SY33-8555-4
File No. S370/S4300-36

Systems

DOS/VSE
Initial Program Load and Job Control Logic

Program Numbers 5745-SC-IPL
5745-SC-JCL

IBM

Summary of Amendments

Edition SY33-8555-4 reflects the changes in program logic for:

- New processor support
  - 3031
  - 4300 processors
- New device support
  - 3310 and 3370 Direct Access Storage Devices (FBA)
  - 8809 Magnetic Tape Unit
  - PRT1 Printers (3289 Model 4 and 3203-5)
  - 3284/3286/3287 Console Printer
  - 5424 MFCU
- Improved supervisor functions
  - More LUBs
  - JIBs no longer required for DASD file protection
  - Pageable supervisor options
  - Use of higher resolution timer
  - Improved method for loading SVA
  - Suppression of channel program scan
- Improved DASD handling
  - DASD volume recognition by means of new VOLUME command
- Extension of label information area and symbolic label access

With DOS/VSE, the following programming support has been removed:

- Support of the IBM 2321 Data Cell
- Support of the IBM 2495 Tape Cartridge Reader

In addition, technical corrections and editorial changes have been made throughout the manual. Changes in contents are indicated by a vertical bar to the left of the change.

Fifth Edition (February 1979)

This is a major revision of, and obsoletes SY33-8555-3 and Technical Newsletter SN33-9237.

This edition applies to the IBM Disk Operating System/Virtual Storage Extended (DOS/VSE) and to all subsequent releases until otherwise indicated. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 Bibliography, GC20-0001, for the editions that are applicable and current.

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

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

In this publication, system and component names as listed below should be read as indicated:

System/component name	To be read as
DOS/VS	DOS/VSE (see Note below)
ECPS:DOS/VS	ECPS:VSE
DOS/VS POWER/VS-E	VSE/POWER
DOS/VS VSAM-E	VSE/VSAM

Note: Unless that name explicitly refers to DOS/VS Release 34 or an earlier DOS/VS release.

This Program Logic Manual (PLM) is a detailed guide to the IBM Disk Operating System IPL and Job Control programs. It supplements the program listings by providing descriptive text and flowcharts. For overall system control logic description, this PLM is used with seven other PLMs:

- DOS/VS Supervisor Logic, SY33-8551.
- DOS/VS Error Recovery and Recording Transients Logic, SY33-8552.
- DOS/VS Logical Transients and Dump Phases Logic, SY33-8553.
- DOS/VS System Serviceability Aids Logic, SY33-8554.
- DOS/VS Linkage Editor Logic, SY33-8556.
- DOS/VS Librarian Logic, SY33-8557.
- DOS/VS Fixed Block Architecture Logical IOCS, SY24-5181.

Prerequisite publications that will aid in the use of this manual are:

- IBM System/370 Principles of Operation, GA22-7000.
- IBM 4300 Processors Principles of Operation, GA22-7070
- DOS/VS System Control Statements, GC33-5376.
- OS/VS, DOS/VS, and VM/370 Assembler Language, GC33-4010.
- Guide to the DOS/VS Assembler GC33-4024.

Publications related in subject matter to the system control PLMs are:

- Introduction to DOS/VS, GC33-5370.
- DOS/VS System Management Guide, GC33-5371.
- DOS/VS Data Management Concepts, GC24-5138.
- DOS/VS Macro User's Guide, GC24-5139.
- DOS/VS Macro Reference, GC24-5140
- DOS/VS Tape Labels, GC33-5374.

- DOS/VS DASD Labels, GC33-5375.
- DOS/VS System Generation, GC33-5377.
- DOS/VS Operating Procedures, GC33-5378.
- DOS/VS Messages, GC33-5379.

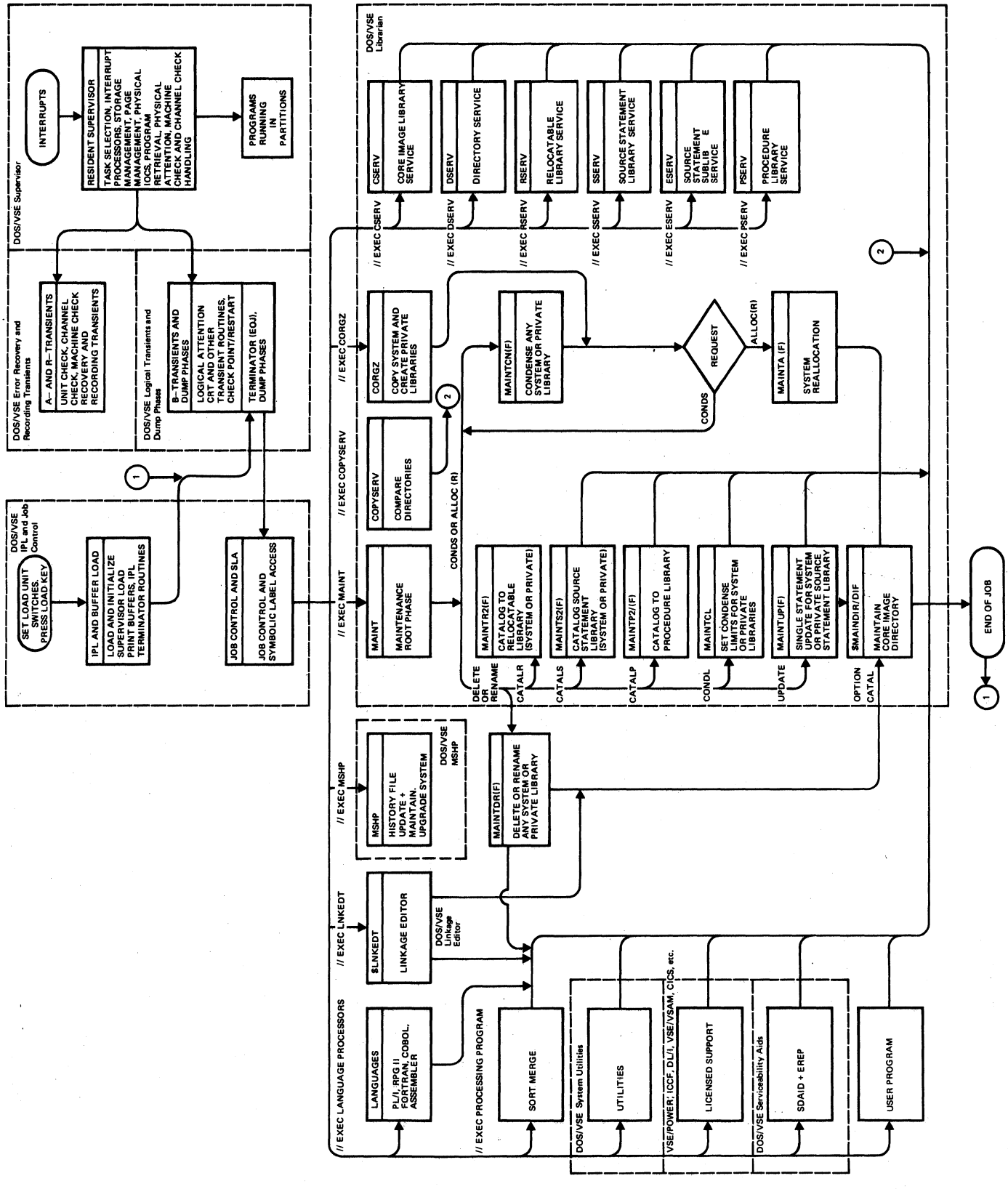
Titles and abstracts of other related publications are listed in the IBM System/370 Bibliography, GC20-0001.

Publication Organization

This manual consists of five major sections. The first section is an introduction to both programs summarizing their functions. The next two sections are dedicated to a detailed description of each of the programs showing their function-to-phase relationship, sequence of operation, control flow and I/O flow graphs, and the storage layout. Then follow general charts of the phases and more detailed charts which are in flowchart form for IPL and Job Control except Symbolic Label Access (SLA). The detailed flowcharts are identified by letters AA through ZZ. Numerals such as 01 identify the general charts.

The last section of the manual, the appendixes, contain label lists, phase to module and error messages cross references, LISTIO examples, and some control blocks and tables.

FIGURES


DOS/VSE Supervisor
INTERRUPTS
RESIDENT SUPERVISOR
TASK SELECTION, INTERRUPT PROCESSORS, STORAGE MANAGEMENT, PAGE MANAGEMENT, PHYSICAL IOCS, PROGRAM RETRIEVAL, PHYSICAL ATTENTION, MACHINE CHECK AND CHANNEL CHECK HANDLING
PROGRAMS RUNNING IN PARTITIONS
DOS/VSE Error Recovery and Recording Transients
A– AND R–TRANSIENTS
UNIT CHECK, CHANNEL CHECK, MACHINE CHECK RECOVERY AND RECORDING TRANSIENTS
DOS/VSE Logical Transients and Dump Phases
B–TRANSIENTS AND DUMP PHASES
LOGICAL ATTENTION CRT AND OTHER TRANSIENT ROUTINES, CHECK POINT/RESTART
TERMINATOR (EOJ), DUMP PHASES
DOS/VSE IPL and Job Control
SET LOAD UNIT SWITCHES, PRESS LOAD KEY
IPL AND BUFFER LOAD
LOAD AND INITIALIZE SUPERVISOR LOAD PRINT BUFFERS, IPL TERMINATOR ROUTINES
JOB CONTROL AND SLA
JOB CONTROL AND SYMBOLIC LABEL ACCESS
DOS/VSE Librarian
CSERV
CORE IMAGE LIBRARY SERVICE
DSERV
DIRECTORY SERVICE
RSERV
RELOCATABLE LIBRARY SERVICE
SSERV
SOURCE STATEMENT LIBRARY SERVICE
ESERV
SOURCE STATEMENT SUBLIB E SERVICE
PSERV
PROCEDURE LIBRARY SERVICE
// EXEC CSERV
// EXEC DSERV
// EXEC RSERV
// EXEC SSERV
// EXEC ESERV
// EXEC PSERV
// EXEC CORGZ
CORGZ
COPY SYSTEM AND CREATE PRIVATE LIBRARIES
MAINTCN(F)
CONDENSE ANY SYSTEM OR PRIVATE LIBRARY
REQUEST
CONDS
ALLOC(R)
MAINTA (F)
SYSTEM REALLOCATION
// EXEC COPYSERV
COPYSERV
COMPARE DIRECTORIES
CONDS OR ALLOC (R)
// EXEC MAINT
MAINT
MAINTENANCE ROOT PHASE
MAINTR2(F)
CATALOG TO RELOCATABLE LIBRARY (SYSTEM OR PRIVATE)
MAINTS2(F)
CATALOG SOURCE STATEMENT LIBRARY (SYSTEM OR PRIVATE)
MAINTP2/(F)
CATALOG TO PROCEDURE LIBRARY
MAINTCL
SET CONDENSE LIMITS FOR SYSTEM OR PRIVATE LIBRARIES
MAINTUP(F)
SINGLE STATEMENT UPDATE FOR SYSTEM OR PRIVATE SOURCE STATEMENT LIBRARY
$MAINDIR/DIF
MAINTAIN CORE IMAGE DIRECTORY
DELETE OR RENAME
CATALR
CATALS
CATALP
CONDL
UPDATE
OPTION CATAL
END OF JOB
// EXEC MSHP
DOS/VSE MSHP
MSHP
HISTORY FILE UPDATE + MAINTAIN. UPGRADE SYSTEM
MAINTDR(F)
DELETE OR RENAME ANY SYSTEM OR PRIVATE LIBRARY
// EXEC LINKEDT
$LNKEDT
LINKAGE EDITOR
DOS/VSE Linkage Editor
// EXEC LANGUAGE PROCESSORS
LANGUAGES
PL/I, RPG II FORTRAN, COBOL, ASSEMBLER
// EXEC PROCESSING PROGRAM
SORT MERGE
DOS/VSE System Utilities
UTILITIES
VSE/POWER; ICCF, DL/I, VSE/VSAM, CICS, etc.
LICENSED SUPPORT
DOS/VSE Serviceability Aids
SDAID + EREP
USER PROGRAM

INITIAL PROGRAM LOAD (IPL)

THE IPL PROCEDURE

The IPL procedure must be performed each time

-- the system is powered up;

-- you want to change the processing mode of your machine on a 4300 processor (switch between ECPS:DOS/VS-mode and 370-mode);

-- you want to reallocate your page data set;

-- you want to reassign
your VSAM master catalog (SYSCAT),
your recorder file (SYSREC),
your system console (SYSLOG);

-- you want to change the load configuration of system programs residing in the SVA;

-- you want to load a new supervisor.

Operating in the supervisor mode, IPL bootstrap processing

-- determines the hardware and SYSRES characteristics,

-- determines the system console address,

-- loads the specified supervisor from SYSRES,

-- builds a three-device system (SYSRES, SYSLOG, SYSUSE).

Now the system is prepared for IPL command processing.

The IPL command processing rootphase is loaded and the IPL commands, entered via SYSUSE, are processed:

-- the system I/O configuration may be completed via ADD and DEL commands;

-- the date and TOD clock may be set via the SET command;

-- SYSREC (and SYSCAT if VSAM will be used) must be assigned via the DEF command;

-- the page data set must be allocated via the DPD command;

-- size and layout of the SVA may be specified via the SVA command which also terminates IPL processing and therefore is mandatory, even if used without operands.

After having performed the functions required, IPL exits to EOJ.

LOADING PRINT CONTROL BUFFERS

For certain printers a buffer has to be loaded. The print control buffers of a printer can be loaded as follows:

-- Automatically during IPL. $$BUFLDR together with $$BUFLD1 and $$BUFLD2 are provided to load the FCB and UCB of a pertinent printer with the standard control information.

-- Dynamically by issuing the LFCB or LUCB attention command. See DOS/VS Logical Transients and Dump Phases Logic.

-- Dynamically by issuing the LFCB macro in a problem program. (This macro can only be used to load the FCB of a printer). See DOS/VS Logical Transients and Dump Phases Logic.

-- As a separate job step by executing the SYSBUFLD program. This program is documented at the end of the IPL description.

Note: For a 5203 printer without the universal character set feature, only the FCB is loaded. For a 1403U, only the UCB is loaded.

$$BUFLDR is executed as part of the IPL procedure. If 3203, PRT1, 3800 or 5203/1403U printers are attached to the system, $$BUFLDR together with the corresponding standard buffer image phases must be available in the core image library. The standard buffer image phases for the individual printers are shown in the following table:

Printer	To Load FCB	To Load UCB
1403U		$$BUCB4
3203	$$BFCB3	$$BUCB3
PRT1 3211	$$BFCB	$$BUCB
3203-4 and -5	$$BFCB00	$$BUCB00
3289-4	$$BFCB10	$$BUCB10
5203	$$BFCB5	$$BUCB5*

* Not required for a 5203 printer without the universal character set feature.

Note: The 3800 printer buffer load is initialized via a special command.

JOB CONTROL PROGRAM

The job control program provides job-to-job transition for all programs, background and foreground.

This program also prepares job steps for execution. (One or more programs can be executed with a single job. Each such execution is called a job step.)

On the basis of information provided in the job control statements, the job control program:

- Prepares the system for execution of programs.
- Assigns physical device addresses to logical units.
- Includes cataloged procedures and procedure modifier statements.
- Sets up fields in the communication region(s).
- Edits and stores volume and file label information.
- Prepares the system for restarting checkpointed programs.
- Clears the partition to binary zeros between job steps, (if job step is in real mode), otherwise it causes the pages to be cleared as they are used.

The job control program is executed in the virtual storage of the partition it is preparing.

SYMBOLIC LABEL ACCESS

Another part of job control is the Symbolic Label Access program (SLA). The SLA program provides device independent access to the DOS/VS label area. It is called each time information for the label area has to be added, deleted, retrieved, or modified.

PHASE-FUNCTION OVERVIEW

IPL is divided into two successive processing steps

- bootstrap processing
- command processing

The function of bootstrap processing is to load a supervisor and build a minimal (3-device) system, the basis for IPL command processing. The 3-device system consists of SYSRES, SYSLOG, and SYSUSE; IPL commands are entered via SYSUSE and since SYSUSE may be assigned to SYSLOG, the 3-device system may physically consist of two devices only. Another characteristic of bootstrap processing is that its main part is performed without supervisor support for input/output or loading modules from the SCIL.

The function of IPL command processing is to bring up the complete system by processing the specified IPL commands; there is now supervisor support available for most of its processing.

Figure 1 shows a list of IPL phases, where they reside, and what their functions are.

	Module/ Macro	Phase	Residence	Phase Description
Bootstrap Processing	$$A$PLBF	$$A$IPL0	0 *	FBA-SYSRES Bootstrap Loader
		$$A$PLBF	3-9 *	FBA-SYSRES Bootstrap Program
	$$A$PLBK	$$A$IPL1	00.00.1, 00.00.2 **	CKD-SYSRES Bootstrap Loader
		$$A$PLBK	00.01.5	CKD-SYSRES Bootstrap Program
	$$A$IPLR	$$A$IPLR	SCIL	IPL Retrieval Program
	IPLBMAC	-	MACLIB	Macro: Contains code and data commonly used by at least two of the bootstrap phases
Command Processing	IJBIPL	-	MACLIB	Macro: Serves to call all command processing phases
	IPLDISK MACIPLR2	$IPLRT2	SCIL	IPL Command processing Root Phase + IPL Command processing initialization
	MACIPLR3	$IPLRT3	SCIL	Process ADD/DEL Commands
	MACIPLR4	$IPLRT4	SCIL	Process SET Command; dynamically allocate SV areas
	MACIPLR5	$IPLRT5	SCIL	Process DEF and DPD Commands
	MACIPLR6	$IPLRT6	SCIL	Process SVA Command
	MACIPLR7	$IPLRT7	SCIL	Termination Processing
Load Print Buffer	IJBSBUFF	SYSBUFLD	SCIL	Load or Modify Print Buffer

Figure 1. IPL Module/Phase/Function Overview

Notes to Figure 1:

* FBA - Physical Block Number
** CC.HH.R of CKD-Device

CKD = Count-Key-Data Device
FBA = Fixed-Block-Architecture Device
SCIL = System Core Image Library

BOOTSTRAP PROCESSING, CHART 01

To load the desired supervisor and build a 3-device system (the base for IPL command processing), special standalone programs from outside the core image library have to be activated. These programs perform the bootstrap processing functions and consist of the following phases:

$$A$IPL0
$$A$IPL1
$$A$PLBF
$$A$PLBK
$$A$IPLR
$$A$IPLE (This phase is distributed only with VSE/Advanced Functions.)

