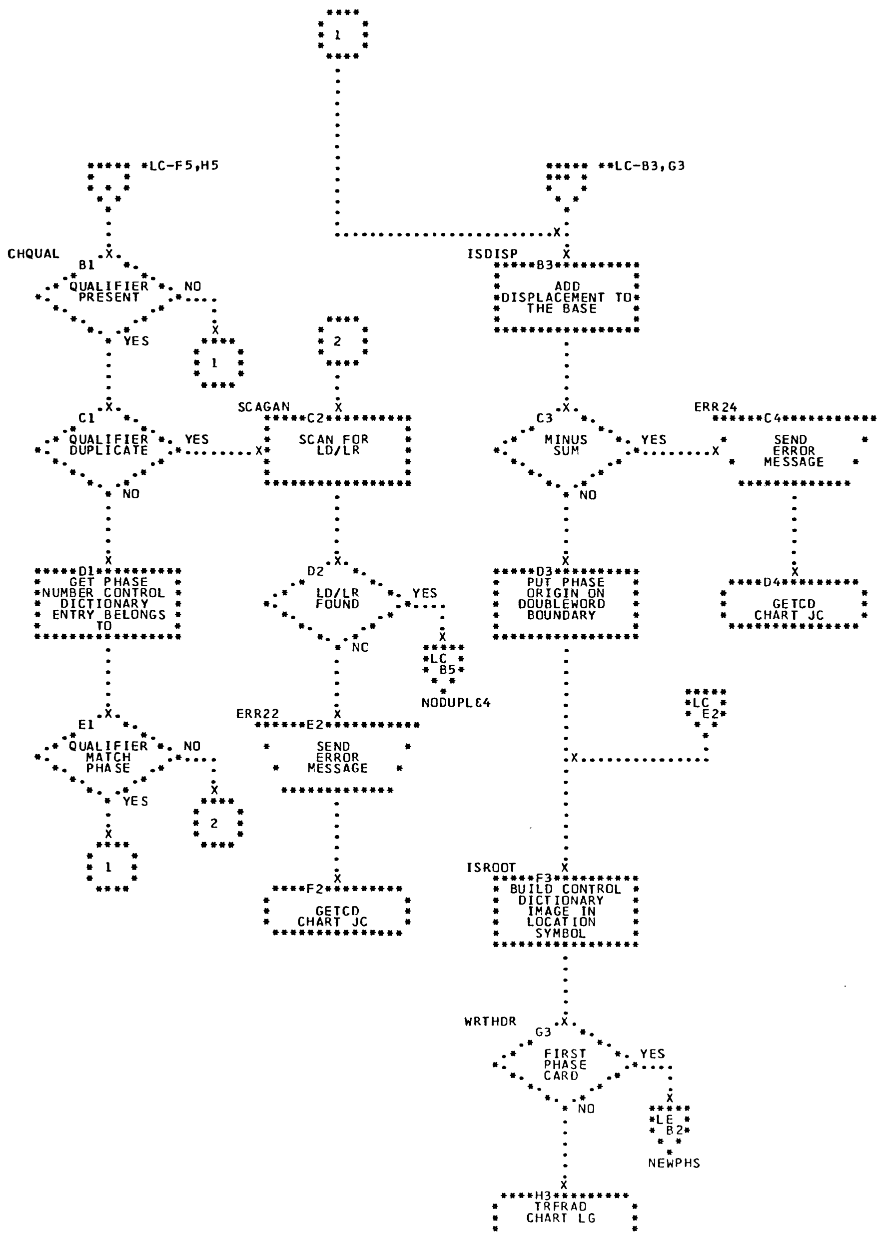


Chart LD. PHASE (Part 4 of 5)

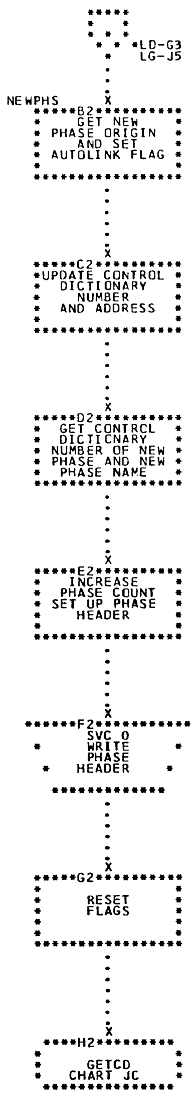


Chart LE. PHASE (Part 5 of 5)

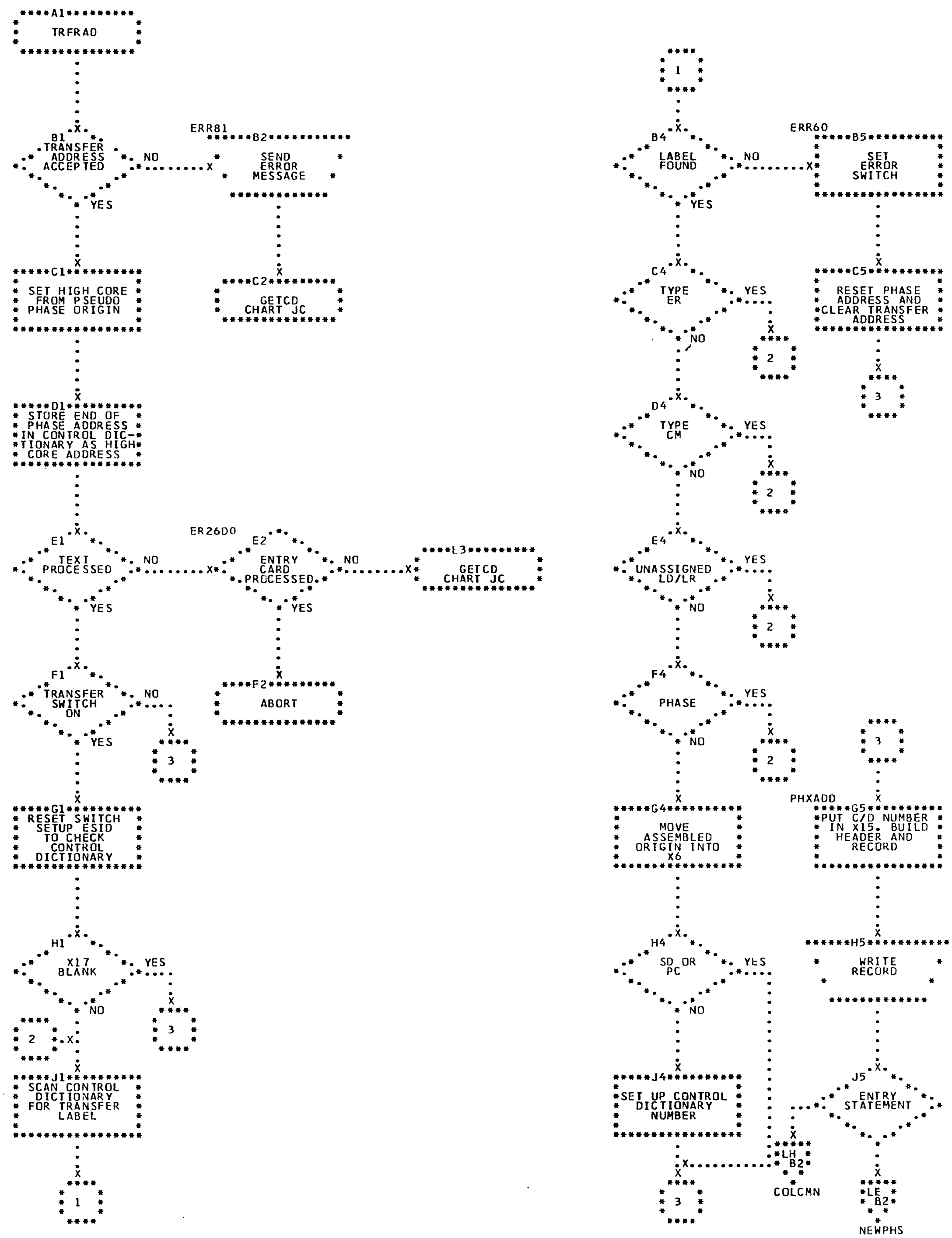


Chart LG. Determine Transfer Address

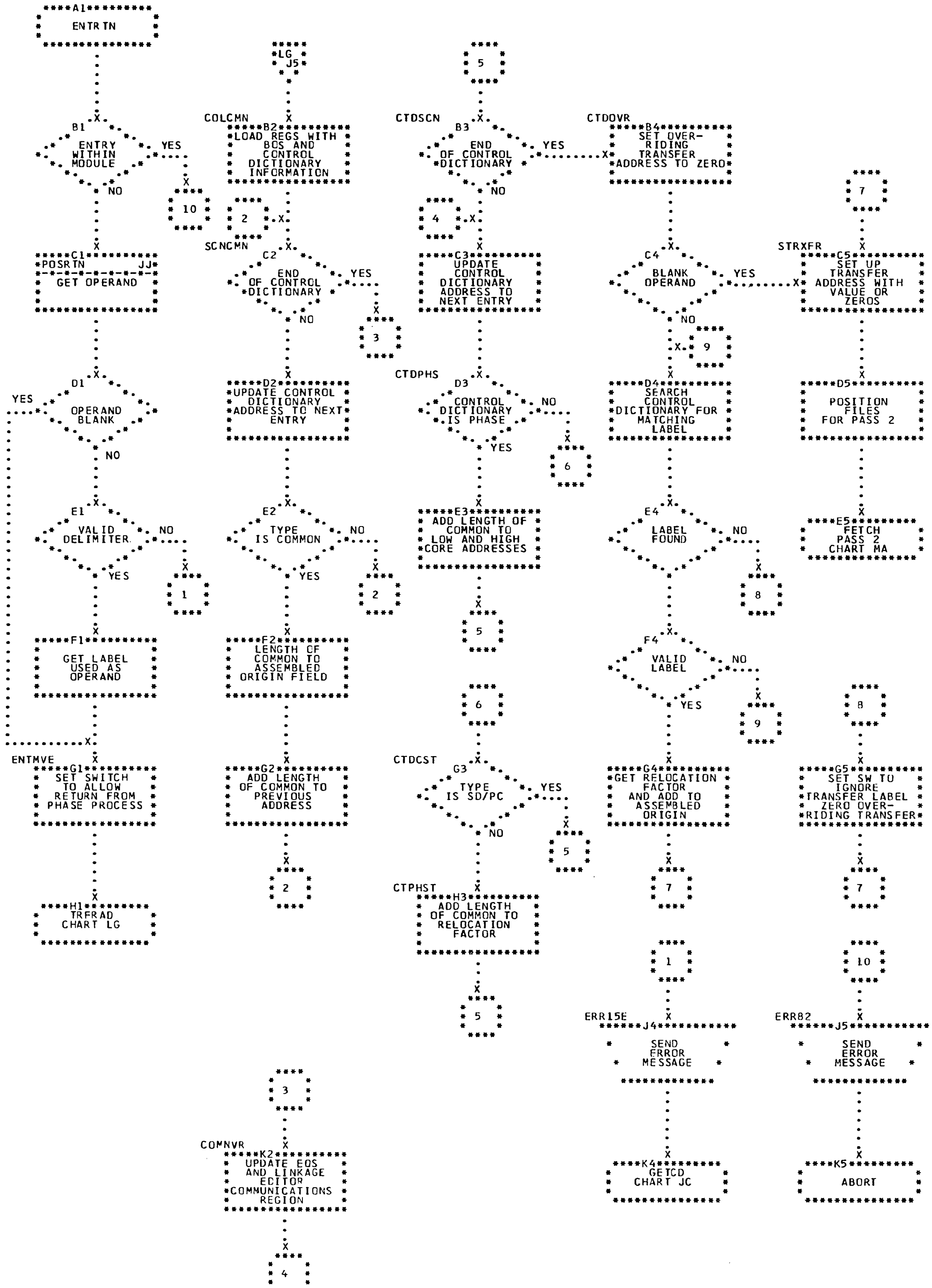


Chart LH. ENTRY

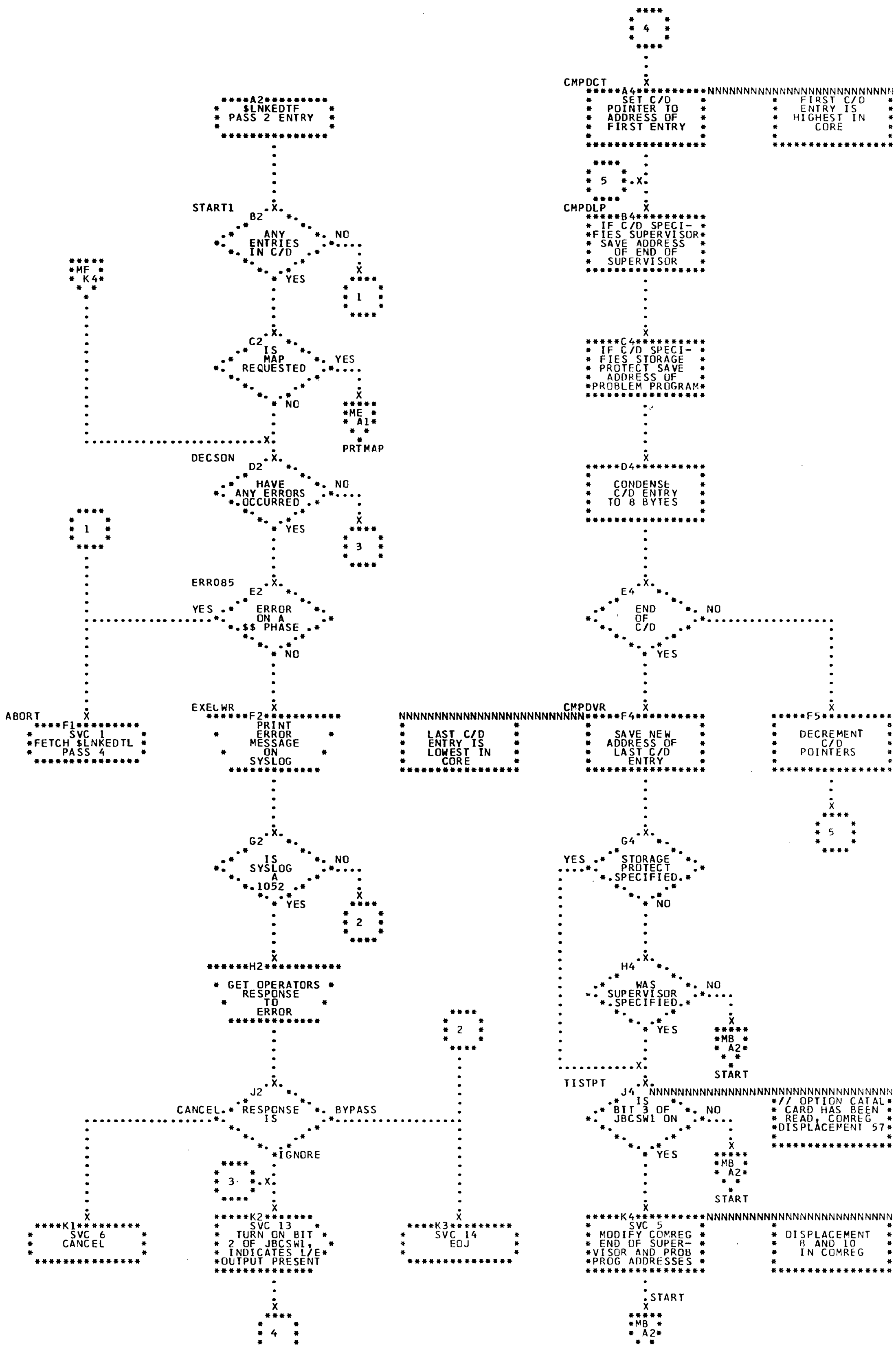


Chart MA. Condense Control Dictionary

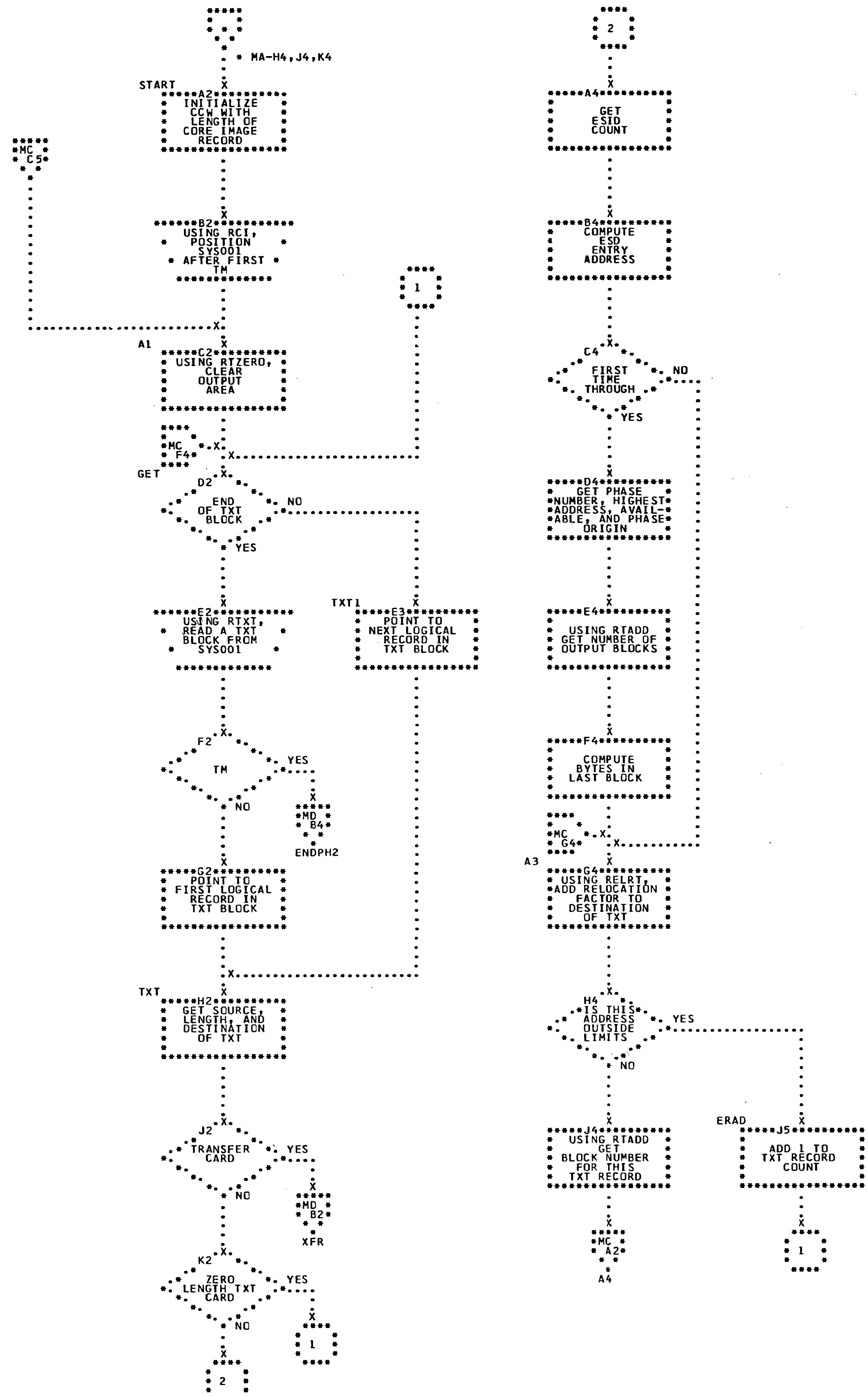


Chart MB. Read TXT Records

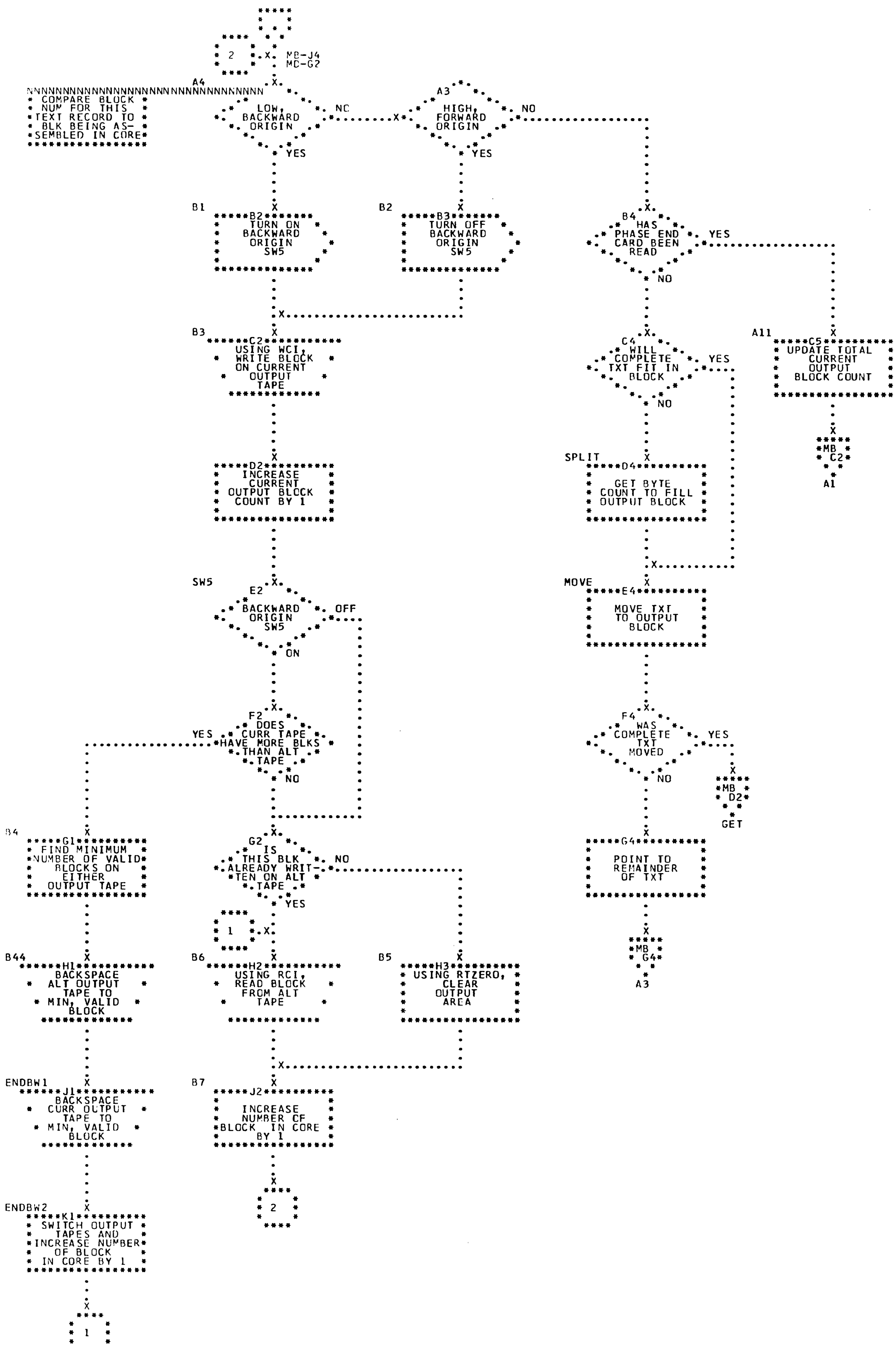


Chart MC. Write Core Image Blocks

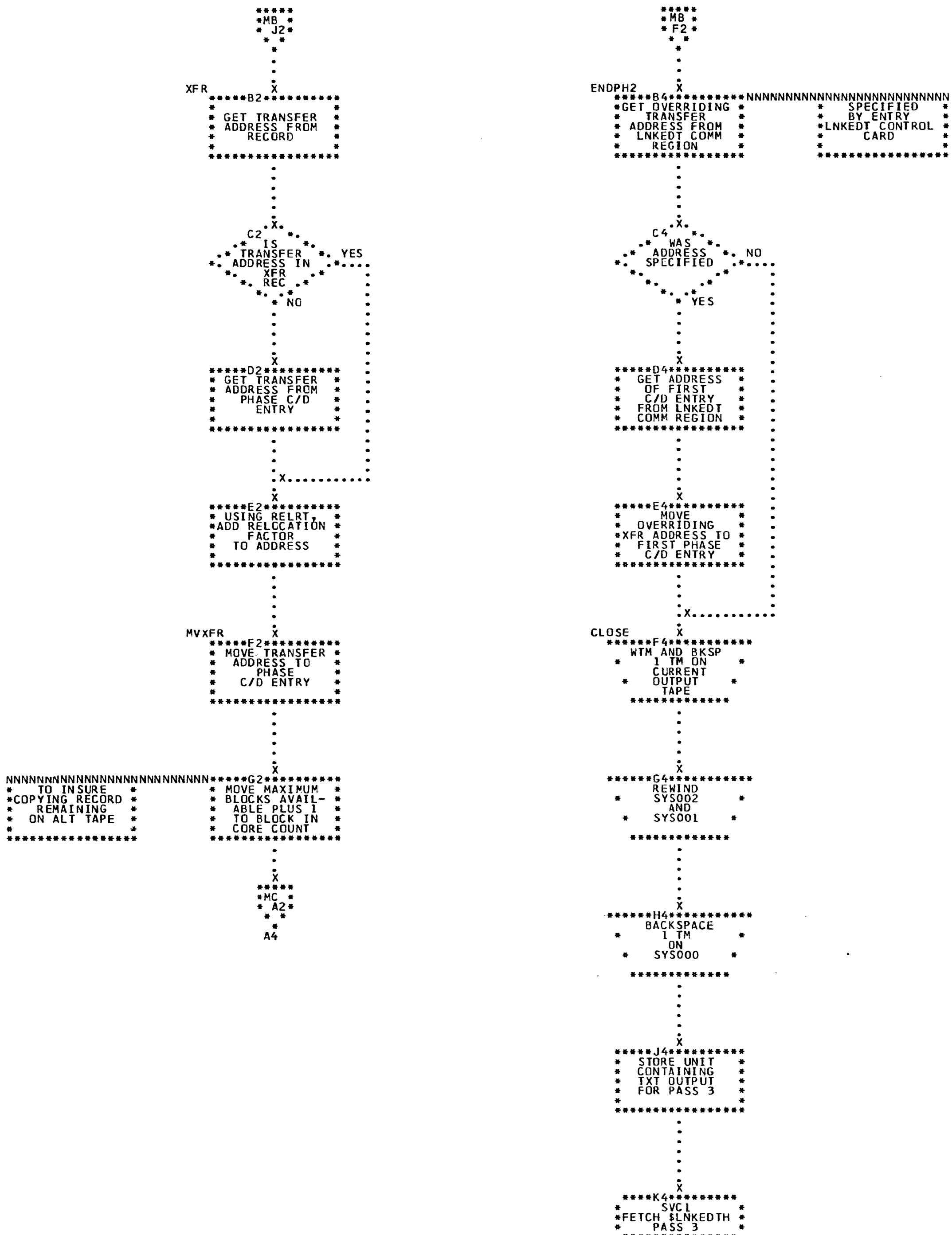


Chart MD. Get Transfer Address

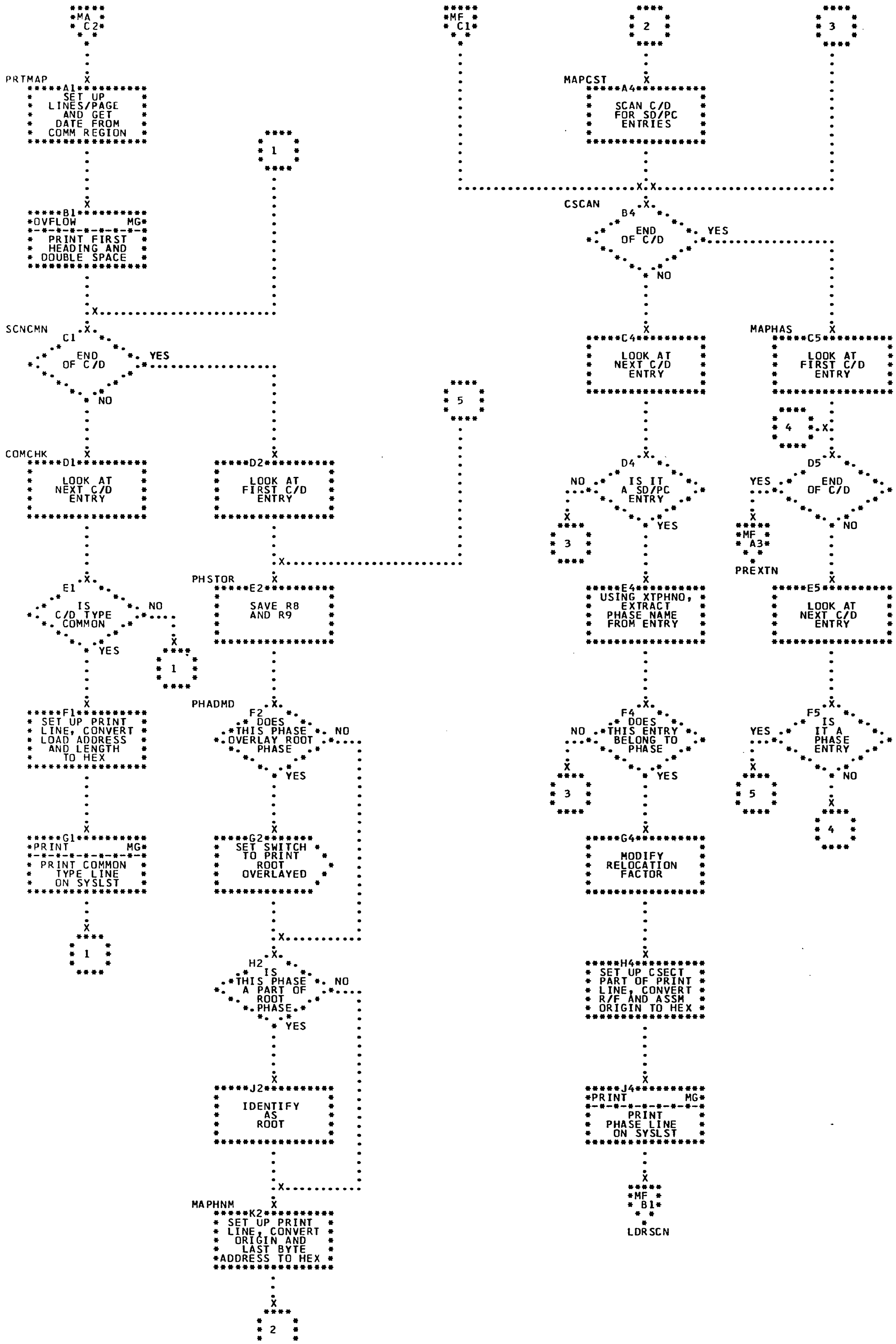


Chart ME. MAP (Part 1 of 2)

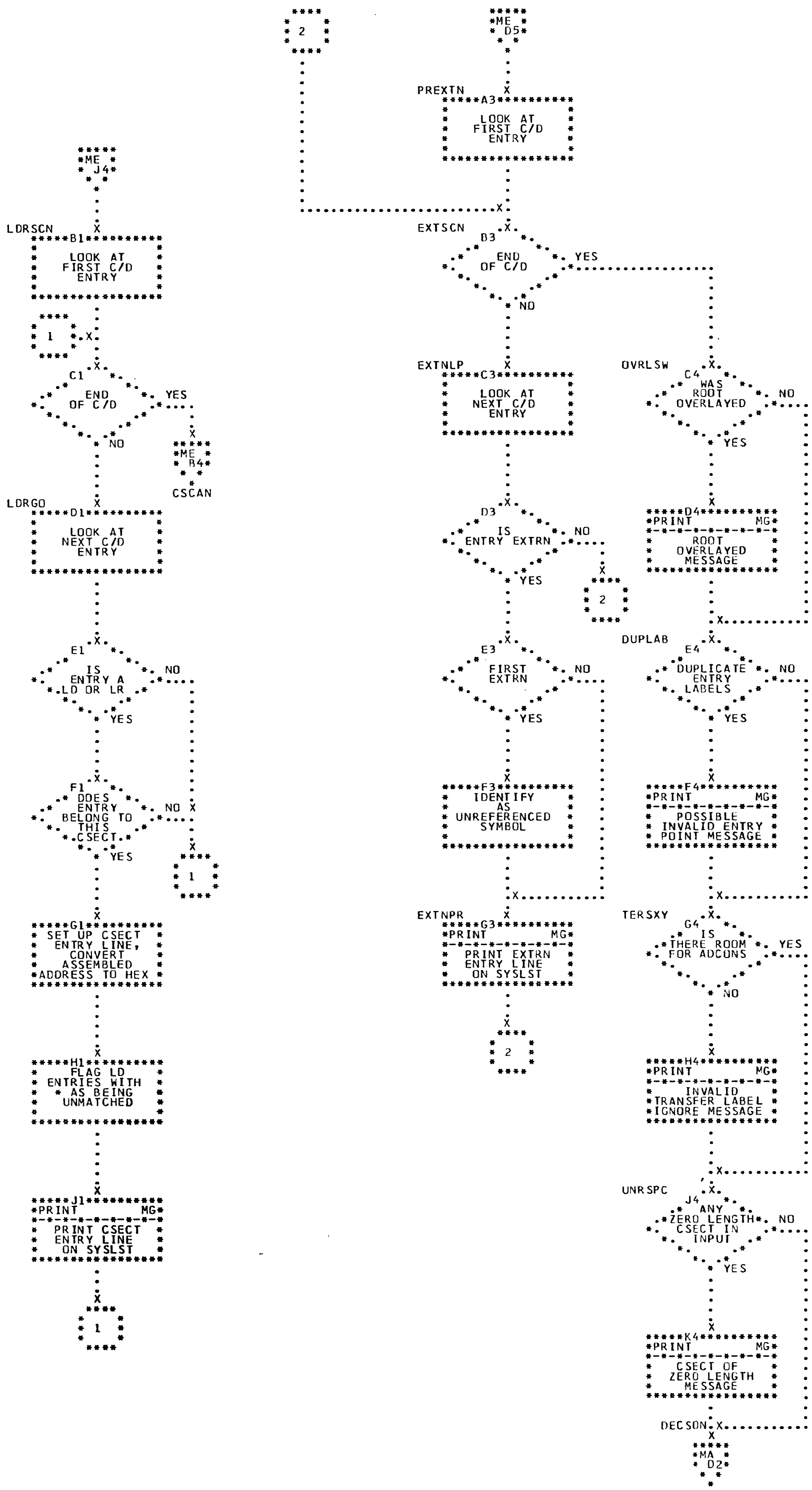


Chart MF. MAP (Part 2 of 2)