Note that all these phases reside originally in the SCIL for APAR and PTF handling; before distributing a system residence, phase $$A$IPL0, $$A$IPL1, $$A$PLBF and $$A$PLBK have to be copied by special utility programs (IJBREPB or RESTORE) to their assigned disk locations. See Figures 1 and 2.

SYSRES Layout

CKD-Device		FBA-Device	
Location	Extent for ...	Location	Extent for ...
00.00.1 00.00.2	Phase $$A$IPL1	Block 0	Phase $$A$IPL0
00.00.3	Volume Label	Block 1	Volume Label
00.01.1 00.01.2 00.01.3 00.01.4	System Directory	Block 2	System Directory
00.01.5	Phase $$A$PLBK	Blocks 3-9	Phase $$A$PLBF
00.02.0 -	Core Image Library	Blocks 10 -	Core Image Library
a -	Relocatable Library	u -	Relocatable Library
b -	Source Stmt. Library	v -	Source Stmt. Library
c -	Procedure Library	w -	Procedure Library
d -	Label Area	x -	Label Area
e -	User Space	y -	User Space
f -	VTOC	z	VTOC

Note: a < b < c < d < e < f u < v < w < x < y < z
FBA block size is 512 Bytes, CKD record length varies.

Figure 2. SYSRES Layout

Hardware IPL will always load $$A$IPL0 and $$A$PLBF (from FBA-SYSRES) or $$A$IPL1 and $$A$PLBK (from CKD-SYSRES) from their assigned disk locations, while $$A$IPLR is always loaded from the SCIL.

Charts AA and AB explain the hardware IPL process, which ends with passing control to $$A$PLBF or $$A$PLBK.

$$A$PLBK, $$A$PLBF - SEQUENCE OF OPERATION

After relocating all address constants in PSW's and CCW's, the program must

-- Determine Type of Hardware

4300-hardware is assumed, if the program successfully performs an STCAP instruction. On 4300-hardware, this instruction also specifies the real and virtual storage size of this model. /370-hardware is assumed, if a STCAP-instruction fails, but a PTLB-instruction is executed successfully. If neither of these instructions could be successfully executed, a flag is set to cancel IPL and the SYSLOG device is still available.

For information on the STCAP and PTLB assembler instructions, see System/370 Principles of Operations.

-- Determine Size of Real Storage

On 4300-hardware, the STCAP instruction returned all necessary information.

On /370-hardware, we get the real storage size by an 'endless-loop' which will clear the real storage from the end of phase $$A$PLBK up to the end of real storage. The loop is interrupted by an addressing-program-check when the real storage is exhausted. The virtual storage size cannot be got from /370-hardware but has to be extracted from the supervisor later on.

On VM systems, DIAGNOSE instructions will be used to determine the size of real storage and to clear it. See IBM System/370 Principles of Operation for DIAGNOSE instructions.

The programs will save all necessary information about hardware and SYSRES device characteristics in an area described by IPLRCOM. The start address of this area and the load address of $$A$IPLR are passed to this phase via special registers.

-- Determine the SYSRES Device Characteristics

We extract the SYSRES volume serial number from the volume label record or block. To get the SYSRES device-type and capacity, we have to perform a READ DEVICE CHARACTERISTICS command only if SYSRES is an FBA device. If it is a CKD-device, we must read the format-4 record (first physical record) of the VTOC whose start address is extracted from the volume label record. The capacity information from the VTOC format-4 record must be found in a device characteristics table to identify the CKD-SYSRES device type.

-- Get Library Extent Information

We read the system directory records 1 and 4 from CKD-SYSRES or the system directory block from FBA-SYSRES respectively. The beginning CC.HH.R of the SCIL and of the procedure library (if present) is saved; for FBA, the beginning and ending physical block number is saved for later use in DEFINE EXTENT commands.

-- Load Phase $$A$IPLR, $$A$IPLE

Phase $$A$IPLE (if residing in the SCIL) is loaded at end of real storage. Space will be allocated for IOFLD (Communication area with IPL command processing) and phase $$A$IPLR, which must be resident in the SCIL, will be loaded in front of IOFLD. See Figure 4. IPLRCOM is saved into IOFLD and control is passed to $$A$IPLR.

$$A$IPLR - SEQUENCE OF OPERATION

Control is passed to this phase from $$A$PLBF or $$A$PLBK. The addresses in PSW's and CCW's are relocated. Now module initialization is performed (message 0I04I built, data areas initialized).

-- Prepare for Operation with/without Phase $$A$CDL0

The directory of the SCIL is scanned for the IPL-phase $$A$CDL0. If found, the text of the phase is read into real storage. If not found, the normal process of system preparation continues.

-- Request a SYSLOG Device

Normally, SYSLOG is specified by the operator via an attention interrupt (pressing REQUEST or ENTER) which transmits the SYSLOG device address.

When the Automatic System Installation feature (ASI) is installed (phase $$A$IPLE resides in the SCIL) the ASI initialization function (ASIFINIT) will transmit the SYSLOG device address from the first statement of the ASI IPL procedure. ASI is in process now.

$$A$CDL0 contains the communication device list (CDL). The list specifies the system console (SYSLOG) and the I/O device (SYSLOG and/or SYSUSE) that may present interrupts to IPL and thus establish themselves as communication device(s) for transmitting IPL commands.

If $$A$CDL0 is in the SCIL, the program checks if the SYSLOG device address is in the CDL. If yes, or if there is no $$A$CDL0 the SYSLOG device is accepted. Otherwise IPL waits for another SYSLOG device address to be transmitted.

Note: IPL may loop indefinitely if the CDL has been specified incorrectly. Therefore it may be useful each time when IPL enters the wait state again to enter manually the device address for SYSLOG and/or SYSUSE into locations X'10' to X'17' in the hexadecimal format 00000cuu. To resume processing press REQUEST/ENTER at SYSLOG.

On acceptance of the SYSLOG device address, a PUB is built and the phase writes message 0I04I.

After having built the PUB, the phase makes SYSLOG available for operator/system communication for the remaining system preparation process; all error messages are written to SYSLOG.

-- Request a Supervisor Name

On an interactive IPL (ASI not in process), the name of the supervisor to be loaded and its paging option are requested via message 0I03A in SCP-format.

When ASI is in process, the name of the supervisor to be loaded and its paging option will be extracted from the first statement of the ASI IPL procedure too. Message 0I03A need not be written.

When ASI has to be restarted (the operator presses the INTERRUPT key of his console), message 0I03A in AF-format will be displayed. The operator may respond either

1. with ASI procedure names for IPL and/or JCL
   i.e. ASI will proceed taking the supervisor name, paging option and IPL commands from the specified IPL procedure.
   or

2. with a supervisor name and paging option
   i.e. ASI will be terminated and IPL will proceed interactively.

The directory of the SCIL is scanned for the specified supervisor entry. If this entry is not found, message 0I02A is issued and the name of the supervisor is requested once more. If the entry is found, the length of the supervisor is computed and it is checked whether the supervisor will fit into real core without overlaying $$A$IPLR.

Note that the maximal supervisor size may not exceed 512K bytes. The supervisor is read into real storage starting from location 0.

If ASI is in process, the ASI-RESTART facility is activated by calling the ASI-enable function (ASIF ENA) via phase $$A$IPLE.

-- Check Supervisor Generation Options

It is verified if the supervisor has been generated for the existing hardware (4300, /370, VM), whether it supports the SYSRES device type (e.g. you may load from FBA-SYSRES a supervisor that has been generated for CKD only), and whether the supervisor has been generated for the correct SYSLOG device type (e.g. you may load from a 3277-SYSLOG a supervisor that has been generated for /370-115 DOS-SYSLOG).

Message 0I68A is displayed at SYSLOG with a code number that specifies the type of incompatibility that has been found between existing environment and supervisor generation.

To obtain supervisor support, we have to initialize its subcomponents such as page manager, input/output supervisor, or timer management.

The paging option is analyzed and if it was N (non-pageable supervisor) it is indicated within the SMCB that there is no pageable supervisor part. Per default, the supervisors are generated with a pageable part.

-- Initiate System

The length of the I/O extended area for this CPU model is calculated.

The RSIZE parameter of the VSTAB-macro specifies the size of the real storage the supervisor is generated for. This real storage is used for dynamically allocatable supervisor tables. The size of these tables must be adjusted to the existing real storage, since the supervisor may have been generated for a real storage that is smaller, equal, or bigger than the existing one. Thus, supervisor table space may be wasted or missing.

In /370 mode, a
page frame table
page table
segment table
are used.

In ECPS:DOS/VS-mode, a
page frame table
is used.

The generated BG begin address and system save area address have to be updated.

A communication area (SVIPL) has been introduced for IPL into the supervisor, described via MAPSVIPL macro. It contains the entry points of routines invoked from IPL for supervisor initialization and pointers to tables that have to be generated or initialized through IPL processing. Its start address is part of SYSCOM (IJBSVIPL). The control registers are initialized.

The DAT-feature and the (location X'50') timer are initialized for /370 only.

The initialization subroutines reside in the supervisor copy-buffer area and are overlaid with table and buffer space during later processing.

Now some indicators are set, signalling that IPL is in progress, disabling the attention routine and permitting write operations on the SYSRES device.

The storage protect key of the supervisor area is set to 0, while the storage protect key of the BG-area (the remaining real storage) is set to 1. The channel control table is built.

-- Build Communication Area for Command Processing (IOFLD)

This area is located at end of real storage. The SYSGEN I/O tables generated into the supervisor are moved into this communication area. The PUBs, FOCL, NICL, FICL, diskette label record (if a diskette is an IPL communication device), and device specific information about SYSLOG and SYSUSE are stored in this area (described by DSECT IOFLD). The tables will be used by the $IPLRT phases to complete the system I/O-configuration process (ADD/DELETE).

-- Locate IPL Communication Device (SYSUSE)

If ASI is in process, a dummy diskette label record is placed into IOFLD, the SYSRES PUB is copied into the SYSUSE PUB and job control is informed that the IPL input is not from SYSLOG. Processing continues building the three-device system.

If there is an interactive IPL, the system is put into the wait state and the operator has the option of selecting the communication device for IPL. If the desired communication device is:

1. A card reader, the operator presses the start key on the reader causing an I/O interrupt (device end).

2. A console printer-keybord, the operator presses the request key causing an I/O interrupt (attention).

Note: Only the device used as SYSLOG may present the attention interrupt.

3. A diskette, the operator presses the start key causing an I/O interrupt (device end).

-- Build 3-Device System I/O Tables

After the operator has selected a communication device, the PUBs built for SYSRES, SYSLOG, and SYSUSE (if SYSUSE is not equal to SYSLOG) are copied into the supervisor PUB table and the BG LUBs for SYSRES, SYSLOG, and SYSUSE are assigned to the coresponding PUB table entries. The FOCL (first-on-channel list) and the channel bucket are built for this 3-device system. Figure 5 below shows an example of the I/O tables for a 3-device system built by $$A$IPLR.

-- Initialize Program Management

The logical transient area (pointed at from SYSCOM via IJBLTA) contains initialization code for the FETCH system task. A temporary AVR (automatic volume recognition) table is allocated and the volume-characteristics of the SYSRES device placed into this table. This information is passed to the FETCH initialization routine.

-- Pass Control to IPL Command Processing

Since FETCH has been initialized, it may now be used for program loading. It is checked whether the command processing root phase $IPLRT2 will fit into real storage. If yes it will be loaded via SVC 4. This SVC has been moved into IOFLD, from where it is executed; thus it is possible to overlay phase $$A$IPLR with $IPLRT2.

A supervisor state PSW is loaded with DAT bit off and I/O and external interrupts enabled. The address of IOFLD is passed via a general register to $IPLRT2; this phase is loaded into the BG-area, behind the BG system save area; it is invoked via BR 1 after SVC 4 from IOFLD.

COMMAND PROCESSING, CHARTS 02 TO 04

For the formats of IPL commands, see DOS/VS System Control Statements. The commands are being processed in the phases $IPLRT2-7. The following gives the sequence of operation in each of these phases:

$IPLRT2 - SEQUENCE OF OPERATION

$IPLRT2 is loaded by the phase $$A$IPLR which calculates the load point behind the supervisor and a reserved area for tables. The phase consists of two CSECTs: IJBIPR50, the root of the program, and IJBIP250, the initialization part, which is overlaid after initialization.

CSECT IJBIP250
- checks for TOD support, stores the TOD clock and, depending on the state of the clock, displays messages on SYSLOG. If the clock is in the 'set' state, the date, time-of-day, and zone are displayed on SYSLOG. A switch is set if the clock is in the 'error' or 'not-set' state.
- checks for RMSR support. Loads $$BCCHHR to build the load list.

CSECT IJBIPR50
- completes the system preparation and processes the IPL commands.
- contains the error subroutines to issue the error messages when necessary.
- loads the appropriate phase into an overlay area behind the root phase:

  $IPLRT3
  for processing an ADD or DEL command

  $IPLRT4
  a. for processing a SET command
  b. for moving I/O tables, PUBs, LUBs immediately before the first DEF, DPD, or SVA command is processed.
  c. for final allocation of tables (PFT, AVR tables, etc.) immediately before the SVA command is processed.

  $IPLRT5
  (with linked Common VTOC Handler) for processing a DEF or a DPD command

  $IPLRT6
  for processing the SVA command

  $IPLRT7
  for termination processing

The ADD, DEL, SET, DEF, DPD, and SVA commands are entered from the IPL communication device (SYSUSE).

When the operator enters the last command (which is always SVA) IPL prompts him to enter

-- a SET command if the TOD has to be set

-- a DEF command for SYSREC, if SYSREC was not assigned by a previous DEF command.

-- a DEF command for SYSCAT if SYSCAT was not assigned by a previous DEF command.

-- a DPD command if the page data set was not allocated by a previous DPD command.

$IPLRT3 - SEQUENCE OF OPERATION

$IPLRT3 is loaded and executed only when ADD or DEL commands are submitted.

$IPLRT3 does one or both of the following:

- Adds one or more devices to the system.

- Deletes one or more devices from the system.

ADD ROUTINE The add routine ensures the device is not already present. It then determines where to add the PUB in the PUB table and moves all the PUB entries beyond this point down one PUB length to make room for the new PUB. The new PUB is then inserted in the area just vacated. The LUB table and FOCL pointers are updated to reflect the change in the PUB table and the routine returns to read another control statement.

DELETE ROUTINE The delete routine first checks to see if the device to be deleted is in the PUB table and then determines the location in the PUB table of the PUB to be deleted. The LUB table and FOCL pointers are updated to reflect this change. The routine returns to to Monitor in the root phase to read another control statement.

$IPLRT4 - SEQUENCE OF OPERATION

$IPLRT4 comprises 3 parts which are executed at different times:

Part 1: Processing of SET command:

- sets the system date.
- sets the system zone if required.
- resets the contents of the TOD clock, if required.

Part 2: Executed before the first DEF, DPD, or SVA command is processed (includes pass 1 of the allocation routine for SV areas).

- checks the channels for file-protect support for each device if DASDFP (DASD File Protect) option was specified during supervisor generation.
- scans the PUB table. If a 3203, 3211, 3289-E, 3800, 1403U, or 5203 is found, $IPLRT4 loads $$BUFLDR.
- checks for RMSR support and if present builds the PUB2 table.
- checks whether RMS support has been generated for CPUs other than the 115 or 125. If not, IPL is terminated. For the models 115 and 125, checks for correct recording support and checks for hardware-recorded devices.
- scans the PUB table for DASD devices. If a DASD device which is not operational is found, the PUB is set to indicate "device down" and the LUB is unassigned.
- allocates an AVR (automatic volume recognition) entry for each DASD PUB and calls the supervisor for initialization of the entry.
- allocates and initializes PUBSCAN tables.
- shifts page and segment tables (both only for 370) and page frame table to their final locations.
- allocates space for some other tables (RTAB, Copy Buffers, extended logout areas).
- The system assignments for SYSRES and the communication device (SYSRDR or SYSLOG) are checked and permanently assigned. The system I/O tables are moved from their temporary location in high real storage to their permanent location in the supervisor area. This move overlays the three-device IPL I/O tables that where built by $$A$IPLR.

After part 2 of $IPLRT4 has been executed no more ADD or DEL commands may be entered.

Part 3: Executed before the SVA command is analyzed. (Includes pass 2 of the allocation routine for supervisor areas.)

- performs final allocations of tables in the dynamic supervisor area after reserving buffer space if SYSREC or HARD COPY FILE is on FBA.
- checks (and prompts) for SYSREC assignment
- warns if SYSCAT is not assigned.

SET ROUTINE The DATIMERT subroutine converts the Date, Clock, and Zone specifications to a 64-bit binary value. This value represents the number of clock units (that is, microseconds times 2 to the 12th power) elapsed since January 1, 1900 at 0.00 a.m. GMT, and is used to set the TOD clock. A GETIME macro is issued to obtain an automatic update of the midnight value in the supervisor.

$IPLRT5 - SEQUENCE OF OPERATION

$IPLRT5 is loaded by the root phase and executed for every IPL. The phase

- diagnoses the DEF command

- assigns the VSAM catalog SYSCAT and/or the system recorder file SYSREC to a physical unit. SYSREC (and SYSCAT, if VSAM will be used) must be assigned via IPL.

- processes the DPD command

- updates parameters for the load leveler

- returns control to the root phase.

DEF ROUTINE The DEF command has to precede the SVA command. The DEF routine reads the physical units specified in the DEF command and assigns SYSCAT and SYSREC to them. SYSCAT may be unassigned (UA).

DPD ROUTINE The DPD command has to precede the SVA command. It causes IPL to do the following:

- Open page data set.

- Format the page data set if it has not been formatted or if it must be reformatted (for CKD devices only).

- If a VOLID was specified a check is made to see if it matches the VOLID on the specified unit. If it does not, a message is issued. If no VOLID has been specified, no check is made.

- Set the load leveller constants based on CPUID and device type of the page data set device.

$IPLRT6 - SEQUENCE OF OPERATION

$IPLRT6 is loaded by the rootphase and executed for every IPL. The phase:

-- diagnoses the SVA command.

-- reads phase $$A$SVA, which contains the names of load lists.

-- loads the load list phases and reads in the SCIL directory entries of all phases that are named in the load lists if they fulfill the requirements for loading.

-- sorts the directory entries.

-- calculates for ALLOC the number of SDL entries and the SVA space needed for the text of the phases (PSIZE).

-- informs the operator if load lists or phases are missing.

$IPLRT7 - SEQUENCE OF OPERATION

$IPLRT7 is loaded by the root phase and executed for every IPL. The phase does the following:

-- Set clock comparator

-- Complete LUBs for foreground partitions; set bits in PUB ownership table

-- Read information on procedure library and label area and store it in system GETVIS area

-- SVC 83 ALLOC (for virtual space)

-- Switch from real to virtual status (for 370 mode)

-- Shift SDL entries into system SVA

-- Load $MAINDIR (or $MAINDIF if FBA) into BG and call this phase for loading system phases into the SVA

-- Allocate LUB2 table and extent blocks

-- SVC 83 ALLOC (for real space, for 370 only).

-- Issue IPL complete message

-- Exit from IPL by SVC 14 (EOJ)

PHASE $$BUFLDR, CHART 05

$$BUFLDR is loaded by phase $IPLRT4. $$BUFLDR scans the PUB table for 3203, 1403U, PRT1, 3800, and 5203 printer entries. When it finds an entry, it tests whether the printer is operational and, if so, loads the UCB. It then loads phase $$BUFLD1, which loads the FCB for this printer (not 1403U). $$BUFLDR then returns to scanning the PUB table for further printer entries. When all buffers have been loaded, $$BUFLDR returns control to $IPLRT4. $$BUFLD2 is the buffer-load error recovery phase and is called by $$BUFLDR. (For the 3800 printer, an initialize command is issued.)

These phases issue two messages:

1. 0I26I indicates that the buffer loads on the currently referenced printer have been completed successfully.

2. 0I27I indicates that the referenced buffer on the referenced printer has failed to be loaded successfully.

After attempts have been made to load all printer buffers, the program returns to $IPLRT4.

THE SYSBUFLD PROGRAM, CHART 06

The SYSBUFLD program is designed to load the appropriate control buffer image into the forms control buffer (FCB) for output in a nonstandard page layout and into the universal character set buffer (UCB) for output with a UCS print train.

SYSBUFLD is invoked by a // EXEC SYSBUFLD statement and can be executed any time after IPL within the user job stream. With one invocation of SYSBUFLD, the FCB and UCB of a single printer, or any combination of buffers on several printers, can be loaded.

The procedure for using SYSBUFLD is described in DOS/VS Operating Procedures.

For information on the physical characteristics and environmental requirements of SYSBUFLD, refer to DOS/VS System Generation.

Input and output requirements and control information is described in DOS/VS System Control Statements.

The program SYSBUFLD requires approximately 3K bytes of storage, where the two phases of the program overlay each other.

SEQUENCE OF OPERATION OF SYSBUFLD

PHASE SYSBUFLD

-- reads control statement

-- checks for errors and incompatibilities

-- loads the UCB buffer

-- calls SYSBUFF1 to load the FCB.

PHASE SYSBUFF1

-- checks number of print positions and phase length

-- loads FCB buffer

-- calls SYSBUFLD to read the next control statement.

IPL CONTROL FLOW

Figure 3 shows an overview of the IPL control flow with the main interface areas. The SYSBUFLD program is included in the overview.

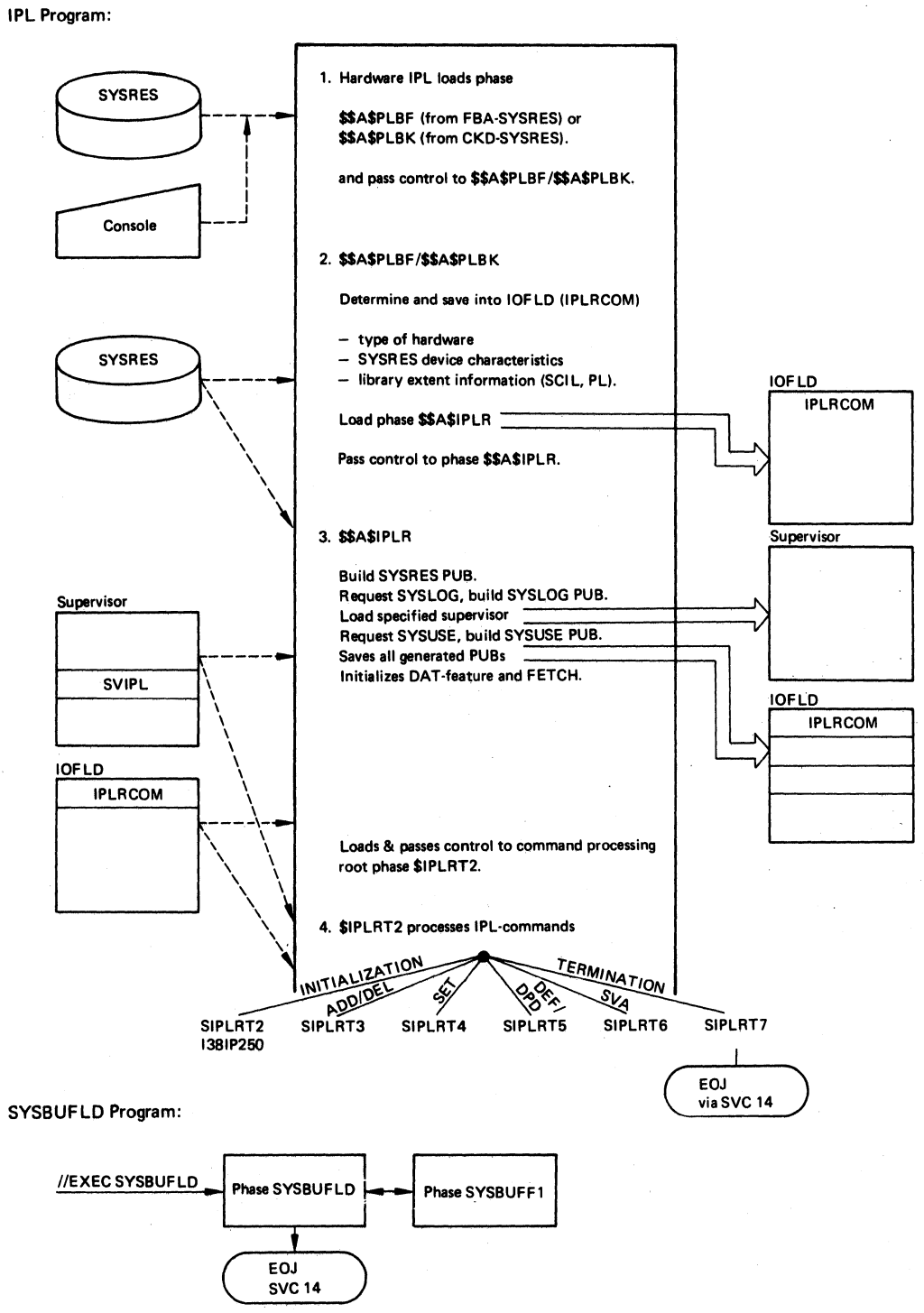


Figure 3. Control Flow of IPL and SYSBUFLD Programs

IPL STORAGE MAP

Figure 4 shows how the IPL program is distributed in storage at 4 different points of the process. Steps 1 and 2 show the different conditions for FBA and CKD.

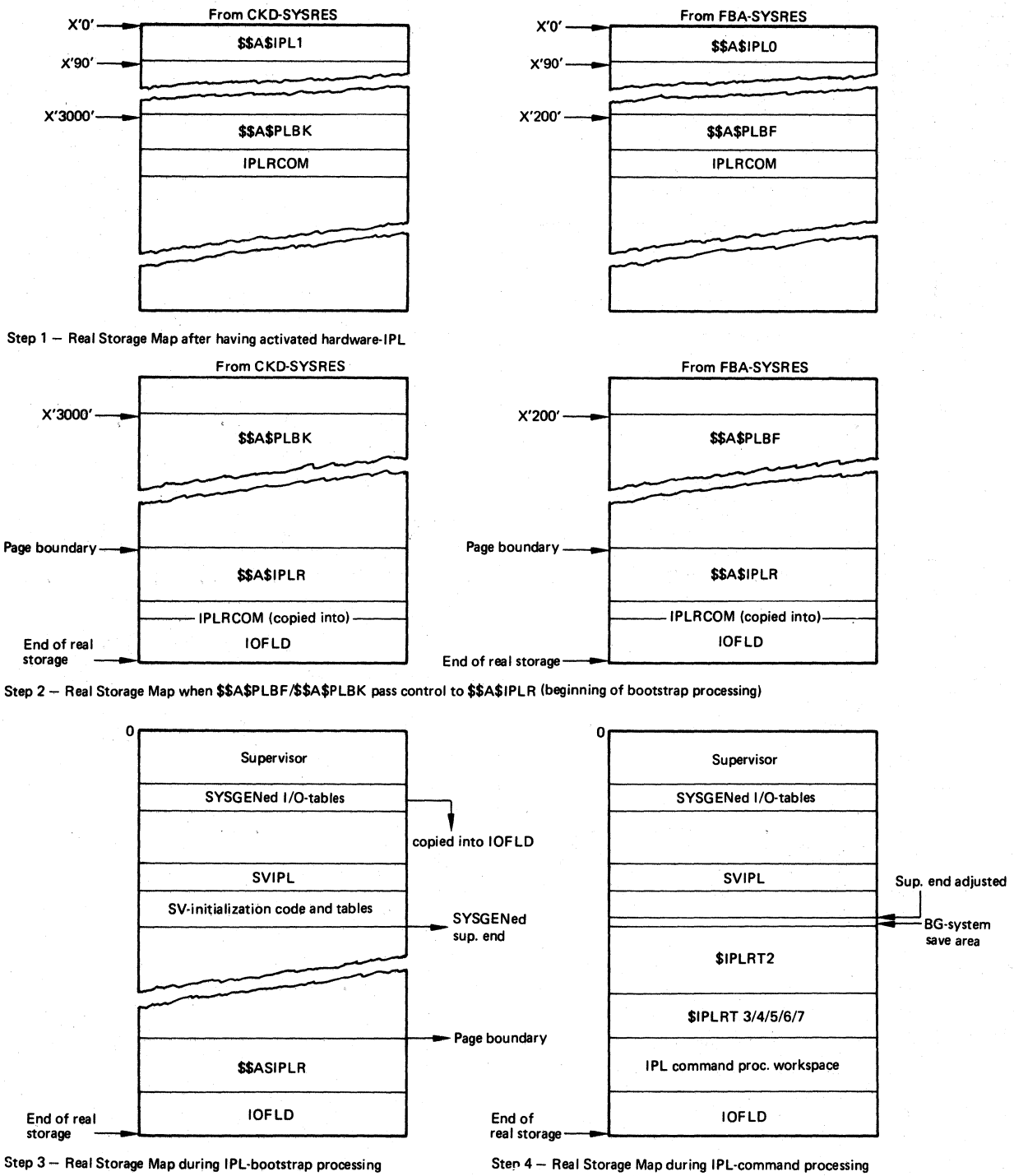
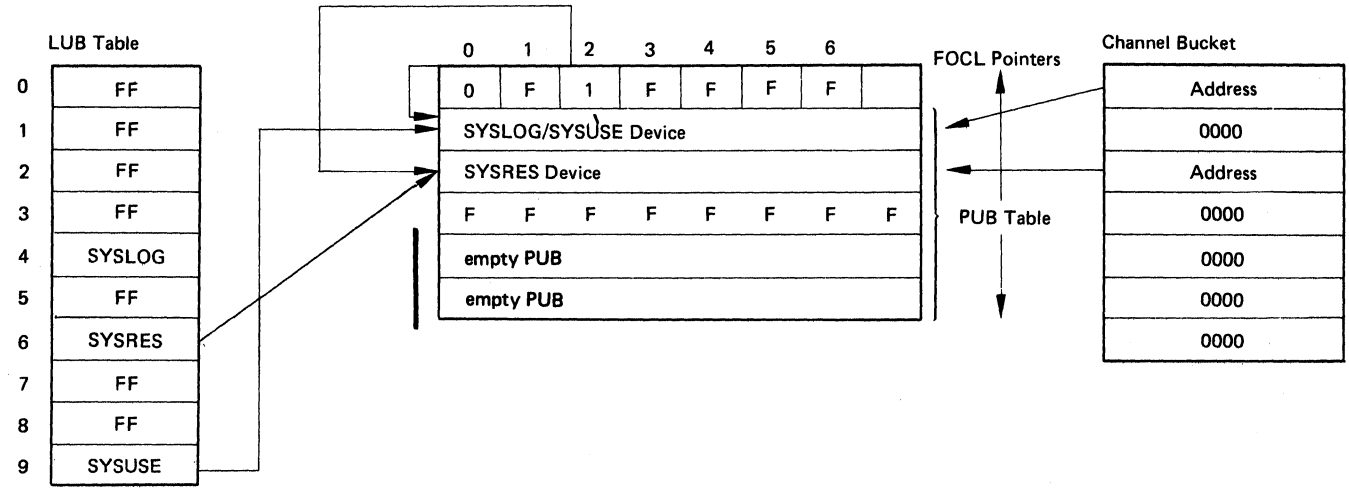


Figure 4. IPL Real Storage Map

I/O TABLES AND HARD WAIT CODES IN THE IPL PROGRAM

Figure 5 shows the I/O tables for the three-device system generated at the beginning of IPL.



Note: It is assumed that SYSRES is on channel 2 and that the system console (SYSLOG) and the communication device (SYSUSE) are on channel 0 and SYSLOG = SYSUSE.

Figure 5. I/O Tables for the Three-Device System

Figure 6 gives an overview of the hard wait states during IPL.

If there is an equipment malfunction during IPL, or if the IPL program cannot be loaded, a message is placed in bytes 0-3. In this state all interrupts are disabled, and you must repeat IPL after displaying these bytes.

MCH/CCH Codes:

Byte 0	Byte 1	Byte 2	Byte 3	Explanation
X'C1'	X'E2'	A,I,S(1)	Not used	Irrecoverable machine check.
X'C2'	X'E2'	Not used	Not used	Irrecoverable channel failure during FETCH.
X'C3'	X'E2'	A,I,S(1)	Not used	Channel failure on SYS1LOG when RMS message scheduled.
X'C4'	X'E2'	A,I,S(1)	Not used	No ECSW stored.
X'C5'	X'E2'	A,I,S(1)	Not used	Channel failure; ERPIB queue exhausted.
X'C6'	X'E2'	A,I,S(1)	Not used	Channel failure; two channels damaged or a damaged channel situation occurred while RMS was executing an I/O operation.
X'C7'	X'E2'	A,I,S(1)	Not used	Channel failure; system reset was presented by a channel.
X'C8'	X'E2'	A,I,S(1)	Not used	Channel failure; system codes in ECSW are invalid.
X'C9'	X'E2'	A,I,S(1)	Not used	Channel failure; channel address invalid (370 mode only).
X'D1'	X'E2'	A,I,S(1)	Not used	Irrecoverable channel failure on SYSVIS.
X'07'	X'E6'	Channel	Unit or X'00'	IPL I/O error or equipment malfunction; condition code 2 during STIDC instruction. Channel and unit indicate whether device in error is SYSRES or communication device. When byte 3=X'00', byte 2 indicates the channel for which STIDC instruction was issued. Re-IPL system.

(1) A=(X'C1') = SYSREC recording unsuccessful. -- Run SEREP.
I=(X'C9') = SYSREC recording incomplete. -- Run SEREP.
S=(X'E2') = SYSREC recording successful. -- Run EREP.

Other Codes:

X'00'	X'00'	X'0F'	X'D0'	IPL is canceled due to an irrecoverable error. The cancel code is placed in byte 1.
X'07'	X'E6'	Channel	Unit or X'00'	IPL input/output error: -- I/O error on SYSRES -- I/O error on communication device -- Equipment malfunction during the STORE CHANNEL ID instruction.
X'F0'	X'C9'	X'F0'	X'F6'	The devices type of SYSRES could not be identified. The volume label (VOL1) or format-4 record of the VTOC contains invalid information. The pack was not initialized correctly. The system enters the hard wait state. Initialize the disk pack.
X'F0'	X'C4'	X'F3'	X'F8'	Refer to message 0D38A in the DOS/VS Message manual.

Figure 6. Hard Wait Codes for IPL in Low Storage

The job control program is automatically loaded into the background partition after IPL. To obtain the job control program in a foreground partition, a BATCH or START command must be issued for the desired partition. The job control program is then loaded into the desired partition if the following minimum requirements are met:

- At least 128K of virtual storage in the partition, including GETVIS space.
- Separate system I/O files for the partition.

The job control program does one or more of the following on the basis of information provided in job control statements:

- prepares programs for execution,
- evaluates control statements and reads cataloged SYSRDR and SYSIPT data for cataloged procedures, and calls the appropriate processing programs,
- assigns device addresses to symbolic units,
- sets up fields in the communication region(s),
- edits and stores volume and file label information,
- prepares for restarting checkpointed programs,
- clears the program area to binary zeros between job steps (real-mode job steps only, the pages of virtual partitions are cleared as they are used),
- prepares input for the linkage editor program if the LINK option has been specified. The statements: ENTRY; ACTION, PHASE, and INCLUDE, when present in the input stream, are copied to SYSLNK as card images. An INCLUDE statement with a blank operand causes the contents of SYSIPT to be copied to SYSLNK until a /* statement is read from SYSIPT. Blank cards from SYSIPT are ignored. This preparation for the linkage editor program is valid for the BG partition only, unless the supervisor includes the PCIL option.
- prepares input for the MAINT program by writing a CATALR statement (if included in the SYSRDR job stream) and any following PHASE statement to SYSPCH.

  This is useful if a compilation or assembler run follows because the SYSPCH output can then be used as input (on SYSIPT) for a subsequent MAINT job step. This is only possible if the link option is not in effect.

The job control program is executed in the virtual storage of the partition it is preparing for use. If the job step being prepared is a virtual-mode job step it overlays the job control program.

A JOB statement in the input stream marks the beginning of a job and a /& statement marks the end of a job. An EXEC statement calls for execution of a job step or of a cataloged procedure. A job step is normally ended with the EOJ macro. A /+ statement marks the end of a cataloged procedure.

The Symbolic Label Access program (SLACC, in program listings referred to as SLA) is used to

read
write, and
modify

DOS/VS label information on label cylinders for CKD devices and in the label area for FBA devices.