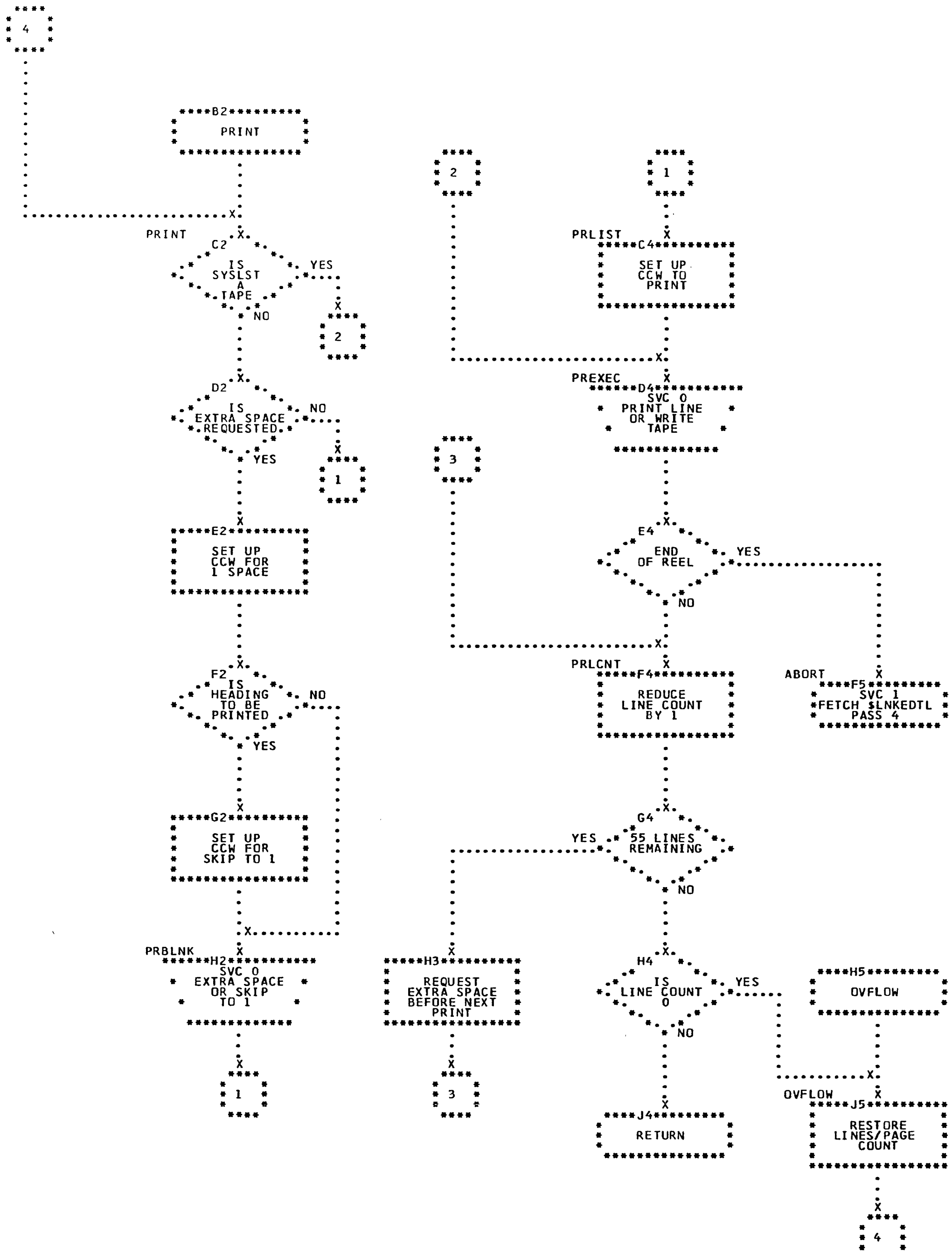


Chart MG. PRINT Subroutine

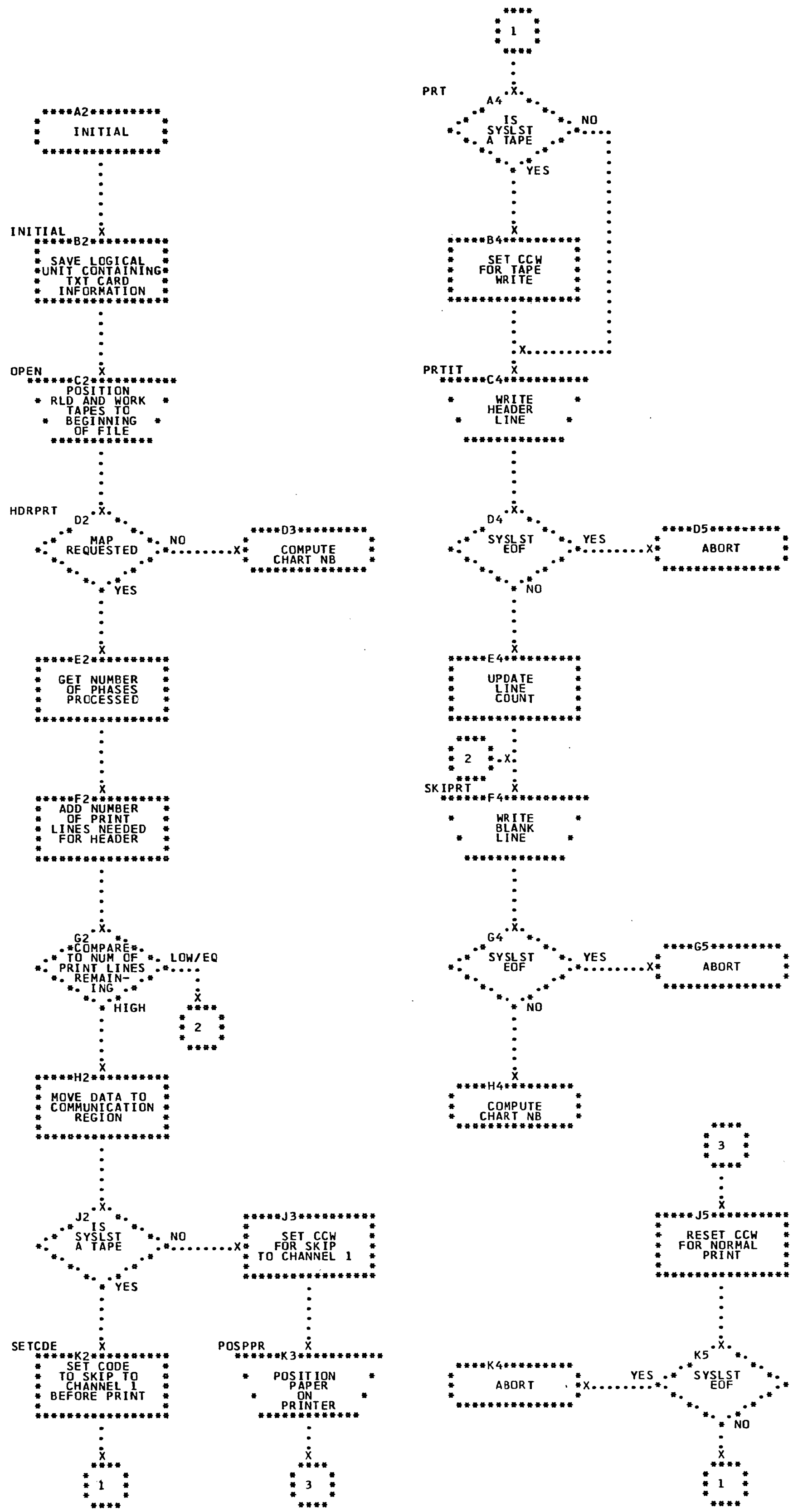


Chart NA. Initialization

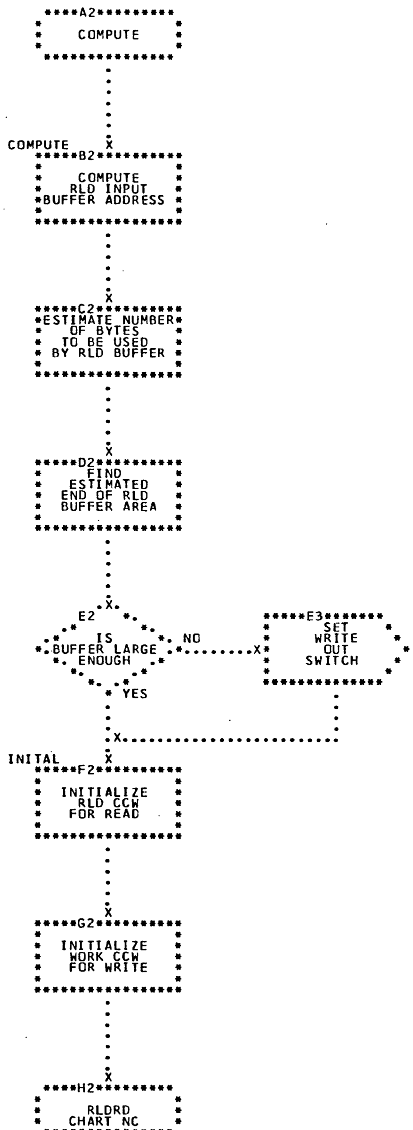


Chart NB. Compute Buffer Size

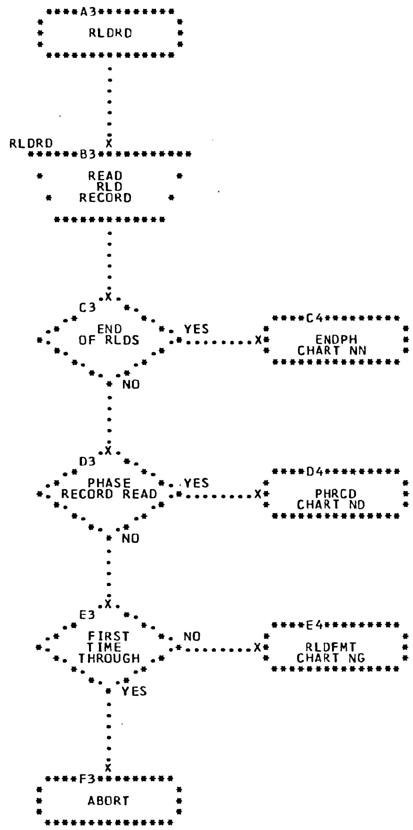


Chart NC. Read RLD Record

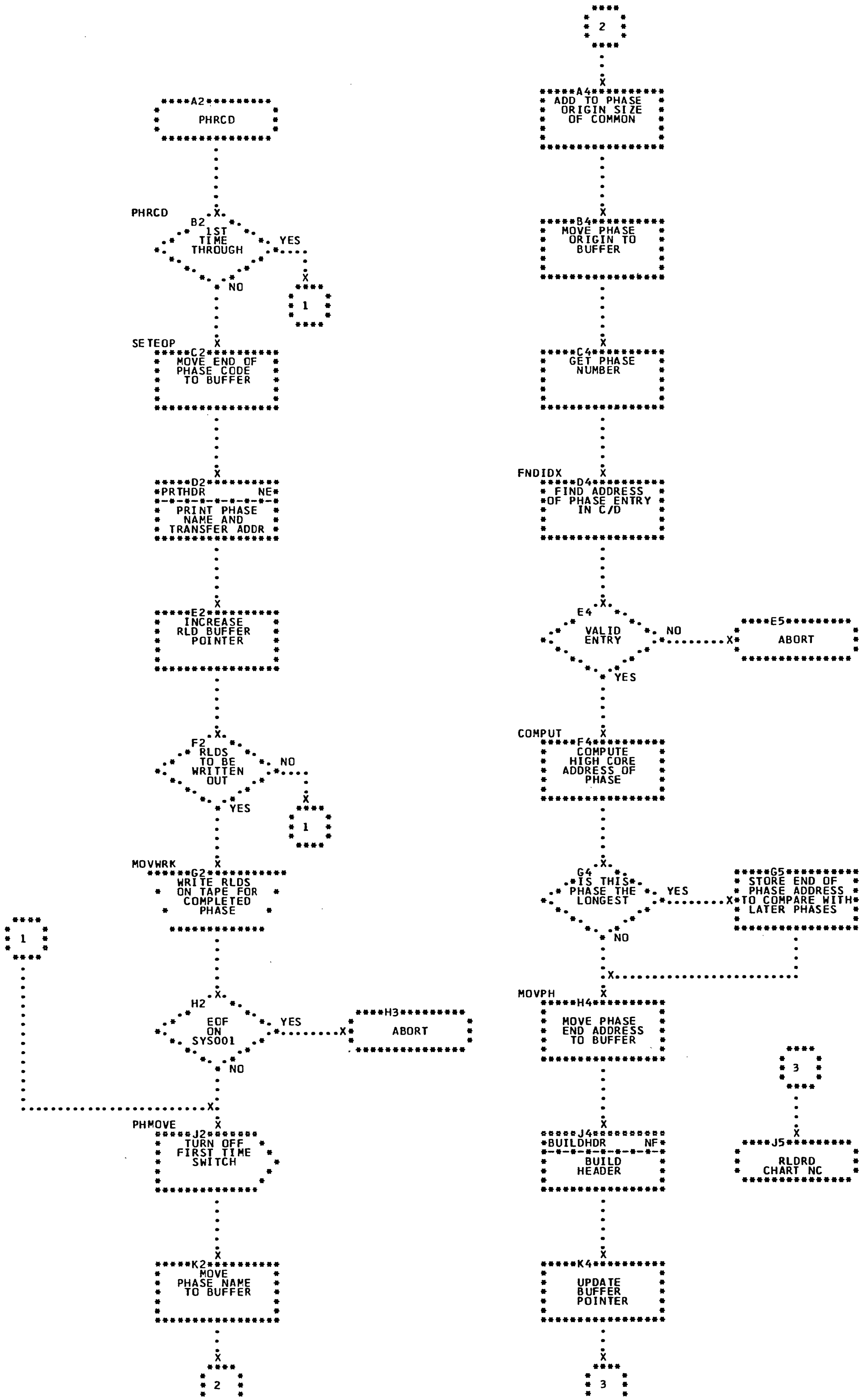


Chart ND. Process PHASE Record

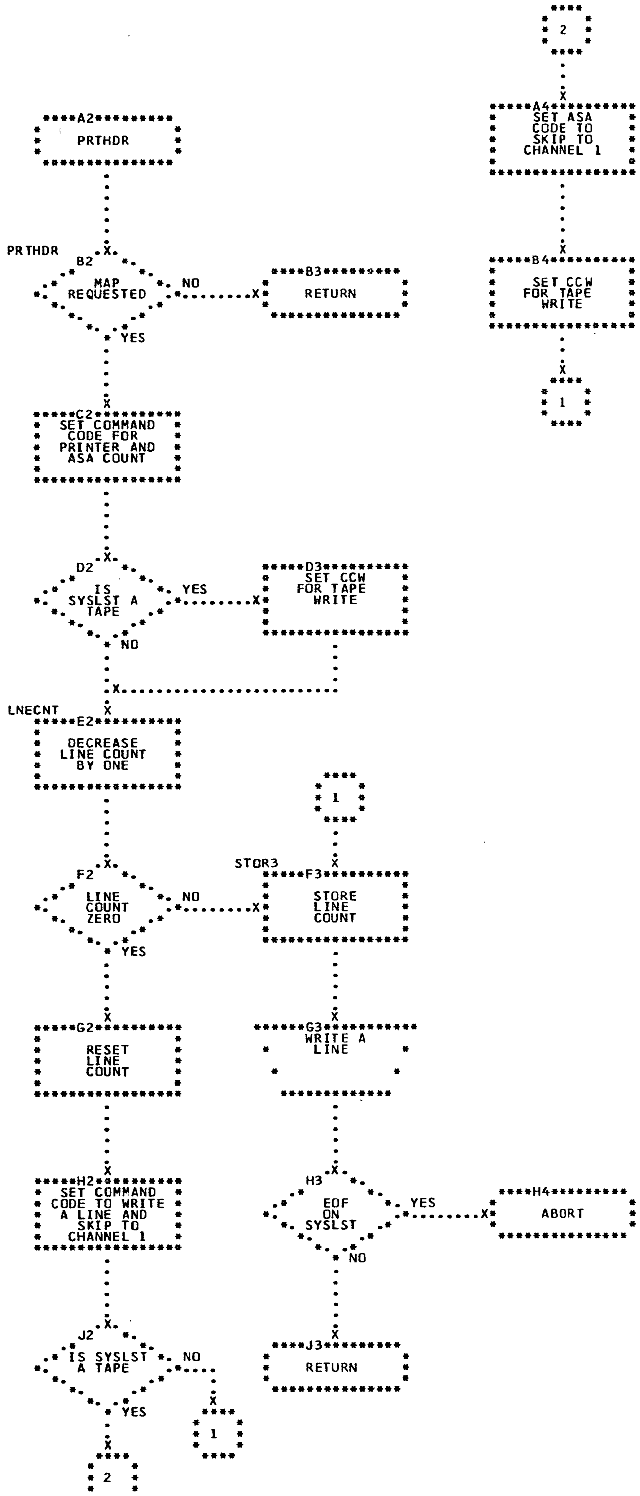


Chart NE. Print Header Subroutine

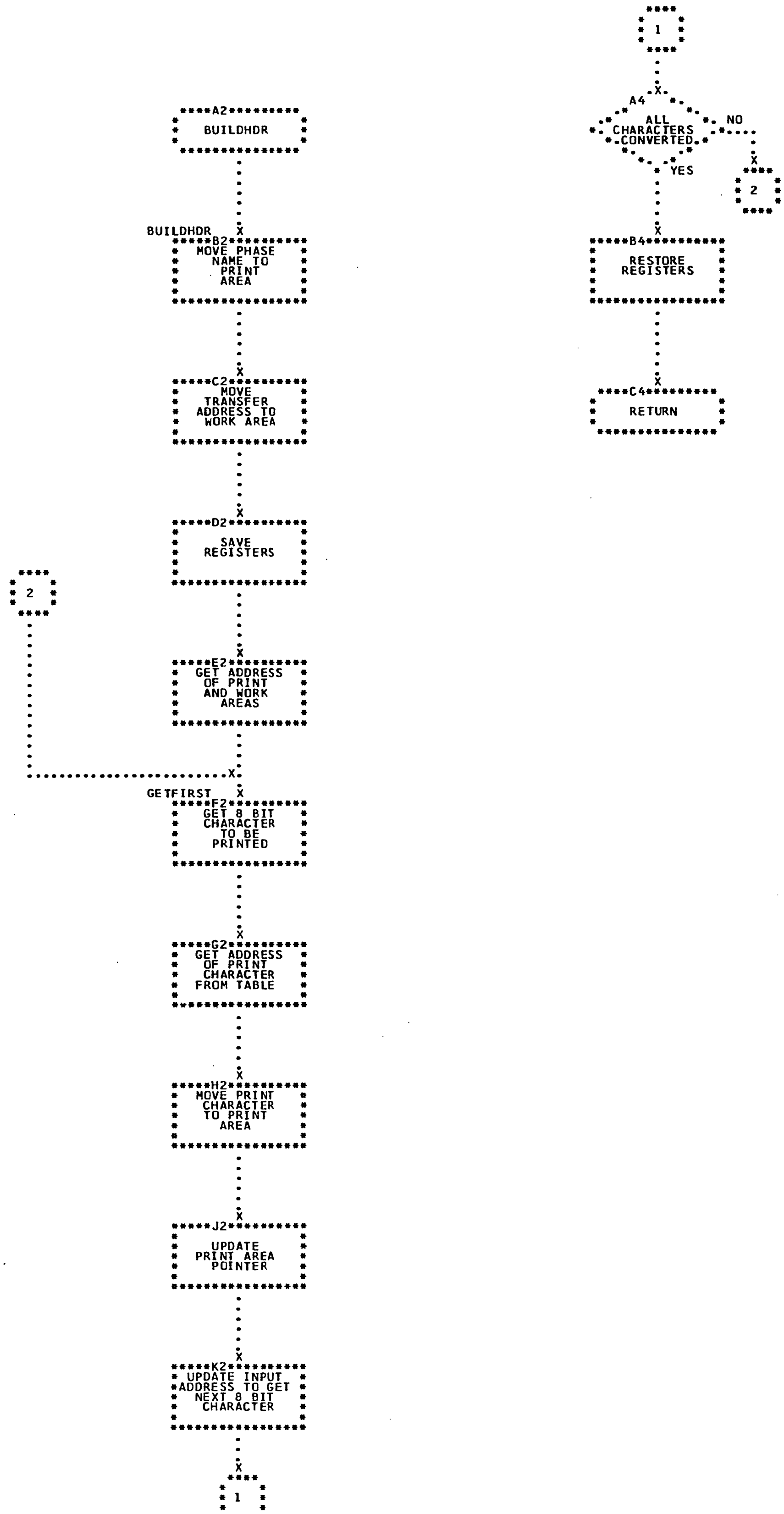


Chart NF. Build Header Subroutine

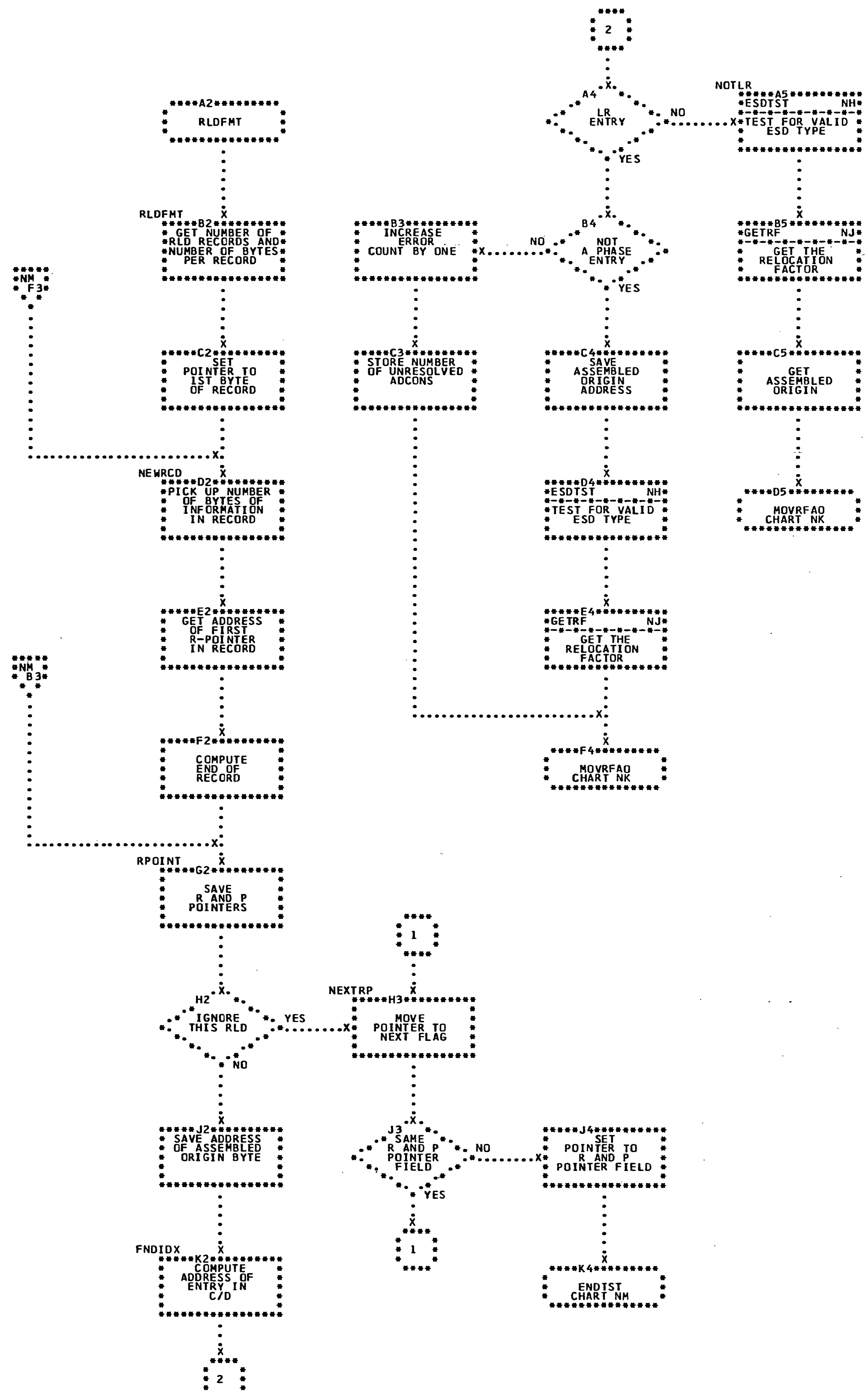


Chart NG. RLD Formatting

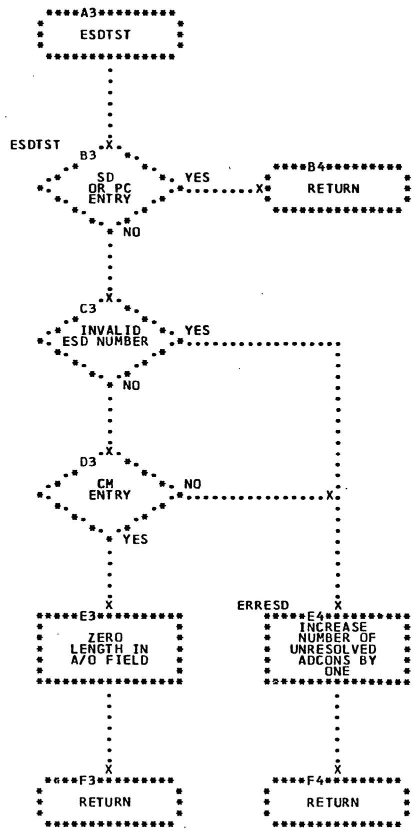


Chart NH. ESD Check Subroutine

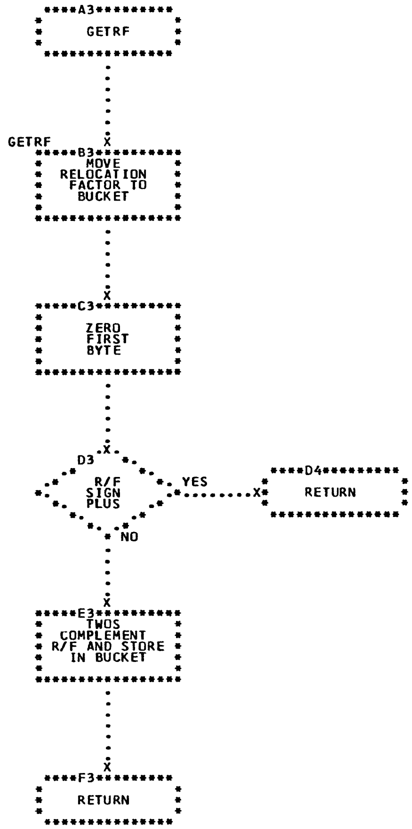


Chart NJ. Get Relocation Factor Subroutine

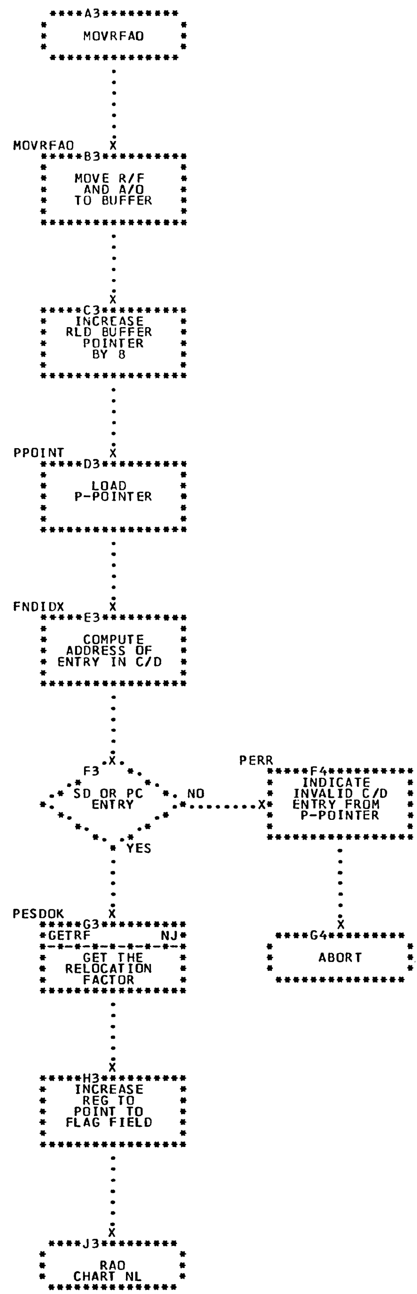


Chart NK. Move Relocation Factor and Assembled Origin

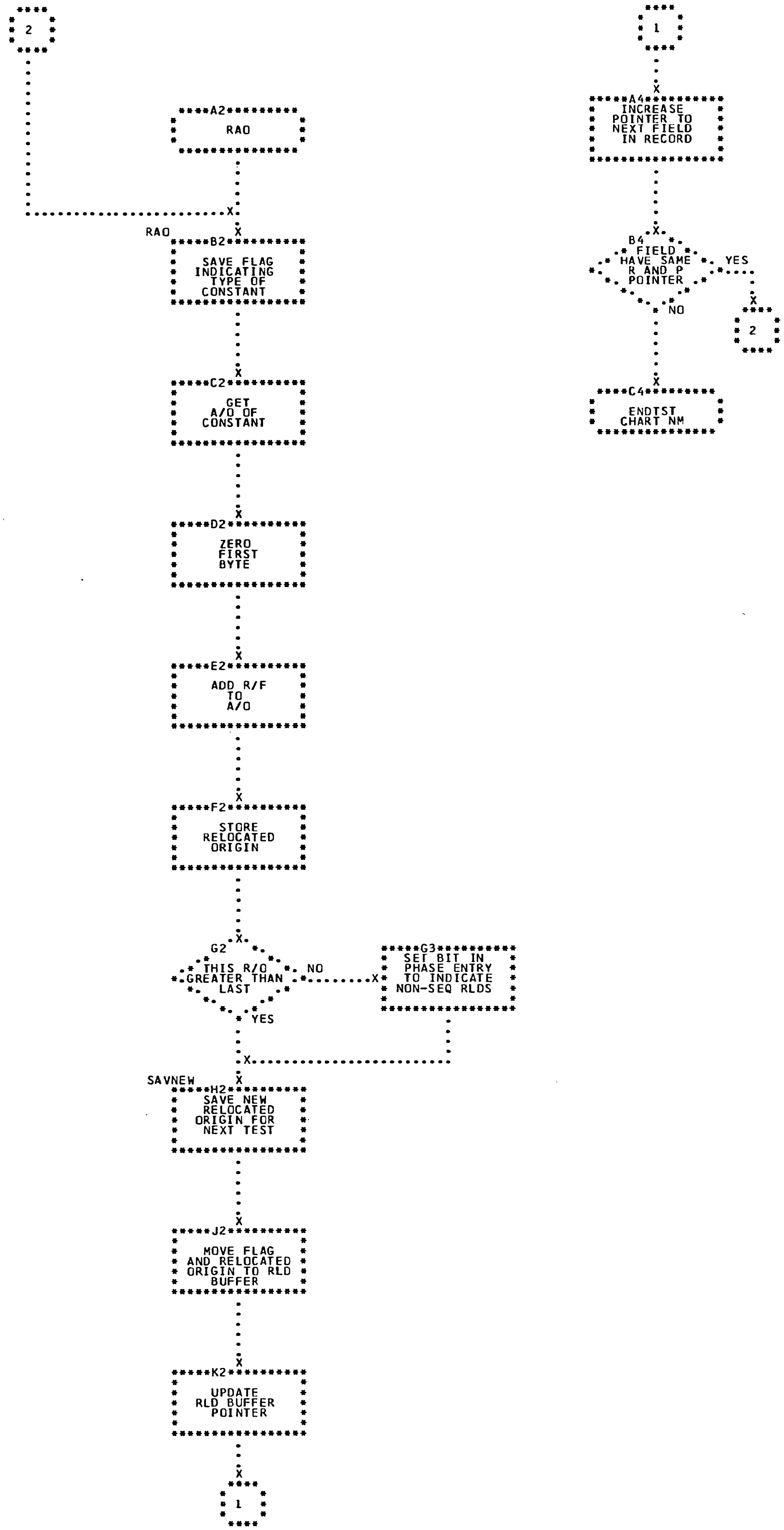


Chart NL. Process Flag

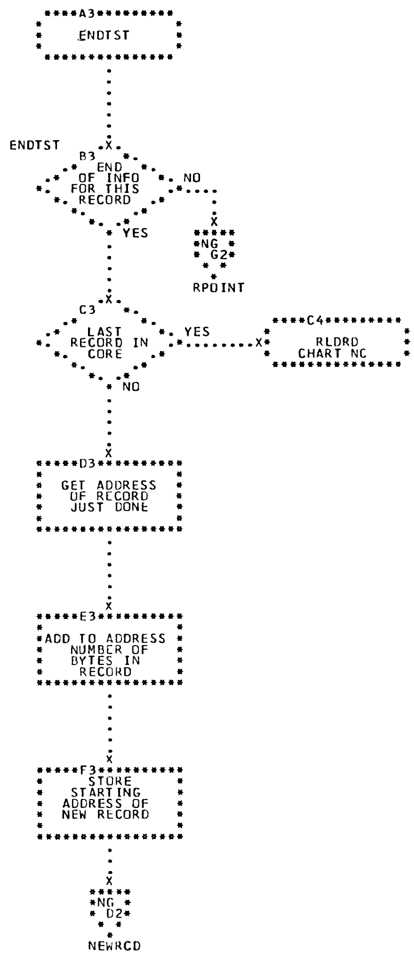


Chart NM. Test for Record End

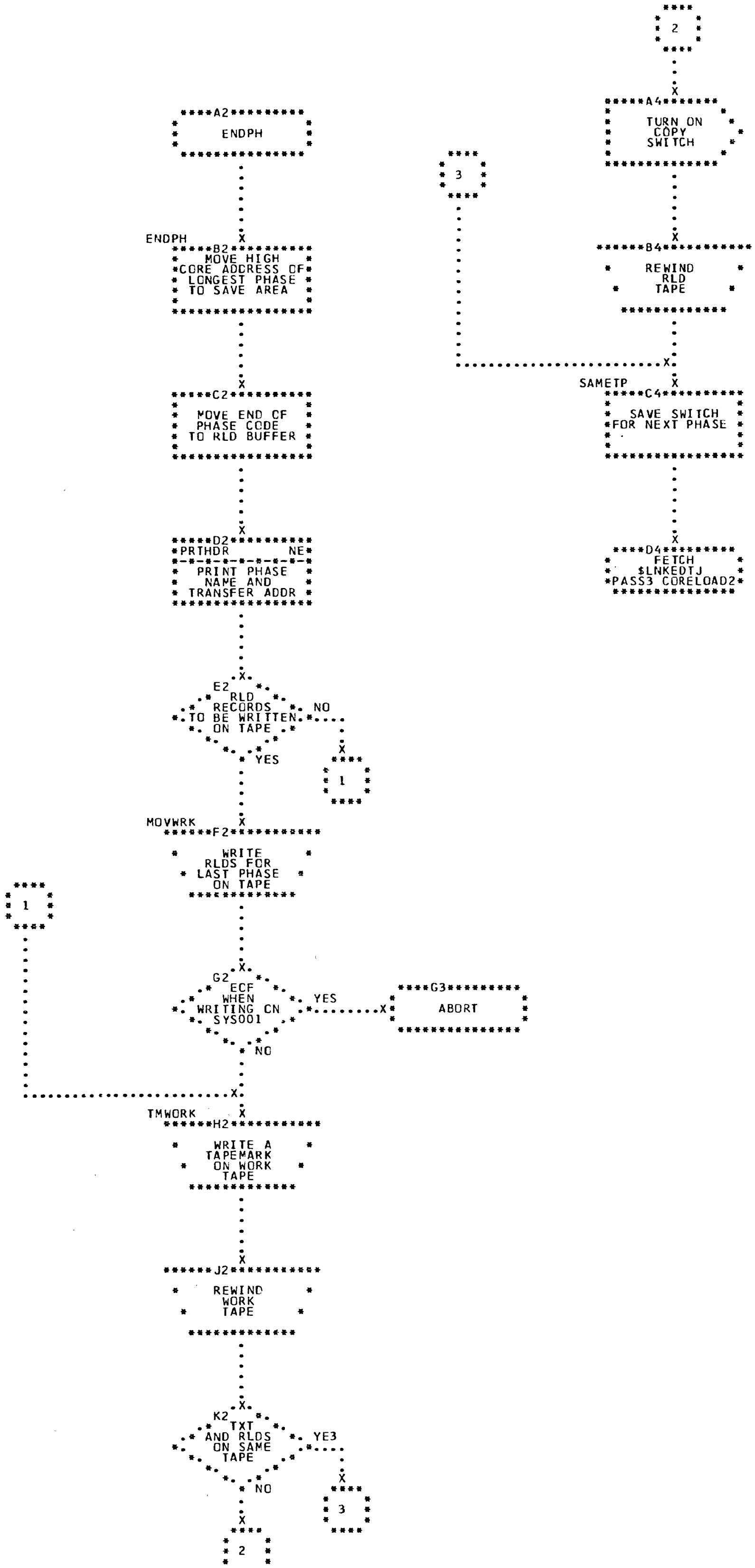


Chart NN. End of RLD's

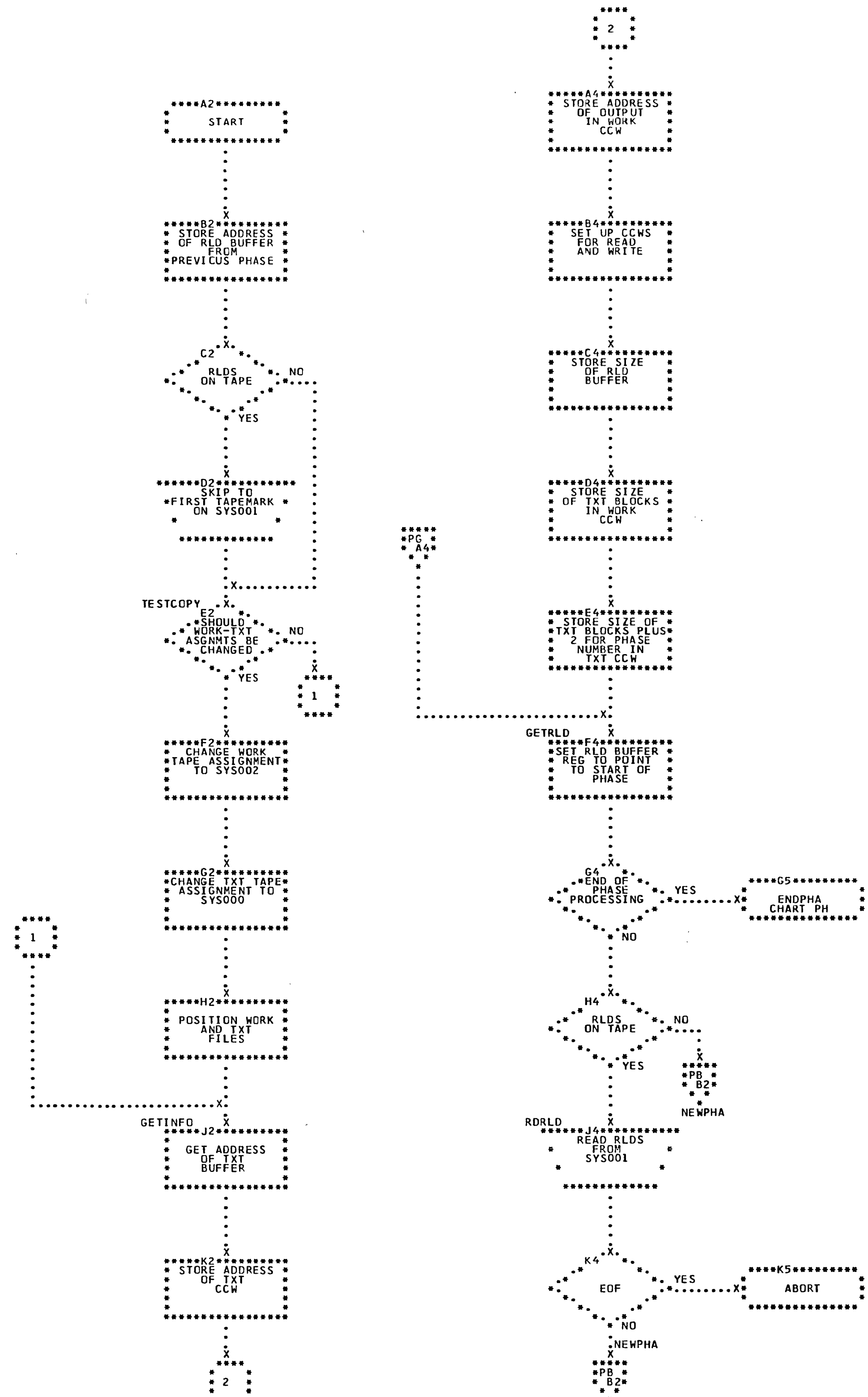


Chart PA. Initialization (Part 1 of 3)

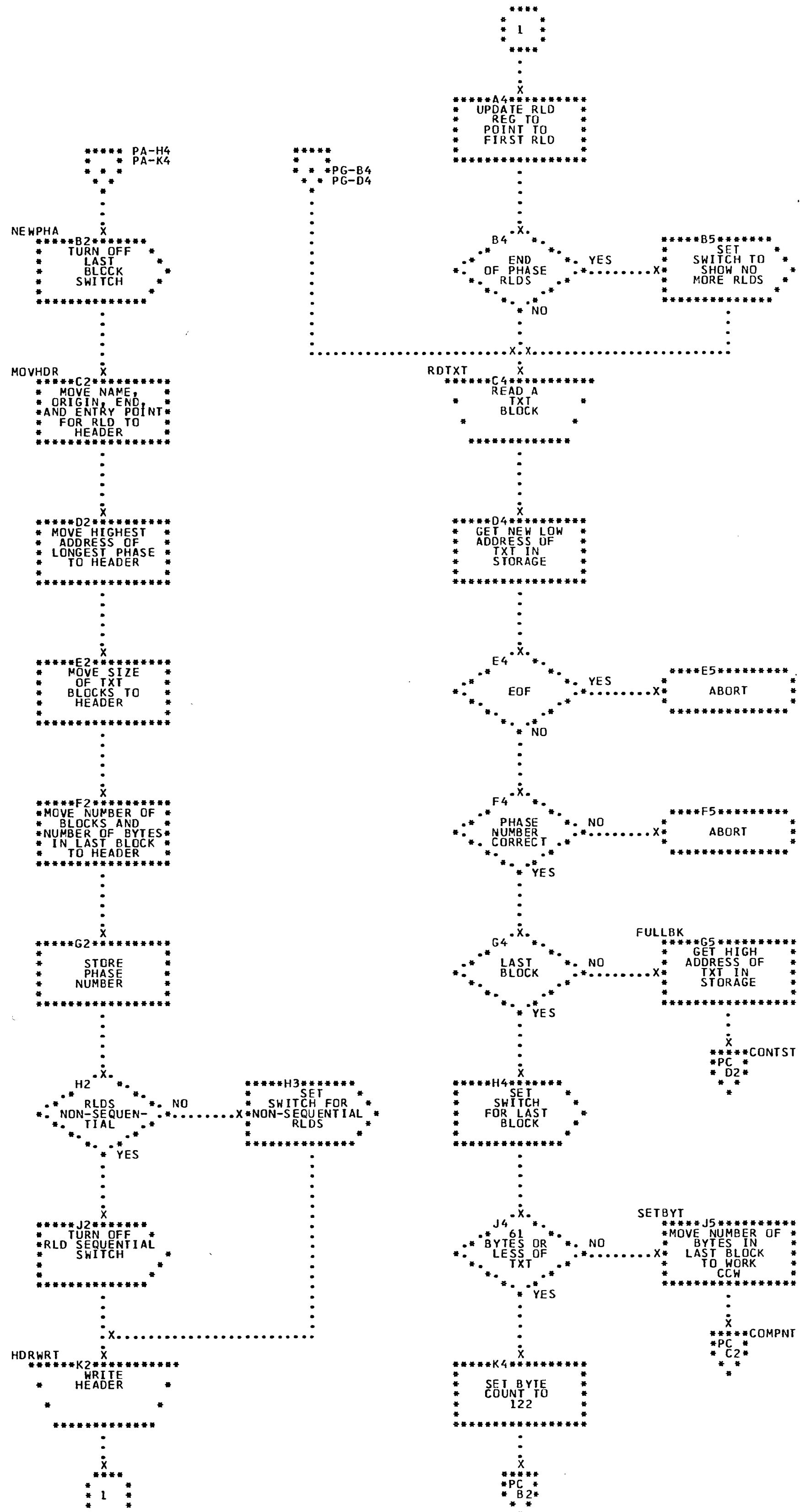


Chart PB. Initialization (Part 2 of 3)

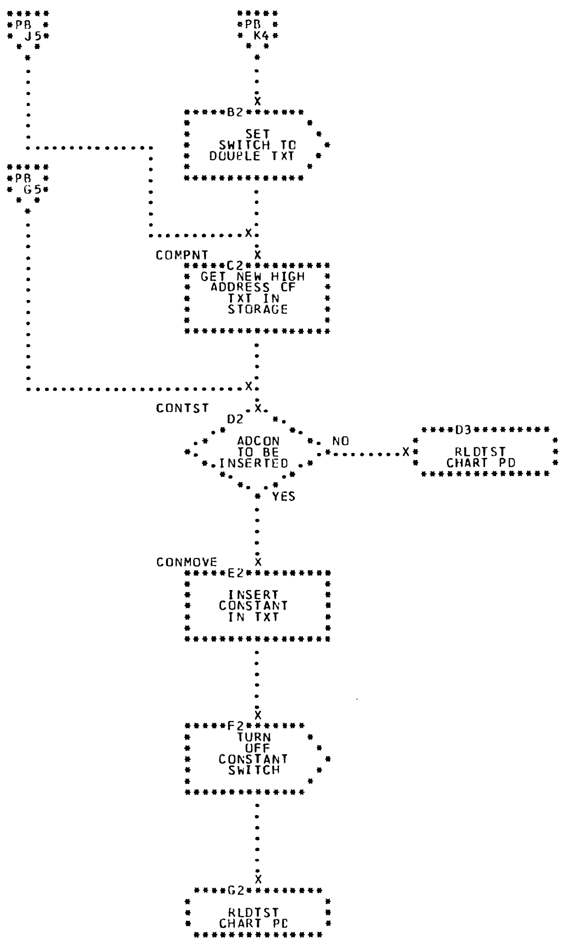


Chart PC. Initialization (Part 3 of 3)