The label area logically is divided into subareas for partition related label information and a general area for partition independent label information.

Partition related information is contained in the following label subareas:

1. User label information (temporary):
   - -- Write-access by job control (// OPTION USRLABEL)
   - -- Cleared by overwriting:
     - -- when a JOB or end-of job statement is read
     - -- at IPL time
2. Partition standard label information (permanent):
   - -- Write-access by job control (// OPTION PARSTD)
   - -- Cleared by overwriting
3. Free-usage label information

   This subarea is used only by the system internally:

-- Write-access and clearing can only be done by direct usage of the label macro
-- Clearing at IPL time

Partition independent label information is contained in the standard (system) label area. This information is valid for all partitions.

-- Write-access by job control (// OPTION STDLABEL)
-- Cleared by overwriting

If label information is required (GETLBL function) the subareas are searched in the following sequence:

-- user labels of partition
-- free usage labels of partition
-- partition standard labels
-- standard labels

PHASE-FUNCTION OVERVIEW

Phase	Function
$JOBCTLA	Root phase
$JOBCTLB	Prepare a checkpointed job for restart
$JOBCTLC	Open hard copy file
$JOBCTLD	Process ASSGN and CLOSE
$JOBCTLE	Process EXEC
$JOBCTLF	Process DVCDN, DVCUP, LISTIO, MAP, RESET, UNBATCH
$JOBCTLG	Process CANCEL, EOJ, /&, /*, EOP, JOB, OPTION
$JOBCTLJ	Process all other commands except those in $JOBCTLK
$JOBCTLK	Process DLAB, DLBL, EXTENT, LBLTYP, RSTRT, SETPRT, TLBL, TPLAB, VOL, XTENT
$JOBCTLM	Process ROD, open recorder file
$JOBCTLN	Job accounting and POWER/VS-E interface
$$BLSTIO	Common functions to process LISTIO and DVCDN
$IJBSLA	Read, write, and modify label information

Note: For the format of job control commands, see DOS/VS System Control Statements.

Figure 7. Job Control Phase-Function Overview

SEQUENCE OF OPERATION

The job control component consists of 11 phases, one B-transient, and the SLA program which are described in the following.

$JOBCTLA, CHARTS 07 AND 08

This phase is the initial entry into the job control program. It is loaded every time the job control program is fetched and is the root phase. It is resident in virtual storage at all times during job control execution and contains routines that are used by the other job control phases.

Job control input is read from SYSRDR or SYSLOG depending on the setting of the job control input switch (COMREG+56 bit 2). As each control statement or command is read, it is analyzed to determine which of the processing routines is to be used. The phase containing the correct processing routine is loaded if it is not already in virtual storage as a result of the previous statement or command.

Phase-Vector Table

The Phase-Vector table contained in the root phase ($JOBCTLA) is used to determine the correct phase and processing routine required to process a given control statement.

The operation field of the control statement is compared to each entry in the table until an equal is found. The equal entry identifies the correct phase and the

displacement within the phase of the branch instruction that directs the program to the correct processing routine. The entry also contains a 1-byte condition switch bank used to control processing for format verification, logging conventions, and cancel procedures for the statement. Figure 8 shows the format of an entry in the Phase-Vector table.

For example, the job control statement may have the following entry in the phase vector table:

```
DC CL7'JOB'
DC X'7A'
DC AL1(12)
DC C'G'
```

The condition switches say:

- The JOB statement is to be processed even if a cancel is being executed.
- Logging on both SYSLOG and SYSLST is suppressed.
- The statement may not start without // and must start in column 1.

Bytes 8 and 9 of the entry say:

The branch-vector table entry is located at a displacement of 12 bytes from the beginning of the phase with suffix 'G' ($JOBCTLG).

Data File Blocks

For its own I/O file handling, job control uses control blocks for which the pattern is located in this phase. These blocks are called DFB's (Data File Blocks) and their format is shown in Figure 9.

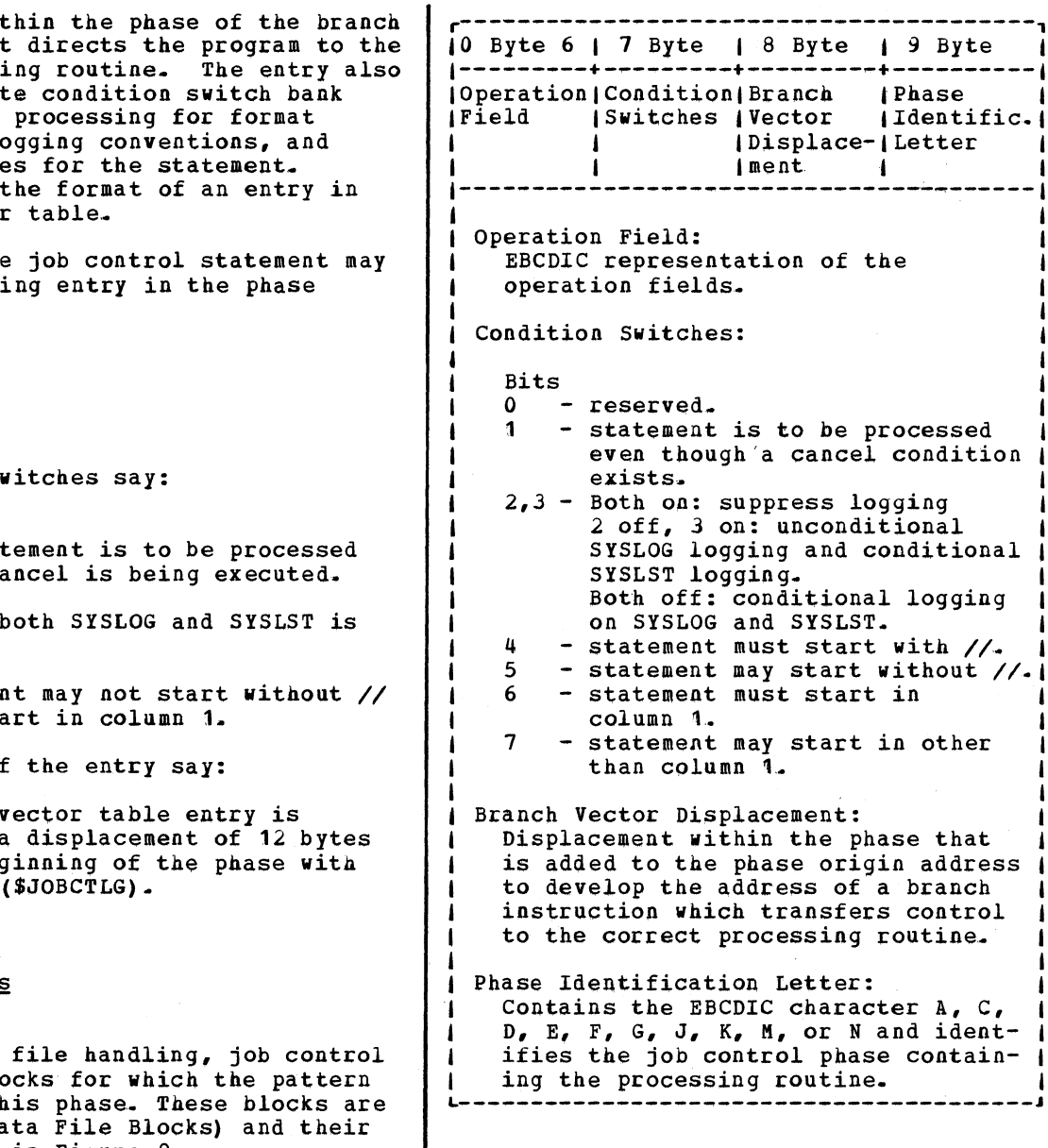


Figure 8. Phase Vector Table Entry Format

```
┌─────────────────────────────────────────┐
|                 ------------        )    |
| DFB      DS CL6              )      )    |
|                              )      )    |
| DFBCBL   DS CL2   Symbolic    )      )    |
|                   unit address) CCB )    |
|                              )      )    |
| DFBCBA   DS CL4   CCW address )      )    |
|                                     )    |
|          DS CL4  ------------       )    |
|                                     ) U/R |
| DFBCWA   DS CL4   ASA CCW            ) TAPE|
|                   address            ) DASD|
|                                     )    |
| DFBCCU   DS CL2   Channel            )    |
|                   and unit           )    |
|                                     )    |
| DFBDVC   DS CL1   Device type        )    |
|                                     )    |
| DFBFLG   DS CL1   Flag byte          )    |
|                   (see Note 1)       )    |
|                                     )    |
| DFBCCW   DS CL8   CCW (Not used      )    |
|                   for DASD)          )    |
|                  ------------            |
|          DS CL8   Seek               )    |
|                                     )    |
|          DS CL8   Set Sector if      )    |
|                   RPS is supported   )    |
|                   TIC if RPS is      )    |
|                   not supported      )    |
|                   (see Note 2)       )    |
|                                     )    |
|          DS CL8   Search             )    |
|                                     )    |
|          DS CL8   TIC                )    |
|                                     )    |
|          DS CL8   Write (count,      )    |
|                   key, and data)     )    |
|                   (chained data)     )    |
|                                     ) DASD|
|          DS CL8   Write (data)       ) Only|
|                                     )    |
|          DS CL8   Set Sector if      )    |
|                   RPS is supported   )    |
|                   TIC if RPS is      )    |
|                   not supported      )    |
|                   (see Note 2)       )    |
|                                     )    |
|          DS CL8   Search             )    |
|                                     )    |
|          DS CL8   TIC                )    |
|                                     )    |
|          DS CL8   Verify (count      )    |
|                    field)            )    |
|                   (chained data)     )    |
|                                     )    |
|          DS CL8   Verify (data)      )    |
|                  ------------       )    |
└─────────────────────────────────────────┘
```

Figure 9. DFB Format

Note 1:

```
DFBFLG - flag byte
         bit 0:1 = Open
         bit 1:1 = Unit Exception
                   processing
         bit 2:1 = U/R device
         bit 3:1 = Overlap requested
         bit 4:1 = ASA is supported
                   on this device
         bit 5:1 = Not used
         bit 6:1 = Attention routine
                   switch
         bit 7:1 = CCB has been
                   initialized
```

Note 2:

With RPS support, the last CCW for a read operation is chained to a read count and a read sector CCW to obtain the sector value of the next record.

<u>Interface $JOBCTLA - $IJBFBA</u>

To have system files supported on FBA devices it is required to specify SYSFIL=YES in the FOPT macro. The specification of

DISK=(devicetype, devicetype, ...)

in the PIOCS macro allows to have system files on CKD and/or FBA devices.

The supervisor call 103 performs the input/output operations from and to system files on FBA. The code of the SVC 103 is split into two parts:

a) a resident part contained in the supervisor (resident SVC 103 code)

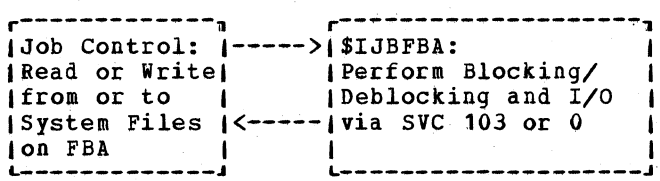
b) a pageable part loaded into the SVA (module $IJBFBA).

```
  -------
 (SVC 103)
  -------
     |
     V
┌────────────┐      ┌──────────────────┐
|Resident    |----->|$IJBFBA           |
|SVC 103 Code|      |Perform Blocking/ |
|            |<-----|Deblocking        |
└────────────┘      └──────────────────┘
```

For performance reasons, $JOBCTLA calls $$JBFBA directly via SVC 4 (LOAD). Also reading from the (FBA) procedure library and writing on the (FBA) SYSLNK file is done via this interface as these files have DIBs like the 'normal' system files (SYSIN, SYSLST, SYSPCH).

Whenever one of these files is to be OPENed or initialized (procedure library is not OPENed) $JOBCTLA requests GETVIS space for I/O buffers and IORB and CCW chains, and it updates the DIB and DIB extension for that file. (See description of DIB and DIB extension in the Appendix.)

When during blocking/deblocking a buffer is exhausted, $IJBFBA performs I/O via SVC 103 if SYSFIL support is available for that file (e.g. SYSPCH) or directly via SVC 0 if no SYSFIL support is available (SYSLNK or procedure library and SYSFIL=NO specified in FOPT).



ENTRY CONVENTIONS: The module $IJBFBA is loaded into the SVA by IPL. Job control issues a load command (SVC 4) for the module $IJBFBA to get its load address returned and transfers control to the module $IJBFBA by a branch and link instruction. Register 14 is used as link register.

ENTRY REGISTERS: Register 0 contains the value zero to indicate that the module $IJBFBA has been entered from the CALL/RETURN interface.

Register 1 contains the CCB address (logical unit = SYSRDR, SYSIPT, SYSPCH, SYSLST, or SYSLNK).

Register 5 contains the DIB address.

Register 13 contains the address of a 64-byte save area to save the caller's registers.

Register 14 is the link register between JCL and IJBFBA.

$JOBCTLB, CHART 09

This phase prepares a partition for restarting a checkpointed job. It

-- finds the checkpoint records

-- checks if the resources are still the same

-- restores control blocks and areas

-- calls phase $$BRSTRT or cancels with an appropriate error message.

$JOBCTLC, CHART 09.1

This phase opens the hard copy file. If SET HC=CREATE has been specified, it creates and formats the Hard Copy file. The CRTSAV table will be initialized. The message "DOS/VS IPL SUCCESSFULLY COMPLETED" will be written to the hard copy file. At processing of the ROD command the current buffers for a blocked hard copy file are written.

$JOBCTLD, CHARTS 10 TO 13

This phase contains the processing routines for the following control statements or commands:

ASSGN
CLOSE

$JOBCTLE, CHART 14

This phase contains processing routines for the EXEC statement or command.

$JOBCTLF, CHART 15

This phase contains the processing routines for the following control statements or commands:

DVCDN
DVCUP
LISTIO
RESET
UNBATCH
MAP

$JOBCTLG, CHARTS 16 AND 17

This phase contains the processing routines for the following control statements or commands:

CANCEL
/& (EOJ)
/+ (EOP)
JOB
OPTION

$JOBCTLJ, CHARTS 18 TO 20

This phase contains processing routines for the following control statements or commands:

ACTION
ALLOC
ALLOCR
CATALR
DATE
ENTRY
HOLD
INCLUDE
LOG
MTC
NOLOG
PAUSE
PHASE
SET
SIZE
STDOPT
STOP
UCS
UPSI
ZONE

$JOBCTLK, CHARTS 21 AND 22

This phase contains processing routines for the following control statements or commands:

DLAB
DLBL
EXTENT
LBLTYP
RSTRT
SETPRT
TLBL
TPLAB
VOL
XTENT

$JOBCTLM, CHART 23

This phase opens the recorder file. If SET RF=CREATE has been specified, it creates and formats the recorder file. The RF-table will be initialized. For a 3031 CPU, the frame records will be written to the recorder file. This phase contains processing routines for the ROD command.

$JOBCTLN, CHART 24

This phase provides interface between the DOS/VS system and $JOBACCT to allow the user to access job accounting information. If POWER/VS-E is running, and controls the partition, it provides also an interface between the DOS/VS-E system and POWER/VS-E to allow creation of an execution account record. For more information on POWER/VS-E, see the logic documentation of that Licensed Program.

$$BLSTIO

This B-transient routine contains subroutines used by the DVCDN and LISTIO processors of $JOBCTLF. When required by these processors $$BLSTIO is fetched (SVC 2) into the supervisor B-transient area.

$IJBSLA, CHARTS 25 TO 37

OVERVIEW

This phase has the following functions which correspond to routines of the same name:

ADDLBL write a label
ADDNXL write additional label information
ENDLBL indicate end of write operations
REPLBL replace modified label information
CLRGRPL clear a group of labels
GETLBL read a label
GETNXL read additional label information
LOCGRPL locate a group of labels
GETNXGL get next label from located group

The root-routine IJBSLA, with the help of two service routines:

INITALL and
INITPART

takes care of initialization, calls one of the function routines listed above, and prepares the exit of the program.

The external interface for programs which want to read or update any label information consists of two macros, LABEL and LPL, which are described in the DOS/VS Supervisor Logic manual.

DESCRIPTION OF LABEL FUNCTIONS

The ADDLBL Function

This function inserts a label from a buffer into a label subarea, both specified in the LPL. Before any label can be added to a label subarea a CLRGRPL function must have been issued against this label subarea to clear it. The CLRGRPL function has to be issued in the same partition as the ADDLBL function.

Contents of LPL:

AREA,AREALEN specify the location and length of the label that is to be inserted into the label area.

FILENAME is the file name contained in the label.

GROUP = PARTITION specifies that the label is to be inserted into a label subarea of the partition.

GROUP = SYSTEM specifies that the label is to be inserted into the system label subarea.

STORE = TEMP specifies that the label is to be inserted into the temporary label subarea of the partition.

STORE = PERM specifies that the label is to be inserted into the permanent label subarea of the partition.

The parameter STORE is not needed and is ignored if GROUP = SYSTEM is specified.

Return codes:

0 Successful request. The label is stored into the specified area.
12 The function does not follow a CLRGRPL function or an other ADDLBL function for the same label subarea.
20 Contents of LPL are invalid.
24 No more space available in label area.

For all return codes other than 0 no label is stored.

The ADDNXL Function

This function is used to insert additional label information into the label area. In DOS/VS, job control produces one label for each '// EXTENT' or '// XTENT' statement for sequential files. These labels contain the same 'DLBL' information (e.g. filename) but differ in their 'EXTENT' information. As DOS/VS must support this function, additional label information is inserted to a label area by the ADDNXL function and accessed by the GETNXL function.

An ADDNXL function must therefore always be preceded by an ADDLBL or may follow an other ADDNXL function. This must be true within one partition.

Contents of LPL:

AREA,AREALEN specify the location and length of the additional label information that is to be stored behind the label specified before.

FILENAME is the file name contained in the label. All other parameters are ignored.

Return codes:

0 Successful request. The label information is stored.
12 Within the partition, this function is not preceded by an ADDLBL or ADDNXL function with the same file name.
20 Contents of the LPL are invalid.
24 No more space available in label area.

For all return codes other than 0, no label information is stored.

The ENDLBL Function

In case of an FBA device, the label blocks are written on the device only if the block is completely filled up with labels. If the block should be written before it is filled up (because no more labels will be added), SLA has to be informed via the ENDLBL function.

Contents of LPL:

The LPL information is not used.

Return codes:

0 Successful request, the labels have been written on FBA.

The REPLBL Function

This function can be used to modify a label that was fetched before by a GETLBL or GETNXL function. It is not possible to change the length of the label or its logical or physical data set name.

The REPLBL function must be preceded by a GETLBL or GETNXL function for the same file label. REPLBL and its preceding function have to be issued within the same task.

Contents of LPL:

Same as for the GETLBL/GETNXL function.

Return codes:

0 Successful request, the label is replaced.
12 The replacement is not for a label just retrieved or changes fields which are not allowed to be changed.
20 Contents of LPL are invalid.

The CLRGRPL Function

With this function, a group of labels can be deleted from the label area.

Contents of LPL:

GROUP = PARTITION specifies that a label subarea of the partition is to be deleted.

GROUP = SYSTEM specifies that the system label subarea is to be deleted.

STORE = TEMP specifies that the temporary label subarea of the partition is to be deleted.

STORE = PERM specifies that the permanent label subarea of the partition is to be deleted.

The parameter STORE is not needed and is ignored if GROUP = SYSTEM is specified. All other parameters of the LPL are also ignored.

Return codes:

0 Successful request. All labels of the specified label subarea are deleted.
20 Contents of LPL are invalid.

For return code 20 no label is deleted.

The GETLBL Function

This function is used to fetch a label from a label subarea. The label is returned in the buffer specified in the LPL and it is either a label of the partition's temporary label subarea, the partitions' permanent label subarea or the system label area, wherever the label is found first.

Contents of LPL:

AREA,AREALEN specify the buffer where the label is to be stored. FILENAM is the search argument for the label. All other parameters of the LPL are ignored.

Return codes:

0 Successful request. The buffer contains the specified label.
4 The specified label does not exist.
8 The buffer length is smaller than the length of the label. But as much of the label as fits in the buffer is moved into it.
20 Contents of LPL are invalid.
28 No GETVIS storage available.

For all return codes other than 0 and 8, the contents of the buffer remain unchanged.

For return codes 0 and 8, the two-byte field LPLLBLEN in the LPL will contain a binary value, specifying the length of the label.

For return code 8, the buffer will contain the first part of the label.

The GETNXL Function

This function is needed if a label information of a sequential file consists of several parts (extents). This function can only follow a GETLBL function or an other GETNXL function, i.e. the first or

only part of a label is always fetched by a GETLBL function and only if the extent information in the label indicates that more extents exist, which are not part of the label just fetched, then the additional information can be obtained by the GETNXL function. (See also description of ADDNXL function.)

GETNXL and preceding GETLBL function have to be issued within the same task.

Contents of LPL:

Same as for the GETLBL function.

Return codes:

0 Successful request. The buffer contains the specified label.
4 No additional information does exist for this label.
8 The buffer length is smaller than the length of the label. But as much of the label as fits in the buffer is moved into it.
12 Within the task, the function is not preceded by a GETLBL or another GETNXL function for the same file label.
20 Contents of LPL are invalid.

For all return codes other than 0 and 8, the contents of the buffer remain unchanged.

For return codes 0 and 8, the two-byte field LPLLBLEN in the LPL will contain a binary value, specifying the length of the label.

For return code 8, the buffer will contain the first part of the label.

The LOCGRPL Function

This function is used to indicate to the label access routine (SLA) that the calling program (normally LSERV) intents to display the contents of the label subarea specified in the LPL.

This function cannot be invoked by more than one task within the same partition.

No actual data transfer occurs. After this function is issued the calling program can retrieve labels of the specified label subarea in their physical sequence by issuing GETNXGL functions (see description below).

Contents of LPL:

GROUP = pn specifies, that a label subarea of a specific partition is to be accessed. pn can be BG | F1 | F2 | etc.

GROUP = SYSTEM specifies, that the system label subarea is to be accessed.

STORE = TEMP specifies, that the temporary label subarea of the partition pn is to be accessed.

STORE = PERM specifies, that the permanent label subarea of the partition pn is to be accessed.

The parameter STORE is not needed and is ignored if GROUP = SYSTEM is specified. All other parameters of the LPL are also ignored.

Return codes:

0 Successful request. The SLA is ready to accept GETNXGL functions for the label subarea specified in the LPL.
4 The specified label subarea is empty.
20 Contents of LPL are invalid.
32 Updating in progress: Area not available.

For all return codes other than 0, a succeeding GETNXGL function cannot be executed successfully.

The GETNXGL Function

This function treats a label subarea as a sequential file of labels. For each request, the function retrieves the label next in sequence from the label subarea which was specified by a preceding LOCGRPL function and stores it into the buffer specified in the LPL.

Contents of LPL:

AREA,AREALEN specify the buffer where the label is to be stored. All other parameters are the same as in the LPL specified with the preceding LOCGRPL function.

Return codes:

0 Normal request. The buffer contains the label next in sequence.
4 No more records in the label subarea.
8 The buffer length is smaller than the length of the label.

12 The function is not preceded by a LOCGRPL function or another GETNXGL function for the same label subarea.
20 Contents of LPL are invalid.
32 Updating in progress: Area not available.

For all return codes other than 0, no label is stored and no further label can be retrieved from the label subarea specified.

JC CONTROL FLOW

Figure 10 shows the control flow of the job control program.

The root phase, $JOBCTLA, can call seven different command processing phases:

$JOBCTLD, E, F, G, J, K, and M. The phase $JOBCTLN has service functions for the following call conditions:

called by A - when the $JOBACCT user routine was canceled,
E - for all but the first EXEC statements of a job,
G - at end-of-job (/&) and simulated EOJ,
J - for a PAUSE EOJ statement.

The phase provides an interface to job accounting and POWER/VS-E.

The phase $JOBCTLK calls a subphase $JOBCTLB. Phase $JOBCTLF calls the B-transient $$BLSTIO to process the LISTIO and DVCDN commands. Phase $JOBCTLM calls a subphase $JOBCTLC.

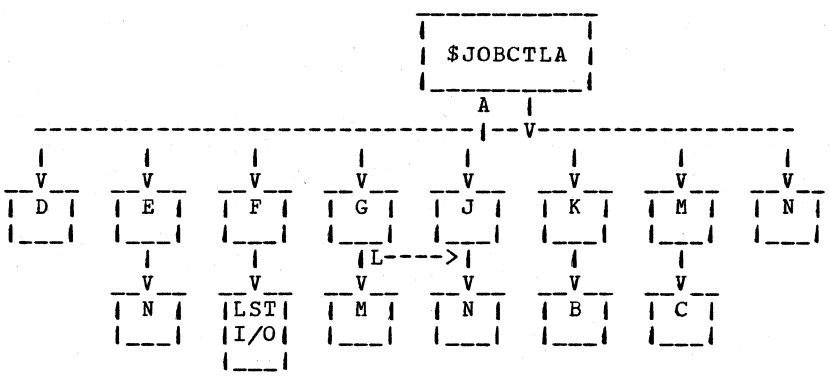


Figure 10. Job Control Program Control Flow

Figure 11 shows the control flow of the SLA program.

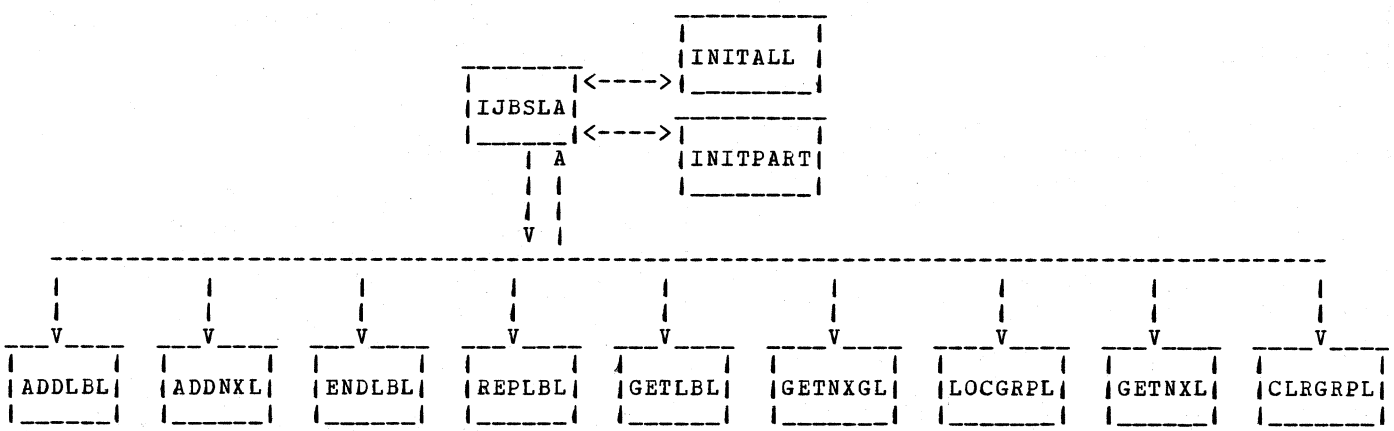


Figure 11. SLA Program Control Flow

JOB CONTROL STORAGE MAP

The job control program may run in the background or a foreground partition. Figure 12 shows how this partition is divided.

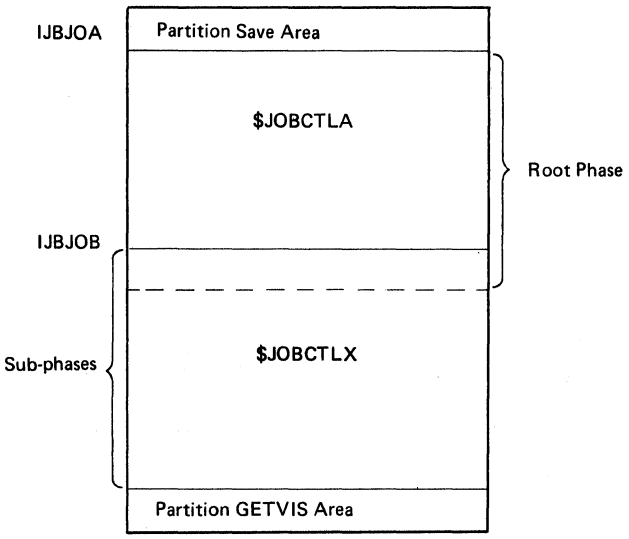


Figure 12. Job Control Storage Map

Note: The part of $JOBCTLA from IJBJOB to the end is only used for initialization purposes when $JOBCTLA is loaded into the partition.

LABEL AREA ORGANIZATION

The label area is divided into label area segments called LASes. For CKD-devices, one LAS is identical to one track. For FBA devices, one LAS has 2K bytes and, with a blocksize of 512 bytes, this contains four blocks.

The LASes are addressed by sequential numbers starting from zero. There is no fixed allocation of LASes to groups of labels. The use of LASes is organized by a label area control block called LACB. It indicates which label groups use which LASes or if a LAS is not in use. The LACB is stored in LAS number zero.

When the first label is to be written, the first free LAS is used. The LAS is then filled up by labels of the same group and if one LAS is not enough for the label group, several LASes are chained together. When a label group is cleared, the LAS or LASes are given back to the pool of free LAS's.

DESCRIPTION OF LACB

LACB Constant

The first four bytes of the LACB contain the four characters 'LACB' to indicate that this is a valid control block.

LAS Pool

The next 248 bits represent each LAS of the the system by one bit, the position of the bit showing the sequential number of the LAS in question and the status showing, if it is available.

Bit is 0: the LAS is used or not existing
Bit is 1: the LAS is free.

When the LACB is initialized, as many bits as there are LASes in the label area are switched ON in the general LAS pool, and the first bit is set OFF again, because LAS number 0 is used to store the LACB itself.

Label Group Entries

Each group of labels has one entry in the LACB:

Entry 1: SYSTEM STANDARD LABELS
Entry 2: BG PERMANENT
Entry 3: BG TEMPORARY
Entry 4: BG FREE USAGE
Entry 5: F1 PERMANENT
Entry 6: F1 TEMPORARY
Entry 7: F1 FREE USAGE

and so on: 3 entries for each partition.

Figure 13 shows the format of a label group entry in the LACB.

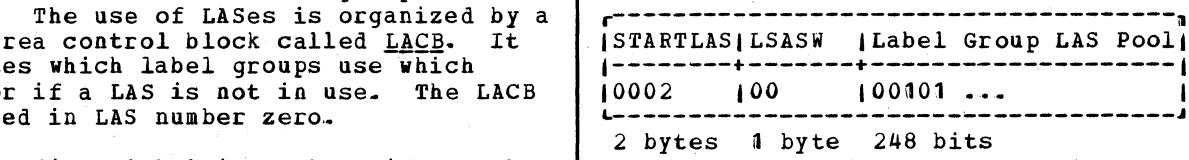


Figure 13. Format of a Label Group Entry

STARTLAS This field has two bytes and contains the sequential number of the first LAS occupied by the label group. If it contains

zeros the label group is empty.

LSASW — This field has one byte and contains FF if the label group is in write process. Otherwise it contains 00.

The rest of the entry is the label group's LAS pool and functions like the general LAS pool.

HANDLING OF THE LACB

How to occupy a free LAS

- In the LAS pool, all bits are scanned for the first bit which is ON ('1'). This bit represents the first free LAS of the label area.
- If the first '1' bit is found, it is switched to 0 to indicate that it is used now.
- In the entry of the group which requests the LAS, the bit with the same position is set to '1'.
- If it was the first LAS for the group of labels, its STARTLAS bytes in the LACB are changed from zero to the number of the LAS.
- If it was not the first LAS for the group, the chaining address in the preceeding LAS has to be updated.

How to free an occupied LAS

- A single label cannot be deleted, only the whole group of labels can be cleared.
- If a group of labels has to be cleared, all LAS bits are OR-ed into the LAS pool: all '1' bits of the entry are switched on in the LAS pool to indicate that the LASes are free again.
- Then the whole entry of the group is switched to '0', the START LAS bytes, the LSASW byte and all LAS bits.

ORGANIZATION OF LABEL AREA SEGMENTS

Contents of a Label Area Segment

The labels are written in the sequence of the DLBL/EXTENT commands.

For CKD label areas, each label is represented by a physical record with count key (8 bytes: first 7 bytes = filename, 8th byte = extent sequence number) and data (data = label information, variable length). For each label writing/reading, an I/O operation occurs. Retrieving a label is done by looking for the hardware key of the record (search key equal operation).

For FBA, the labels are gathered in the I/O area, until the size of an LAS is reached. Then an I/O operation occurs for the whole LAS. To retrieve a label, the whole LAS is read. The search within the LAS has to be done by program logic.

For internal organization of an LAS on FBA see next chapter.

Chaining of LASes

- If there is not enough space in one LAS for the labels of a group, several LASes are chained together.
- The number of the first LAS in a chain is contained in the STARTLAS bytes in the LACB. If these bytes are zero, the group of labels is empty (each group of labels has its own chain and start address).
- LASes are chained together with a chaining address which contains the sequential number of the next LAS in the chain.
- The chaining address is part of the LAS:
  - For FBA: first two bytes of each LAS (containing zero for last LAS in chain).
  - For CKD: each not-last-LAS in chain contains a chaining record as last record of the track. The chaining record is identified by a special 8-byte key and has a data length of one byte, containing the chaining address.

If there is no chaining record on track, the LAS is the last in chain.

Format of a LAS on FBA

Figure 14 shows the format of a LAS on FBA.

```
01 23 45 67 .... bytes
┌──────────────────────────┐ ┌──────┐
|CA LL L1 LAB1 L2 LAB2 L3 LAB3|---Ln LABn|
└──────────────────────────┘ └──────┘
 |  |  |
 |  |  L---2 bytes: label length
 |  L------2 bytes: LAS length used
 L---------2 bytes: chaining address
```

Figure 14. LAS Format on FBA

Chaining Address (at the beginning of each LAS)

- Zero, if it is the last LAS in chain.
- Non-zero: containing sequential number of next LAS in chain.

LAS length used: indicates how many bytes are used within the LAS (after chaining address of each LAS).

Label length: before each label, there are 2 bytes, containing the length of the label information (not including the length field).

For detailed layout of label see DOS/VS DASD Labels.

INITIALIZATION FOR SLA

PROVIDE SYSTEM GETVIS STORAGE

- SLA needs system GETVIS storage for work buffers and I/O buffers. A special area is installed in the SVA during IPL: When the SVA and the system GETVIS area is created, the highest storage area is reserved for SLA. This area is divided into:
  - a general workarea with key 0.
  - one workarea for each partition with the appropriate storage key in each area.
- Each of those areas has a size of 6K.
- The start address of the SLA area is stored in the 4-byte field IJBSLACB of the system communication region.
- The layout of the SLA GETVIS storage is shown in Figure 15.

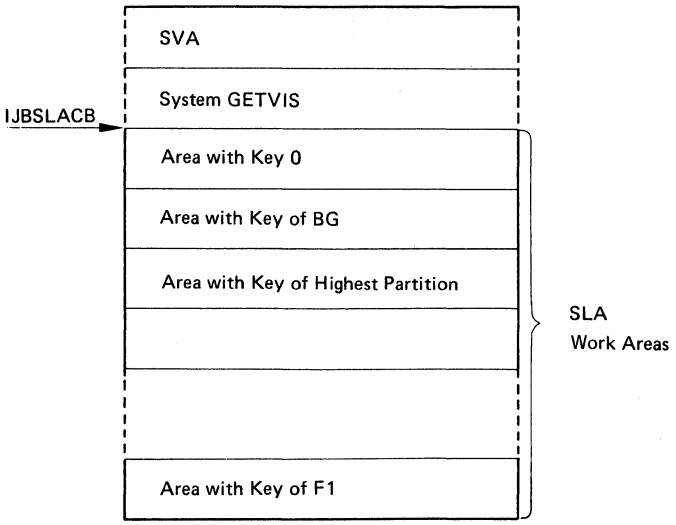


Figure 15. SLA GETVIS Storage Layout

STORE SYSTEM INFORMATION IN WORKAREA

IPL also stores the information about the physical location of the label area and the procedure library for SLA and JCL in a workarea. MAINT modifies this information if necessary.

The address of this workarea is contained in the four byte field 'IJBSLACB' in the system communication region.

This workarea has protection key zero. This means that all components which have to store or modify something in the area, also have to have key 0.