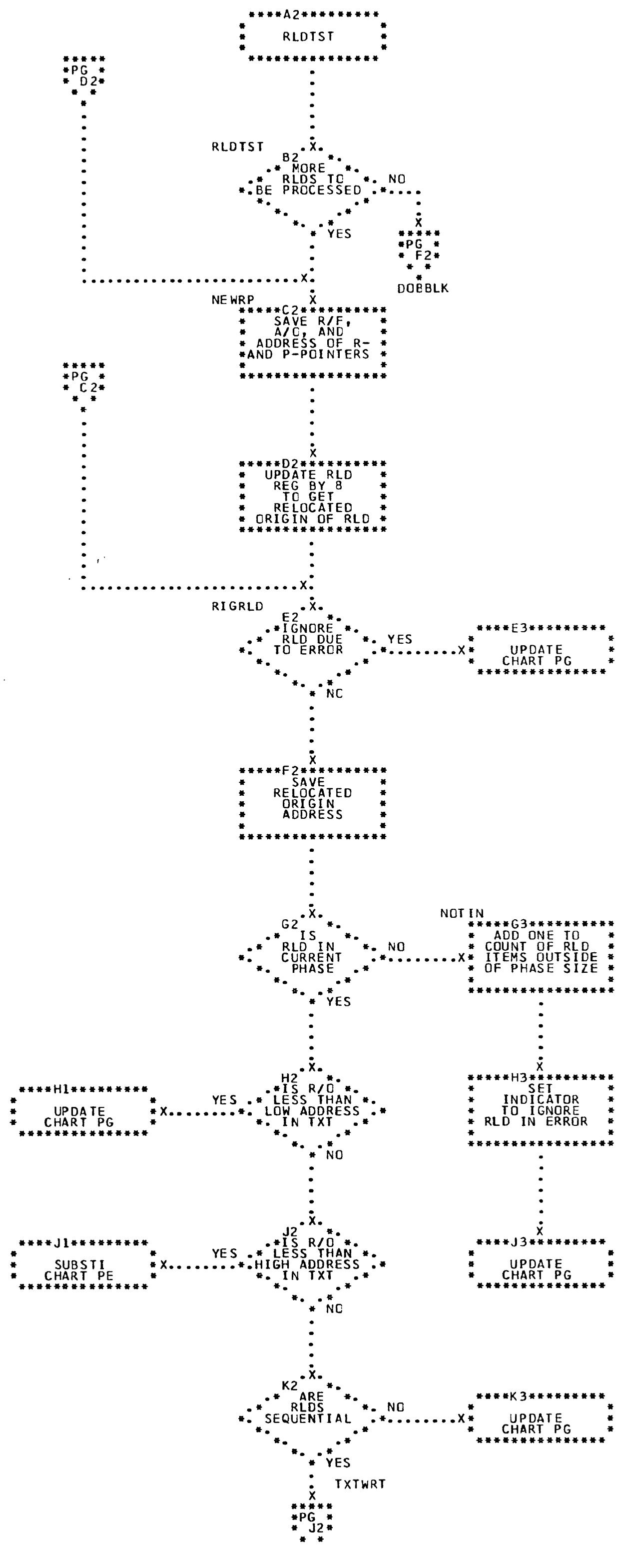


Chart PD. Match RLD to TXT

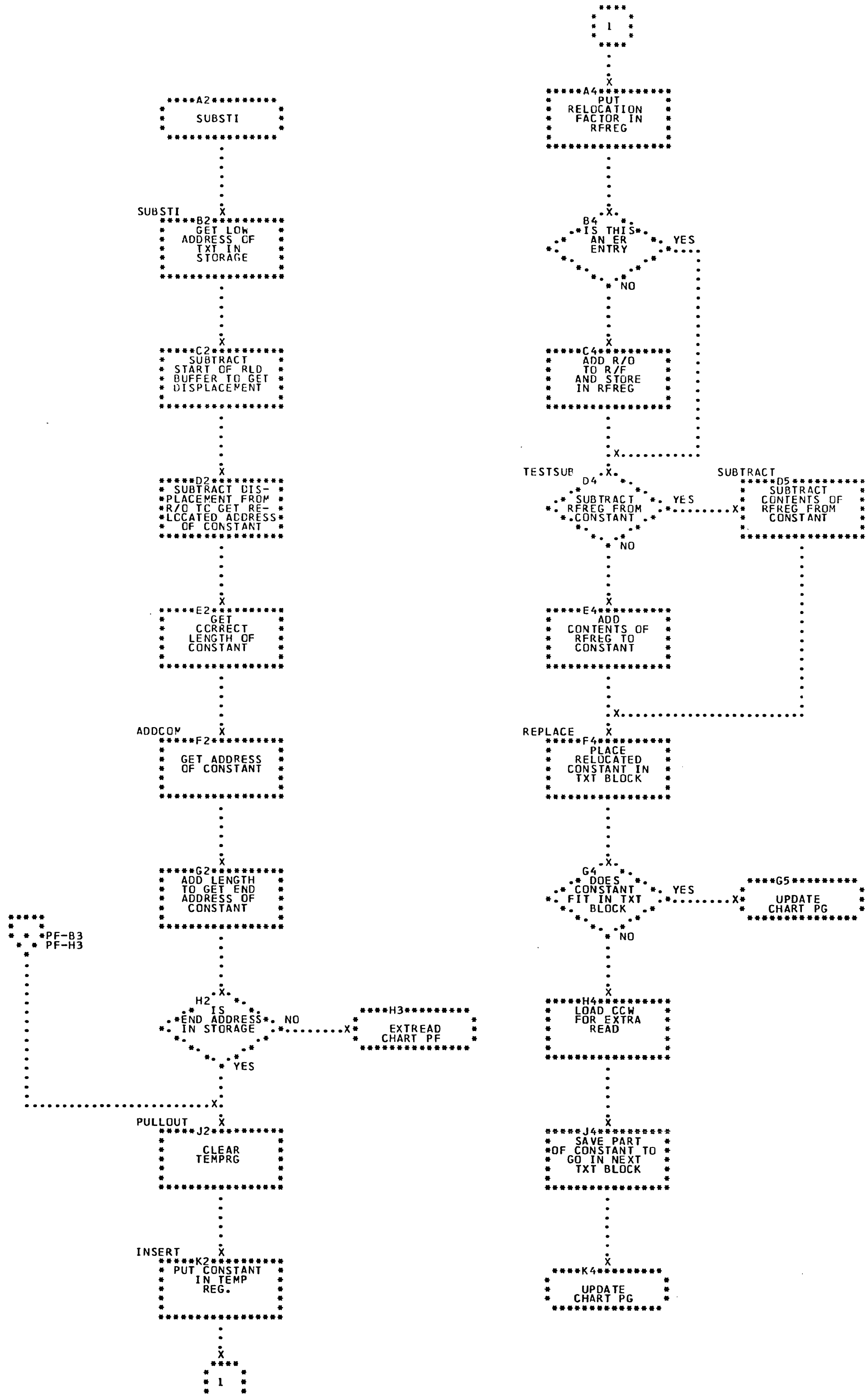


Chart PE. Relocate Constant

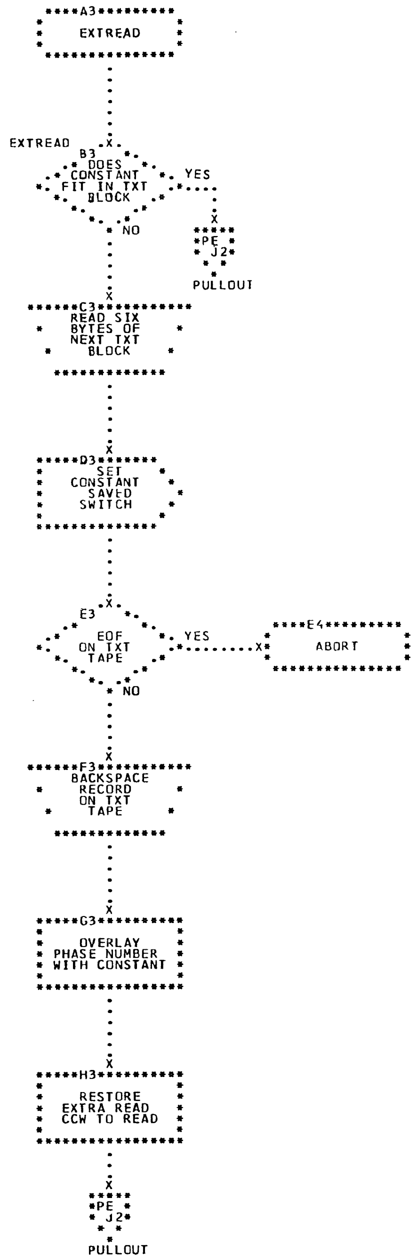


Chart PF. Extra Read

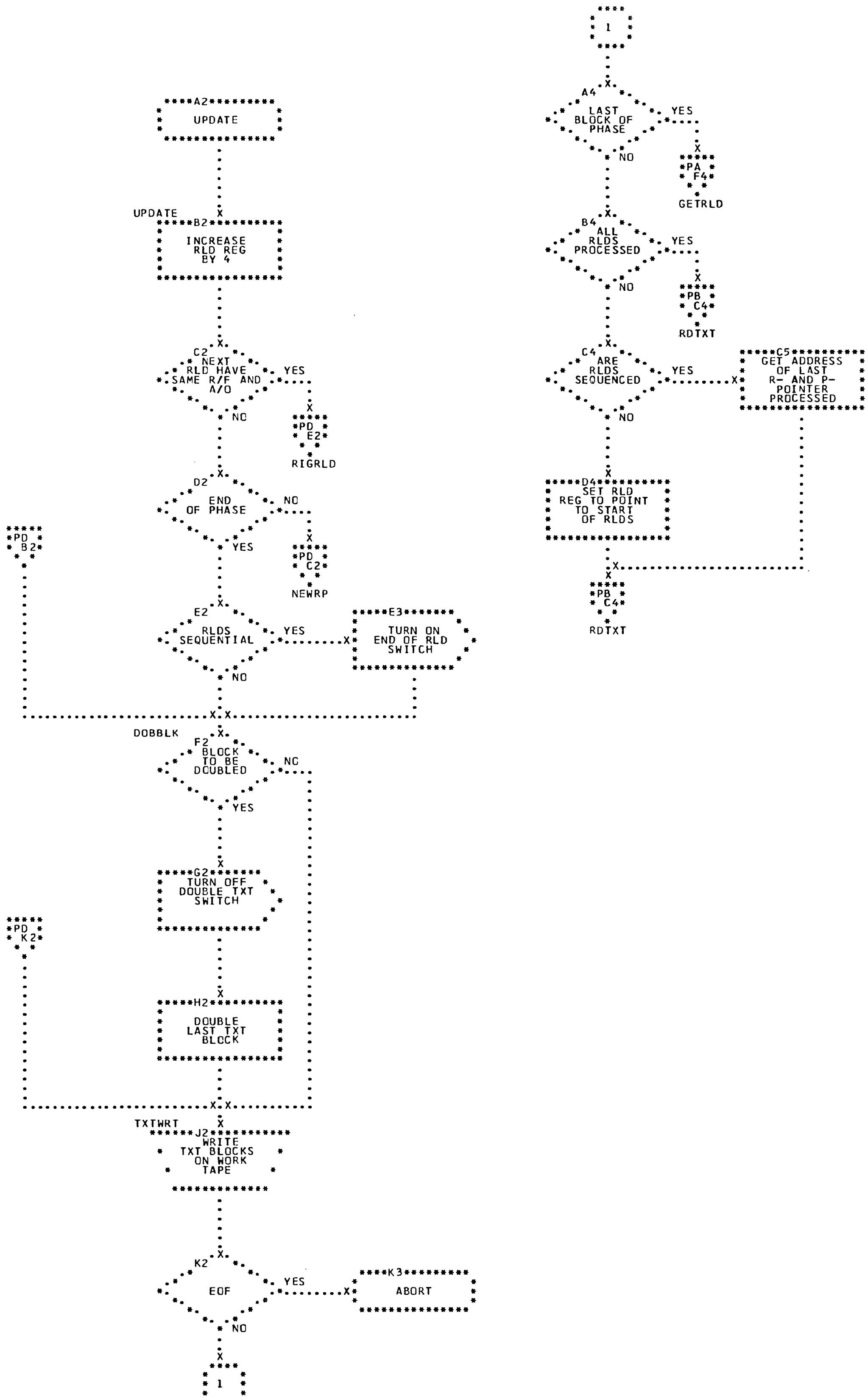


Chart PG. Get Next RLD

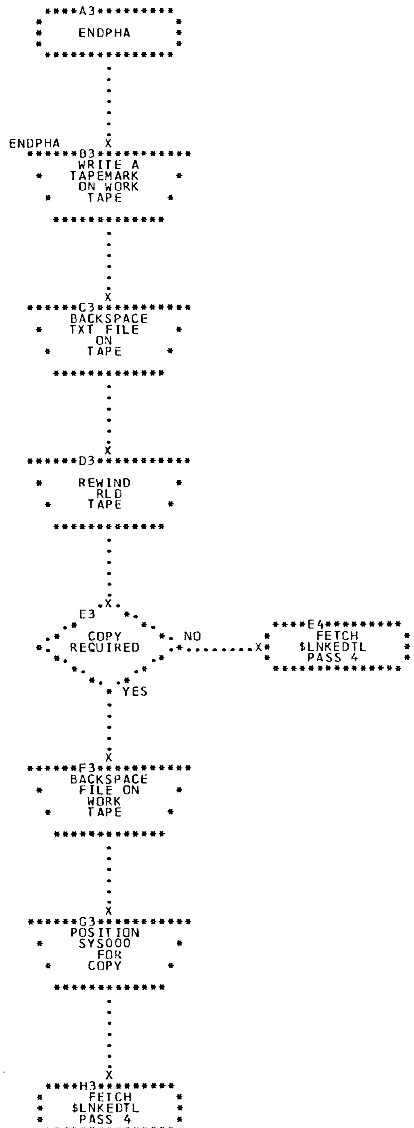


Chart PH. End of Processing

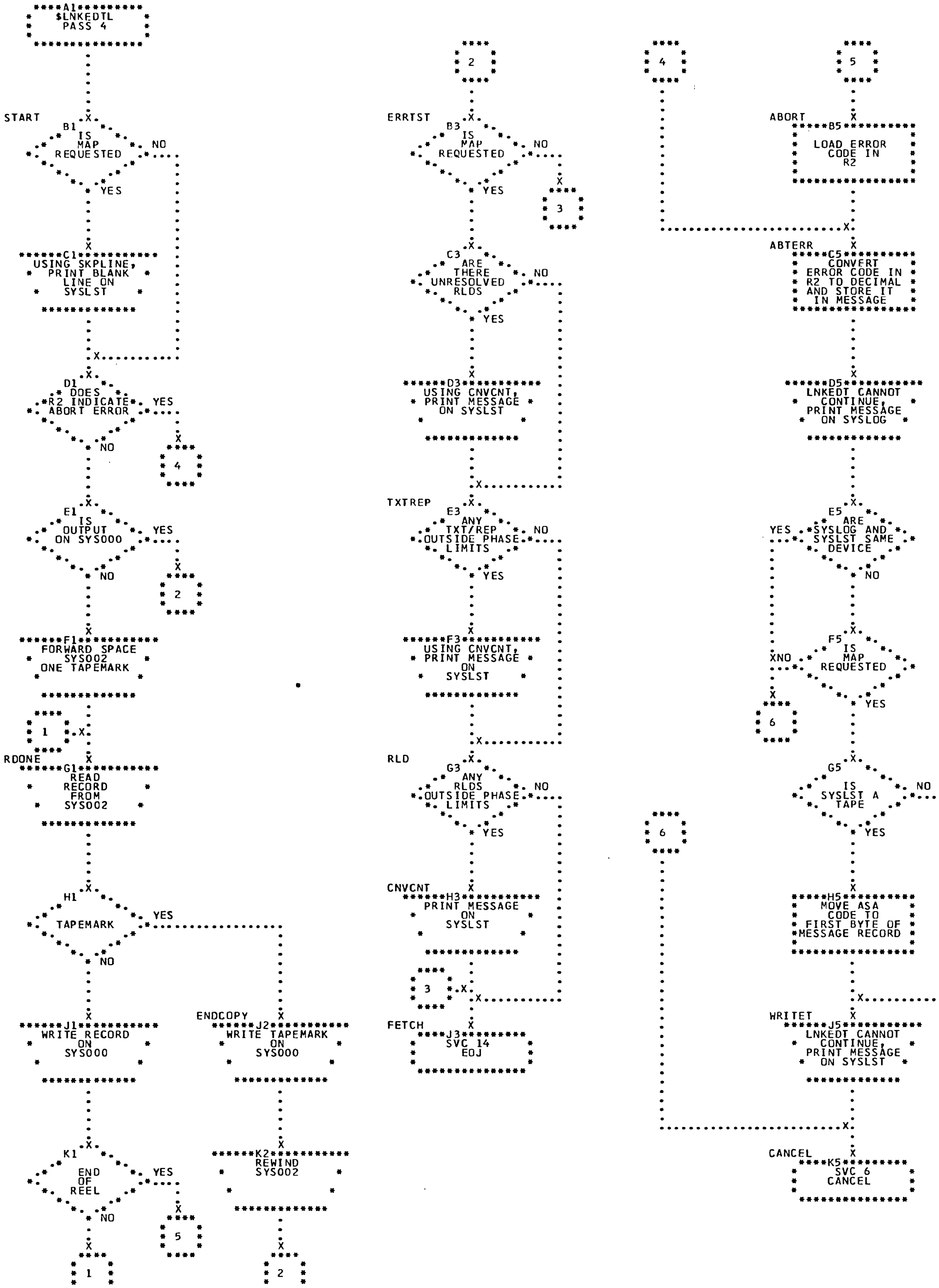


Chart QA. Pass 4

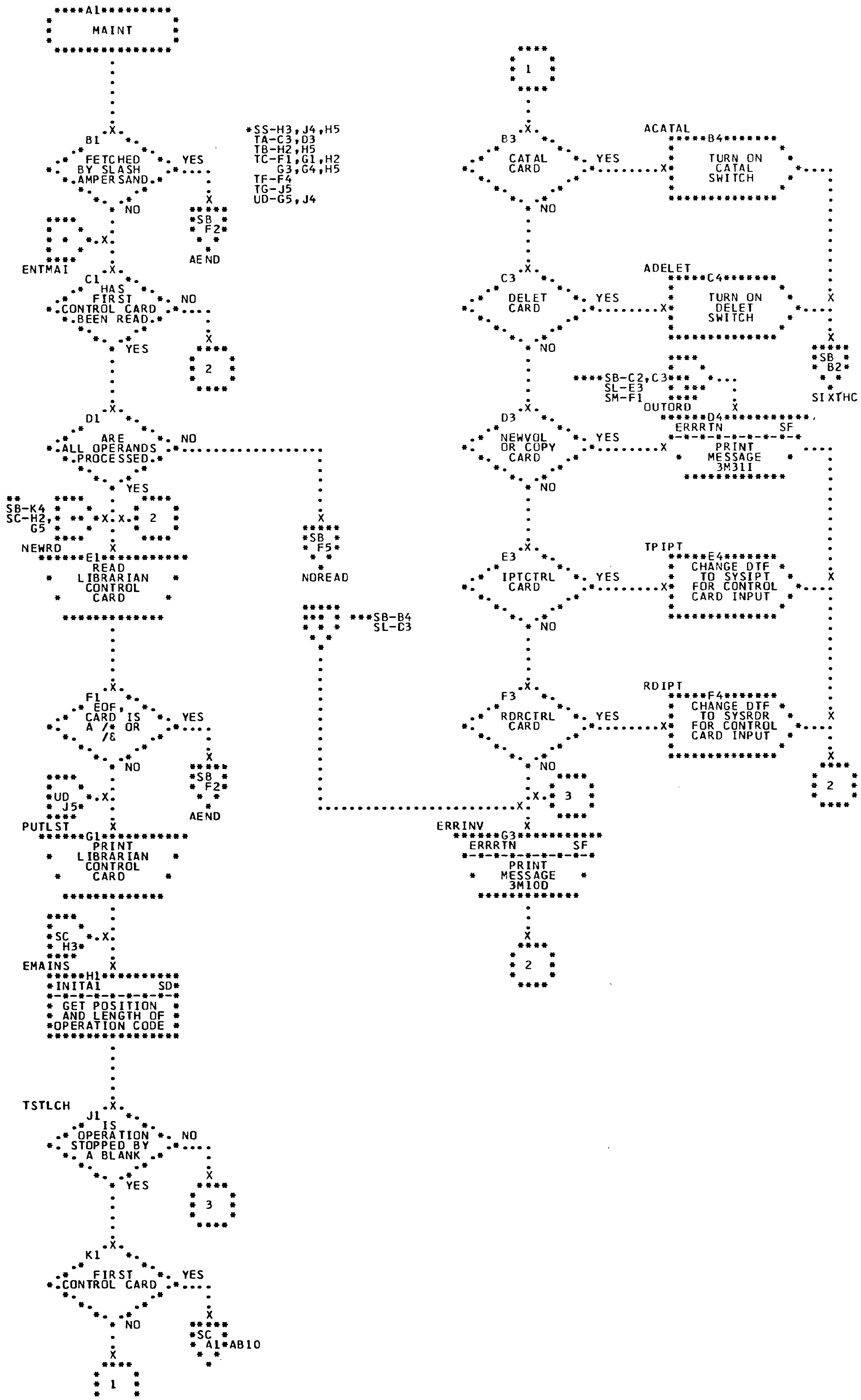


Chart SA. Read Control Card

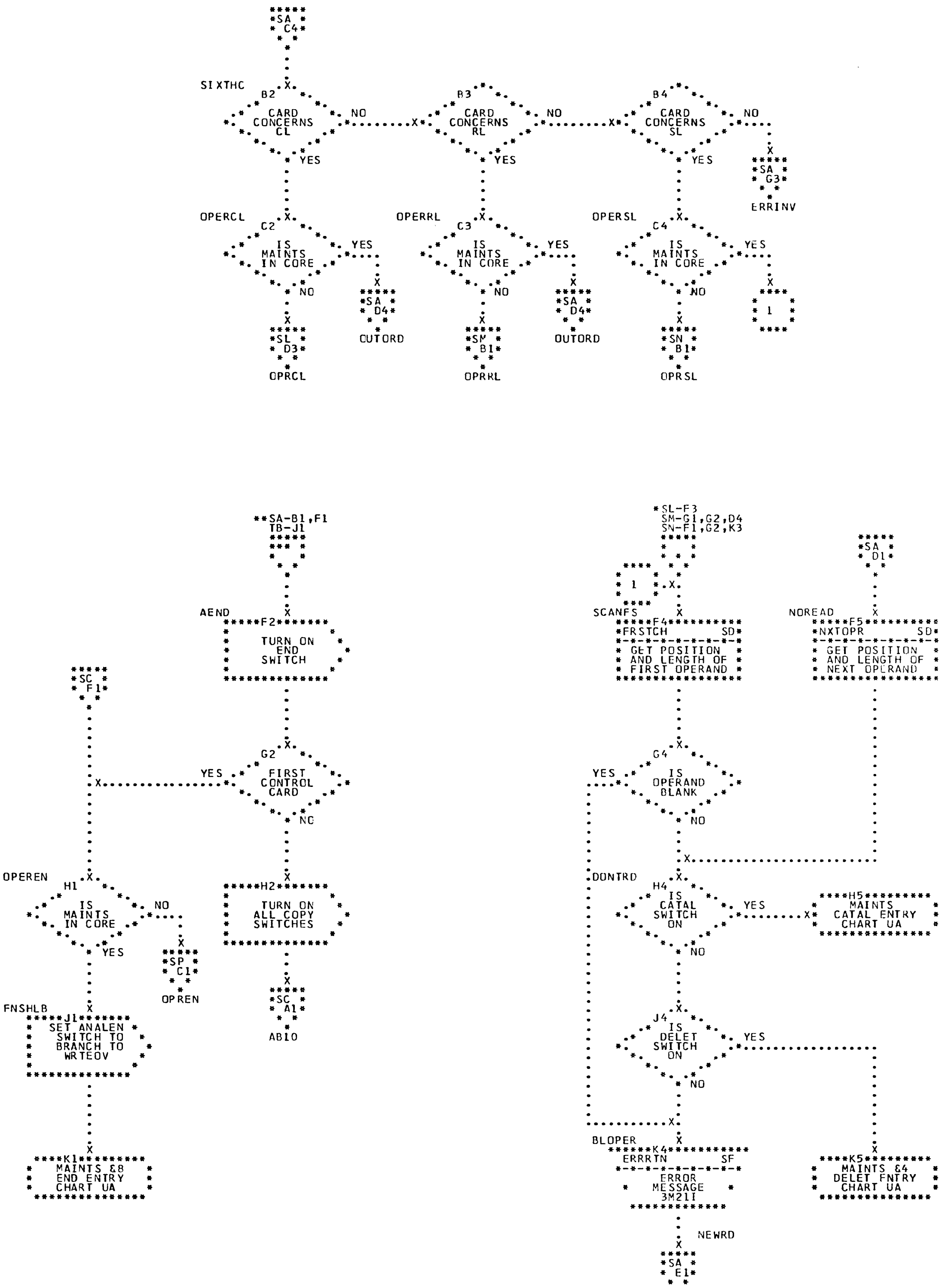


Chart SB. Determine Exit

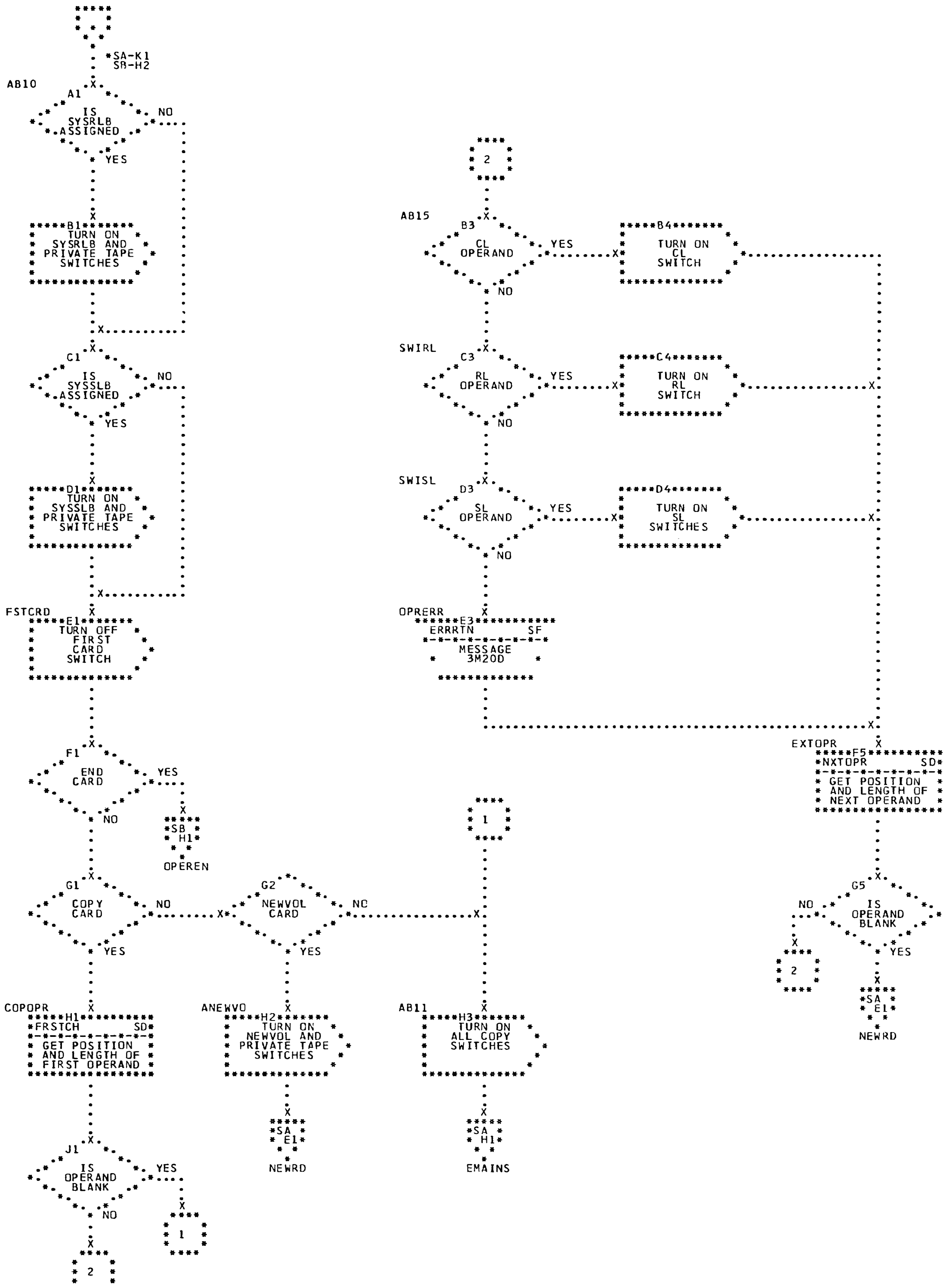


Chart SC. Process First Card

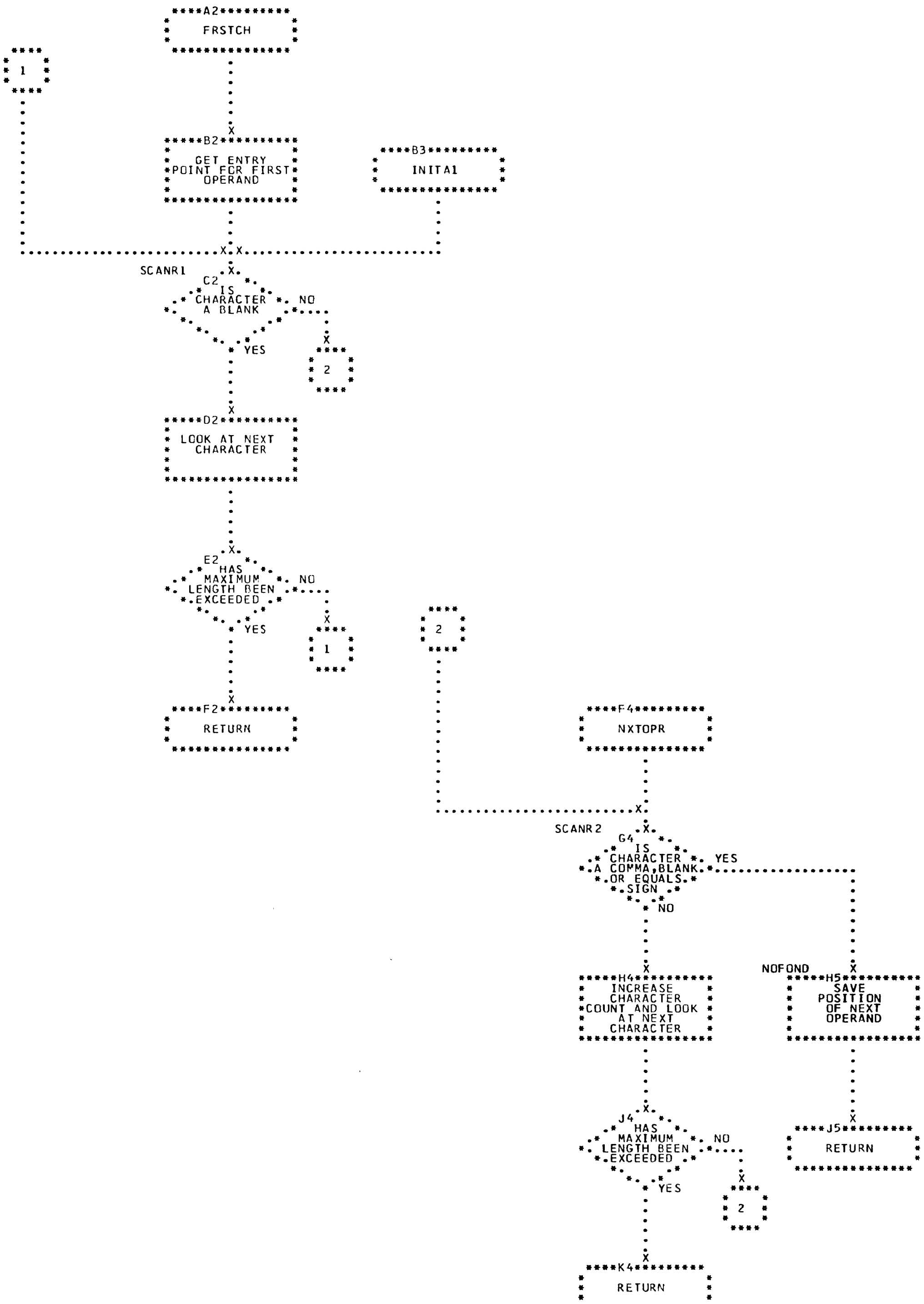


Chart SD. Scan Subroutines

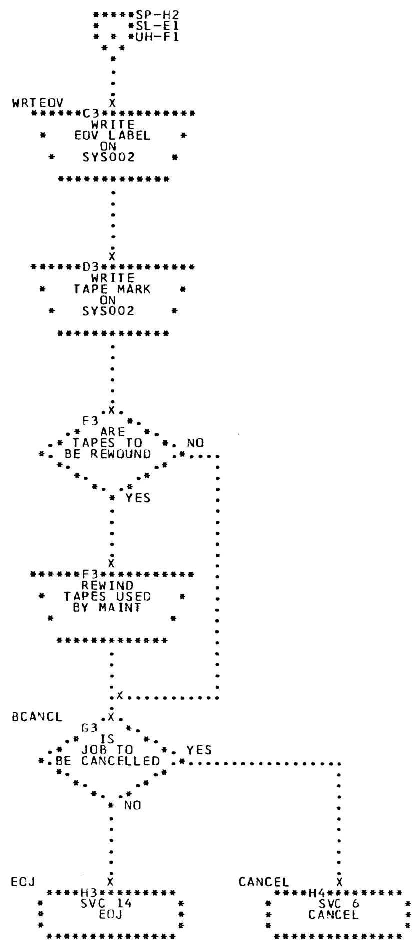


Chart SE. End of Job

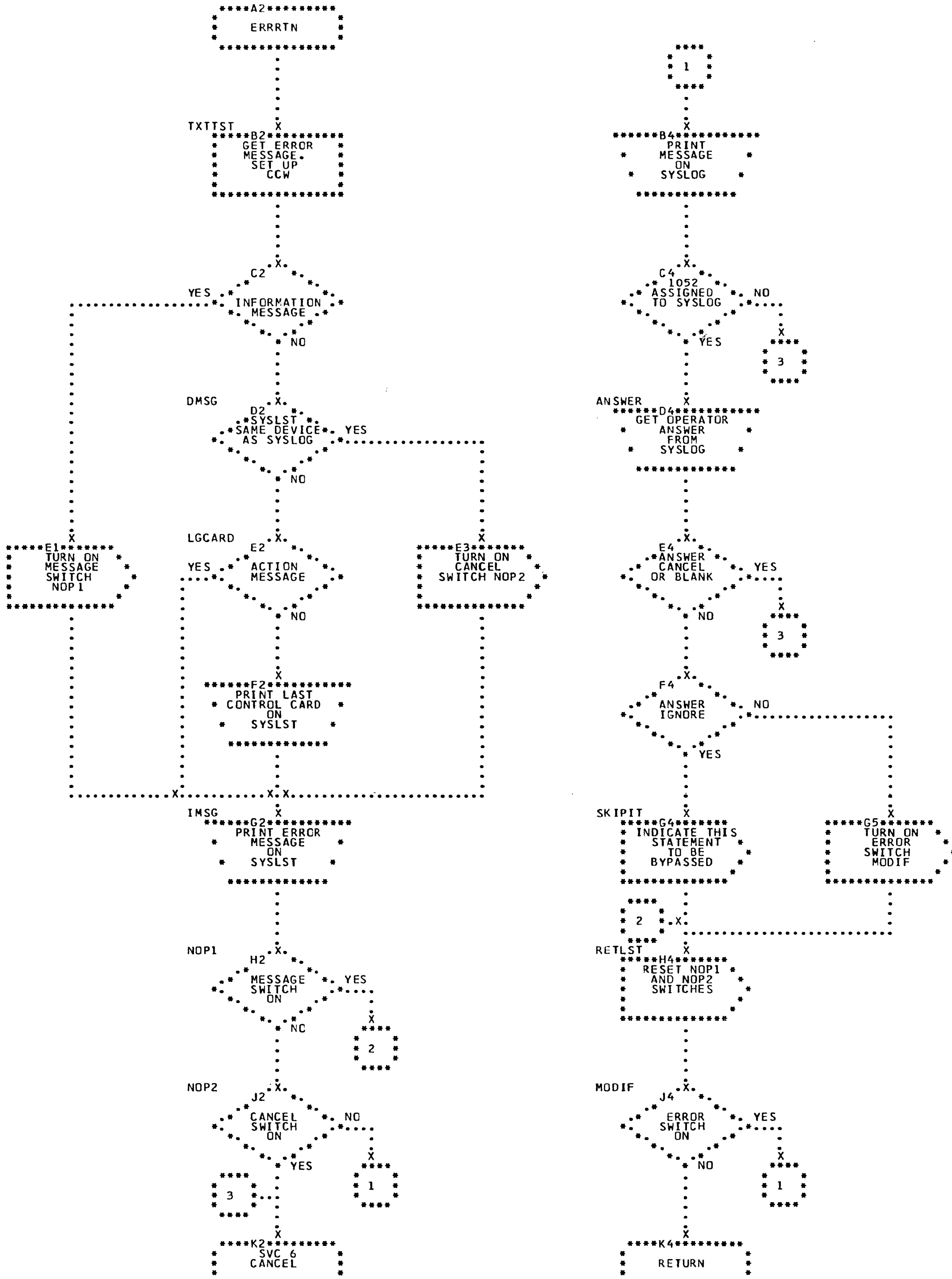


Chart SF. Error Message Subroutine

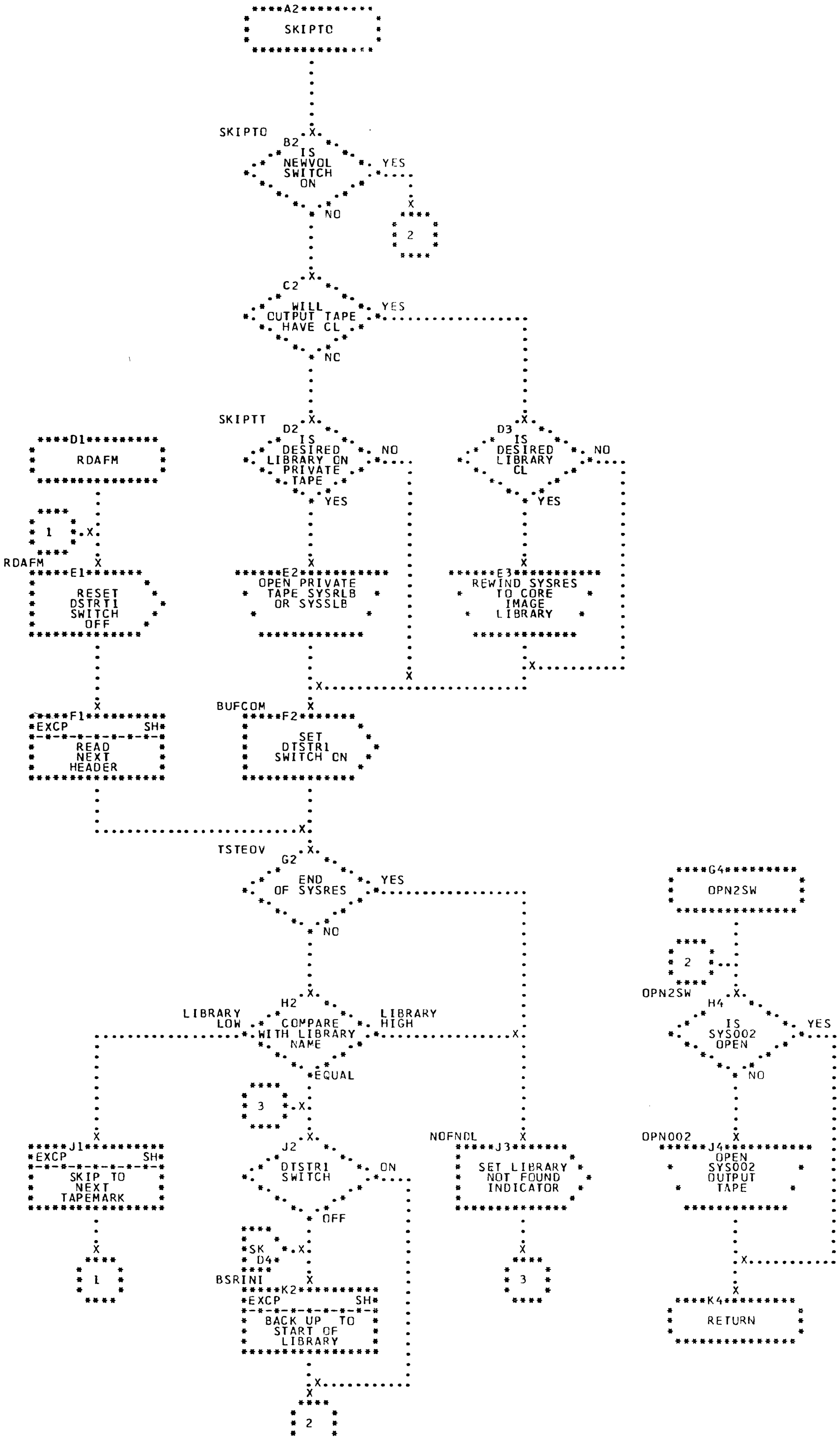


Chart SG. Find Library Subroutine

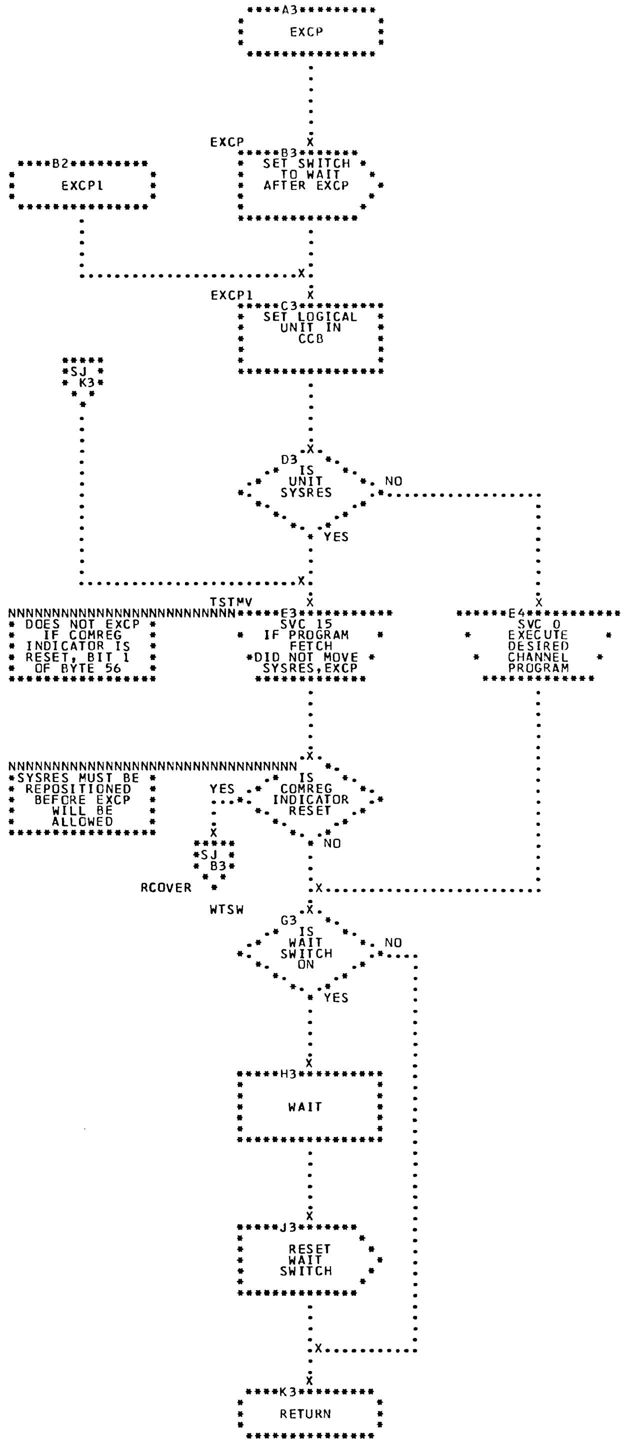


Chart SH. Execute Channel Program Subroutine

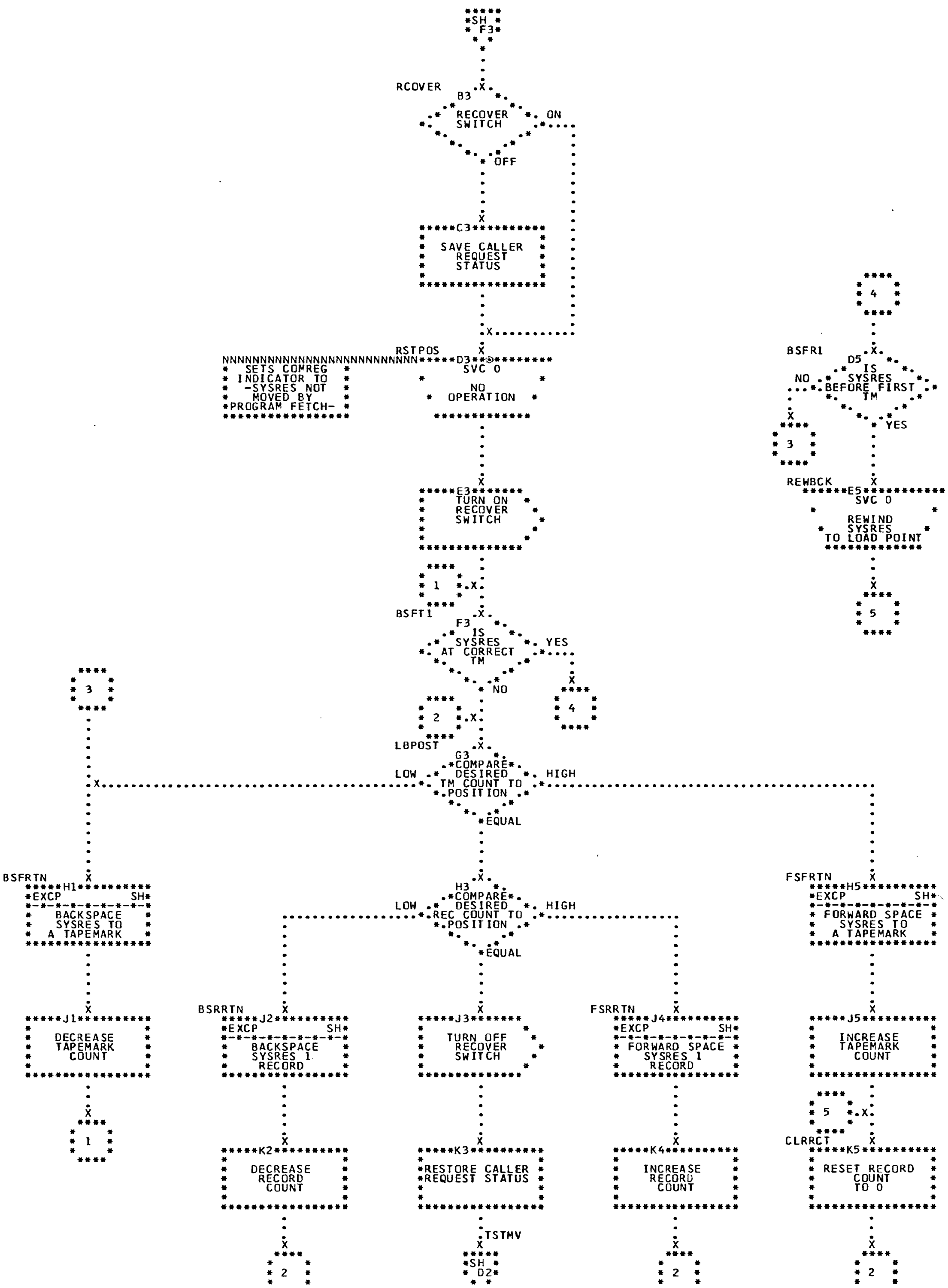


Chart SJ. Reposition SYRES for EXCP