The Layout in the Information Area is as follows:

1. For FBA:

Procedure Library

0- 3	NNNN	Start Address of Procedure Library	)
4- 7	NNNN	Relative displacement to first block	) *)
8-11	NNNN	Relative displacement to last block	)

Label Information Area

12-15	NNNN	Start Address of Label Information Area	
16-19	NNNN	Relative displacement to first block	
20-23	NNNN	Relative displacement to last block	*)
24-29	NNNNNN	Number of partitions of supervisor last IPLed (bytes 28, 29)	

*) corresponds to bytes 36-65 of the FBA system directory block

2. For CKD:

Procedure Library

0- 6	BBCCHHR	Starting address of the procedure library	**)

12-13	NN	Number of label cylinders on this device	
14-15	CC	Address of the file definition storage cylinder(s)/label cylinder(s)	***)
24-29	NNNNNN	Number of partitions of supervisor last IPLed (bytes 28, 29)	****)

**) corresponds to bytes 0-6 of the system directory record no. 4

***) corresponds to bytes 76-79 of the system directory record no. 1

****) corresponds to bytes 60-65 of the system directory record no. 4

I/O FLOW OF THE JOB CONTROL PROGRAM

The job control program uses the following logical I/O units:

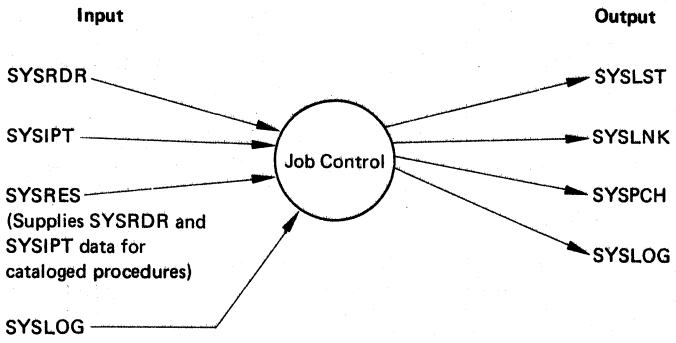
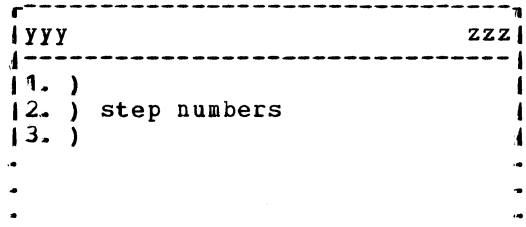


Figure 16. Job Control I/O Flow

GENERAL CHARTS

GENERAL CHARTS CONVENTIONS:

1. A unit of programming, routine, CSECT, or phase, is contained in one box like this:



Where: xxx marks the label and routine name
yyy says shortly what the routine does
zzz is the reference to the detail chart(s)

2. On-page connectors are such:

3. Off-page connectors are such:

Where: The number in the frame marks the chart from or to which we go.

The word above (incoming) or below (outgoing) marks the label (routine) on that chart.

The number under the word marks the step within the routine to which we go if it is not step 1.

Chart 01. IPL Bootstrap Processing (AA-AD)

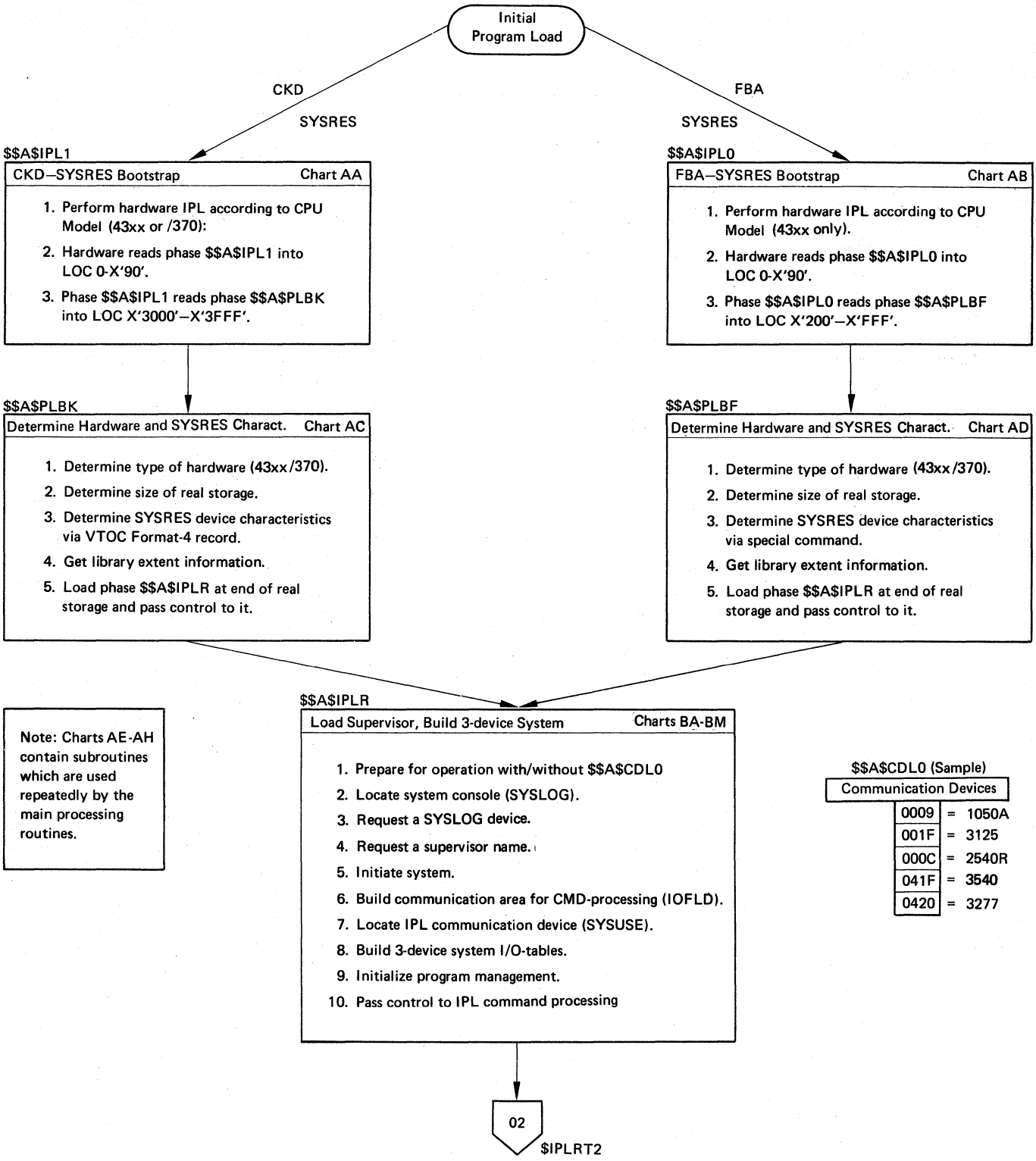
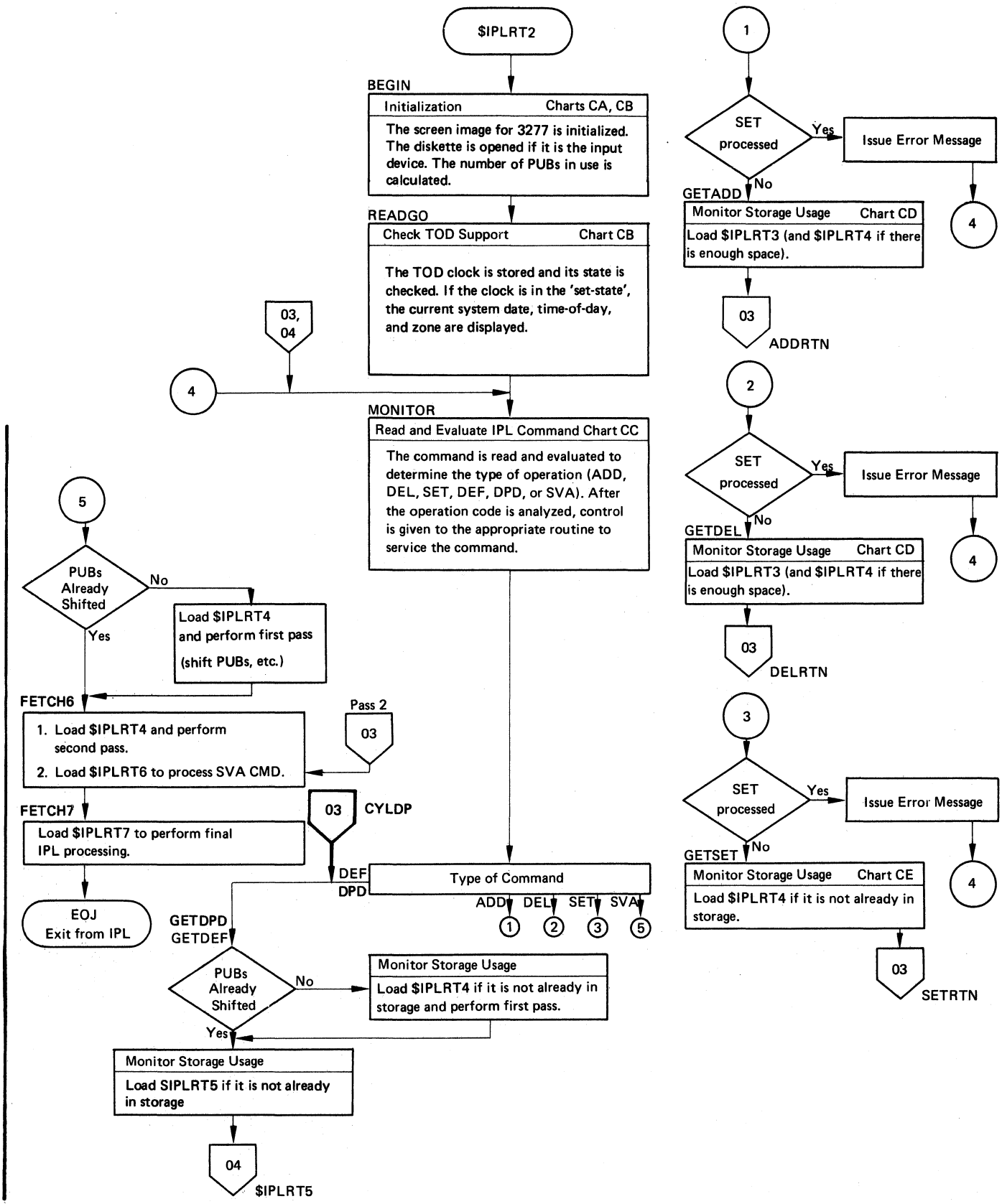


Chart 02. $IPLRT2 - Command Processing Root (CA-CL)



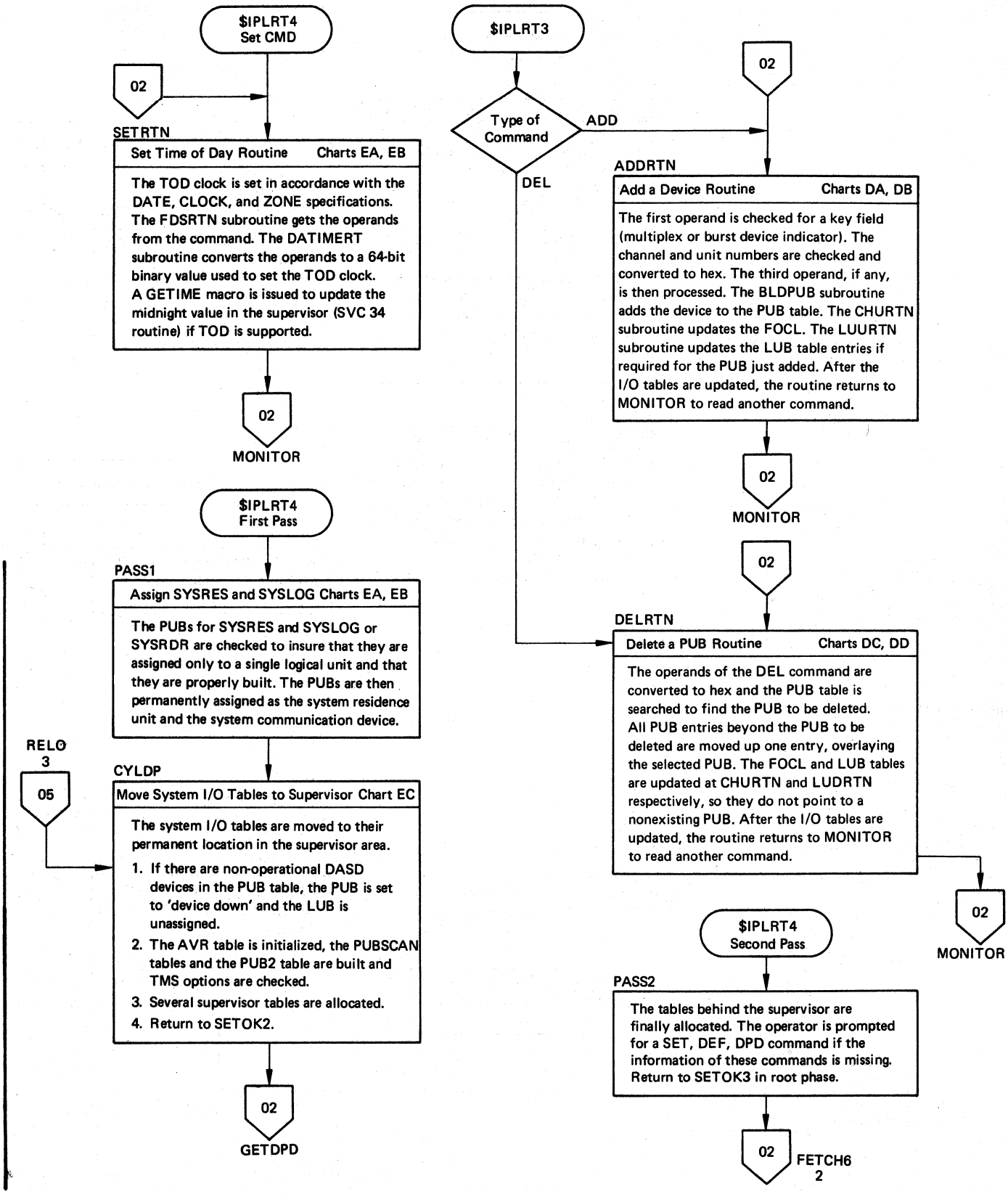
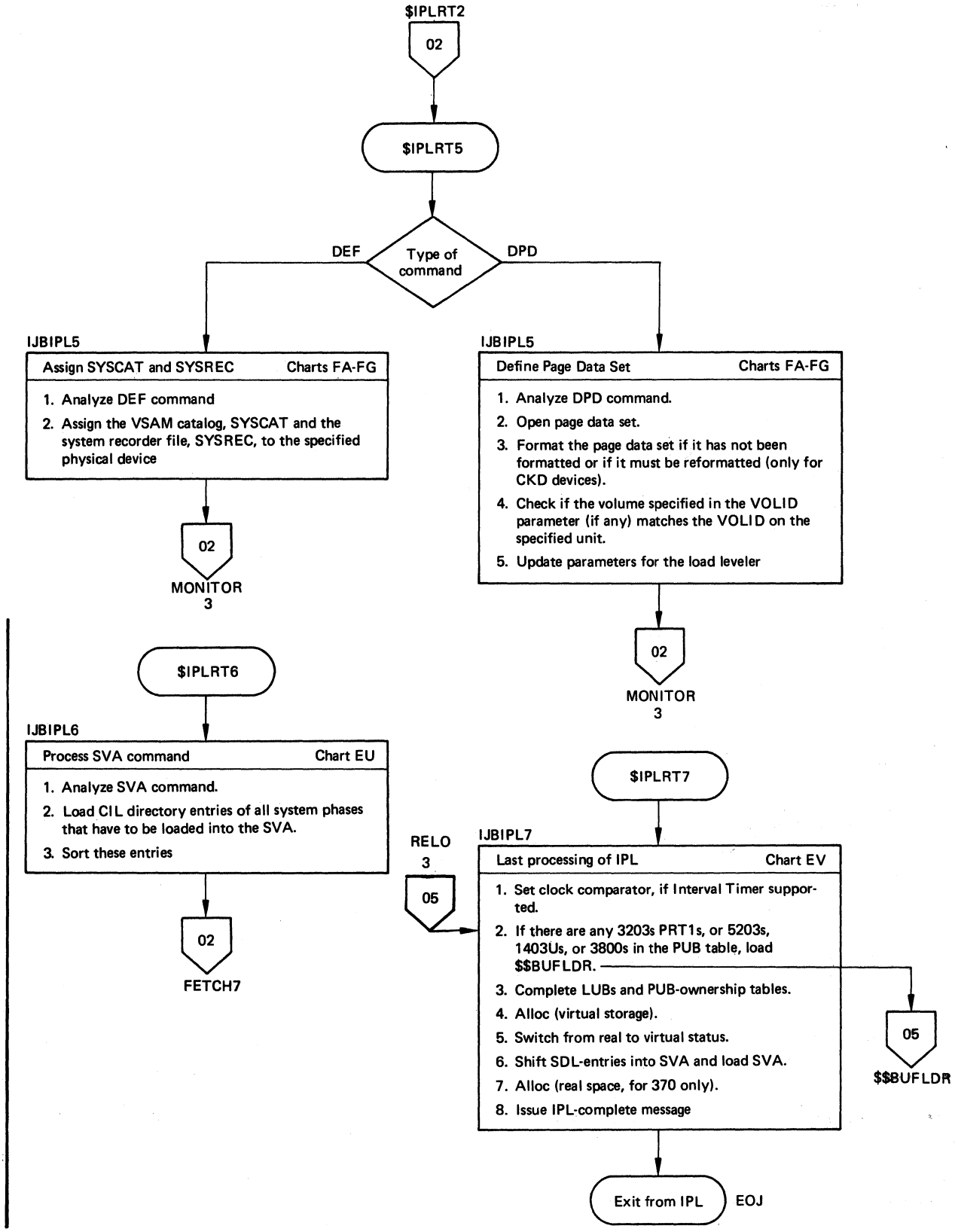

$IPLRT4
Set CMD
02
SETRTN
Set Time of Day Routine Charts EA, EB
The TOD clock is set in accordance with the DATE, CLOCK, and ZONE specifications. The FDSRTN subroutine gets the operands from the command. The DATIMERT subroutine converts the operands to a 64-bit binary value used to set the TOD clock. A GETIME macro is issued to update the midnight value in the supervisor (SVC 34 routine) if TOD is supported.
02
MONITOR
$IPLRT4
First Pass
PASS1
Assign SYSRES and SYSLOG Charts EA, EB
The PUBs for SYSRES and SYSLOG or SYSRDR are checked to insure that they are assigned only to a single logical unit and that they are properly built. The PUBs are then permanently assigned as the system residence unit and the system communication device.
RELO
3
05
CYLDP
Move System I/O Tables to Supervisor Chart EC
The system I/O tables are moved to their permanent location in the supervisor area.
1. If there are non-operational DASD devices in the PUB table, the PUB is set to 'device down' and the LUB is unassigned.
2. The AVR table is initialized, the PUBSCAN tables and the PUB2 table are built and TMS options are checked.
3. Several supervisor tables are allocated.
4. Return to SETOK2.
02
GETDPD
$IPLRT3
Type of Command
ADD
DEL
02
ADDRTN
Add a Device Routine Charts DA, DB
The first operand is checked for a key field (multiplex or burst device indicator). The channel and unit numbers are checked and converted to hex. The third operand, if any, is then processed. The BLDPUB subroutine adds the device to the PUB table. The CHURTN subroutine updates the FOCL. The LUURTN subroutine updates the LUB table entries if required for the PUB just added. After the I/O tables are updated, the routine returns to MONITOR to read another command.
02
MONITOR
02
DELRTN
Delete a PUB Routine Charts DC, DD
The operands of the DEL command are converted to hex and the PUB table is searched to find the PUB to be deleted. All PUB entries beyond the PUB to be deleted are moved up one entry, overlaying the selected PUB. The FOCL and LUB tables are updated at CHURTN and LUDRTN respectively, so they do not point to a nonexisting PUB. After the I/O tables are updated, the routine returns to MONITOR to read another command.
02
MONITOR
$IPLRT4
Second Pass
PASS2
The tables behind the supervisor are finally allocated. The operator is prompted for a SET, DEF, DPD command if the information of these commands is missing. Return to SETOK3 in root phase.
02
FETCH6
2

Chart 04. $IPLRT5, $IPLRT6, $IPLRT7 - DEF, DPD, SVA Command Processing and IPL Termination (EU-FG)



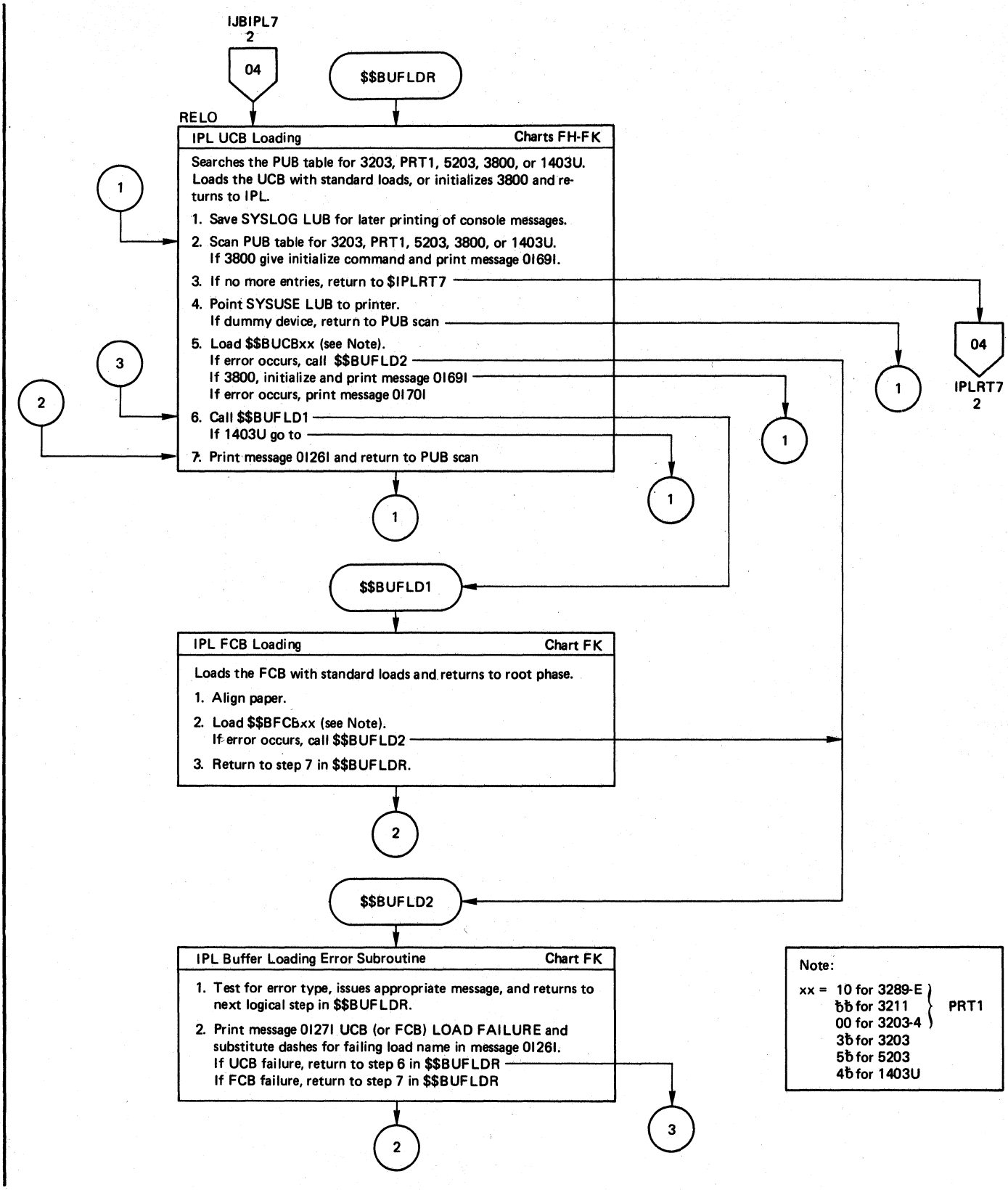
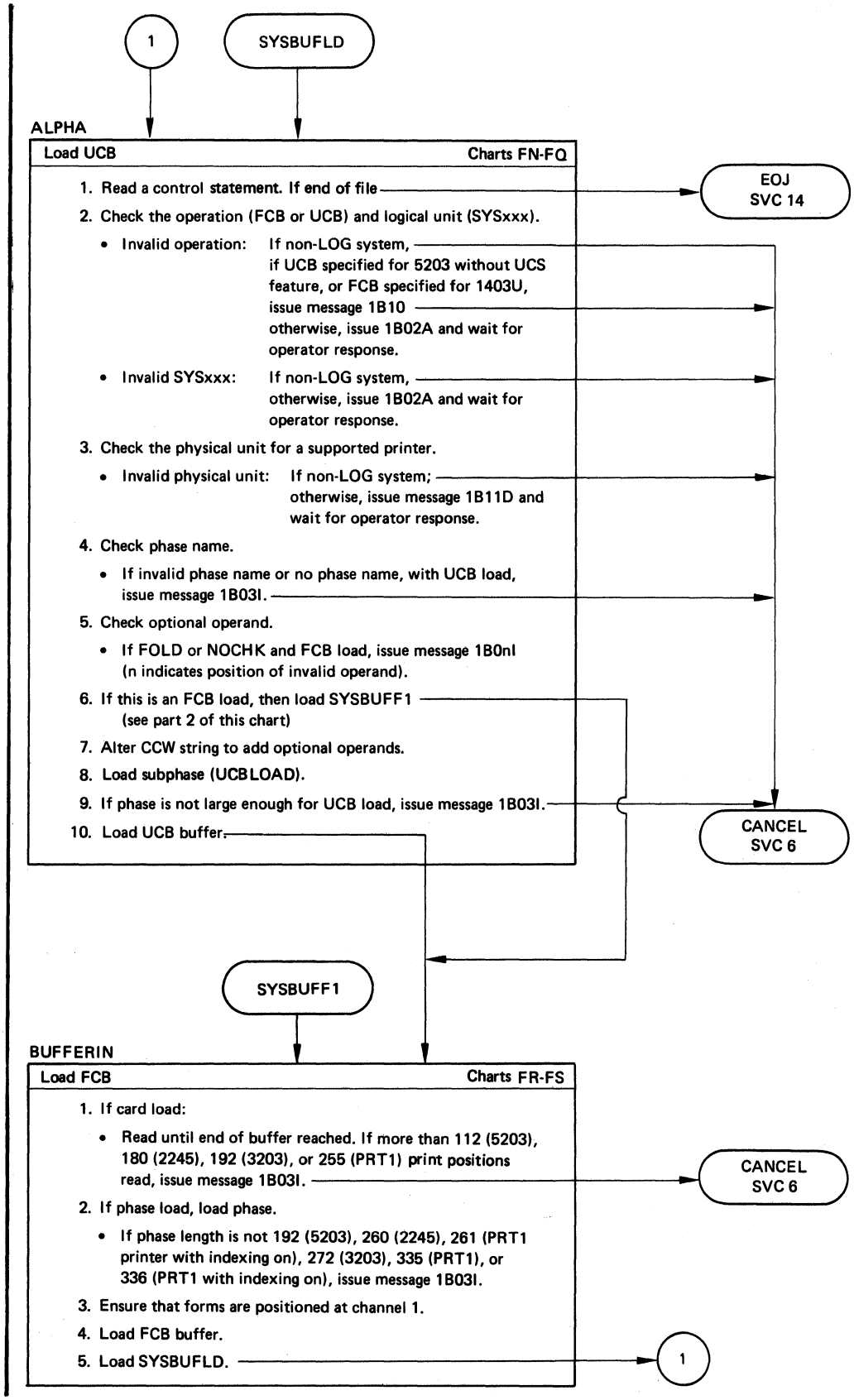

IJBIPL7
2
04
$$BUFLDR
RELO
IPL UCB Loading
Charts FH-FK
Searches the PUB table for 3203, PRT1, 5203, 3800, or 1403U. Loads the UCB with standard loads, or initializes 3800 and returns to IPL.
1. Save SYSLOG LUB for later printing of console messages.
2. Scan PUB table for 3203, PRT1, 5203, 3800, or 1403U. If 3800 give initialize command and print message 01691.
3. If no more entries, return to $IPLRT7
4. Point SYSUSE LUB to printer. If dummy device, return to PUB scan
5. Load $$BUCBxx (see Note). If error occurs, call $$BUFLD2 If 3800, initialize and print message 01691 If error occurs, print message 01701
6. Call $$BUFLD1 If 1403U go to
7. Print message 01261 and return to PUB scan
1
2
3
04
IPLRT7
2
$$BUFLD1
IPL FCB Loading
Chart FK
Loads the FCB with standard loads and returns to root phase.
1. Align paper.
2. Load $$BFCBxx (see Note). If error occurs, call $$BUFLD2
3. Return to step 7 in $$BUFLDR.
$$BUFLD2
IPL Buffer Loading Error Subroutine
Chart FK
1. Test for error type, issues appropriate message, and returns to next logical step in $$BUFLDR.
2. Print message 01271 UCB (or FCB) LOAD FAILURE and substitute dashes for failing load name in message 01261. If UCB failure, return to step 6 in $$BUFLDR If FCB failure, return to step 7 in $$BUFLDR
Note:
xx = 10 for 3289-E
bb for 3211
00 for 3203-4
PRT1
3b for 3203
5b for 5203
4b for 1403U

Chart 06. SYSBUFLD - Load Print Buffers Program (FM-FS)



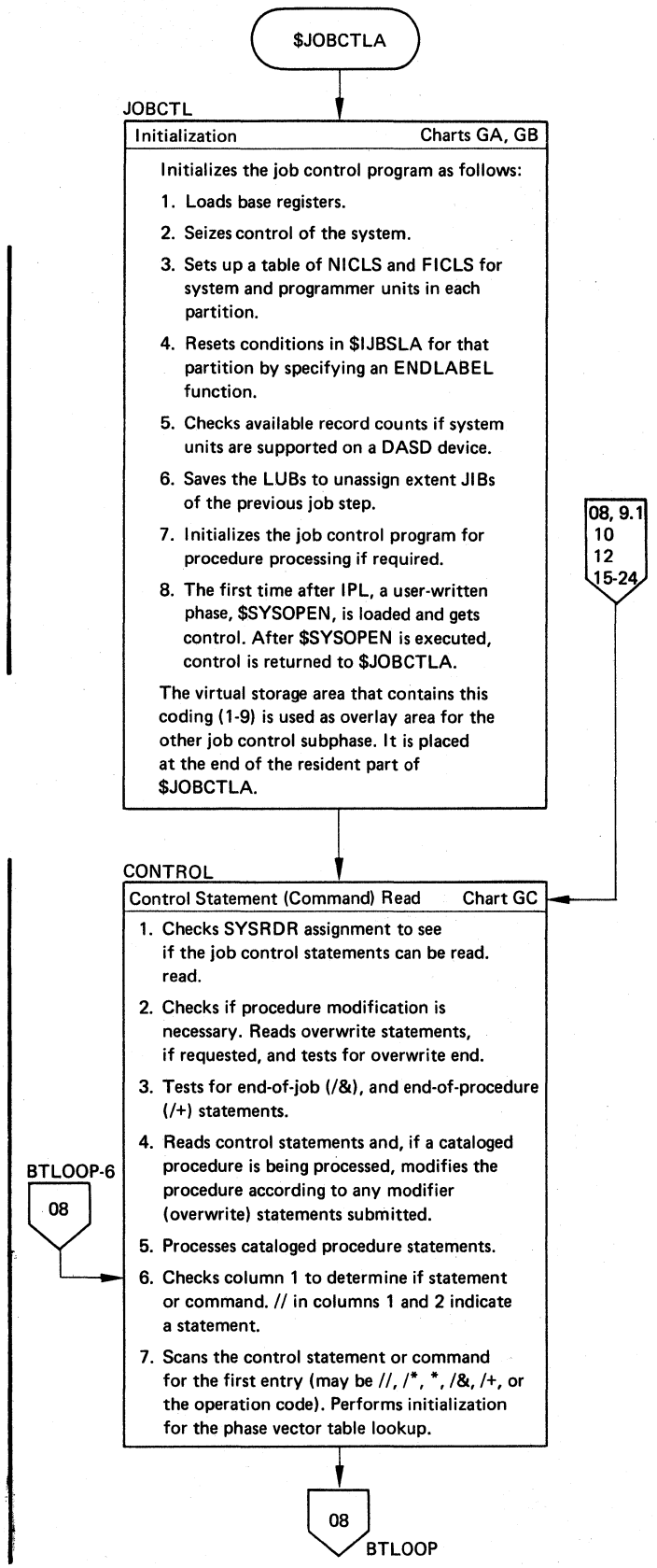
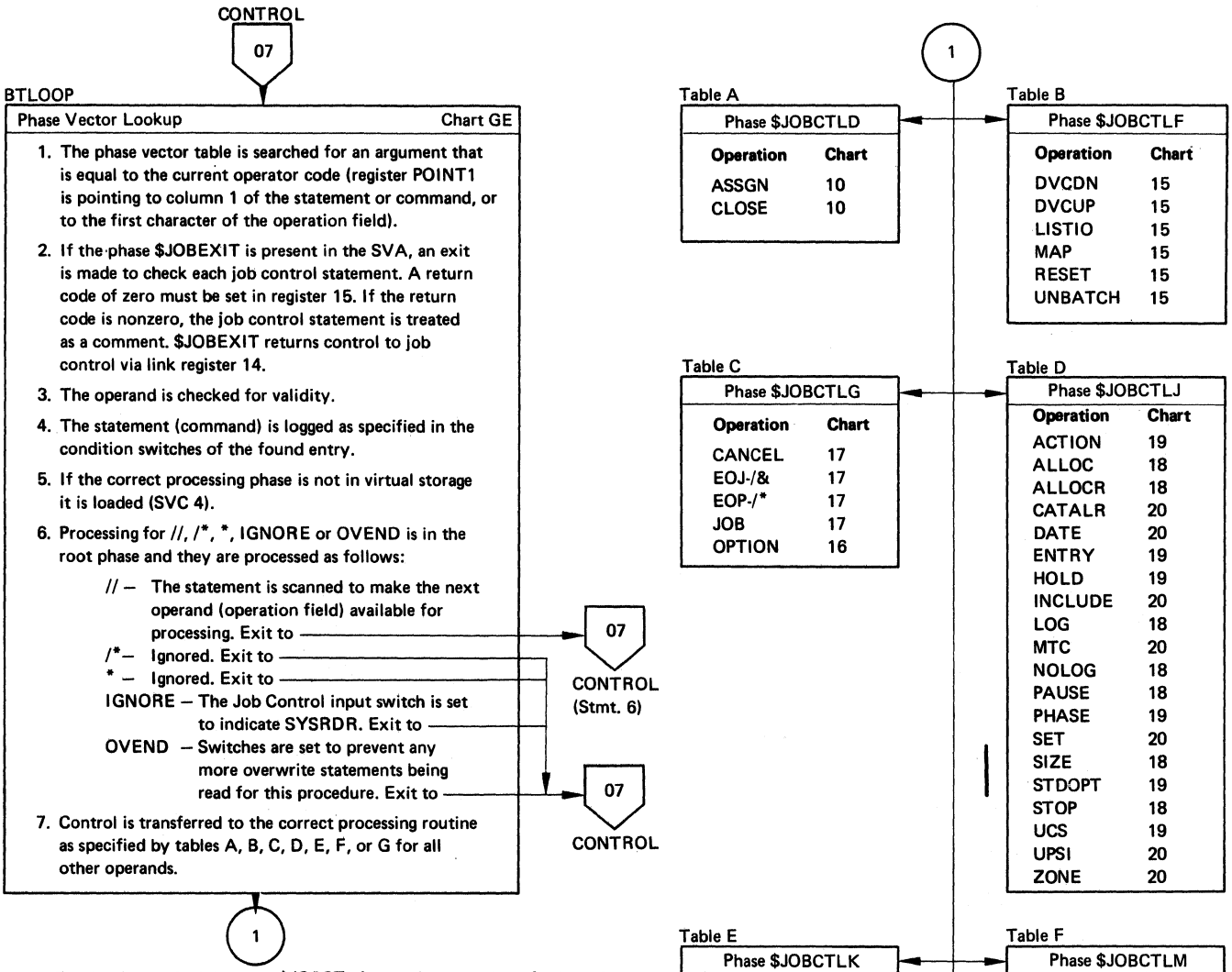