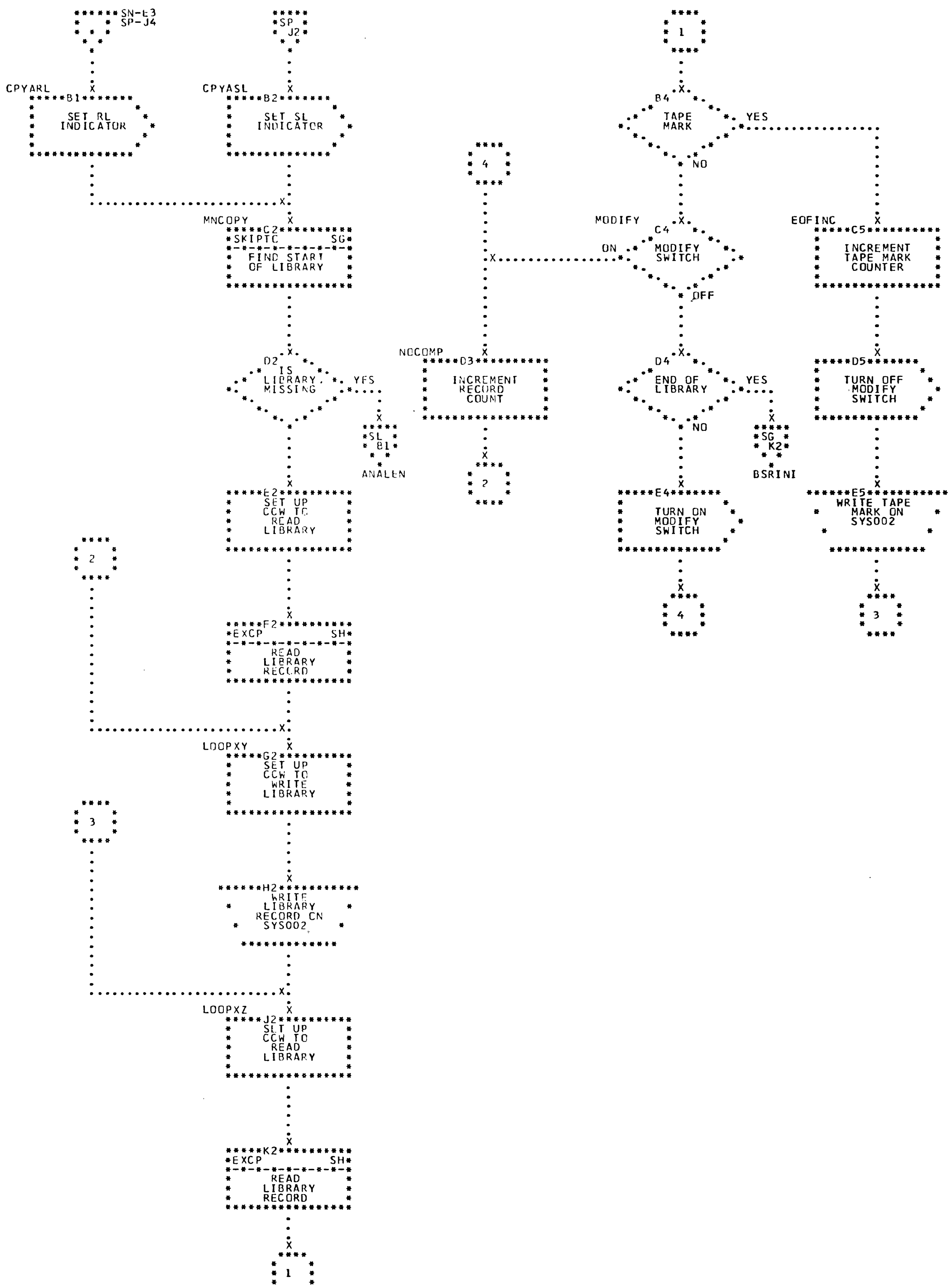


Chart SK. Copy Complete Library

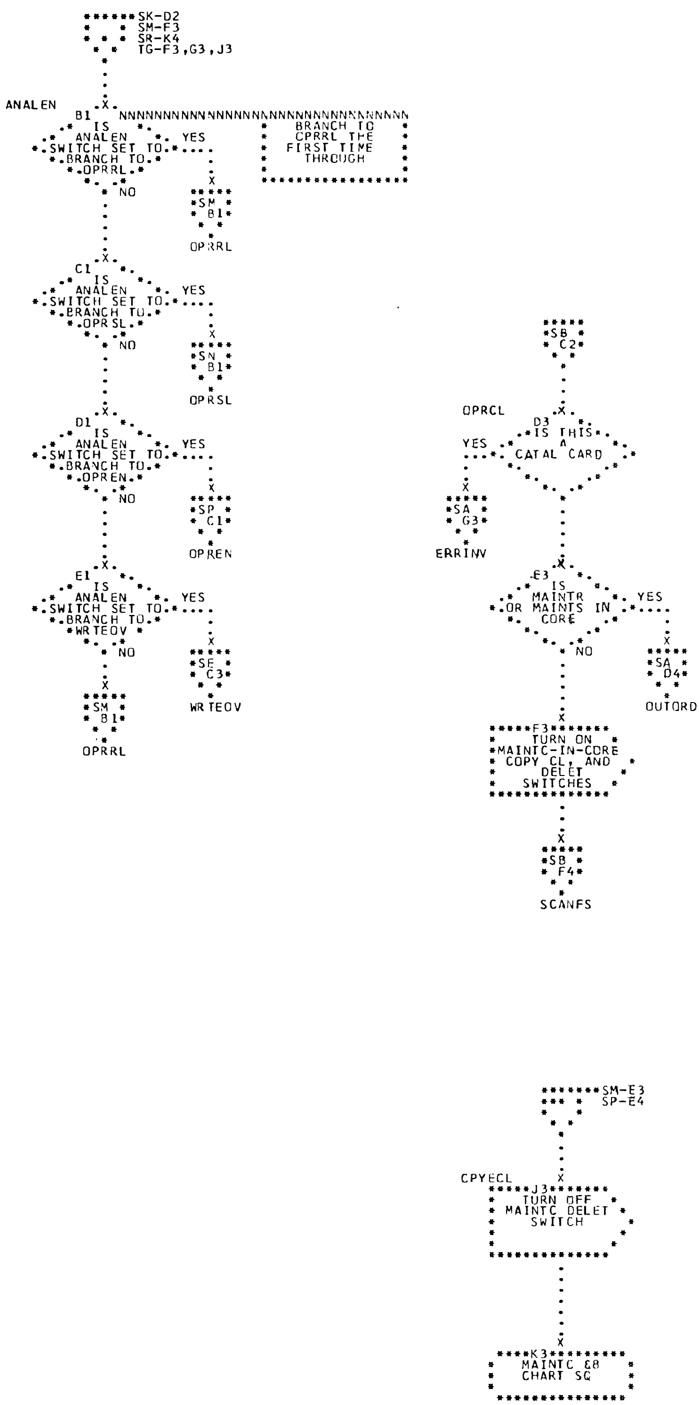


Chart SL. Fetch for Core Image Library

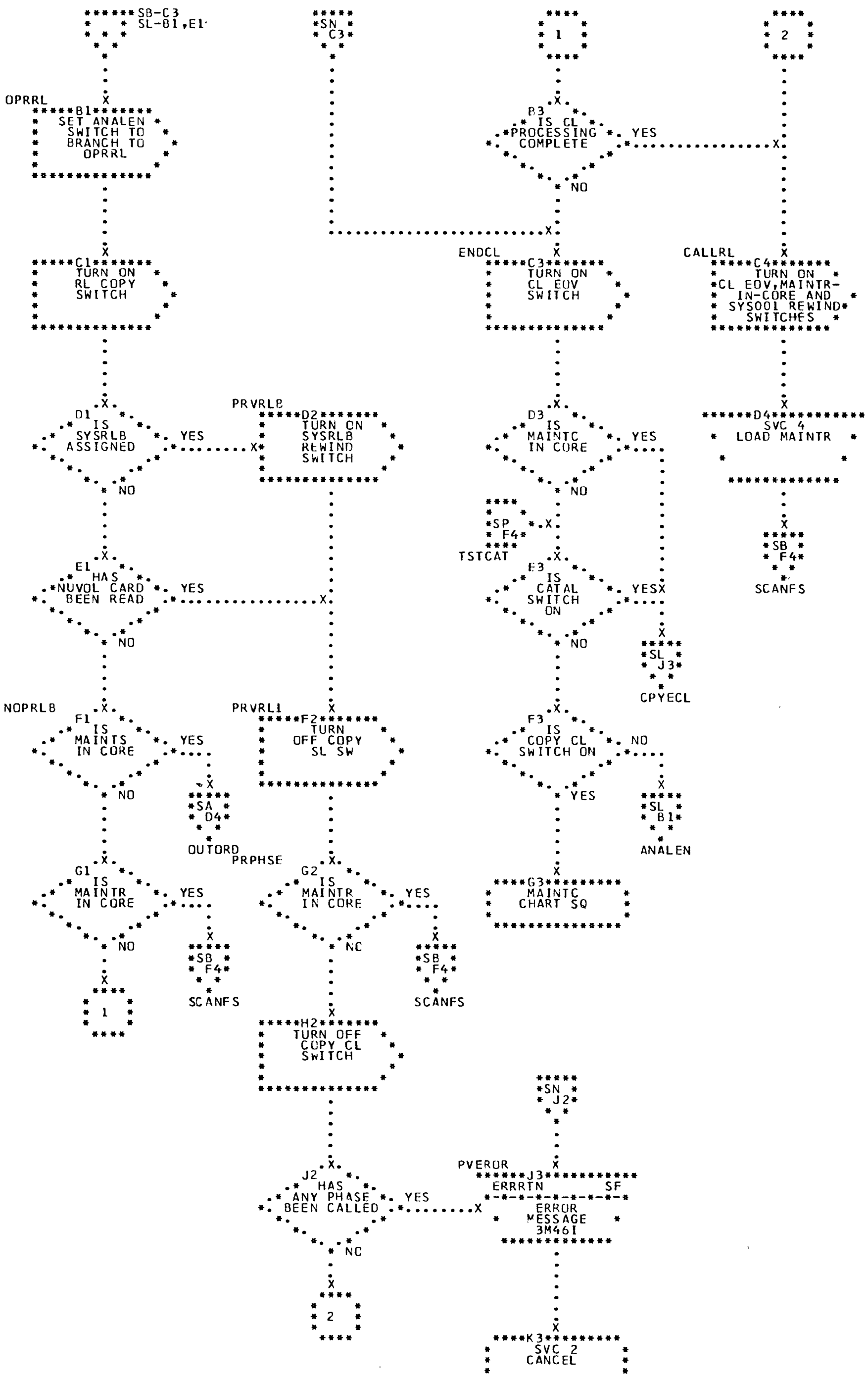


Chart SM. Fetch for Relocatable Library

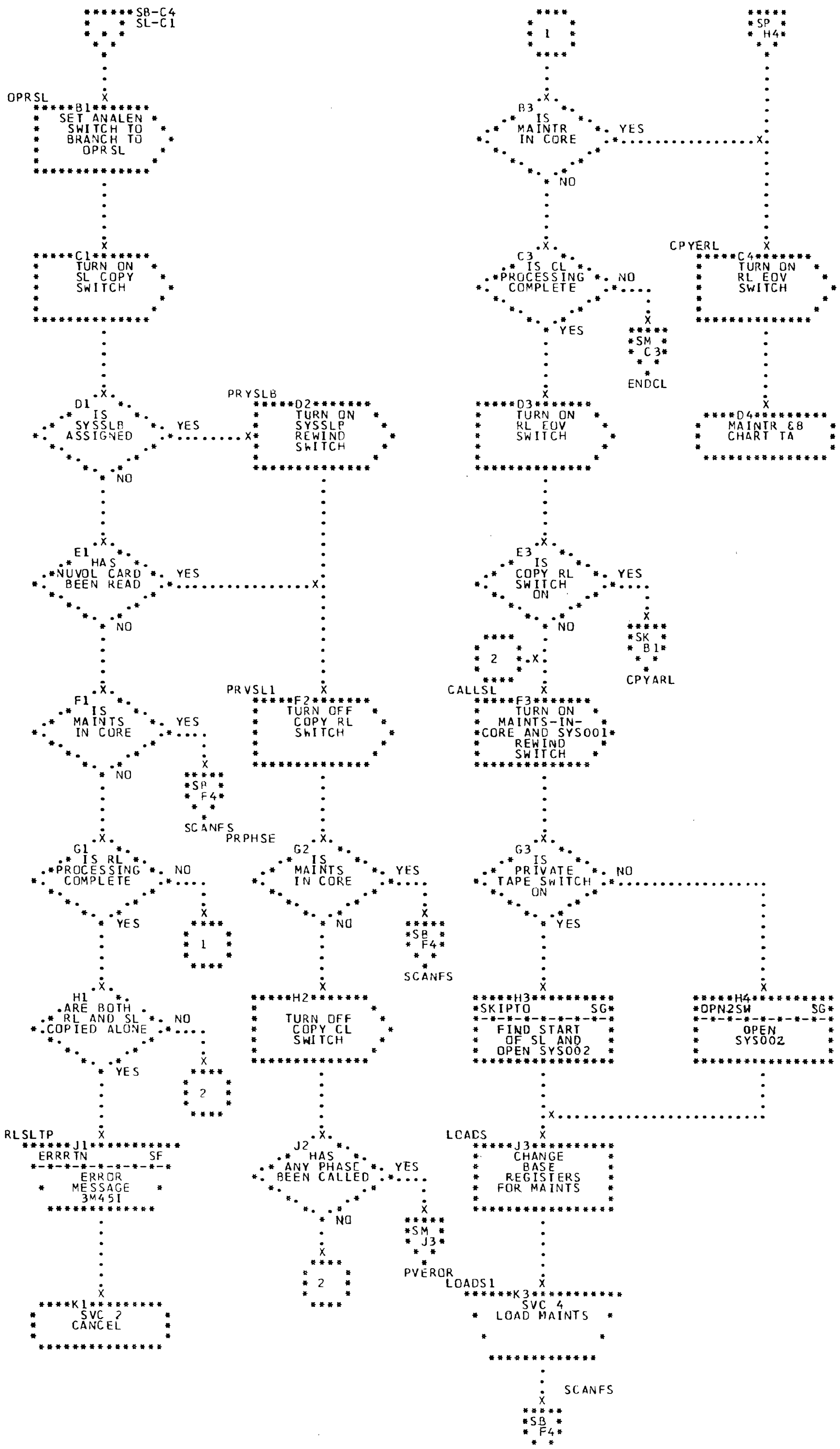


Chart SN. Fetch for Source Statement Library

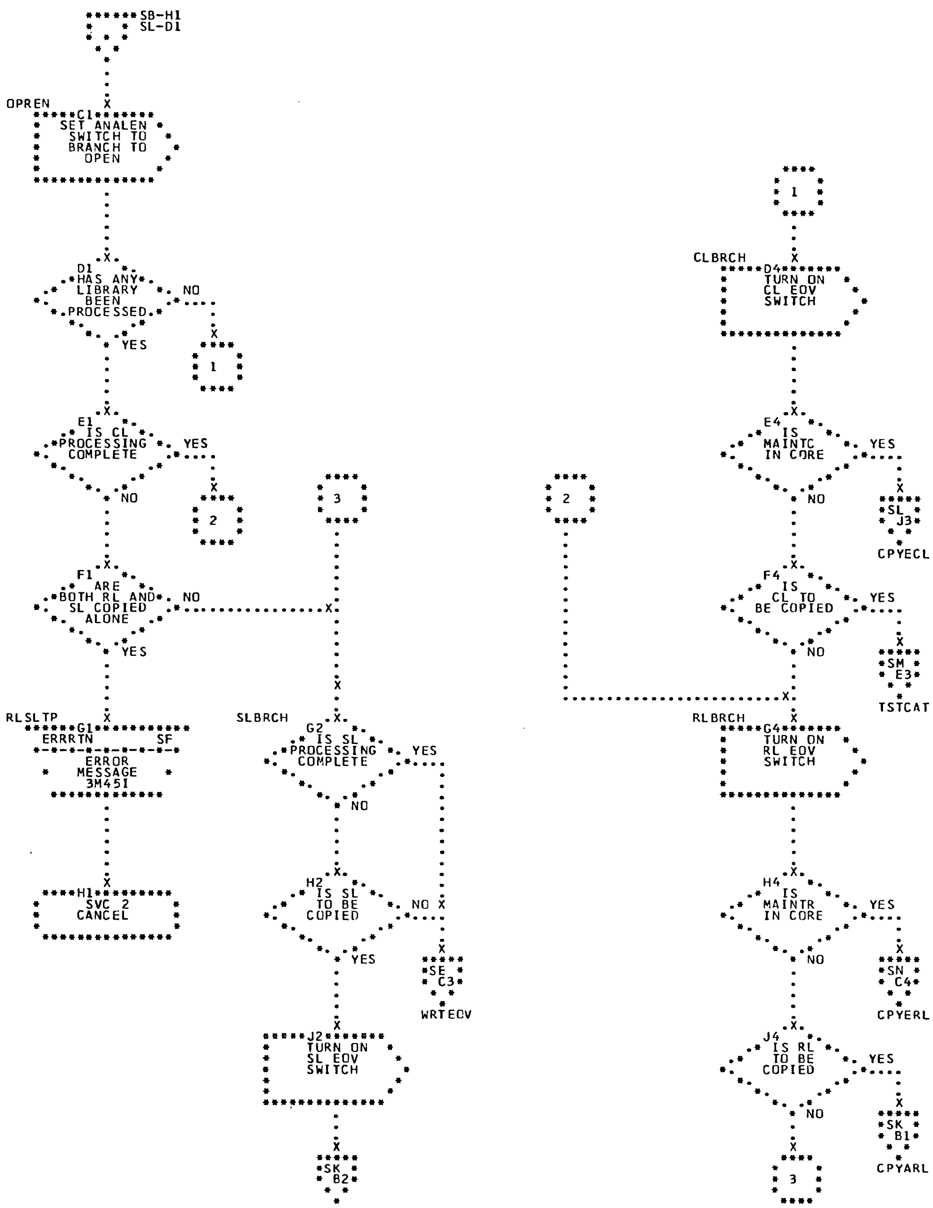


Chart SP. End of Fetch

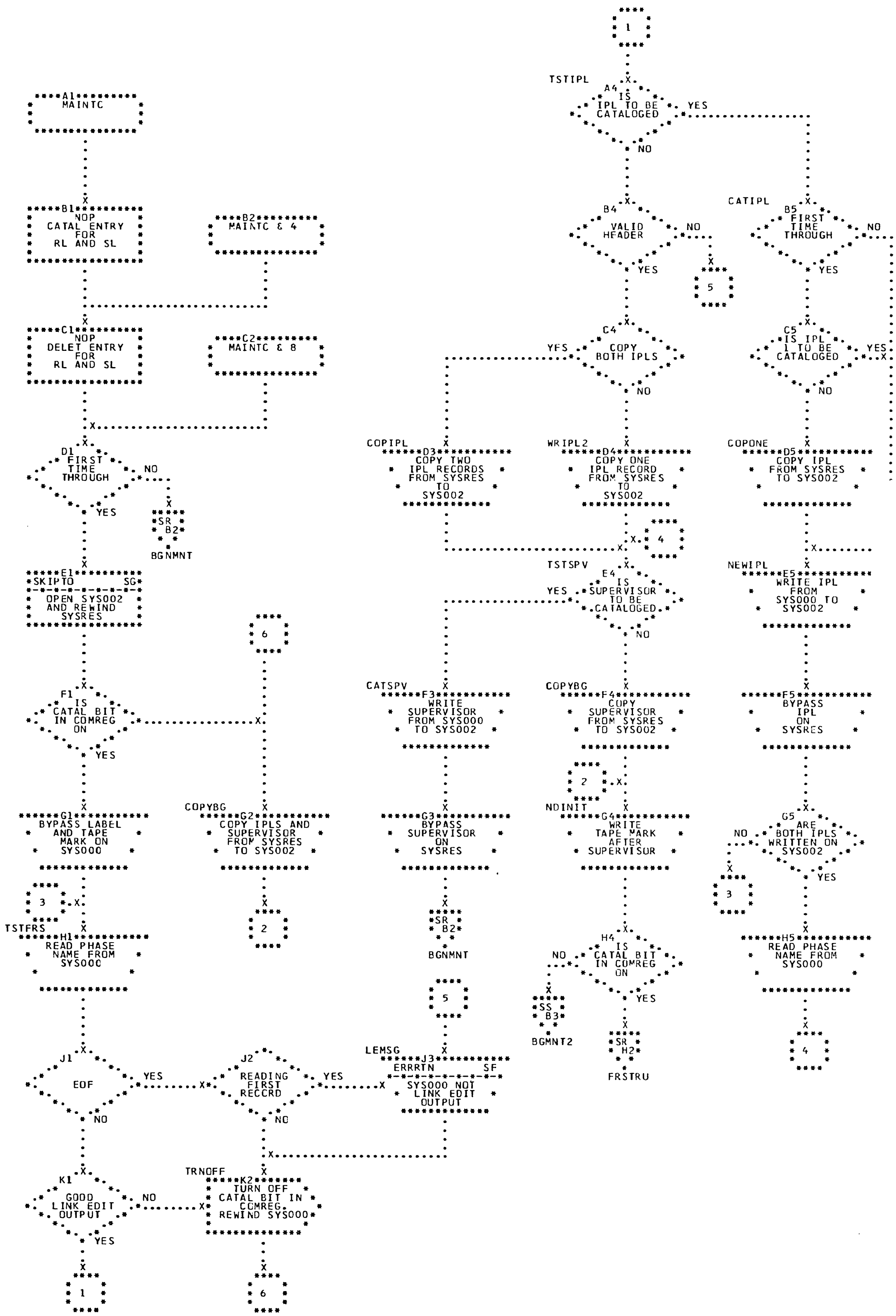


Chart SQ. Catalog IPL's and Supervisor

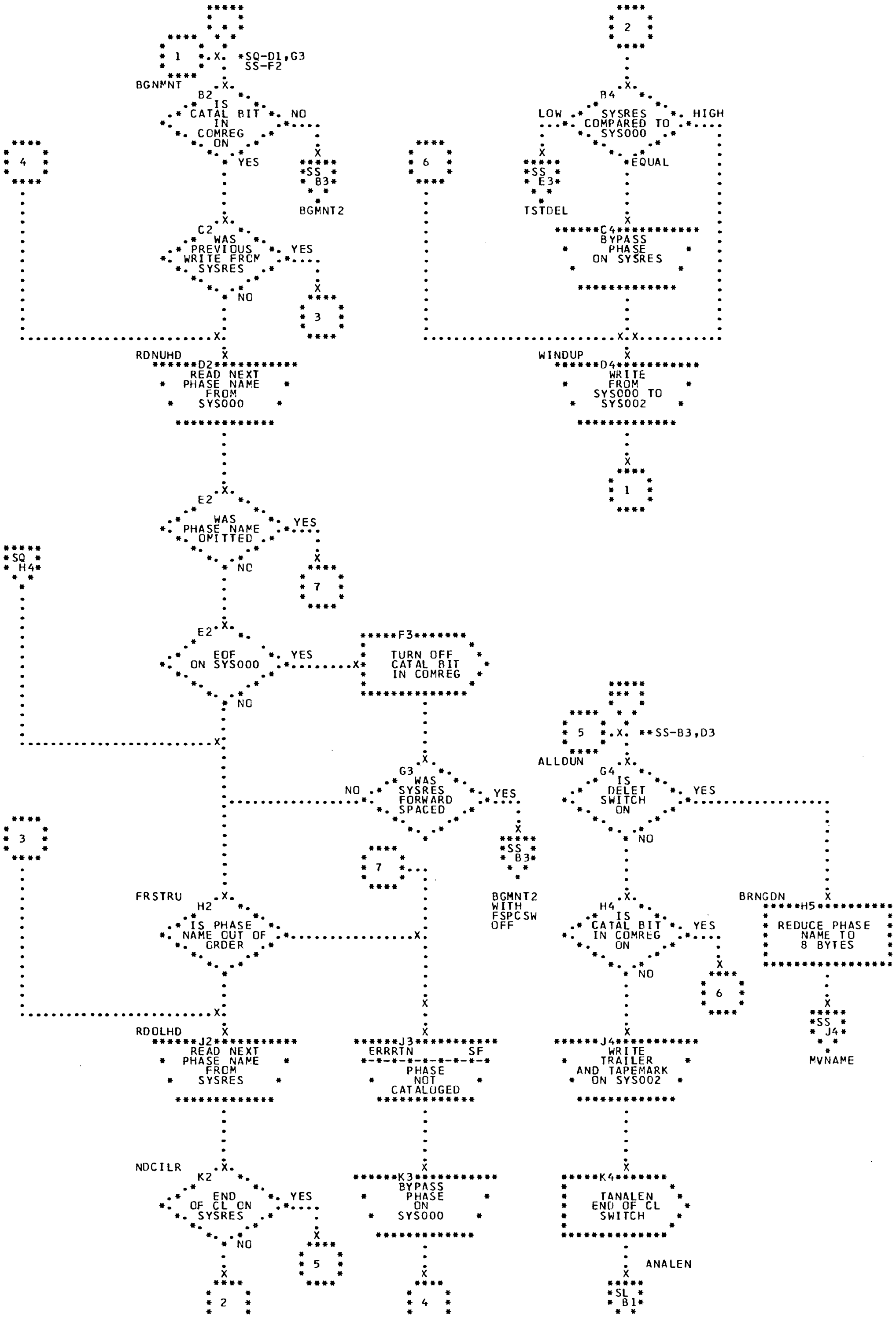


Chart SR. Catalog

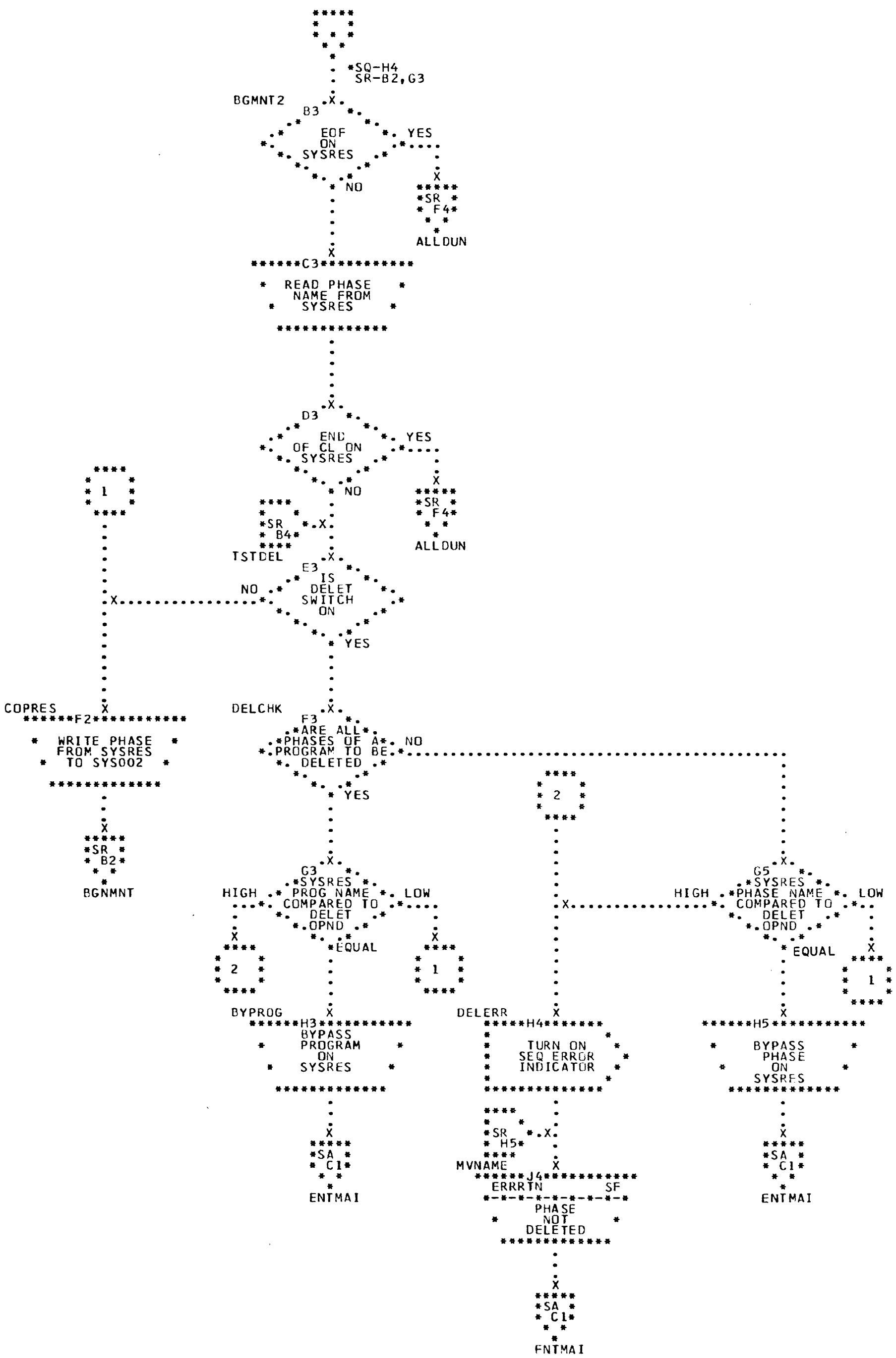


Chart SS. Delete

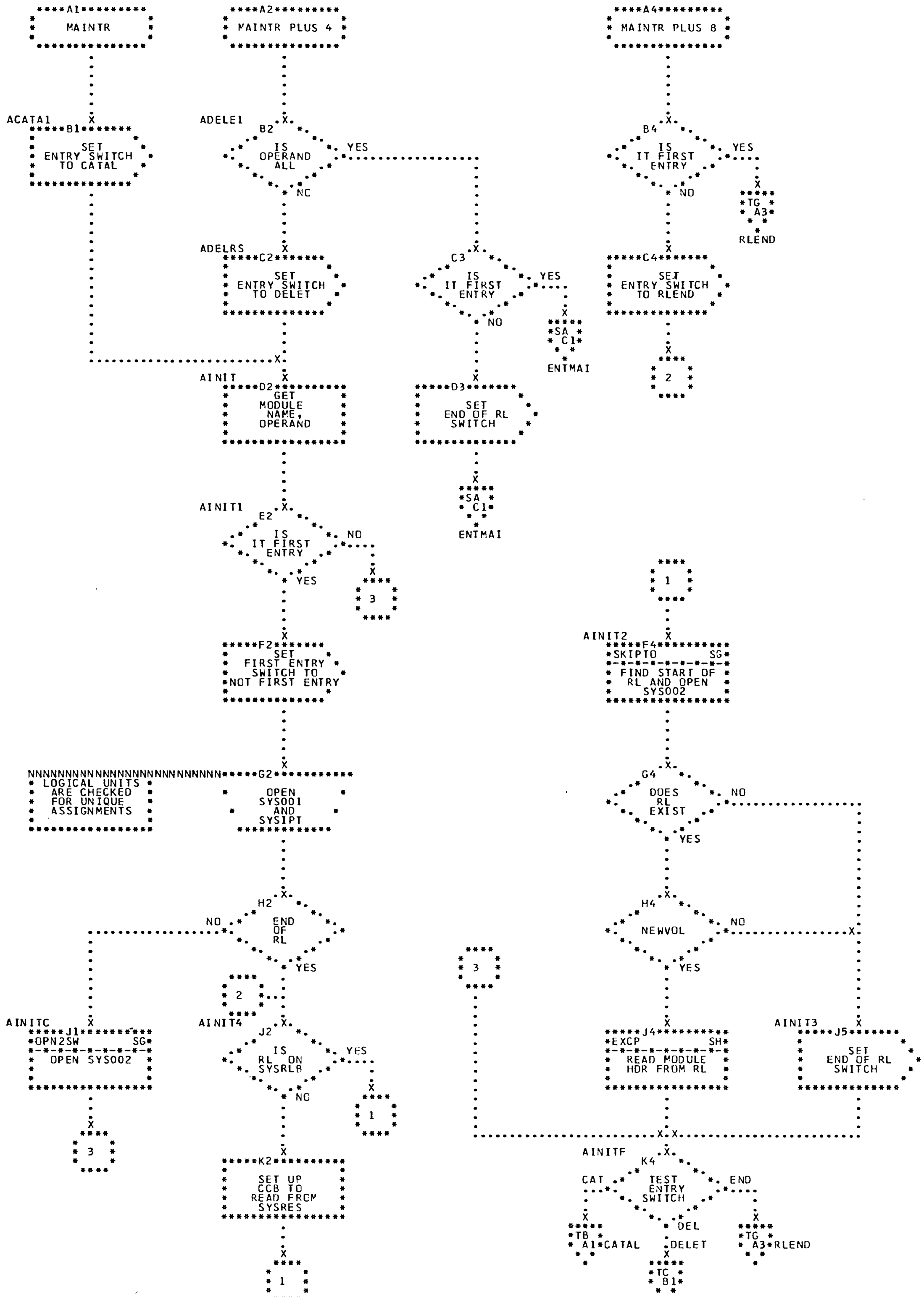


Chart TA. Initialization

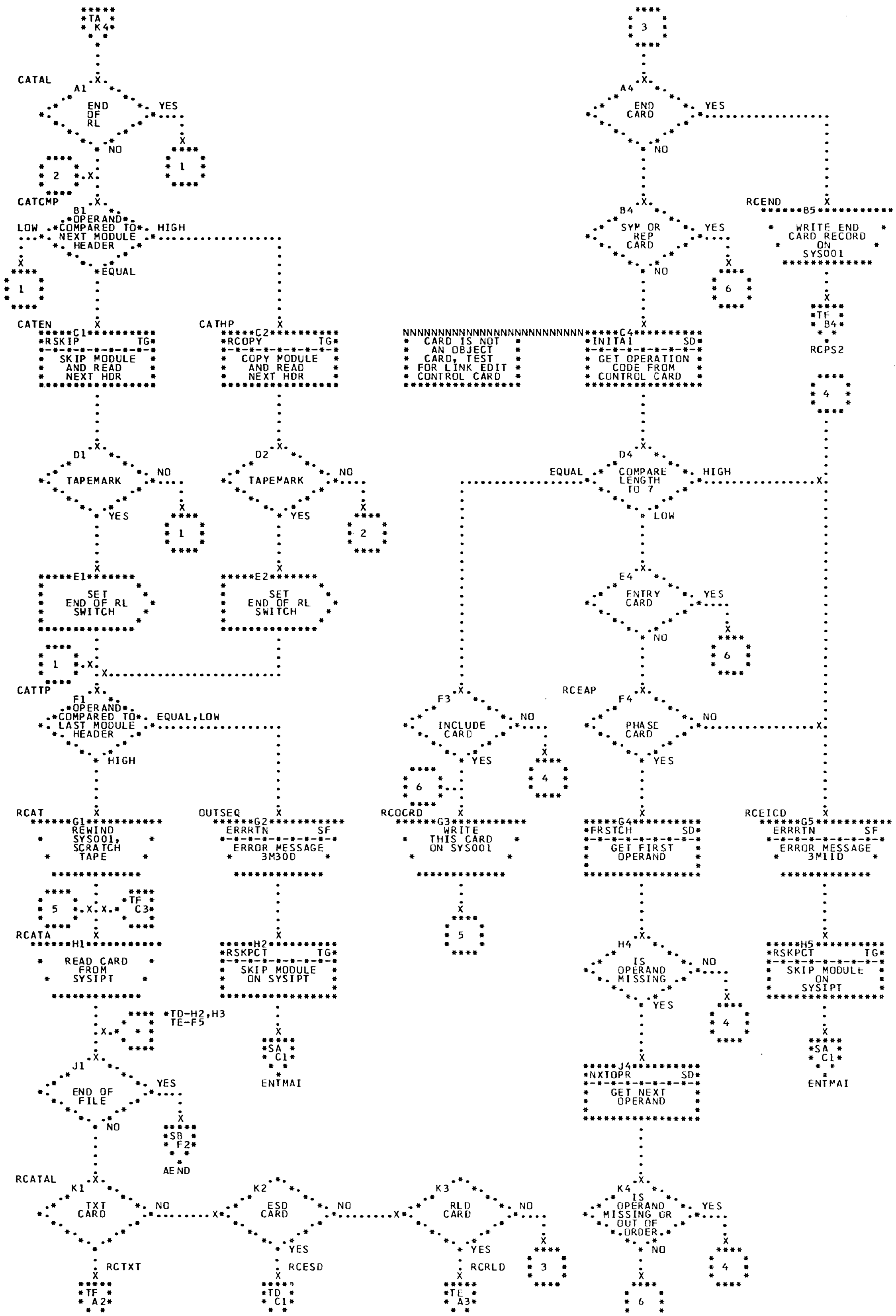


Chart TB. Catalog

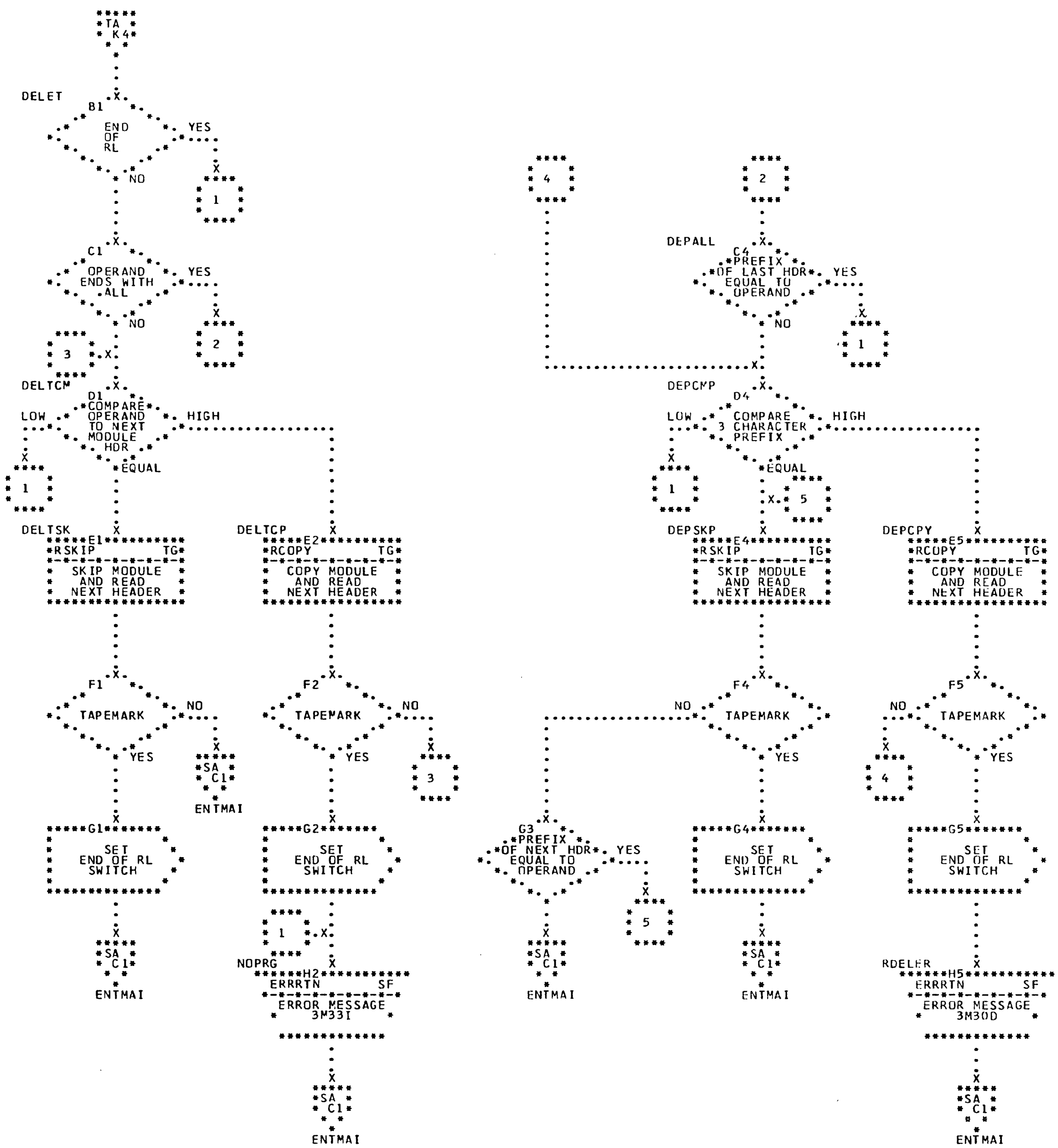


Chart TC. Delete

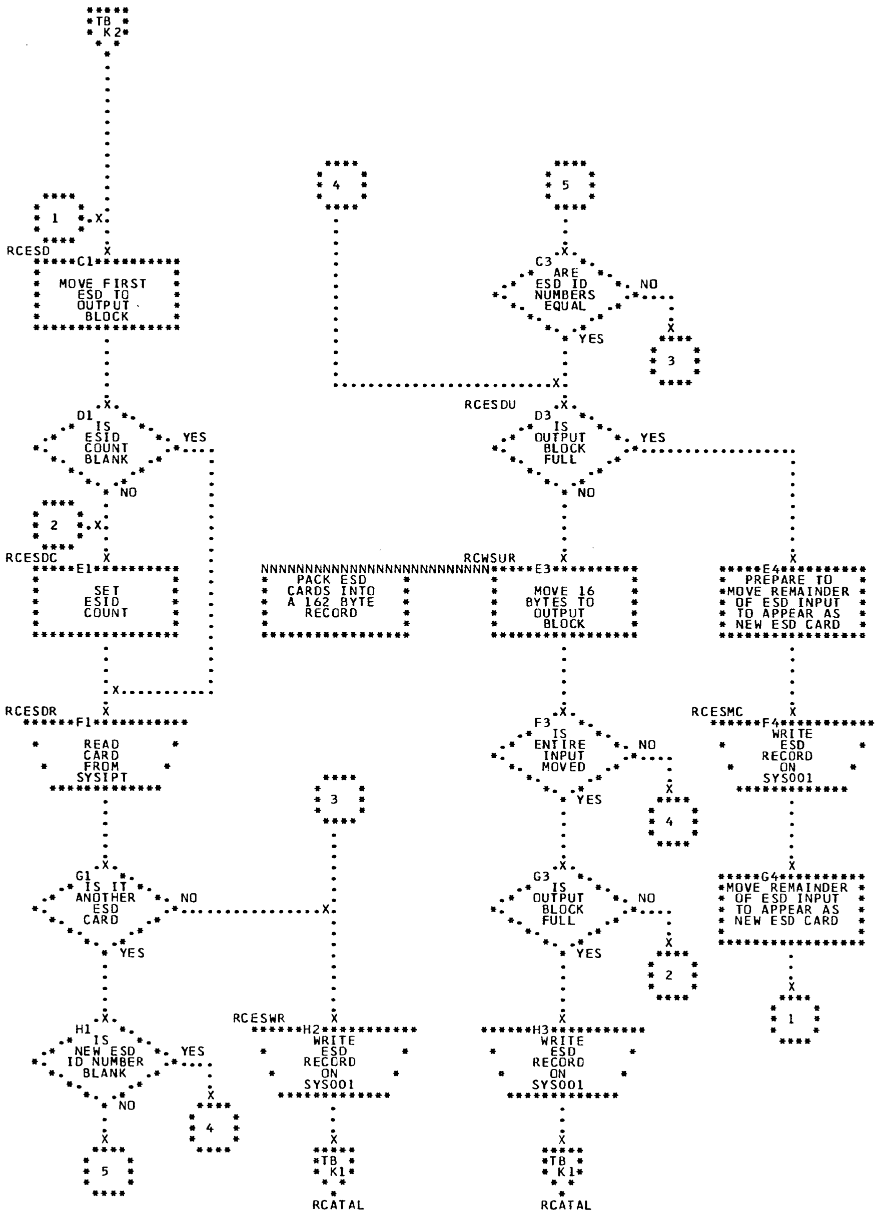


Chart TD. Catalog ESD

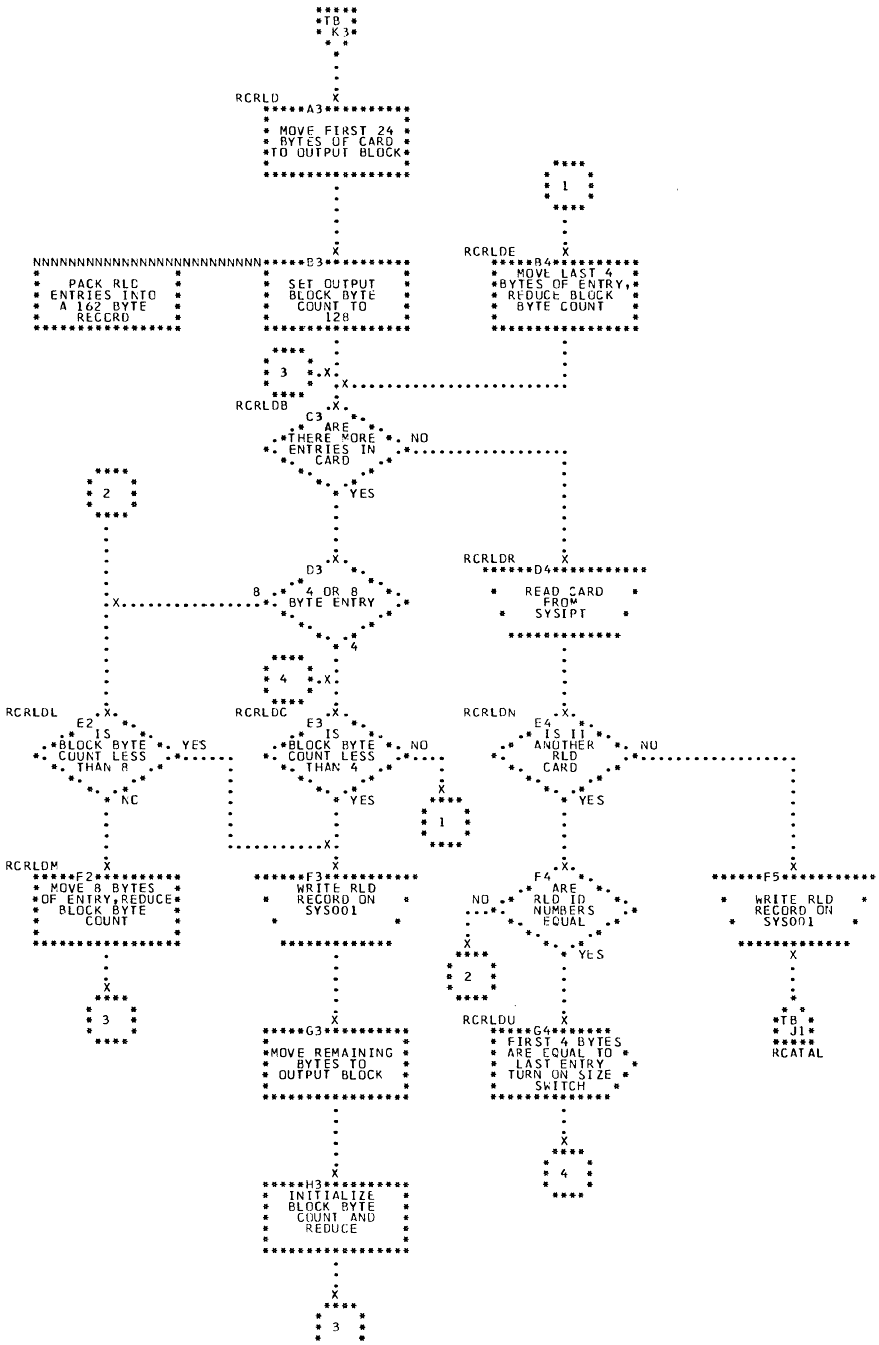


Chart TE. Catalog RLD

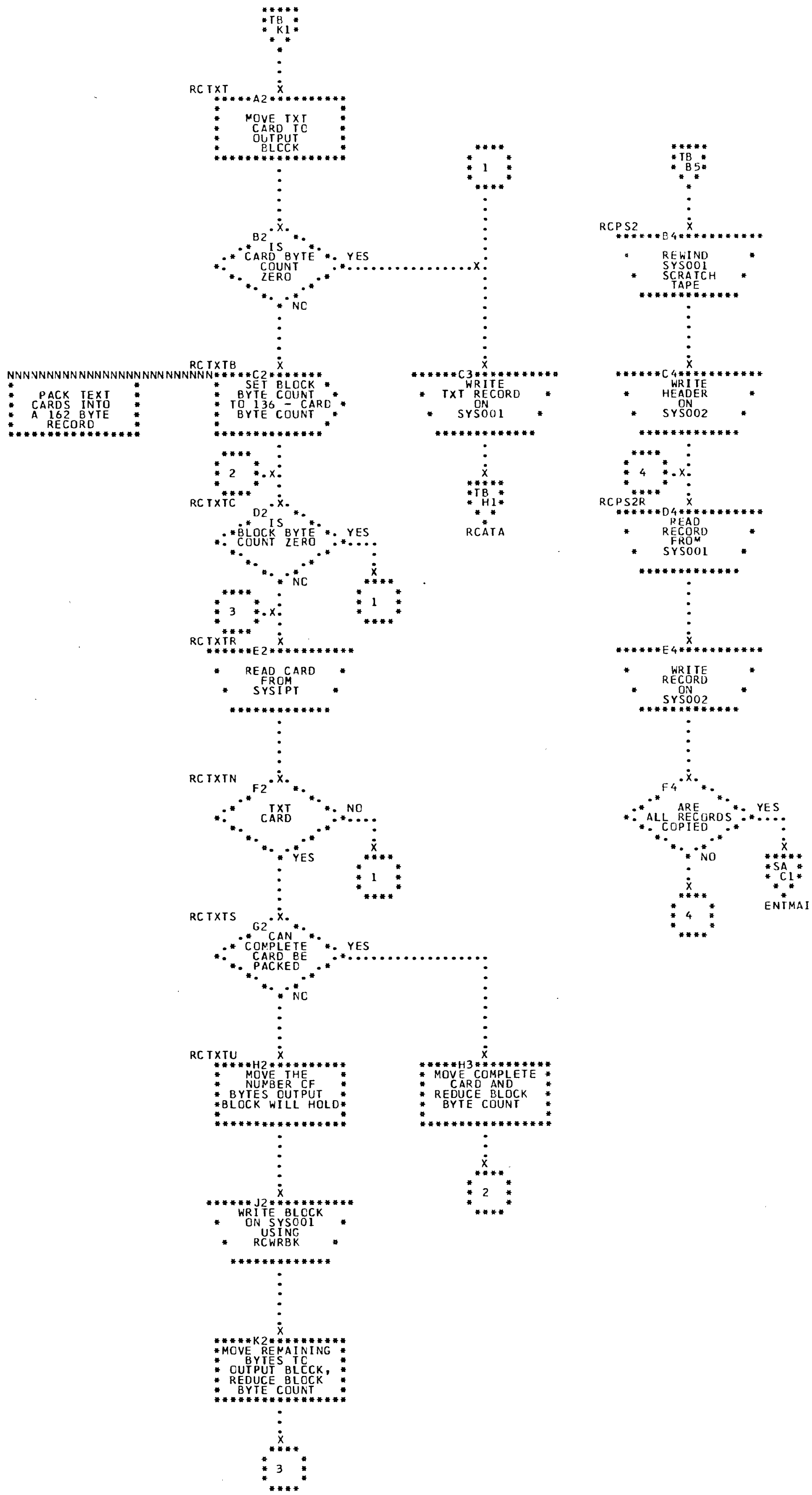


Chart TF. Catalog TXT

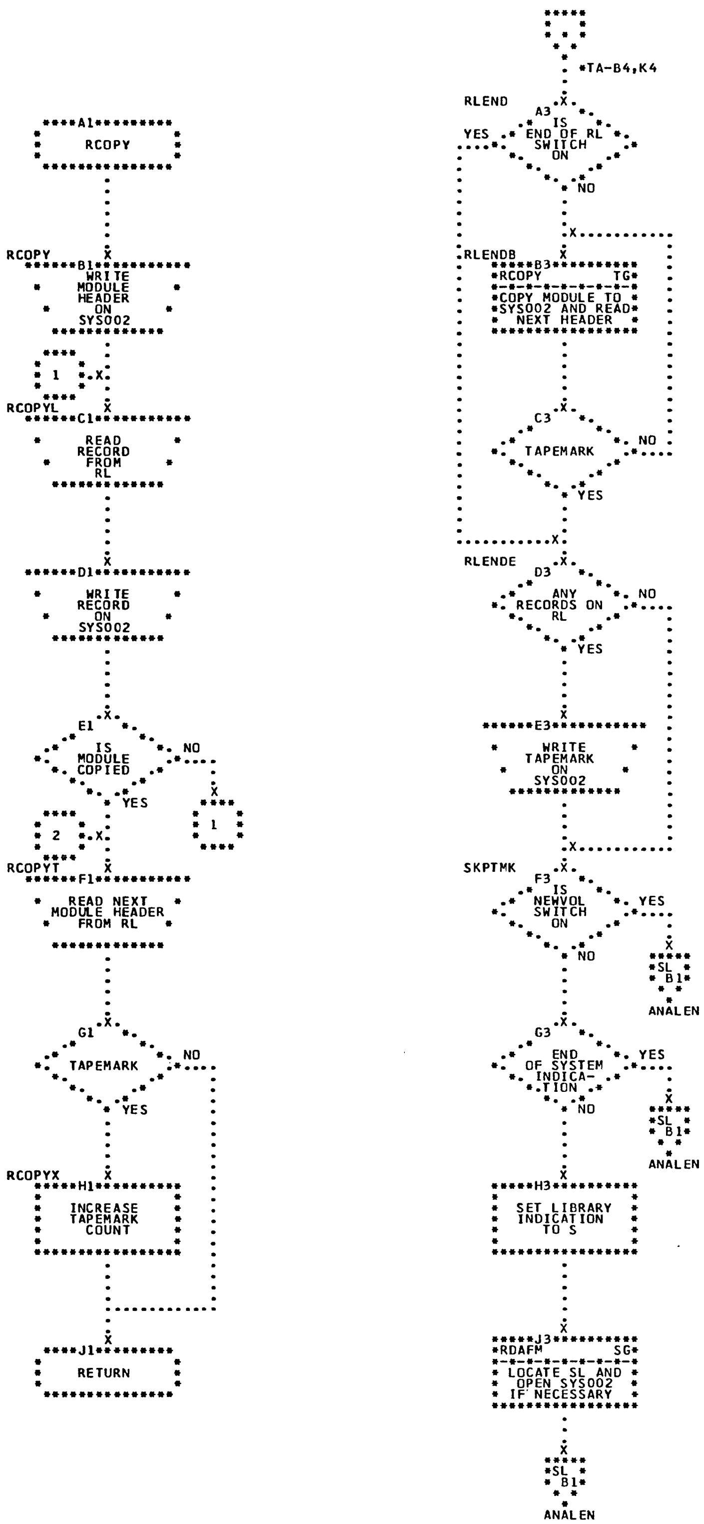


Chart TG. I/O Subroutines

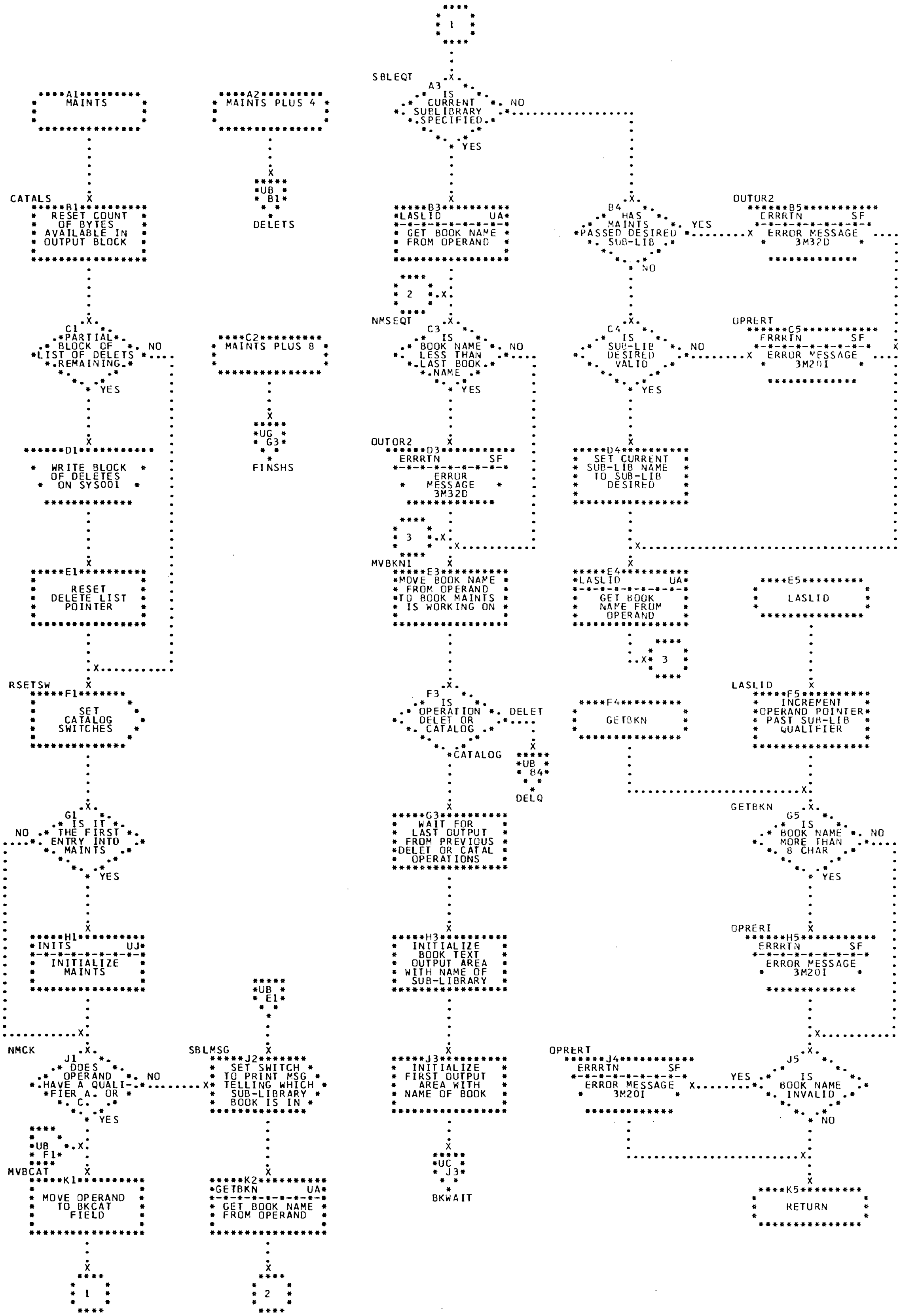