$JOBCTLA
JOBCTL
Initialization
Charts GA, GB
Initializes the job control program as follows:
1. Loads base registers.
2. Seizes control of the system.
3. Sets up a table of NICLS and FICLS for system and programmer units in each partition.
4. Resets conditions in $IJBSLA for that partition by specifying an ENDLABEL function.
5. Checks available record counts if system units are supported on a DASD device.
6. Saves the LUBs to unassign extent JIBs of the previous job step.
7. Initializes the job control program for procedure processing if required.
8. The first time after IPL, a user-written phase, $SYSOPEN, is loaded and gets control. After $SYSOPEN is executed, control is returned to $JOBCTLA.
The virtual storage area that contains this coding (1-9) is used as overlay area for the other job control subphase. It is placed at the end of the resident part of $JOBCTLA.
08, 9.1
10
12
15-24
CONTROL
Control Statement (Command) Read
Chart GC
1. Checks SYSRDR assignment to see if the job control statements can be read. read.
2. Checks if procedure modification is necessary. Reads overwrite statements, if requested, and tests for overwrite end.
3. Tests for end-of-job (/&), and end-of-procedure (/+) statements.
4. Reads control statements and, if a cataloged procedure is being processed, modifies the procedure according to any modifier (overwrite) statements submitted.
5. Processes cataloged procedure statements.
6. Checks column 1 to determine if statement or command. // in columns 1 and 2 indicate a statement.
7. Scans the control statement or command for the first entry (may be //, /*, *, /&, /+, or the operation code). Performs initialization for the phase vector table lookup.
BTLOOP-6
08
08
BTLOOP

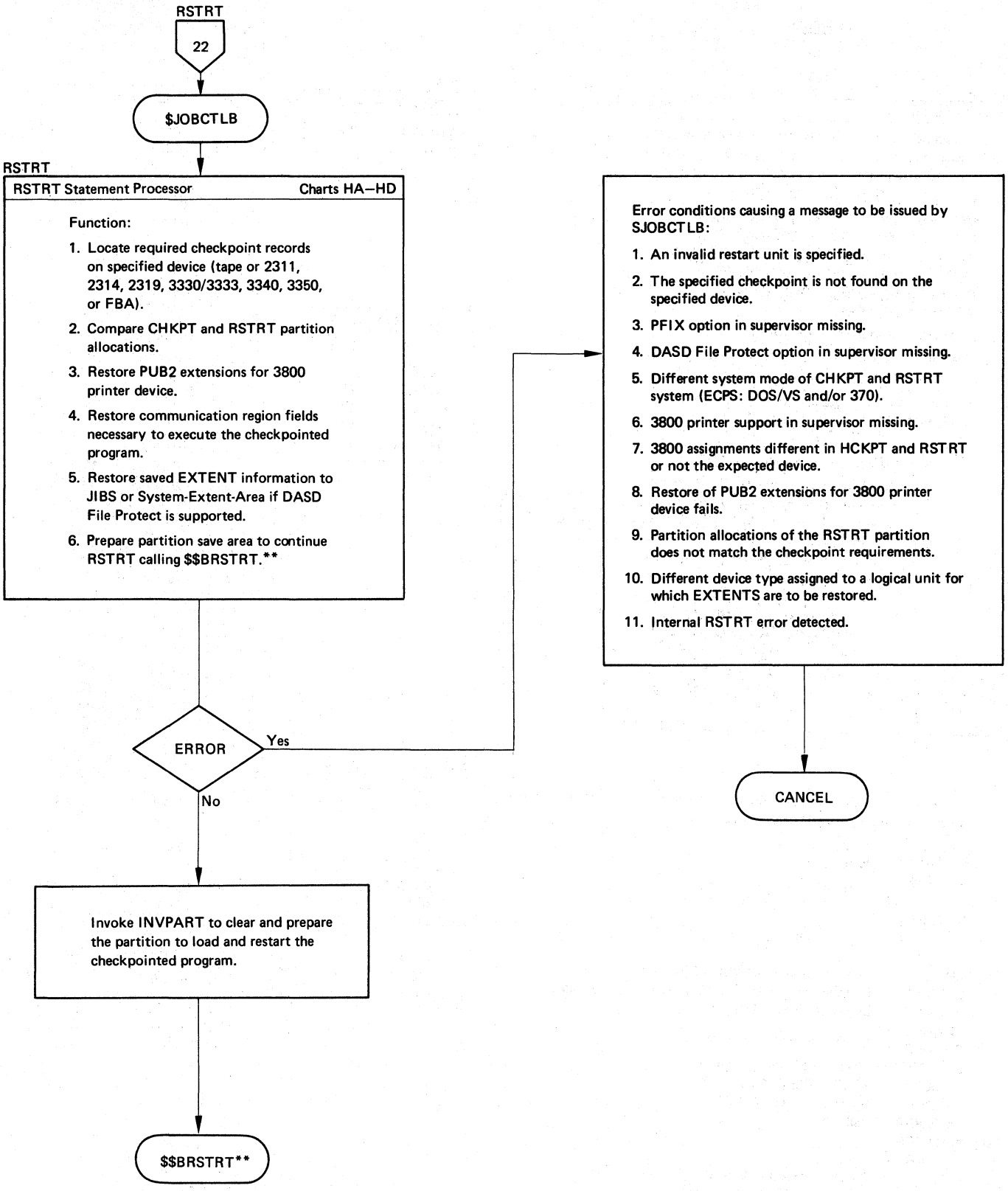
Chart 08. $JOBCTLA (Part 2 of 2) - Job Control Root Phase (GA-GY)



Besides these principle parts, $JOBCTLA contains a number of subroutines which are called by the rest of job control:

RLINDT	Read a statement
EXCPRG	Perform I/O
SCANR2, SCANR3	Scan for operand in statement
NVSERR, ERRRTN, OERRTN	Error routines
LOGOUT	Log a statement on SYSLOG (if not logged already)
LSTOUT	List a message/statement on SYSLST (if not listed already)
MSGOUT	Log a message on SYSLOG
CHKASG, CHKASG3	Check if a SYSnnn is assigned
LOGCHK	Check for SYSLOG device type
MTNCNT	SEIZE/RELEASE control
IGNORE	Switch input to SYSRDR
FETCHRF	Call subphase
WAITERR, SYSERR	Error routines
OVEND	Process OVEND statement
SETAPRT	Call 3800 buffer load routines
INITDIB	Initialize FBA DIB
CONTROL2	Process a generated statement
ERRRTN0	Error routine

Chart 09. $JOBCTLB - Restart Processor (HA-HD)



** See DOS/VS Logical Transients and Dump Phases Logic.

Chart 09.1 $JOBCTLC - Write Hard Copy File (HM-HS)

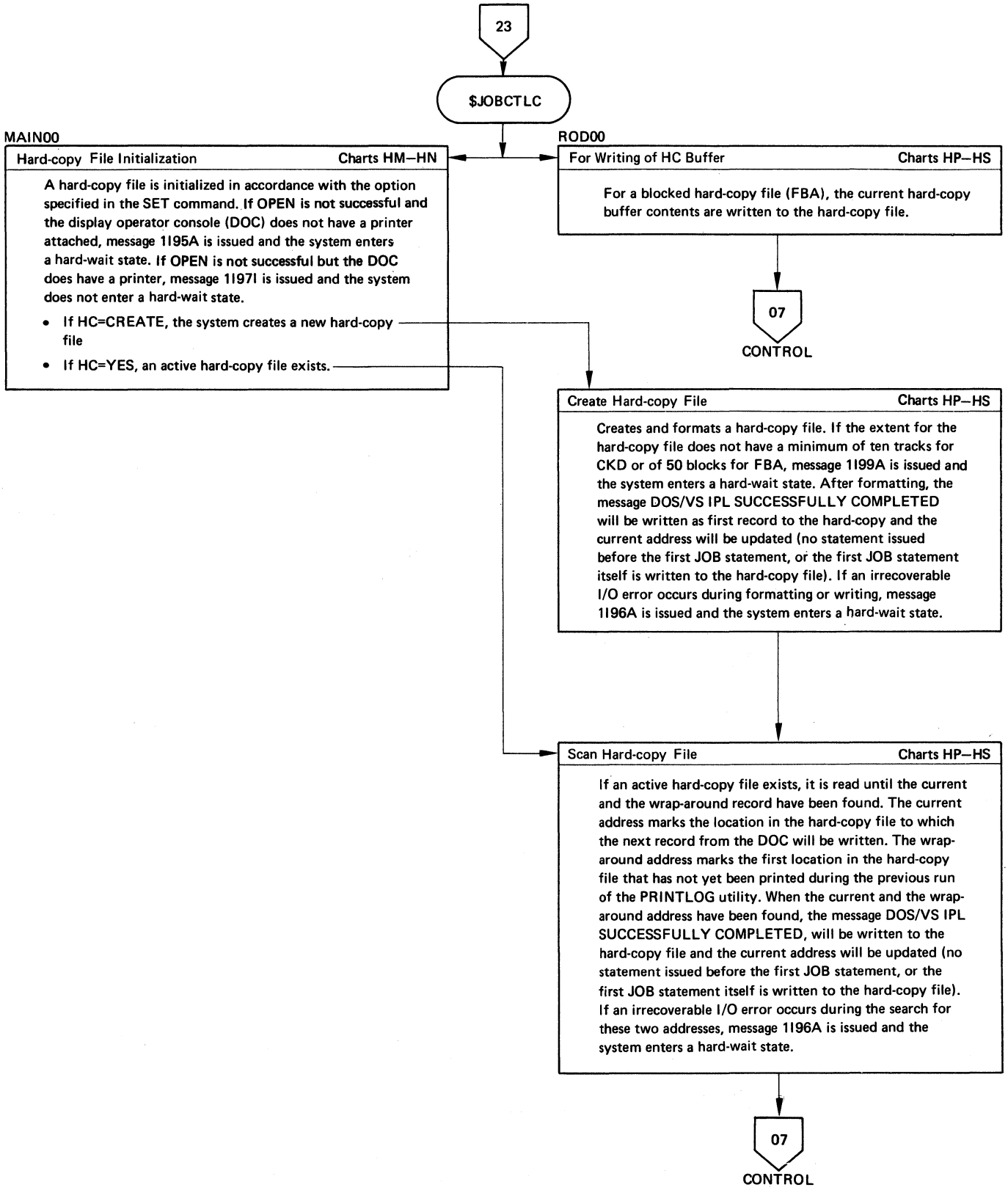


Chart 10. $JOBCTLD - Main Routines ASSGN and CLOSE

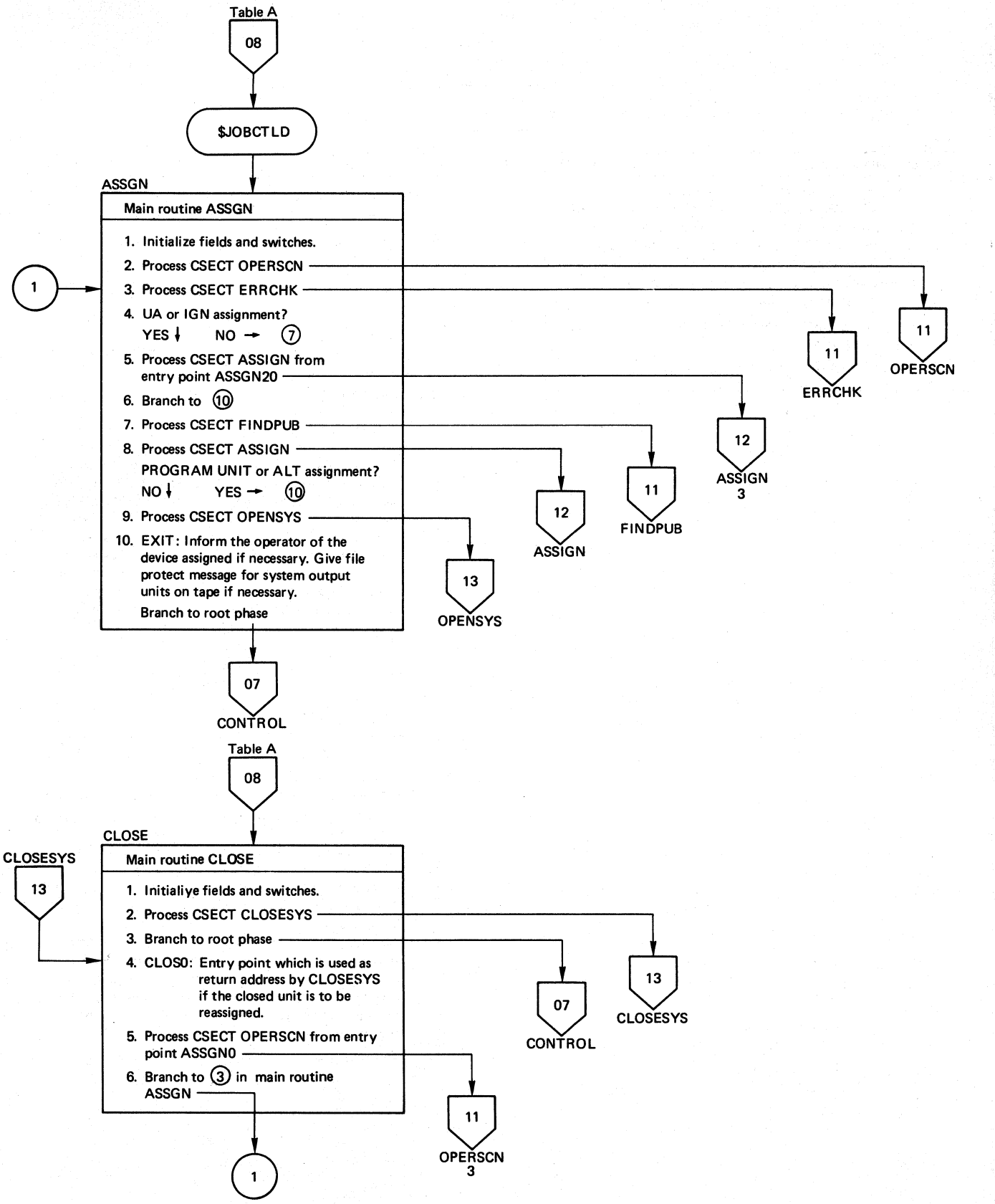


Chart 11. $JOBCTLD - CSECTS OPERSCN, ERRCHK, and FINDPUB

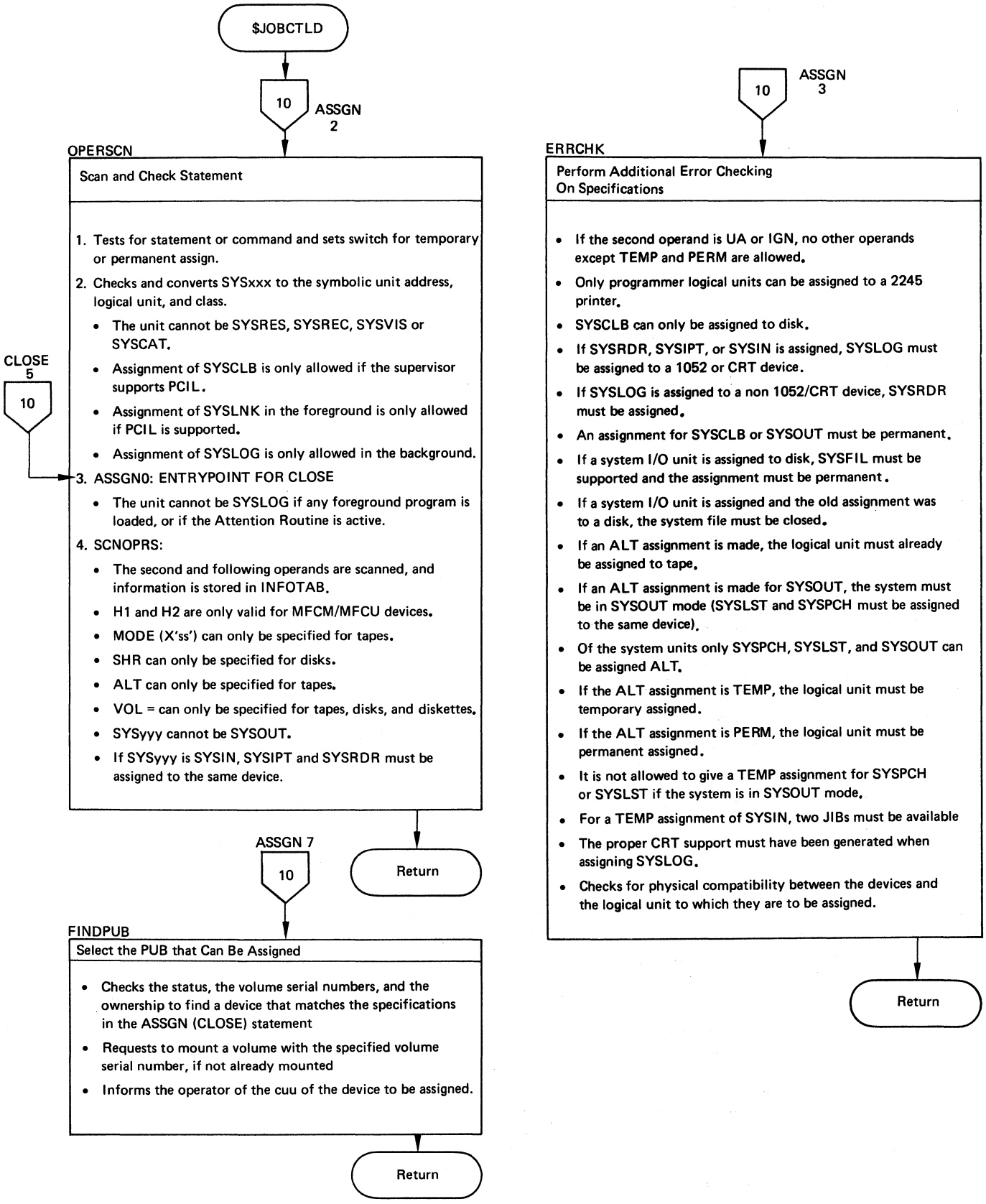


Chart 12. $JOBCTLD - CSECT ASSIGN