Chart UA. Catalog

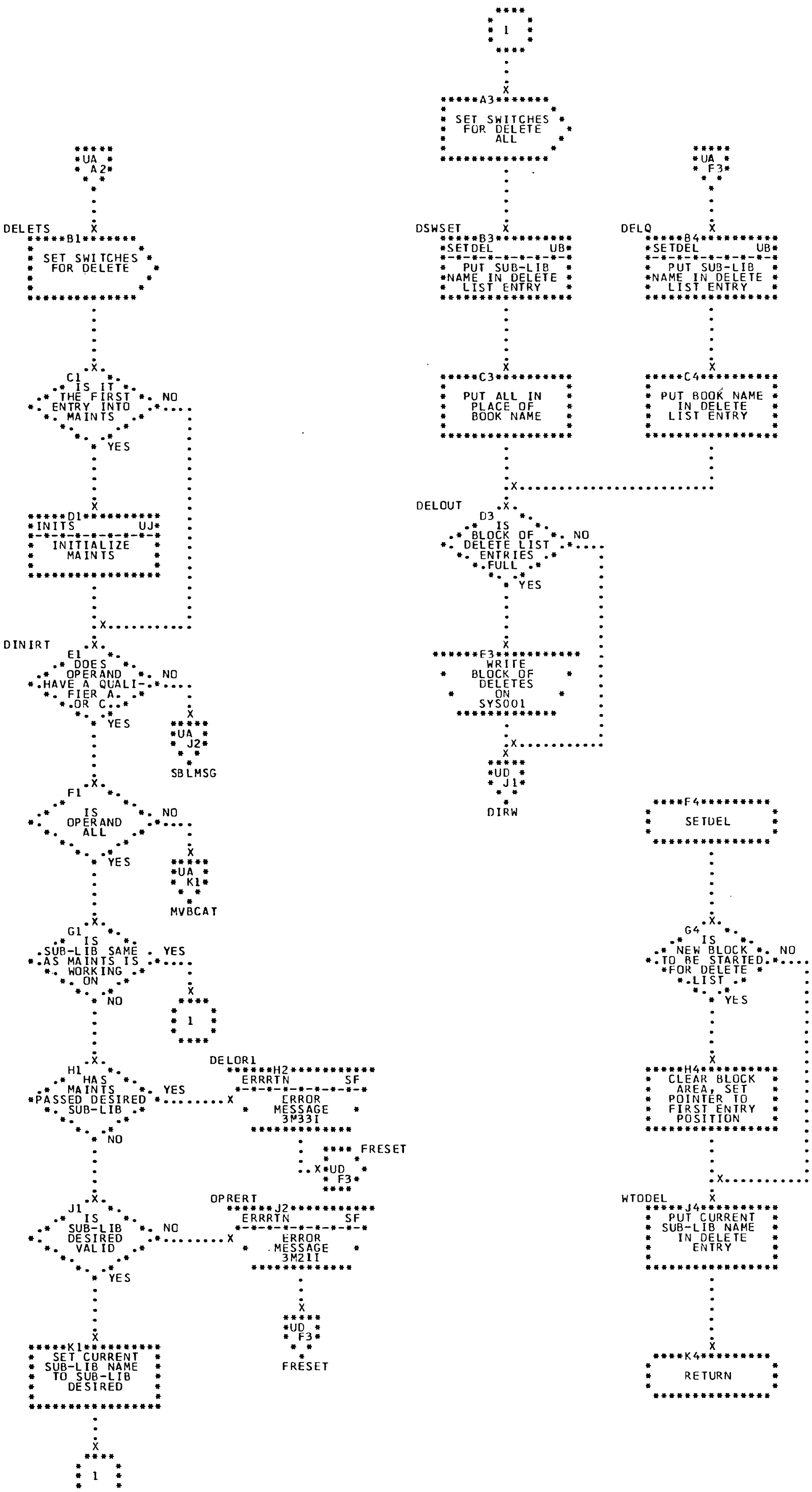


Chart UB. Delete

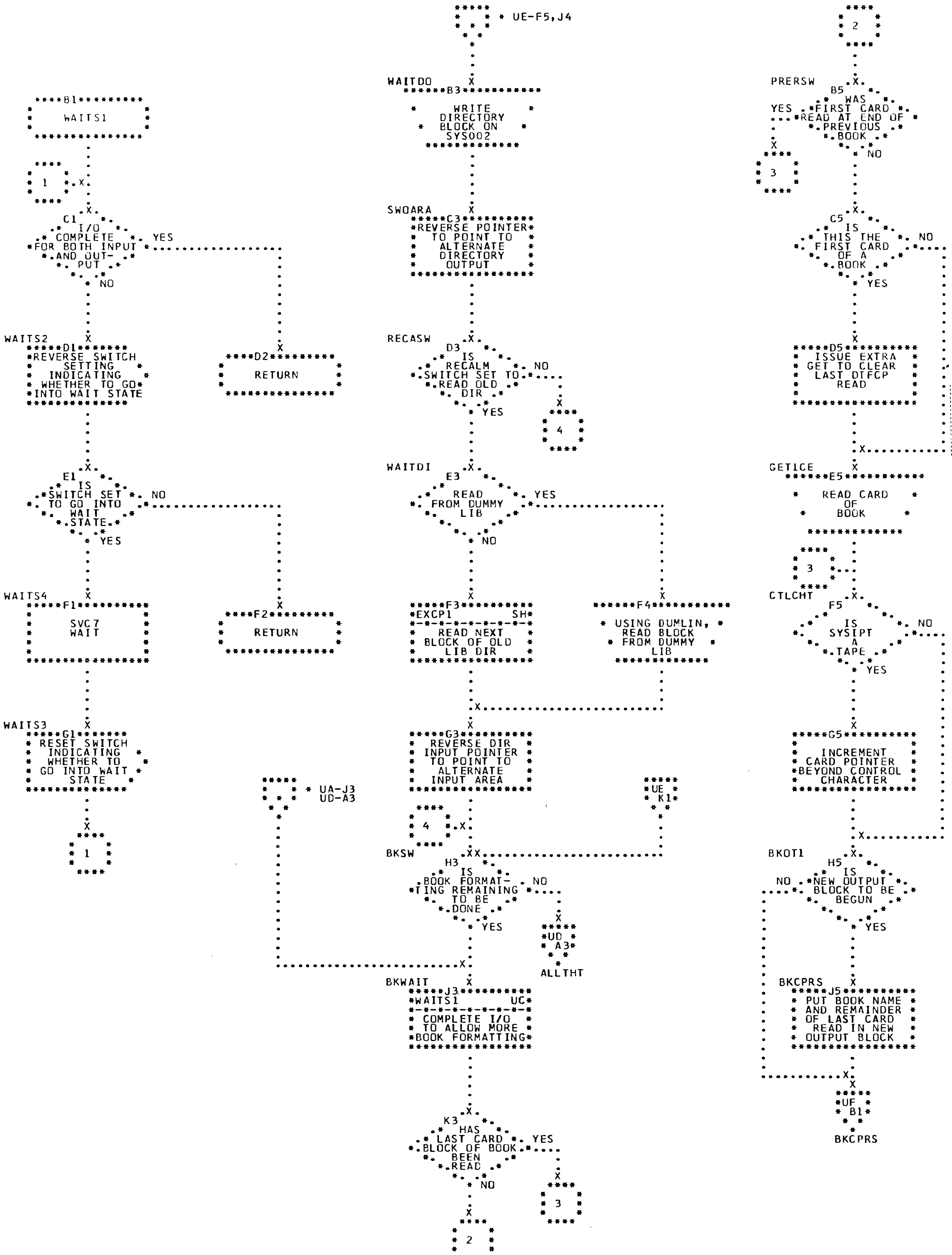


Chart UC. Do I/O

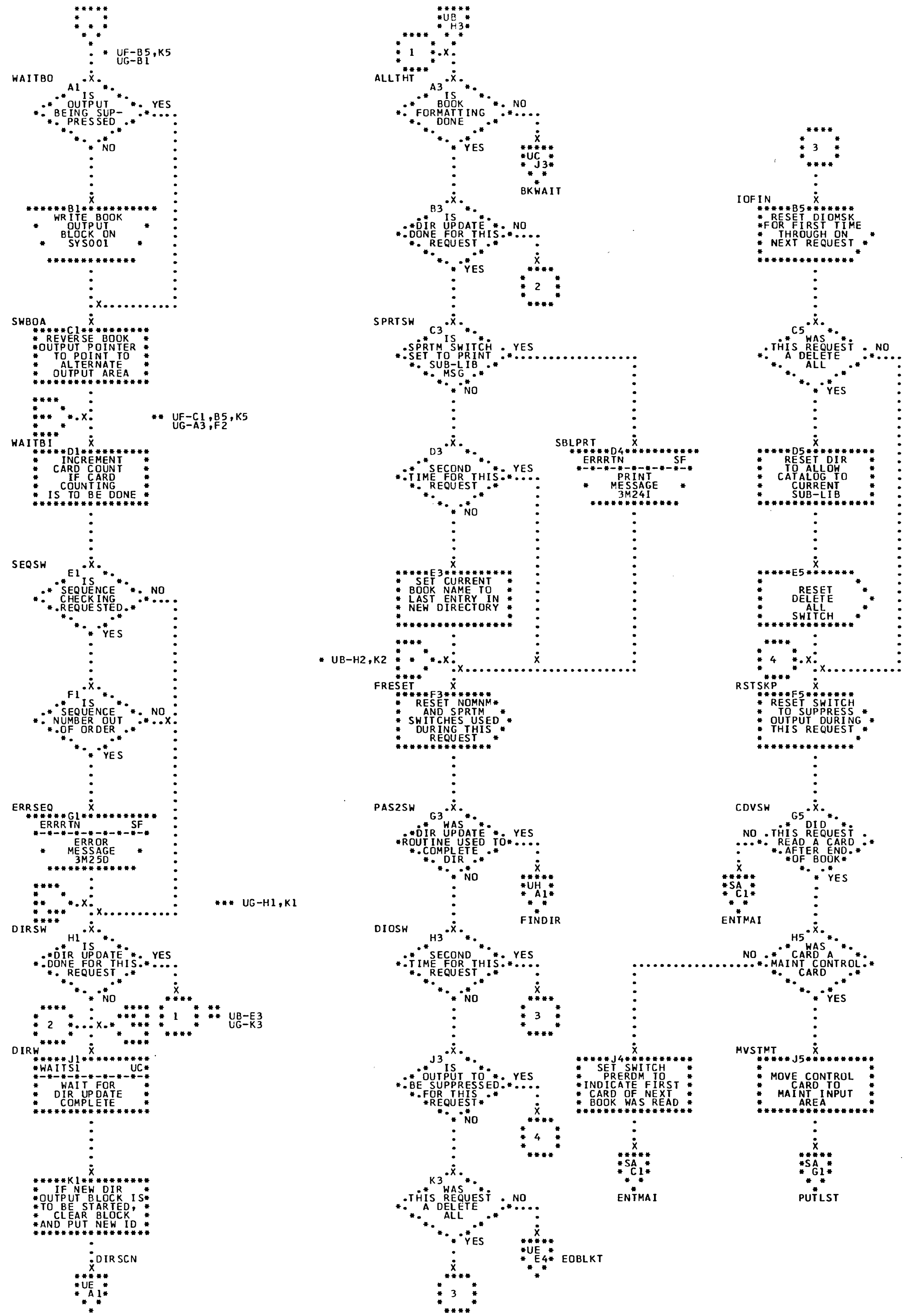


Chart UD. End of Book

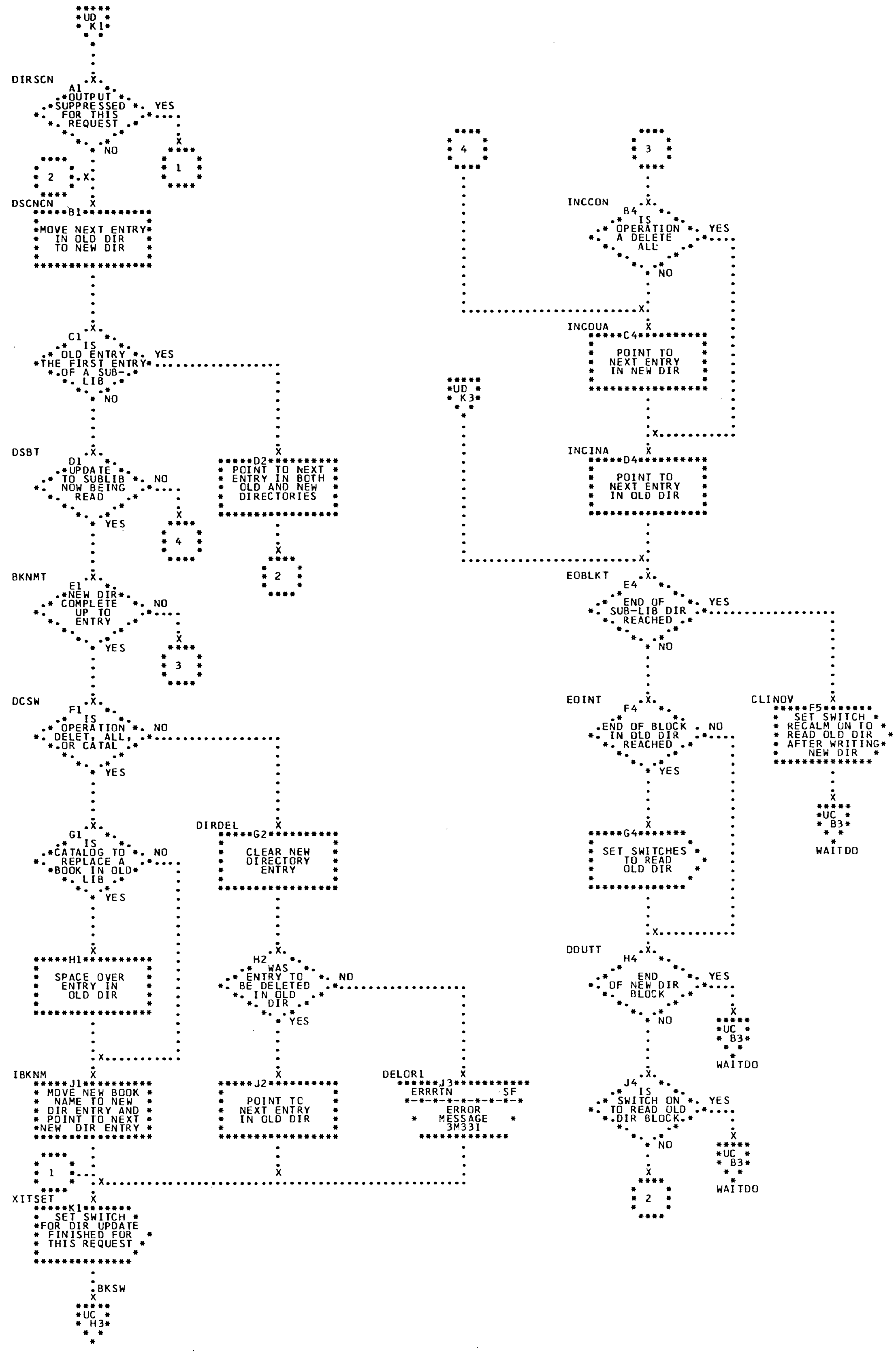


Chart UE. Update Directory

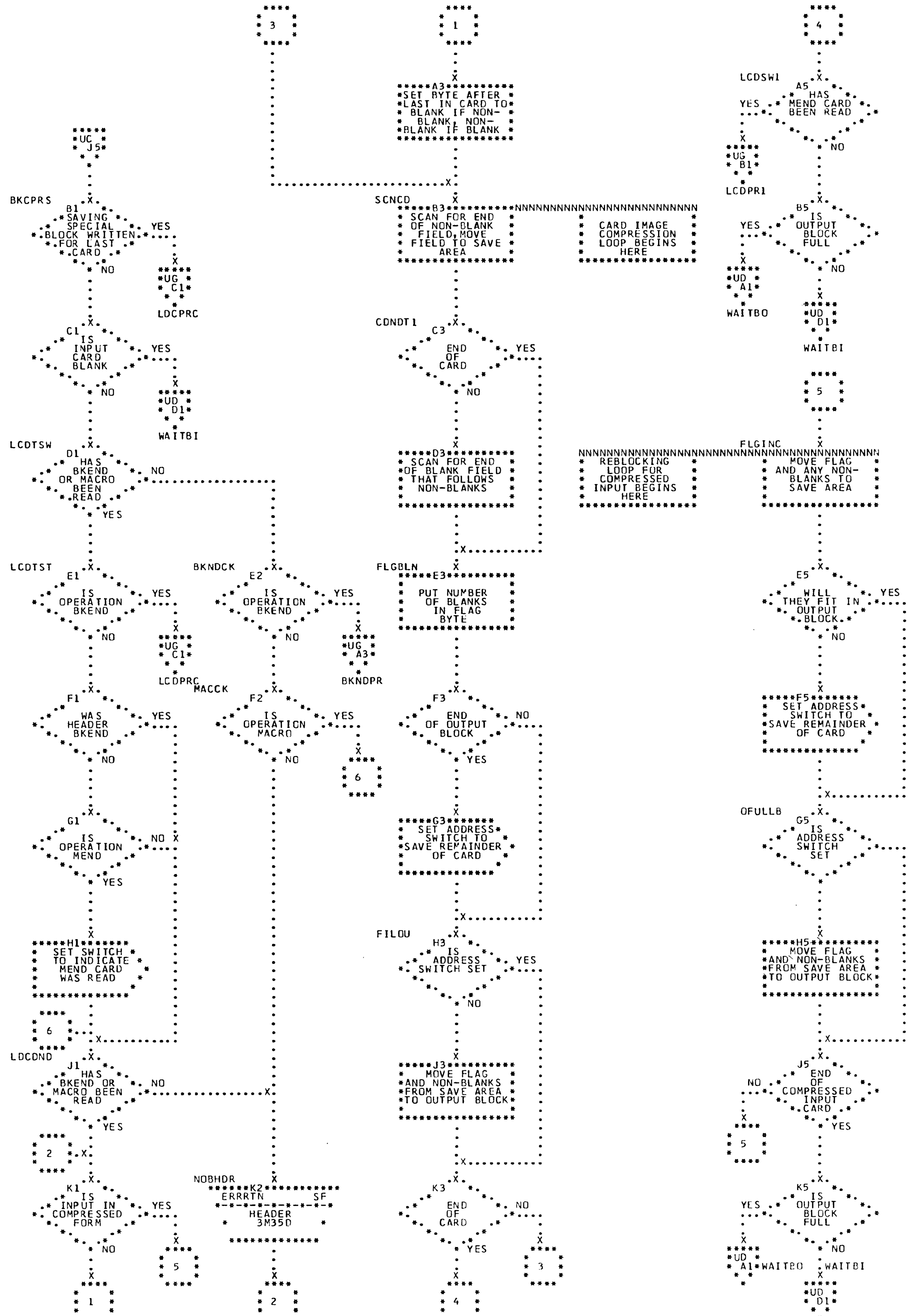


Chart UF. Compress Book

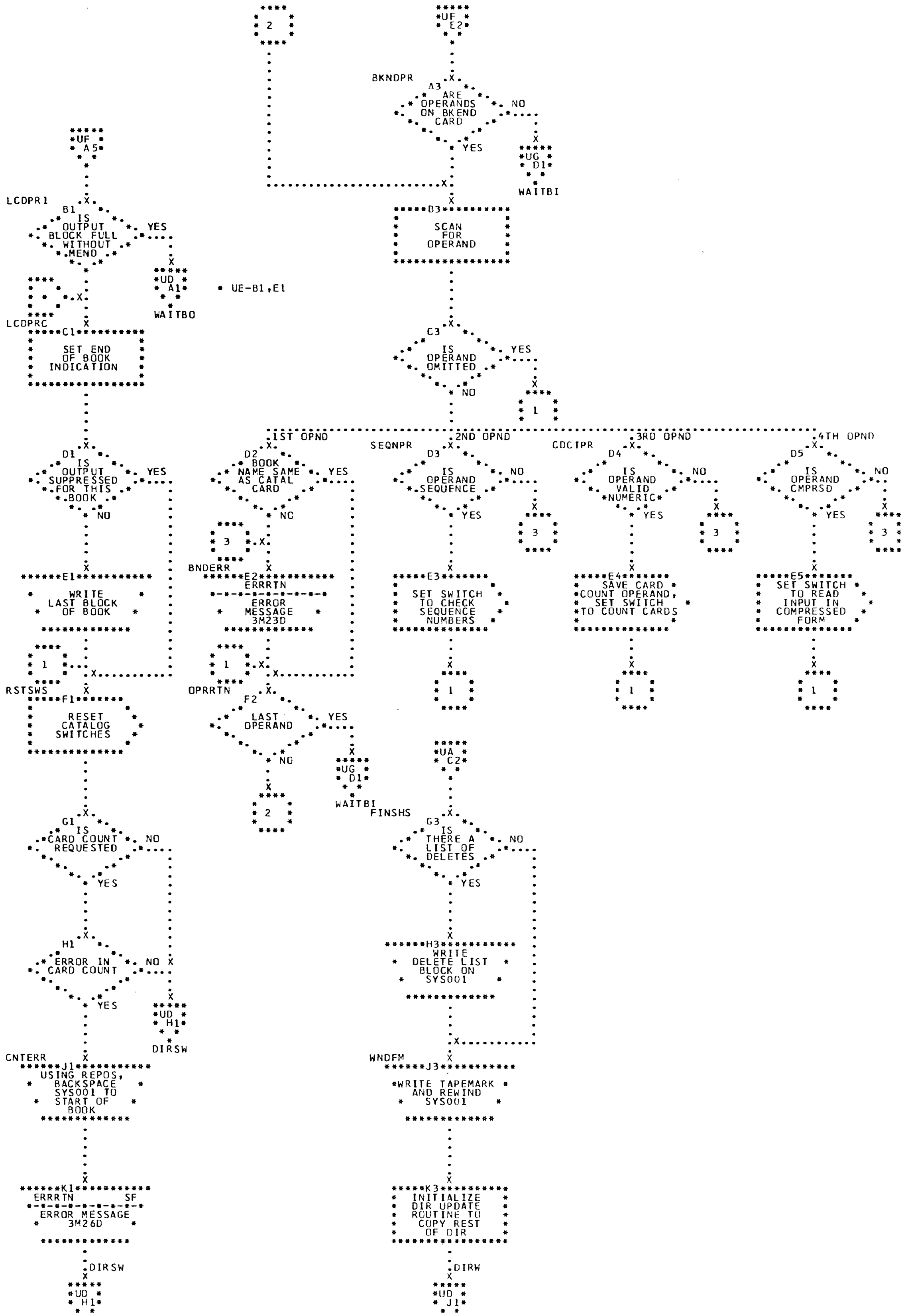


Chart UG. Analyze BKEND

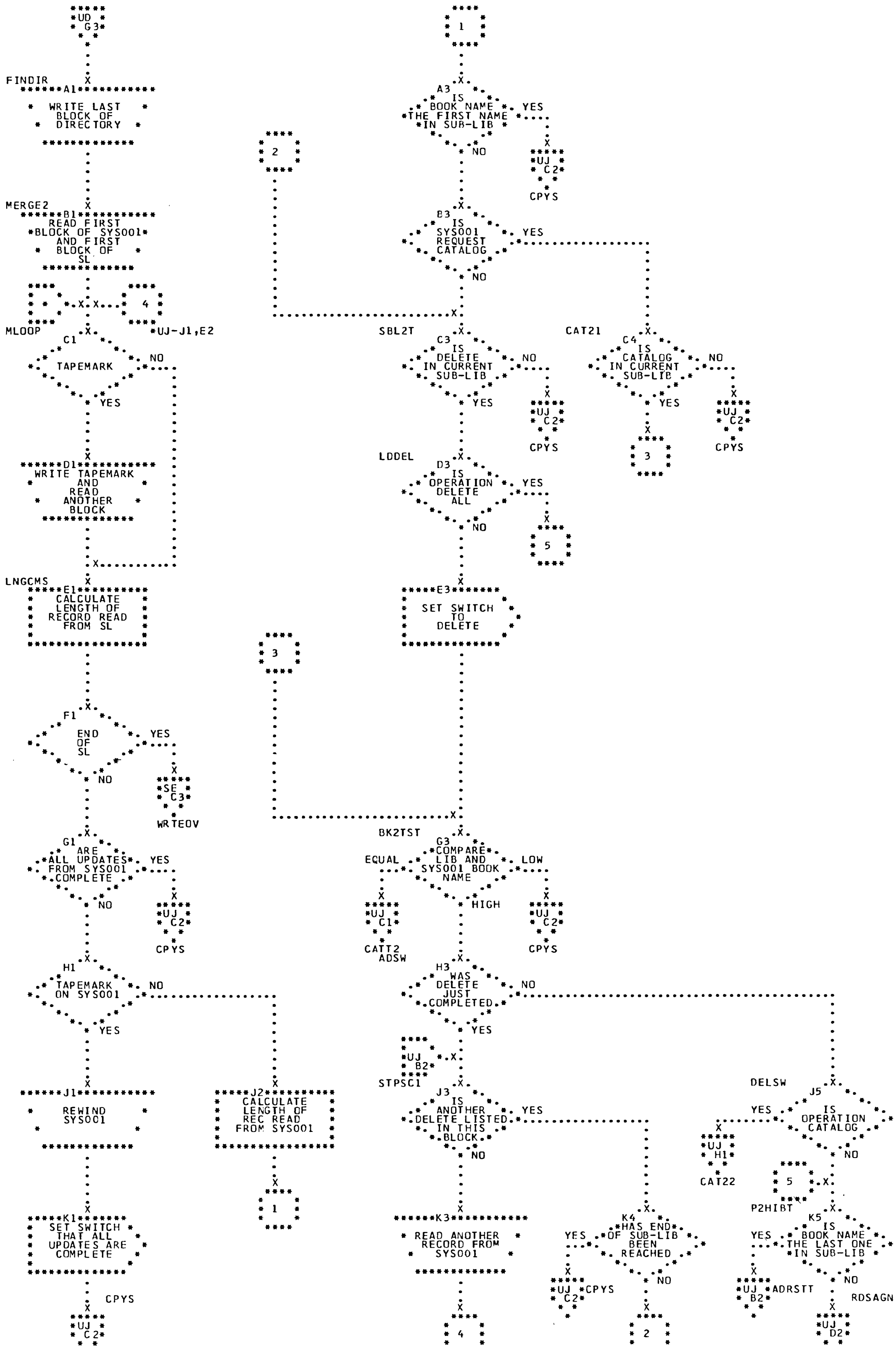


Chart UH. Finish Directory

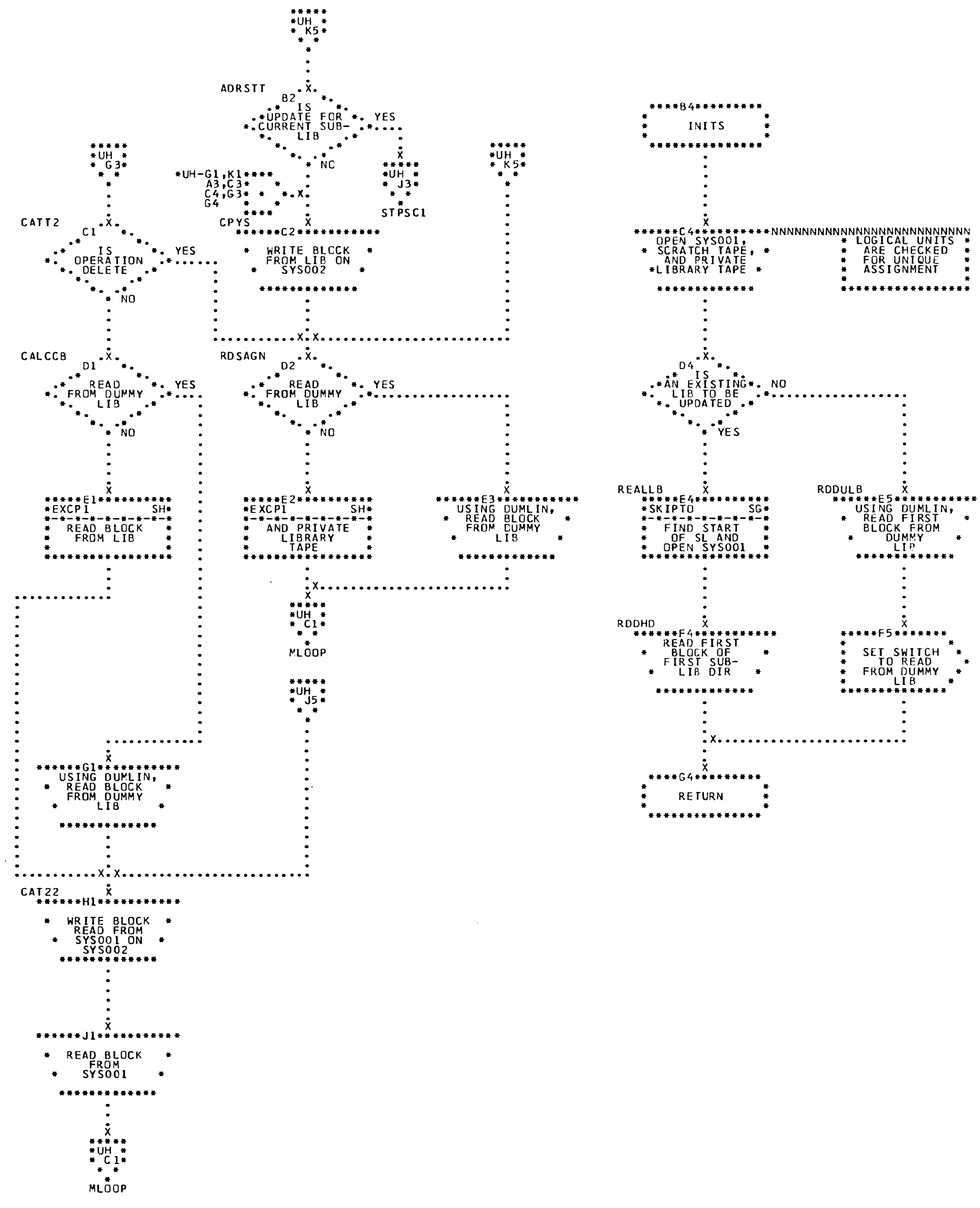


Chart UJ. Build New Library

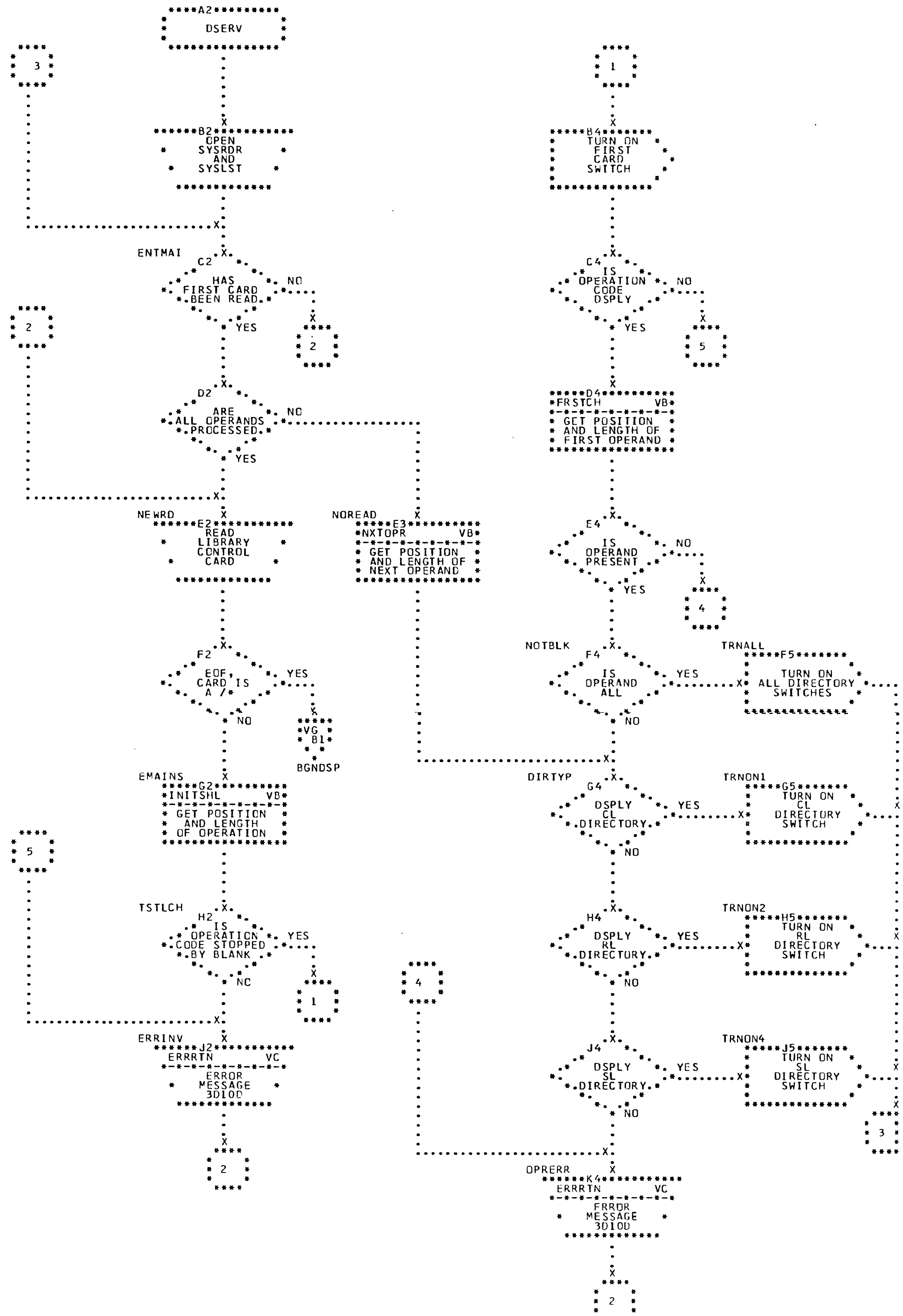


Chart VA. Read Control Card

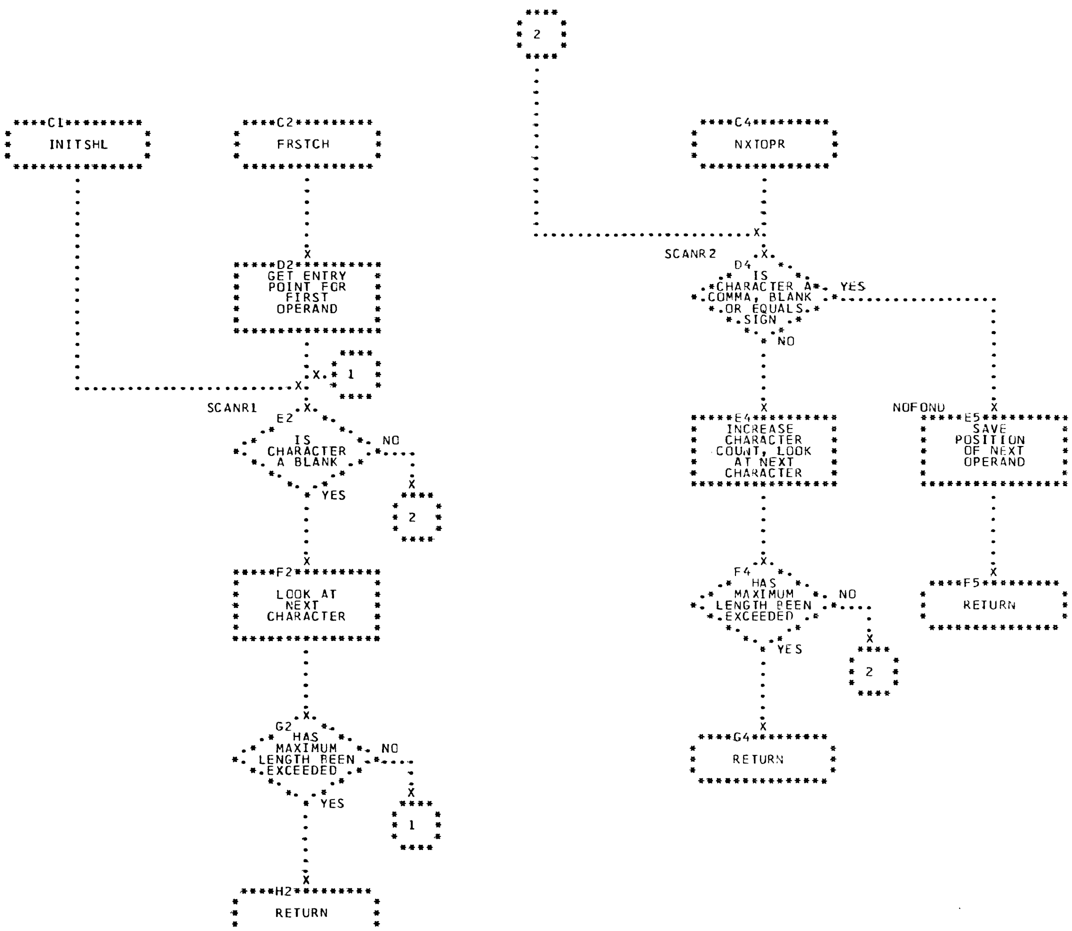


Chart VB. Scan Subroutines

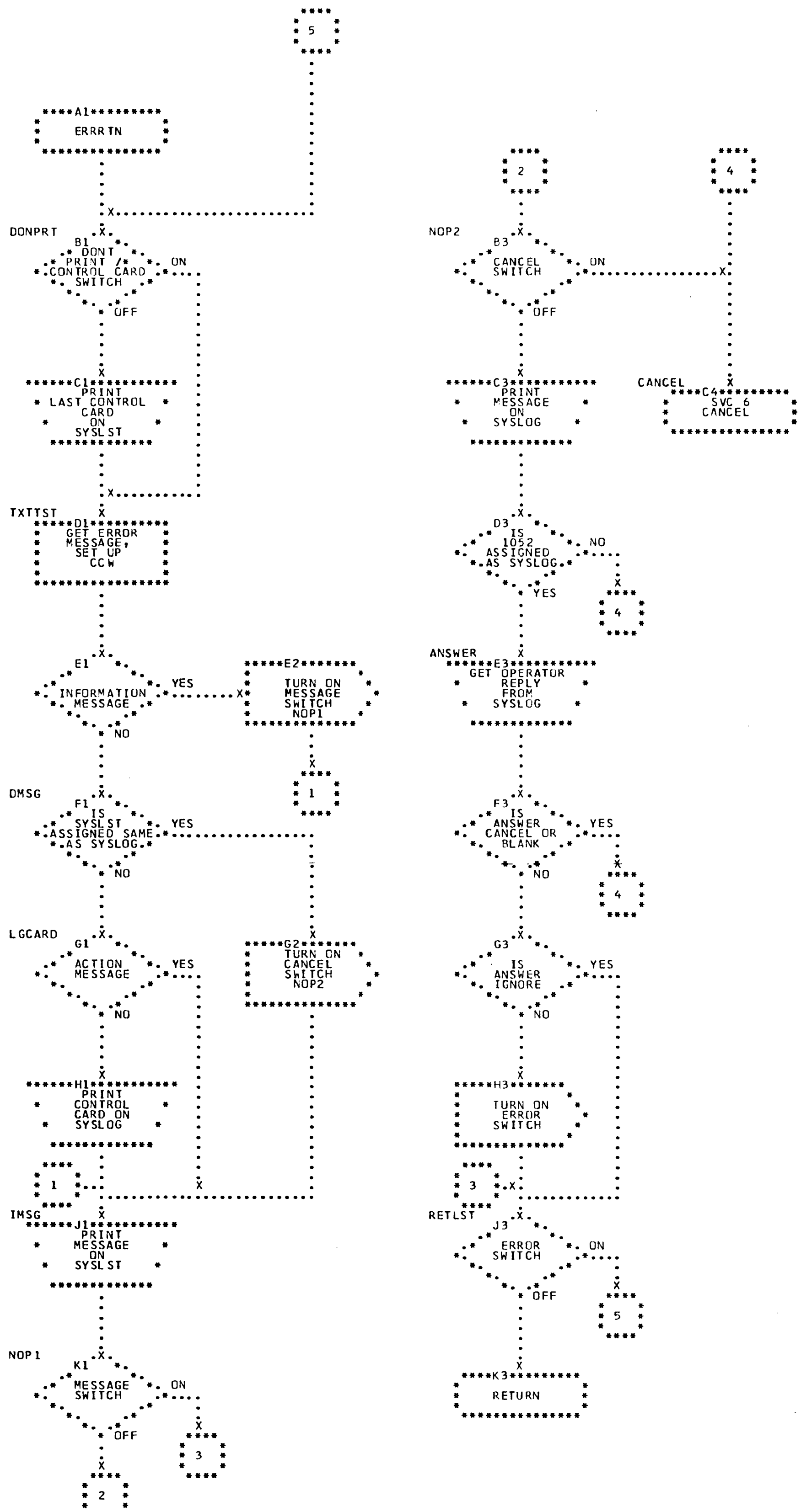


Chart VC. Error Message Subroutine

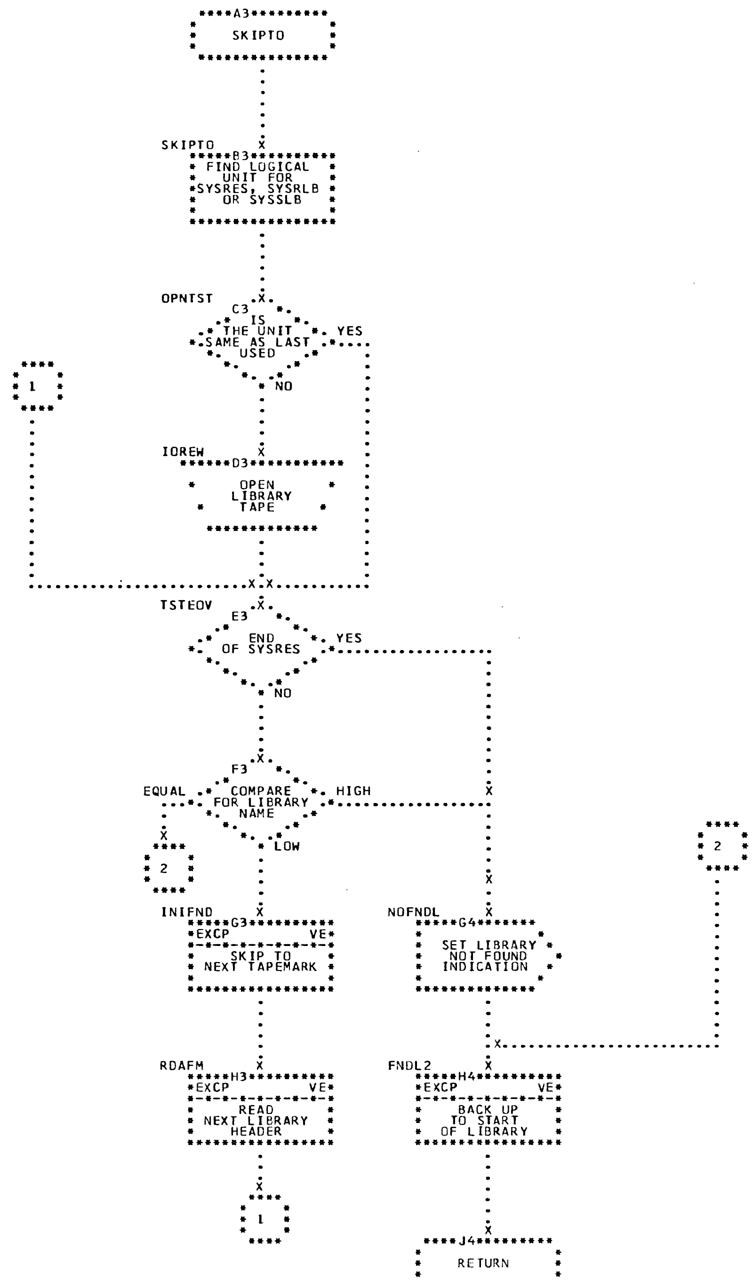


Chart VD. Find Library Subroutine

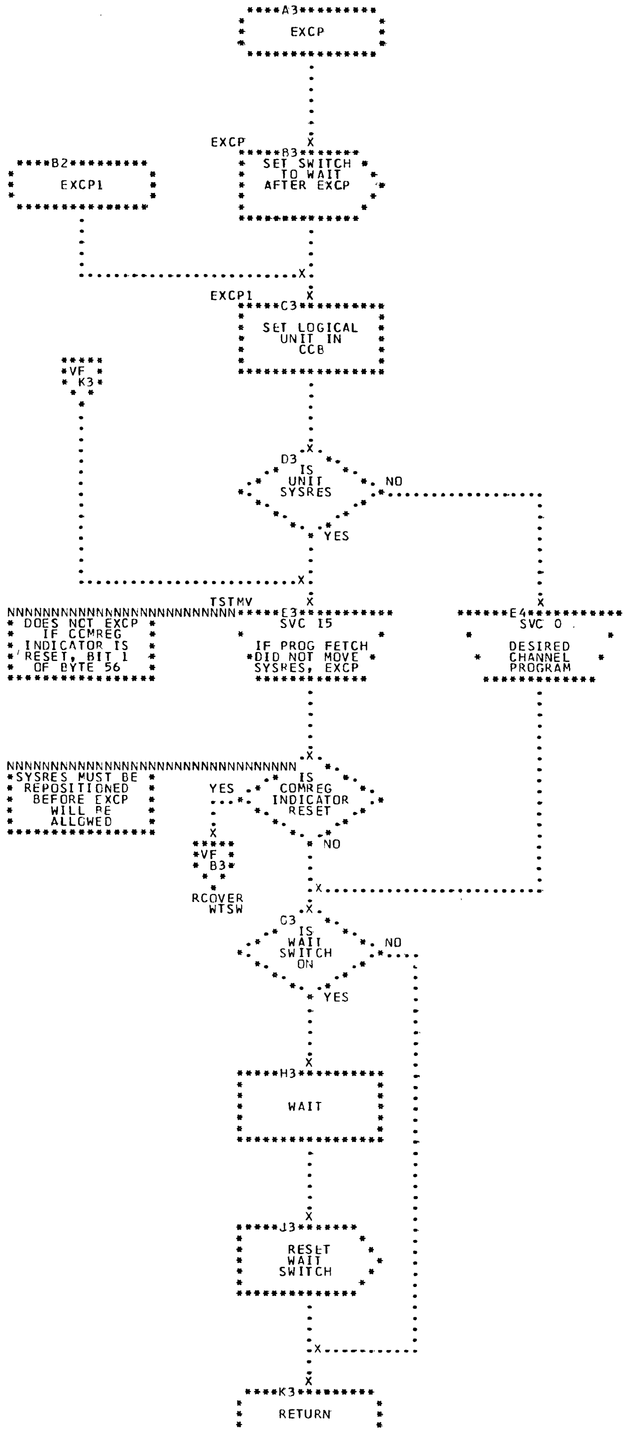


Chart VE. Execute Channel Program Subroutine

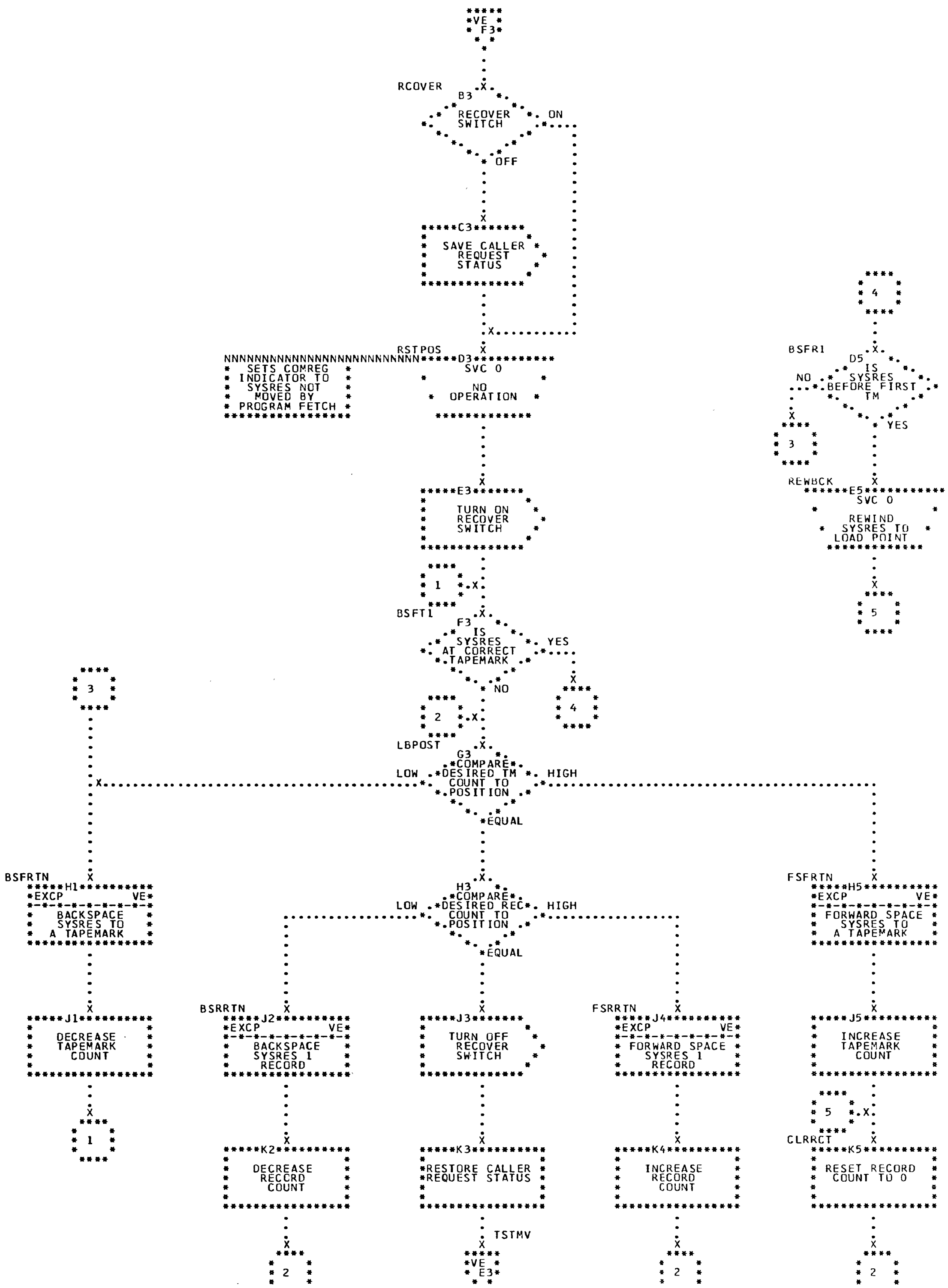


Chart VF. Reposition SYRES for EXCP

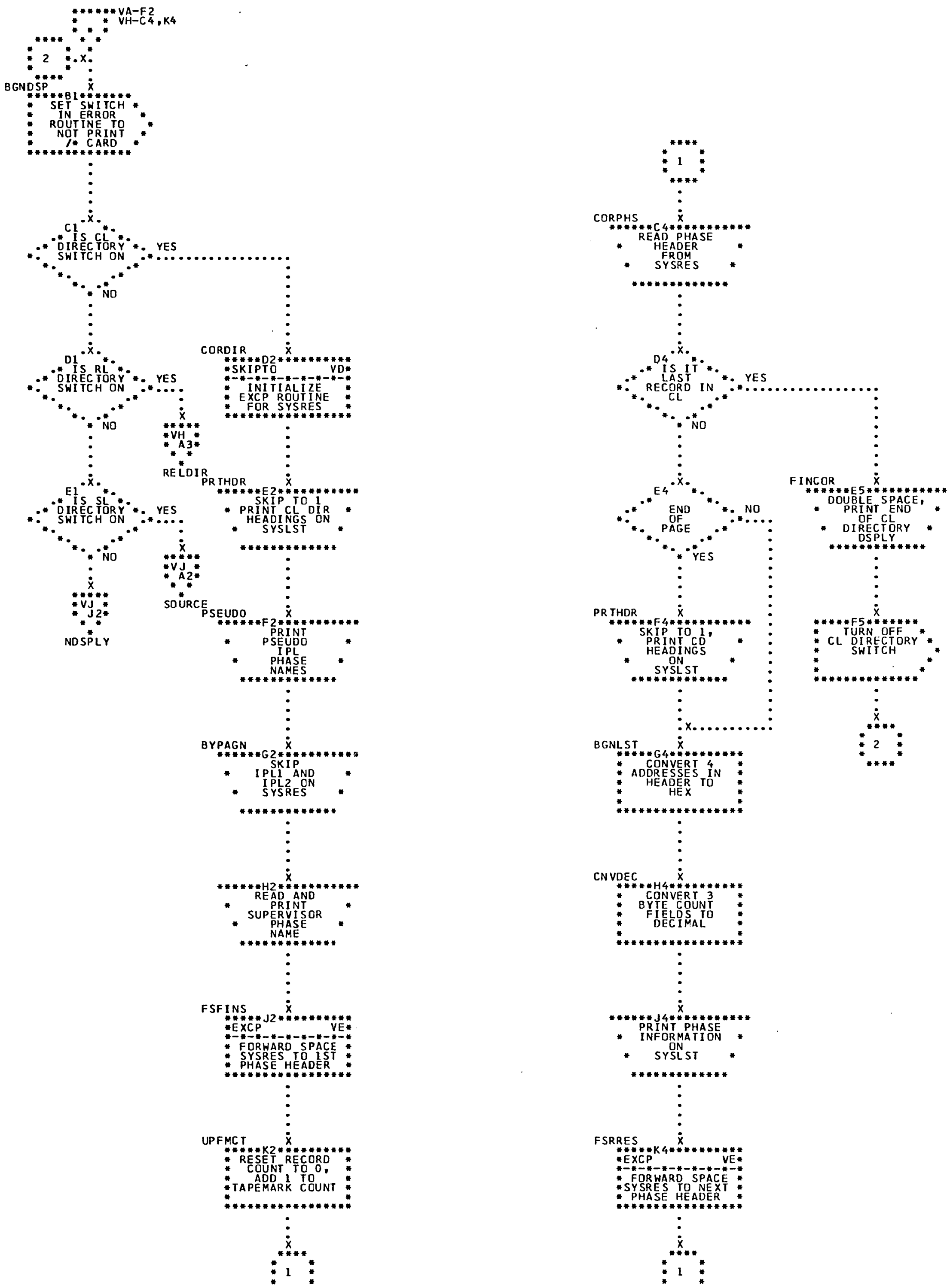


Chart VG. Display Core Image Directory

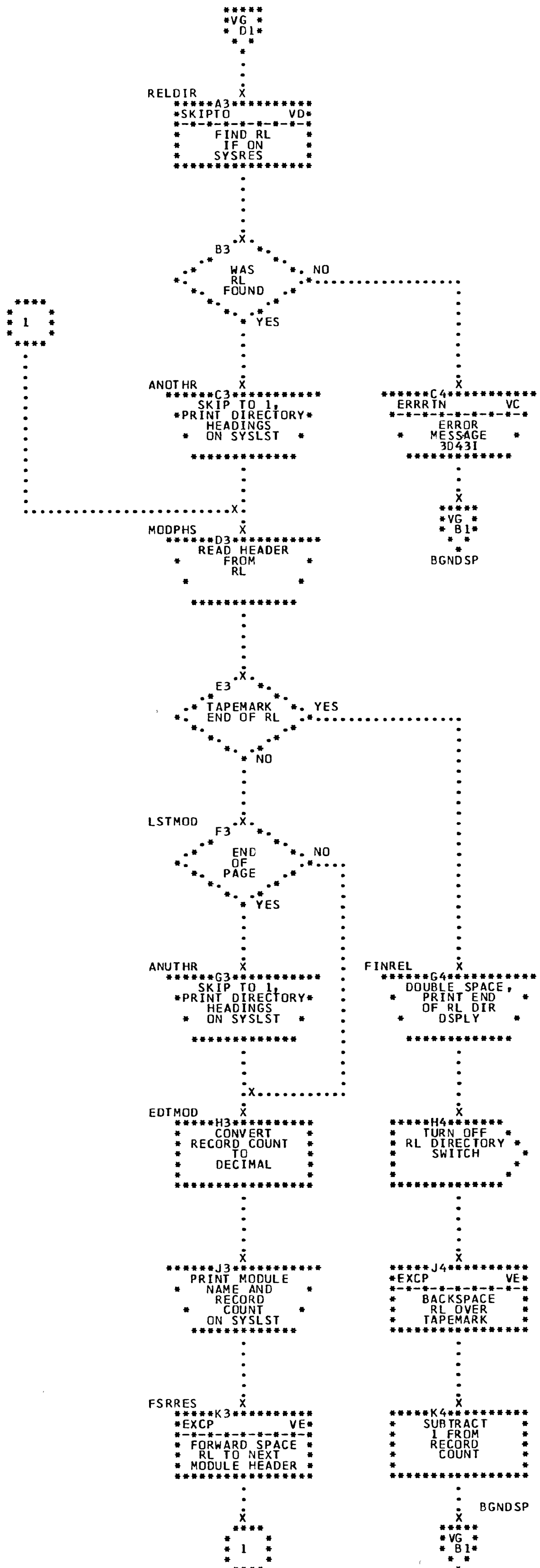


Chart VH. Display Relocatable Directory

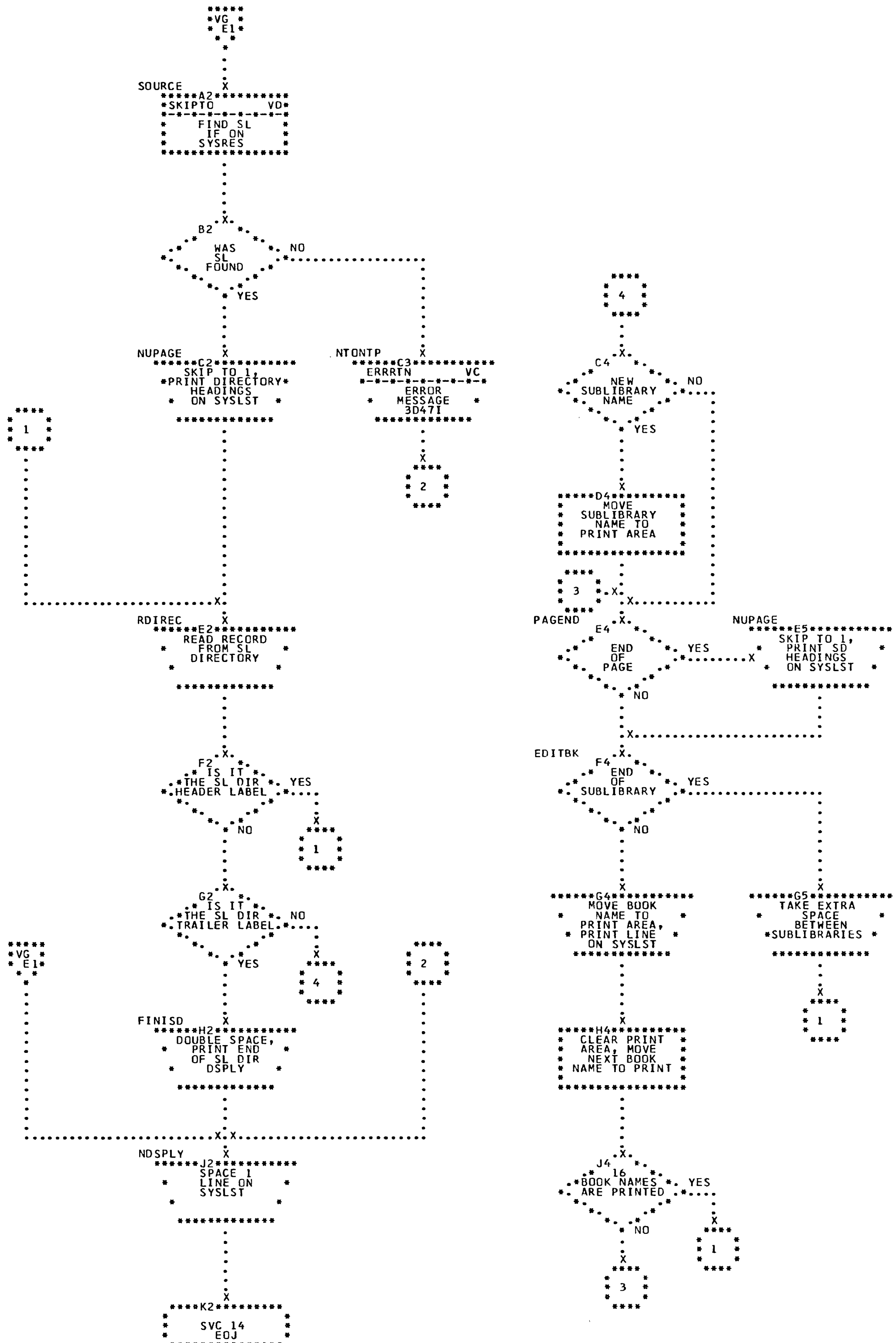


Chart VJ. Display Source Statement Directory

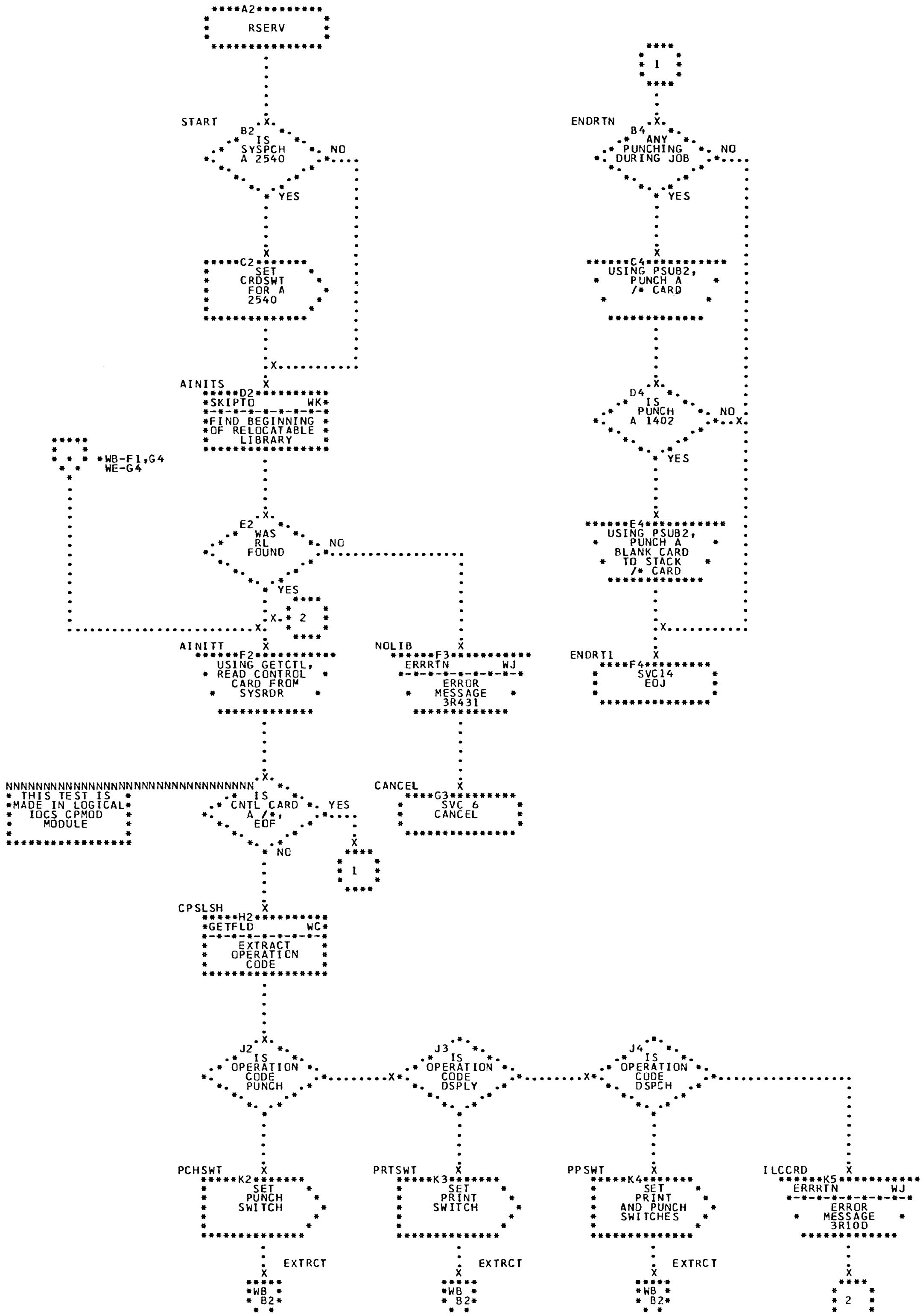


Chart WA. Read Control Card

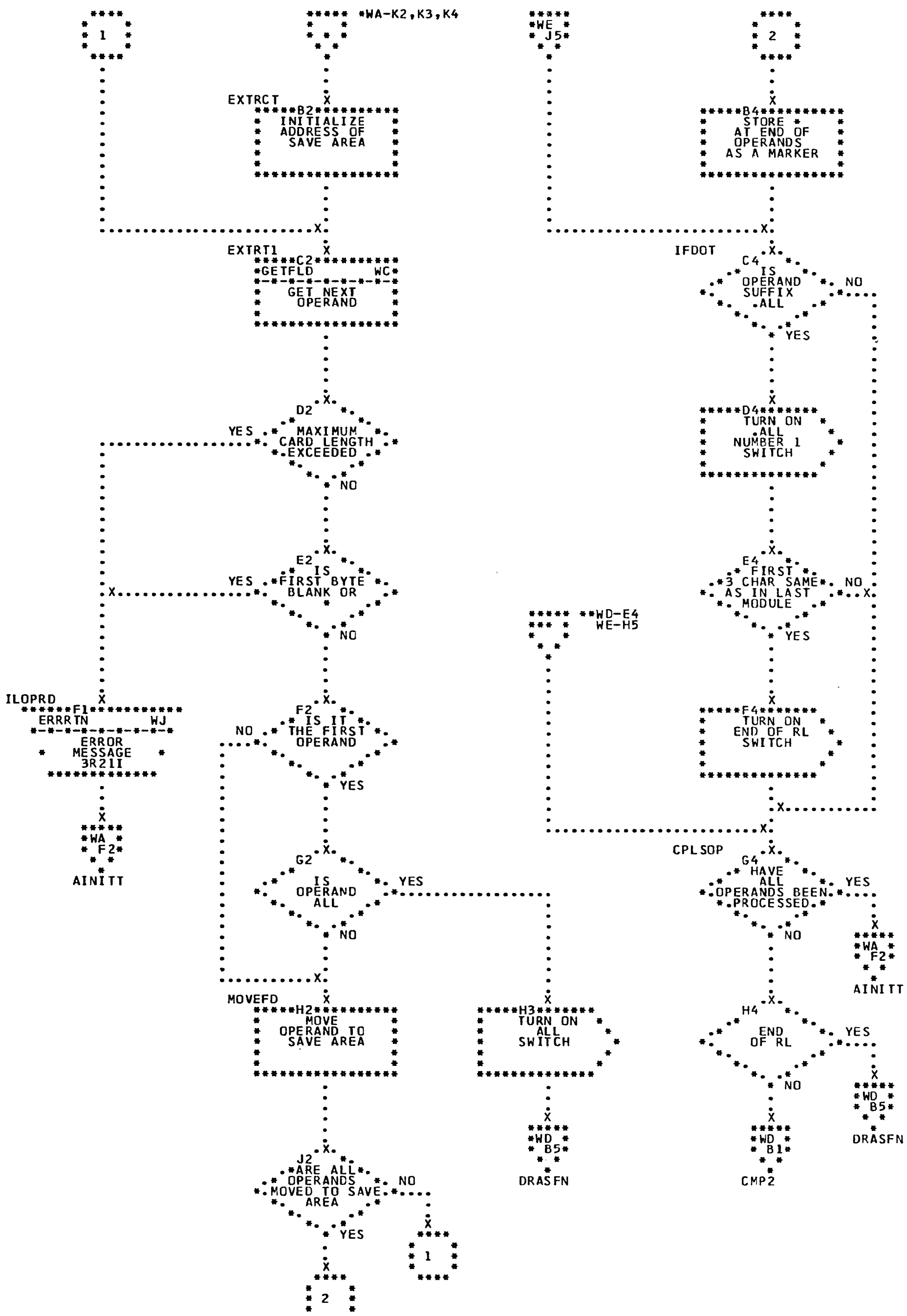


Chart WB. Analyze Operands

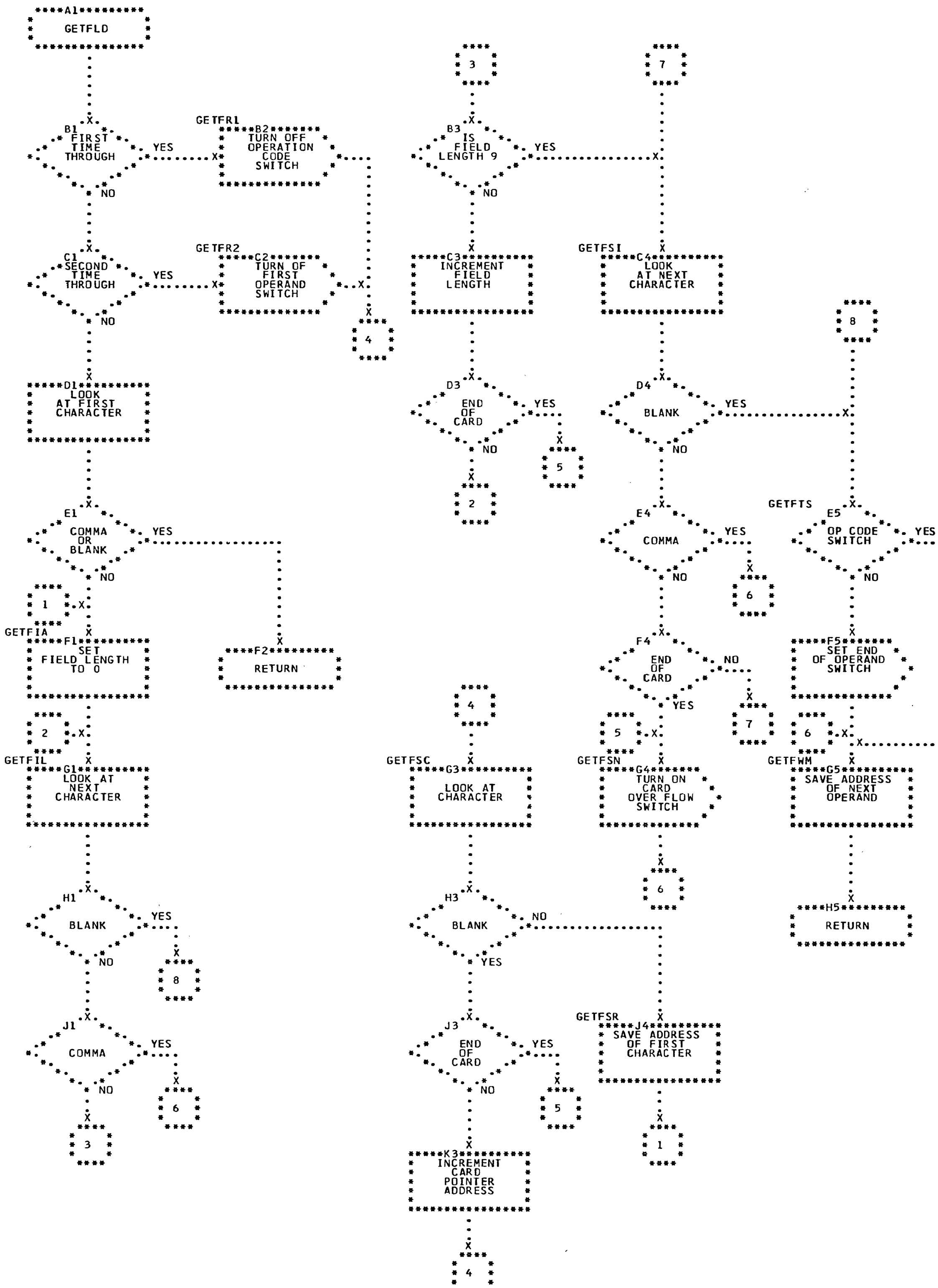


Chart WC. Scan Subroutine

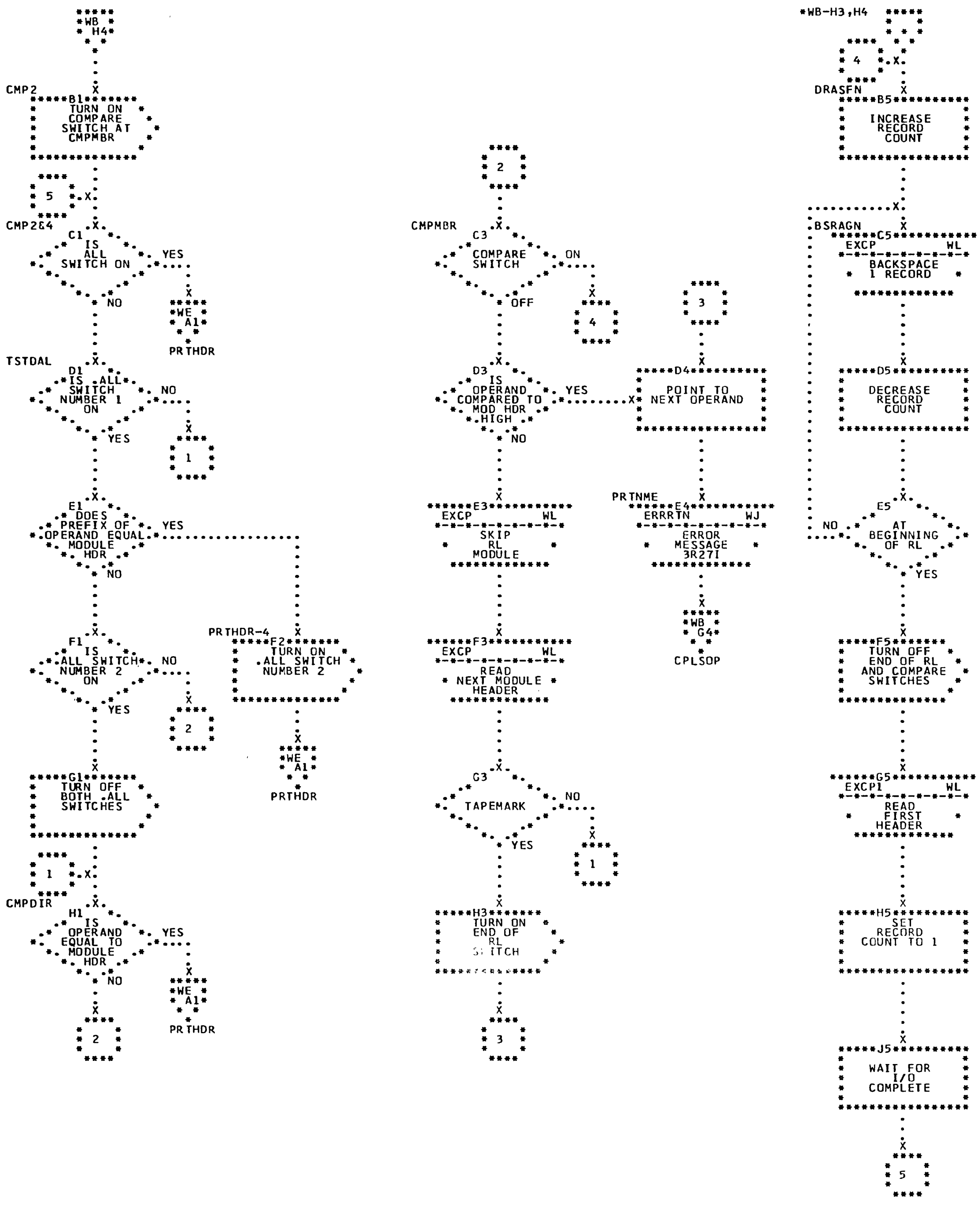


Chart WD. Locate Module Header

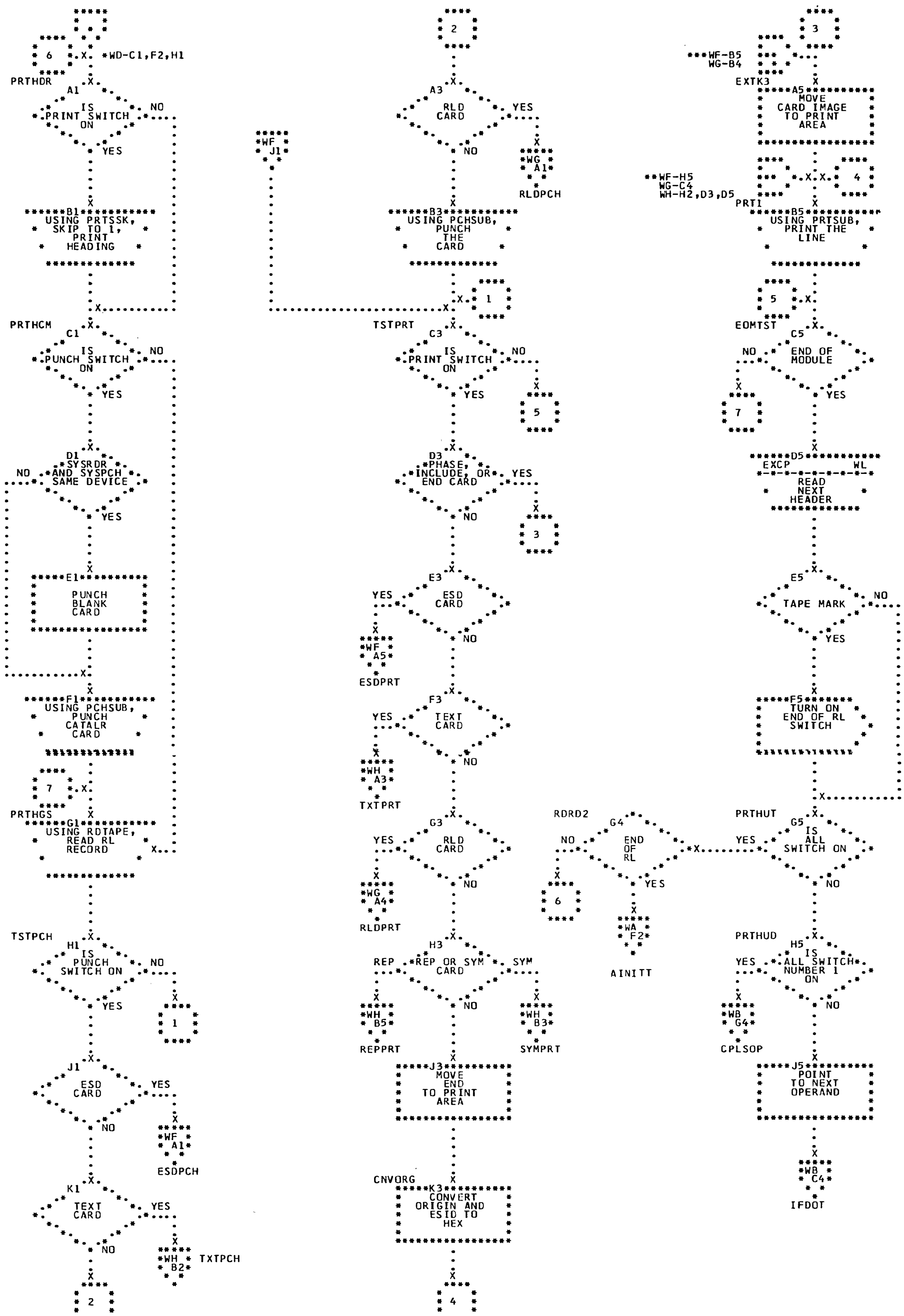


Chart WE. Analyze Relocatable Library Record

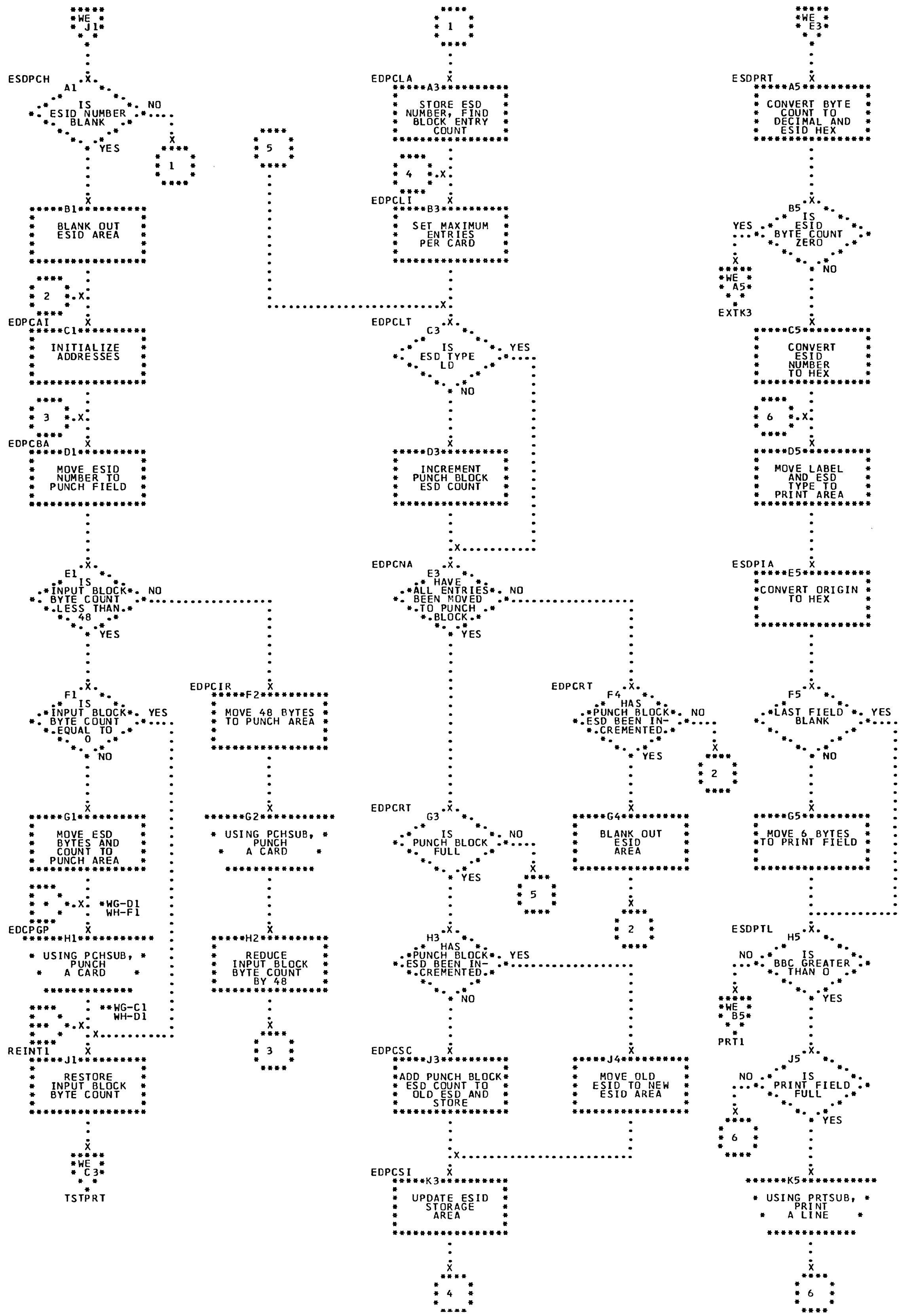


Chart WF. Process ESD

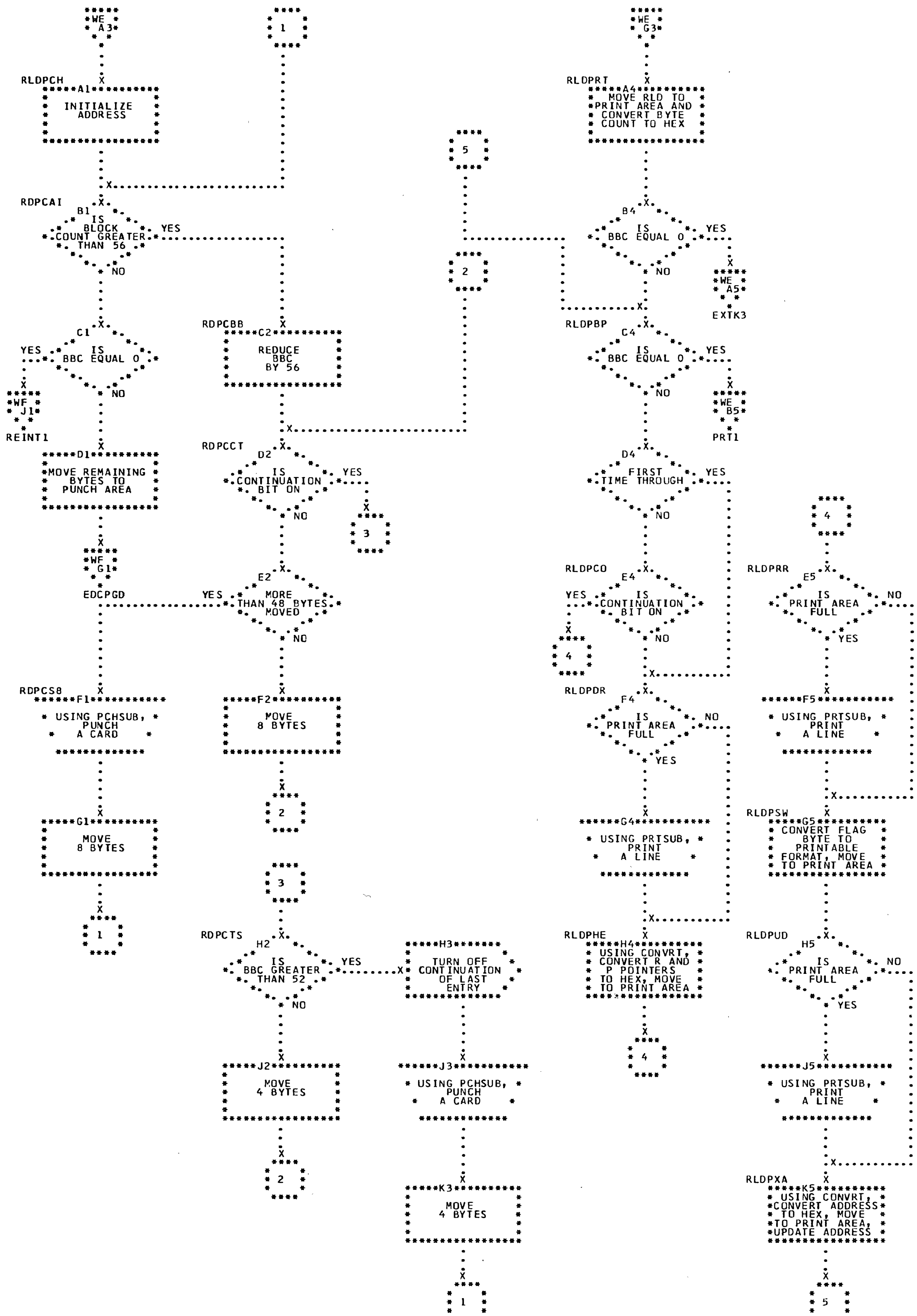


Chart WG. Process RLD

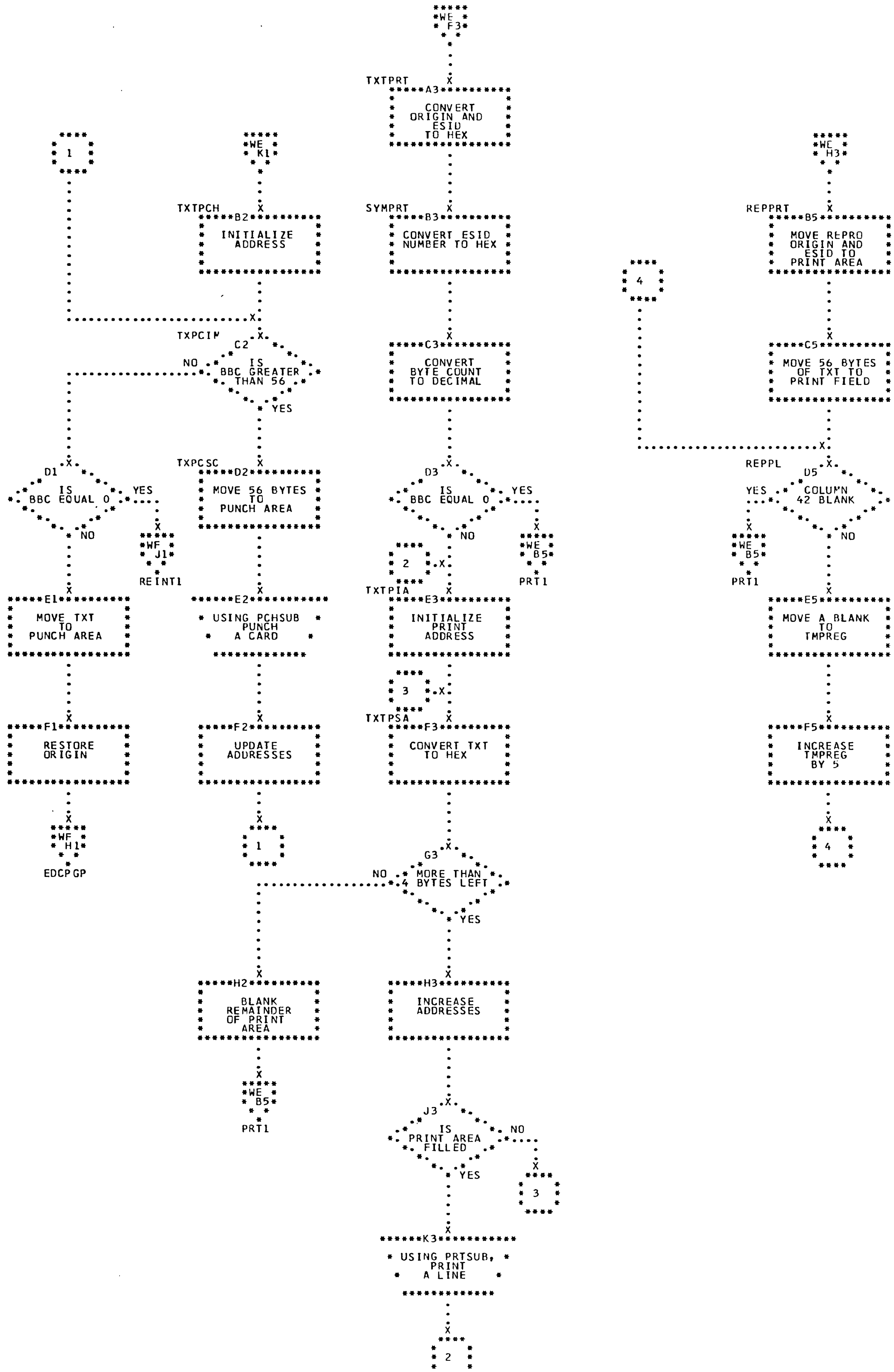


Chart WH. Process TXT and REP

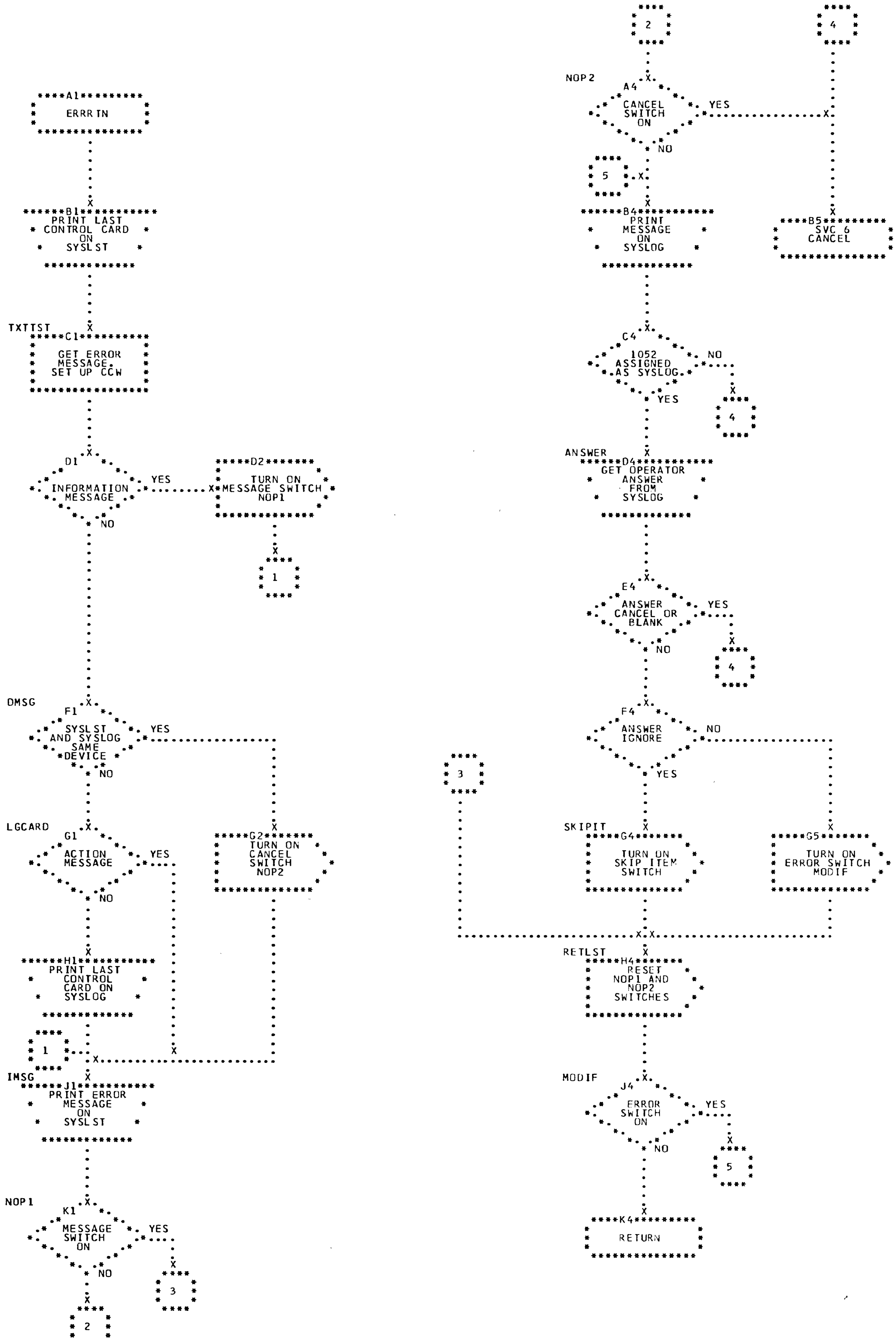


Chart WJ. Error Message Subroutine

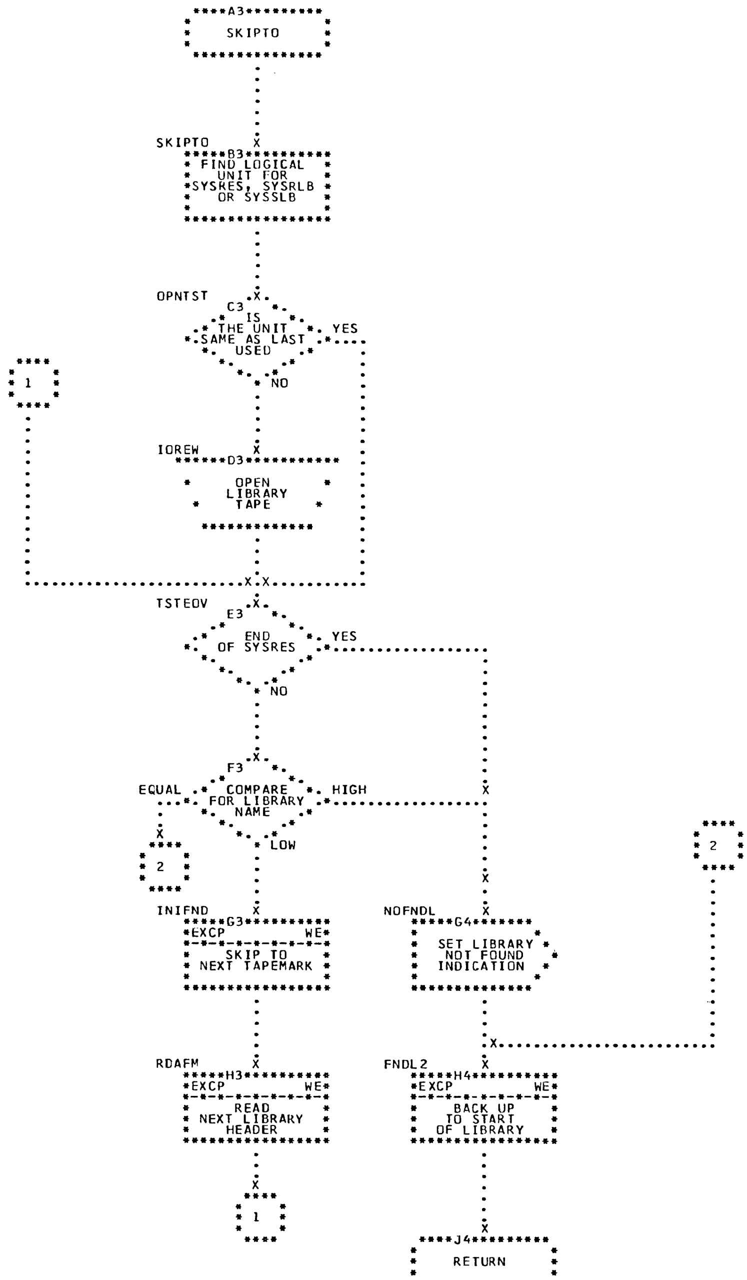


Chart WK. Find Library Subroutine

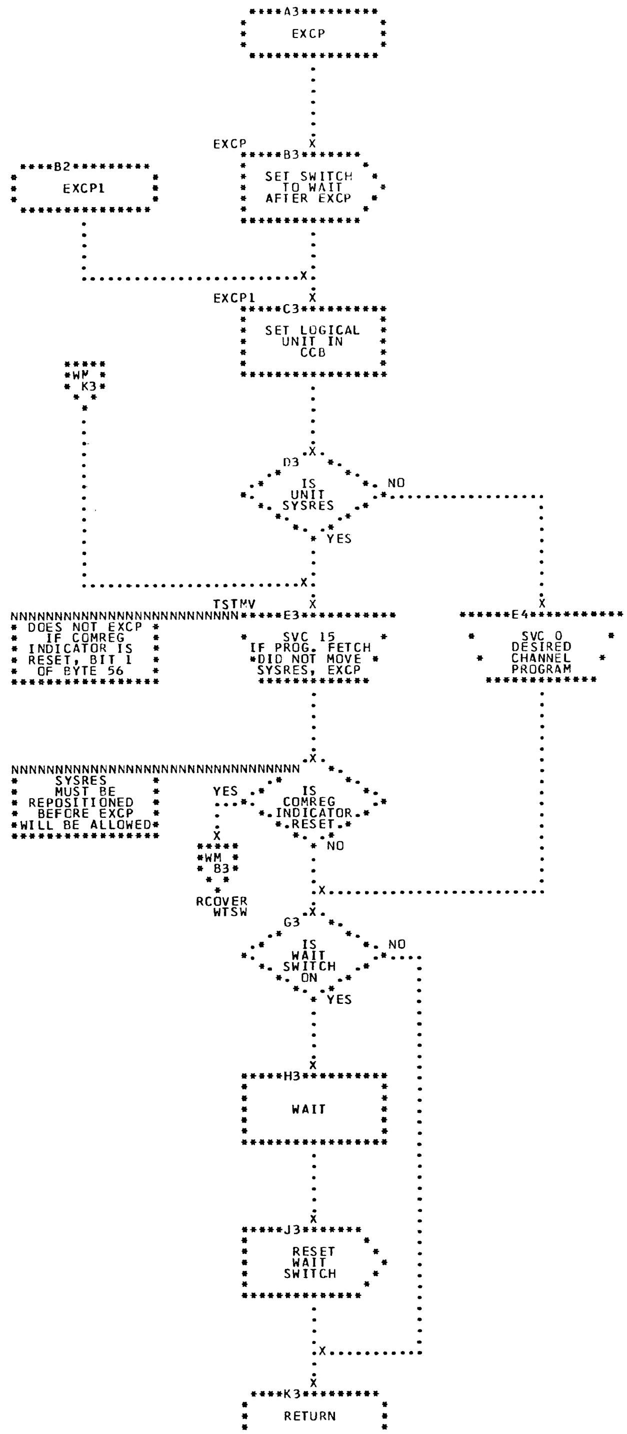


Chart WL. Execute Channel Program Subroutine

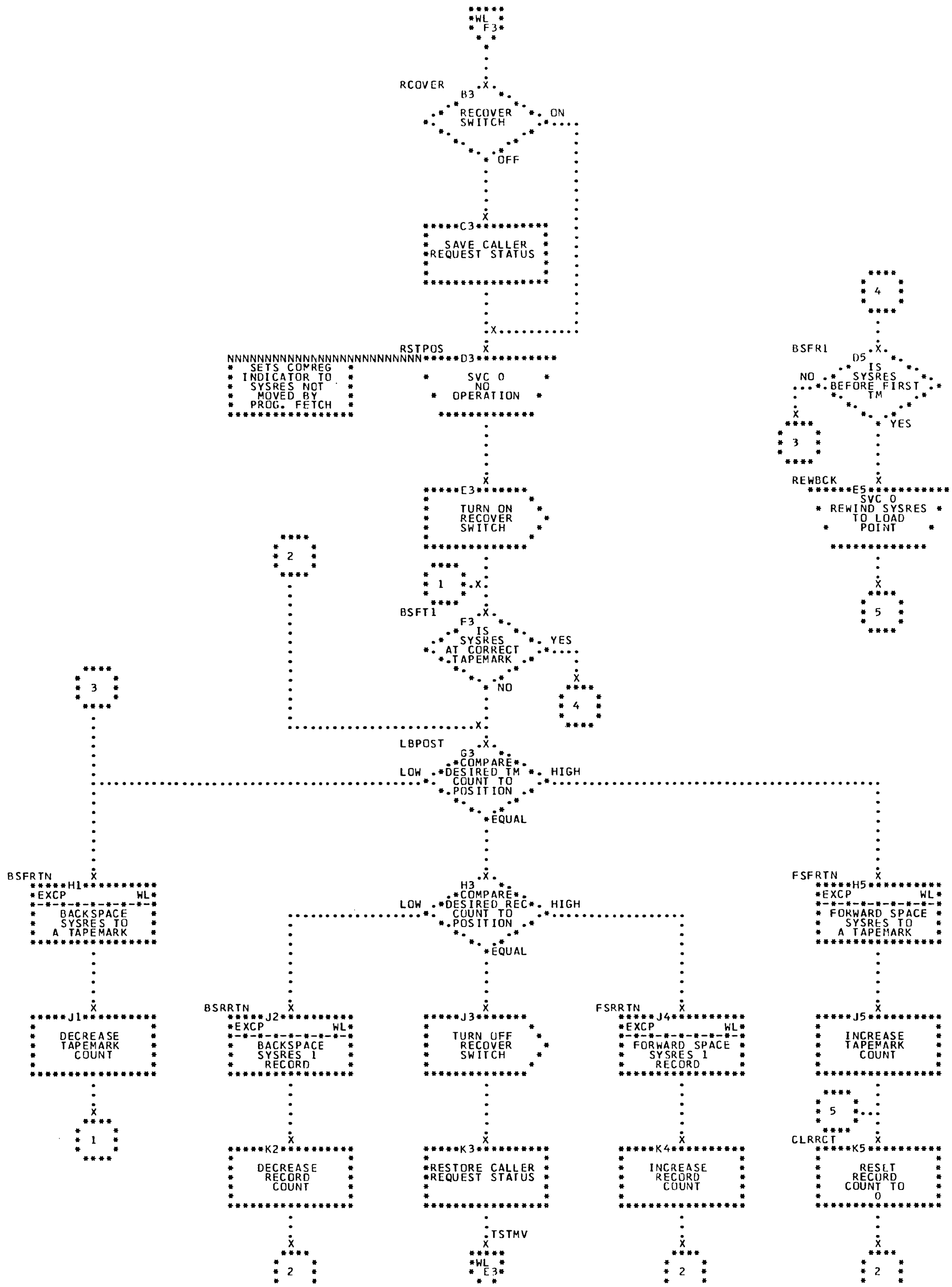


Chart WM. Reposition SYRES for EXCP

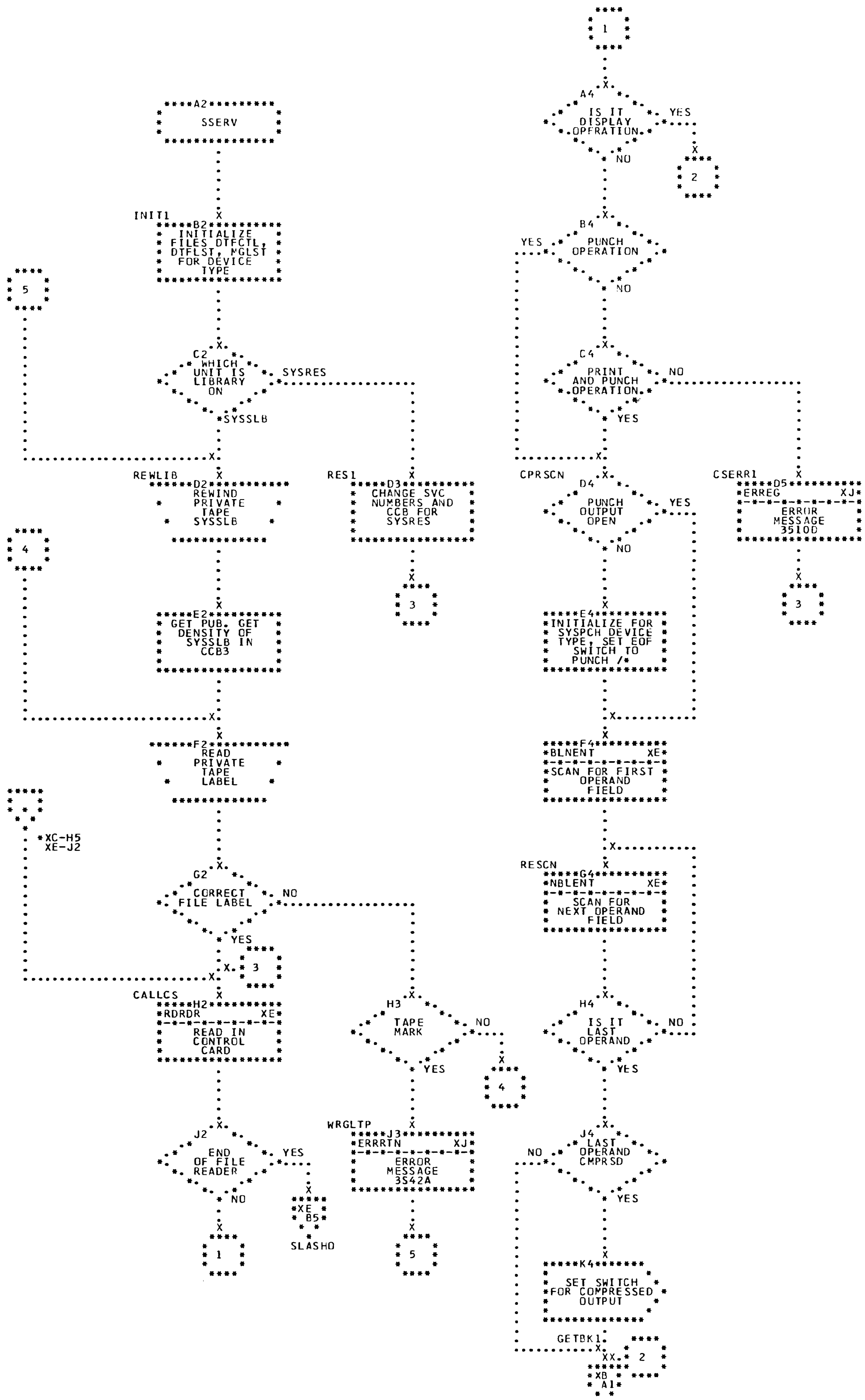


Chart XA. Read Control Card; Determine Operation

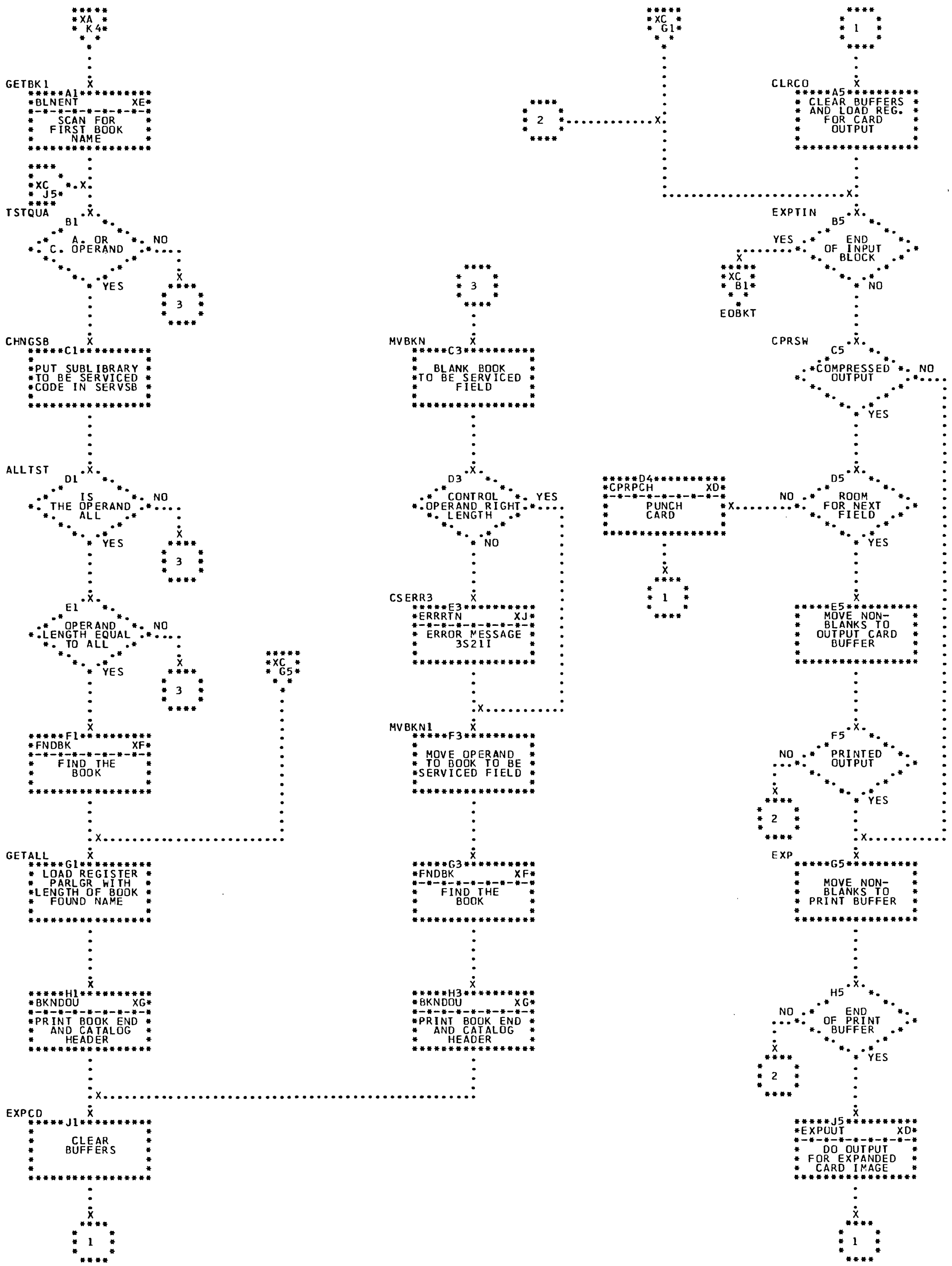


Chart XB. Fill Buffers

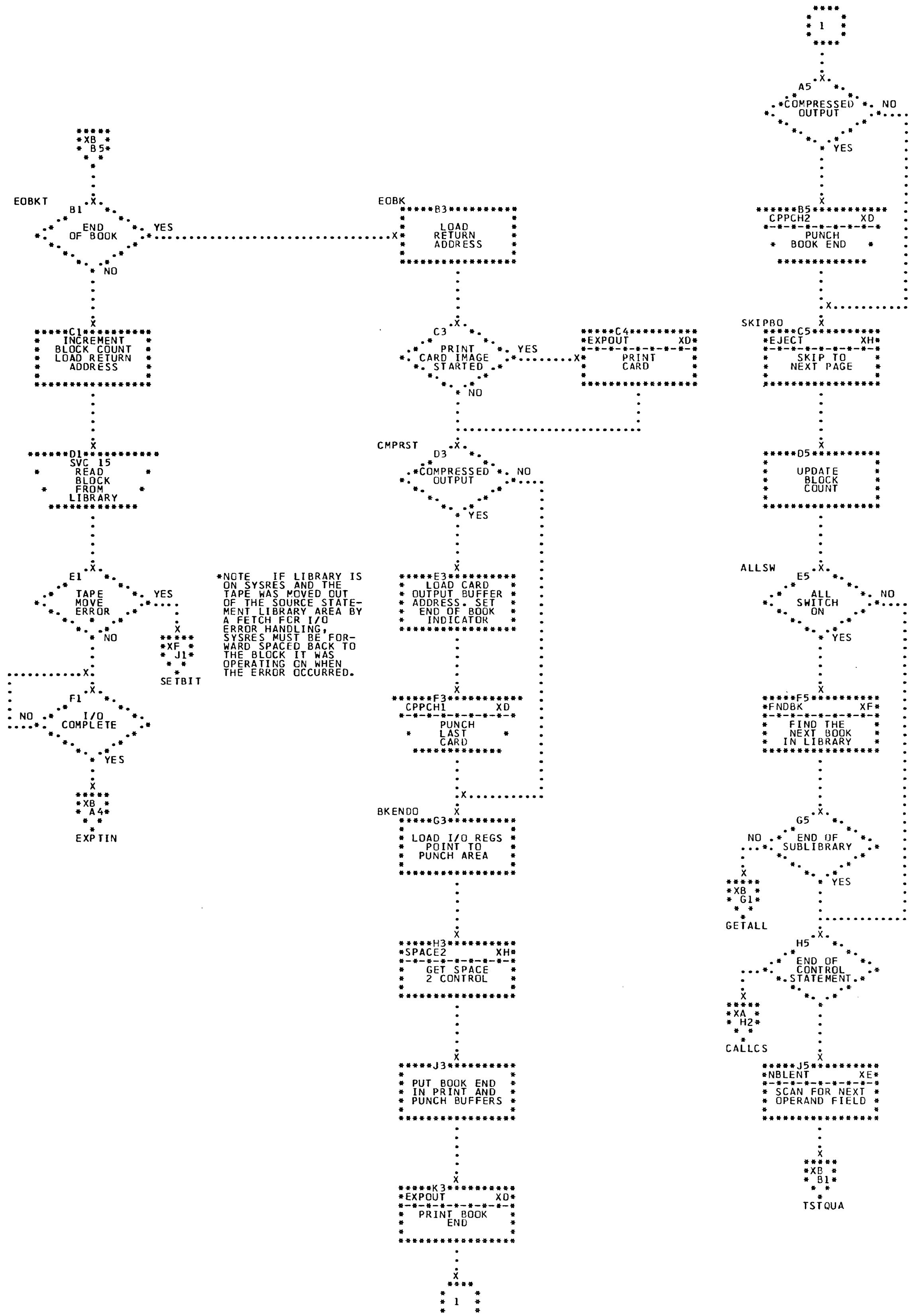


Chart XC. End of Book

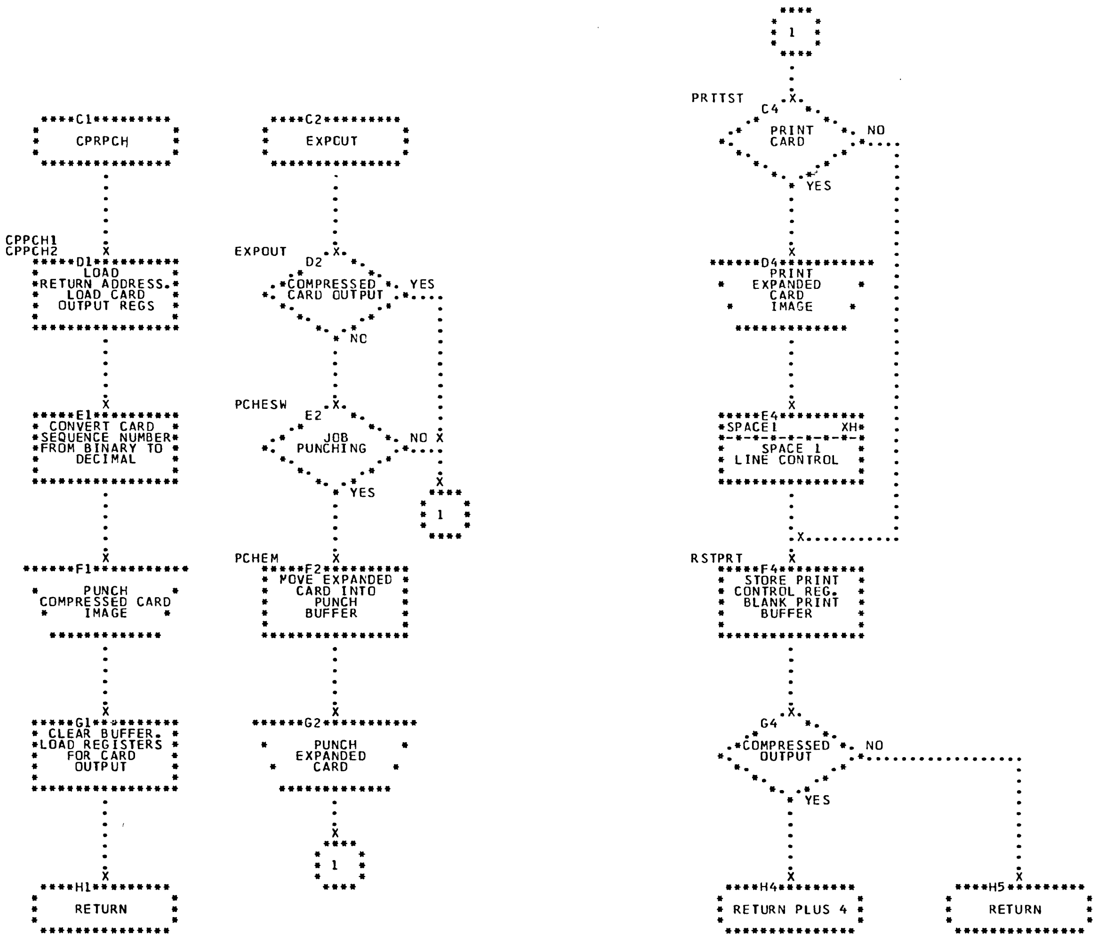


Chart XD. Output Subroutines

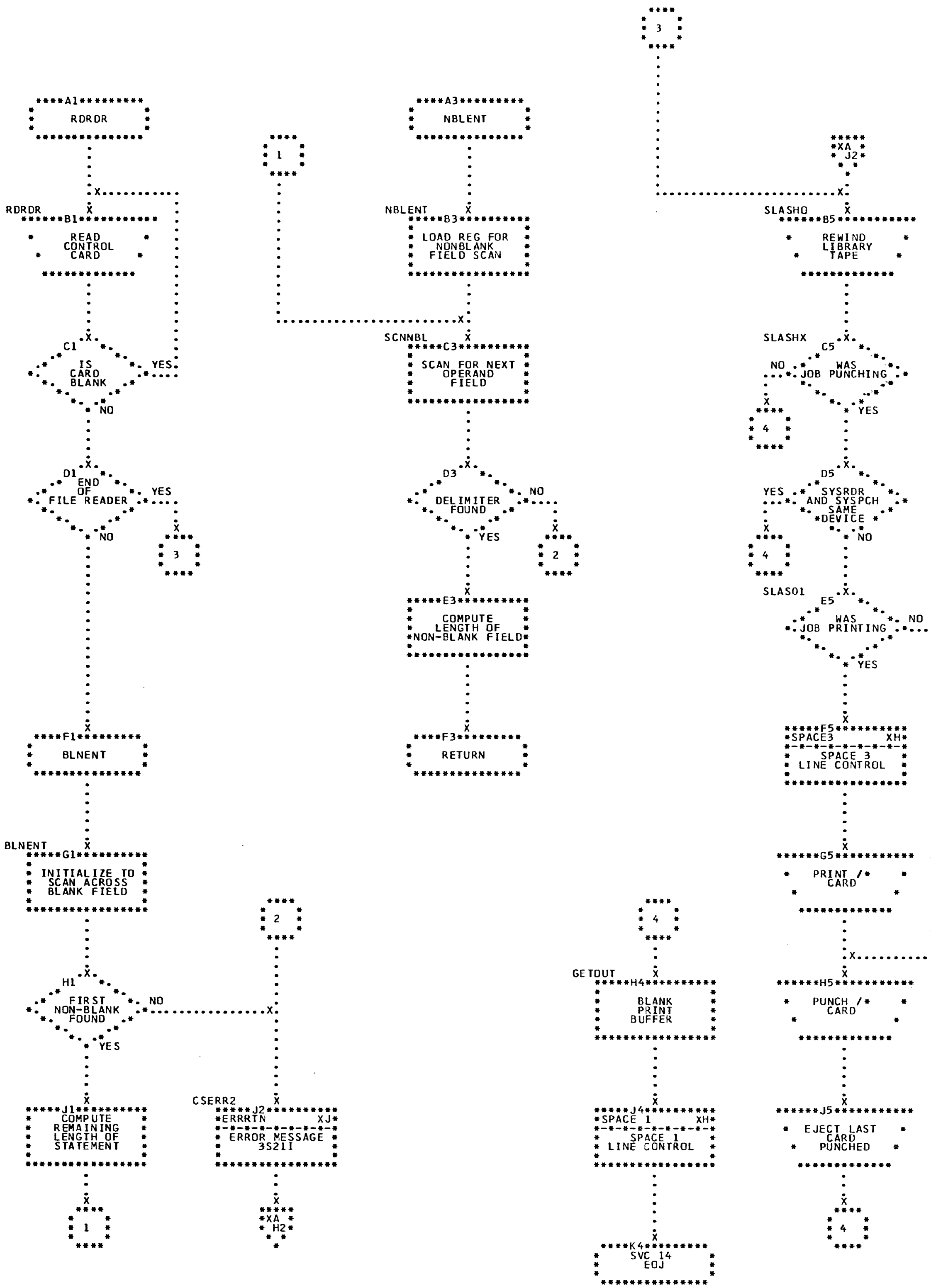


Chart XE. Input Subroutines

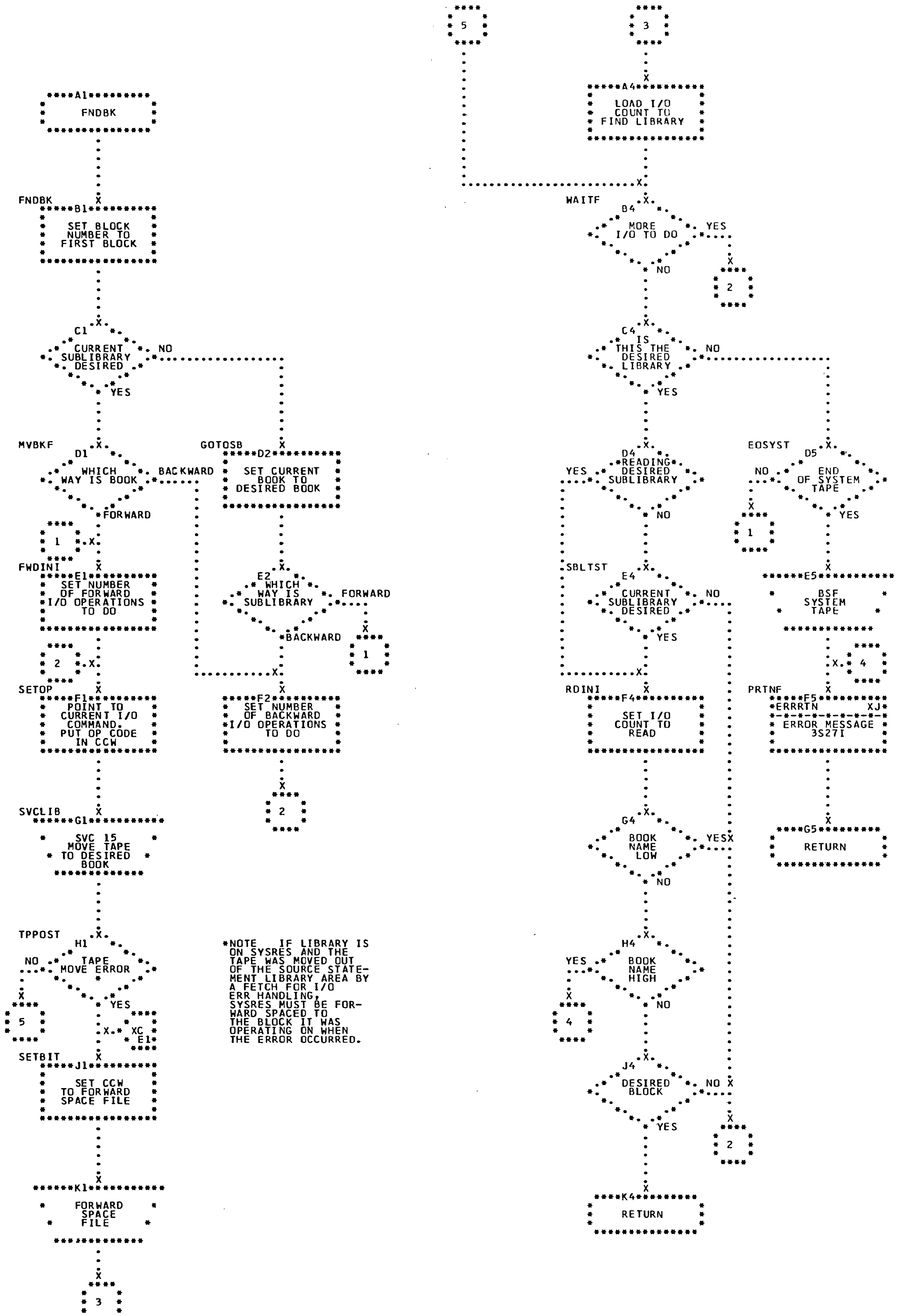


Chart XF. Find Book Subroutine

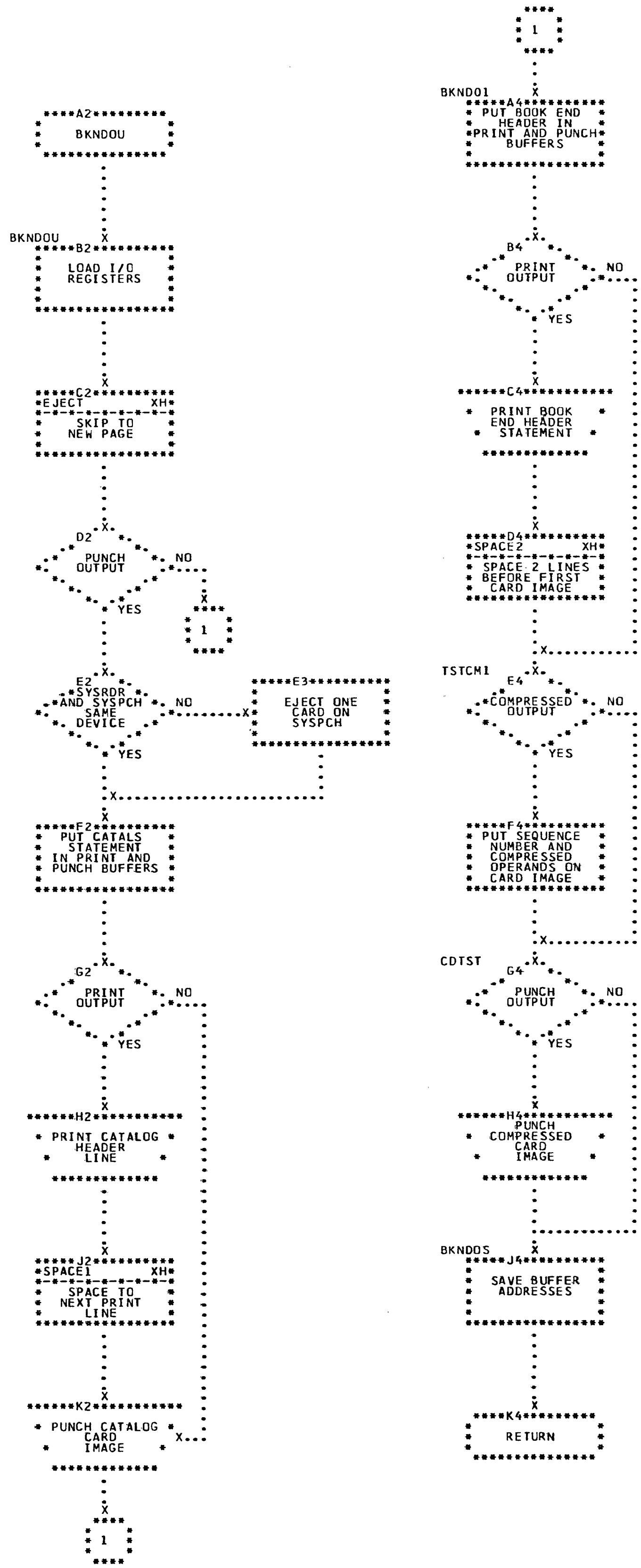


Chart XG. Write Header Subroutine

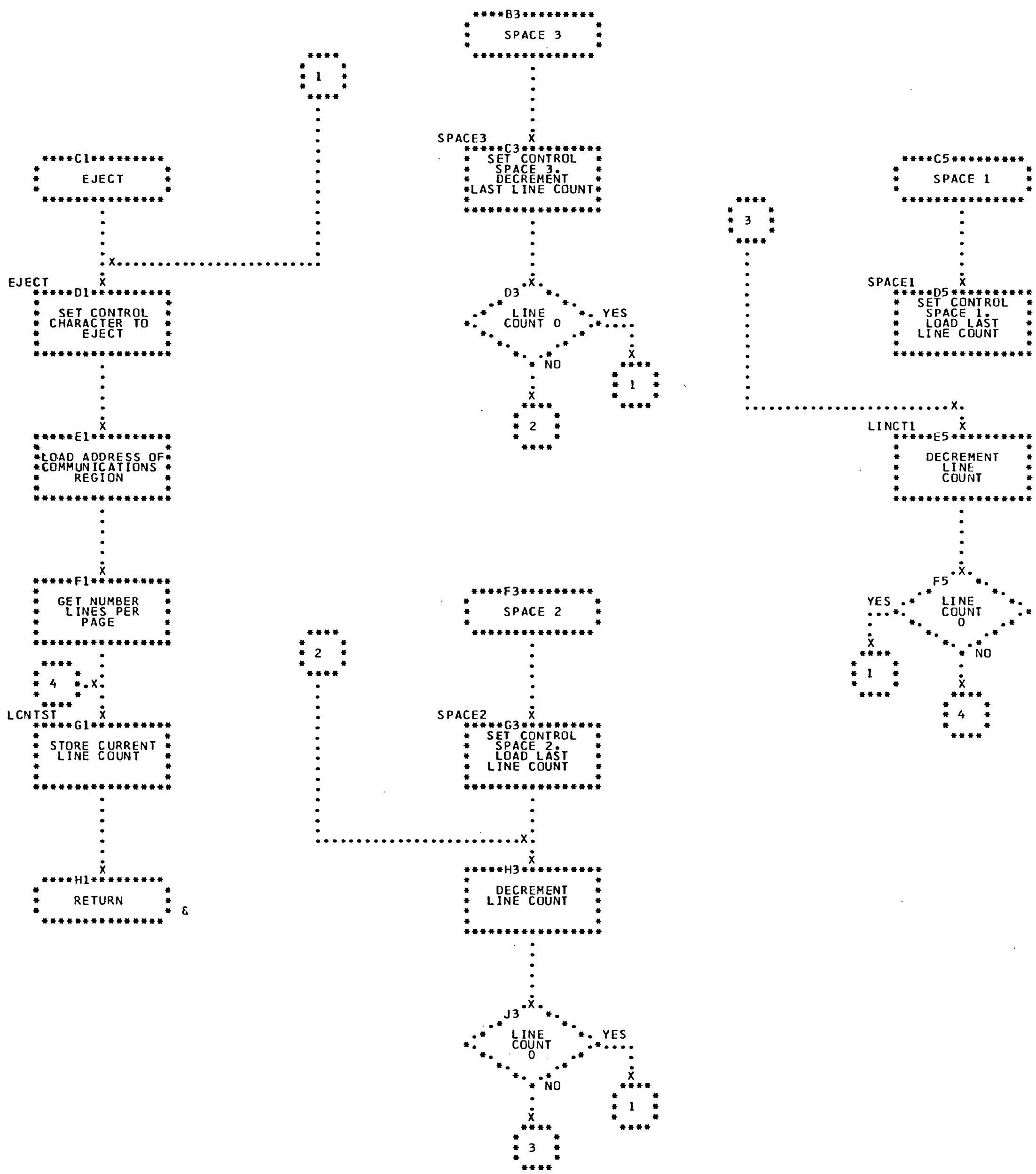


Chart XH. Space Control Subroutines

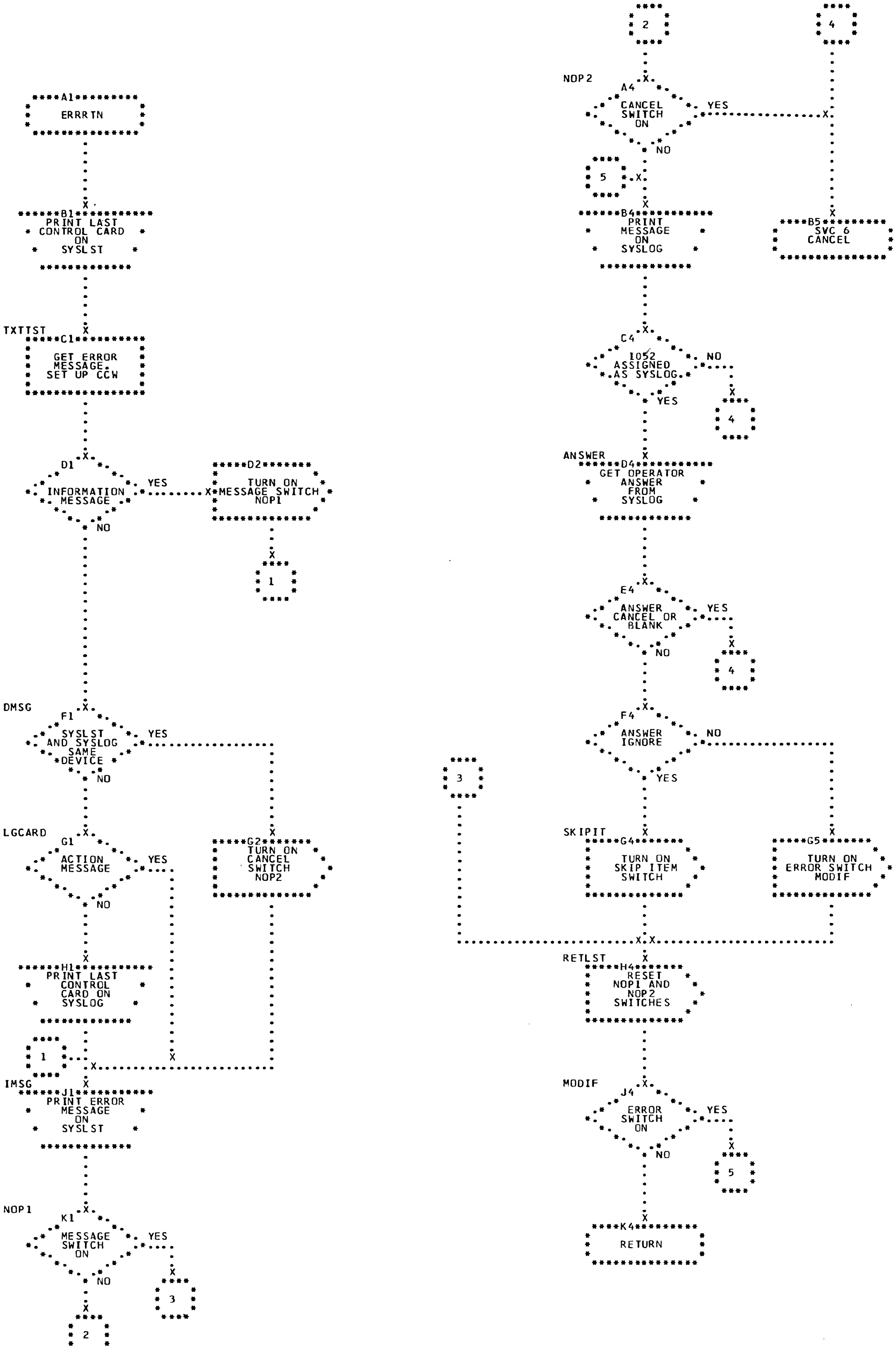


Chart XJ. Error Message Subroutine

Z24-5022-0

IBM S/360 Printed in U.S.A. Z24-5022-0

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N. Y. 10601

READER'S COMMENT FORM

IBM System/360 Basic Operating System
System Control (16K Tape)
Program Logic Manual

Z24-5022-0

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. All comments will be handled on a non-confidential basis. Copies of this and other IBM publications can be obtained through IBM Branch Offices.

	Yes	No
Does this publication meet your needs?	☐	☐
Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

- What is your occupation? ____________________
- How do you use this publication?

As an introduction to the subject?	☐	As an instructor in a class?	☐
For advanced knowledge of the subject?	☐	As a student in a class?	☐
For information about operating procedures?	☐	As a reference manual?	☐

Other ____________________

- Please give specific page and line references with your comments when appropriate. If you wish a reply, be sure to include your name and address.

COMMENTS:

- Thank you for your cooperation. No postage necessary if mailed in the U. S. A.

Staple

Z24-5022-0

Fold

Fold

FIRST CLASS
PERMIT NO. 170
ENDICOTT, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
P. O. Box 6
Endicott, N. Y. 13760

Attention: Programming Publications, Dept. 157

Fold

Fold

Cut Along Line

IBM S/360 Printed in U.S.A. Z24-5022-0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N. Y. 10601

Additional Comments: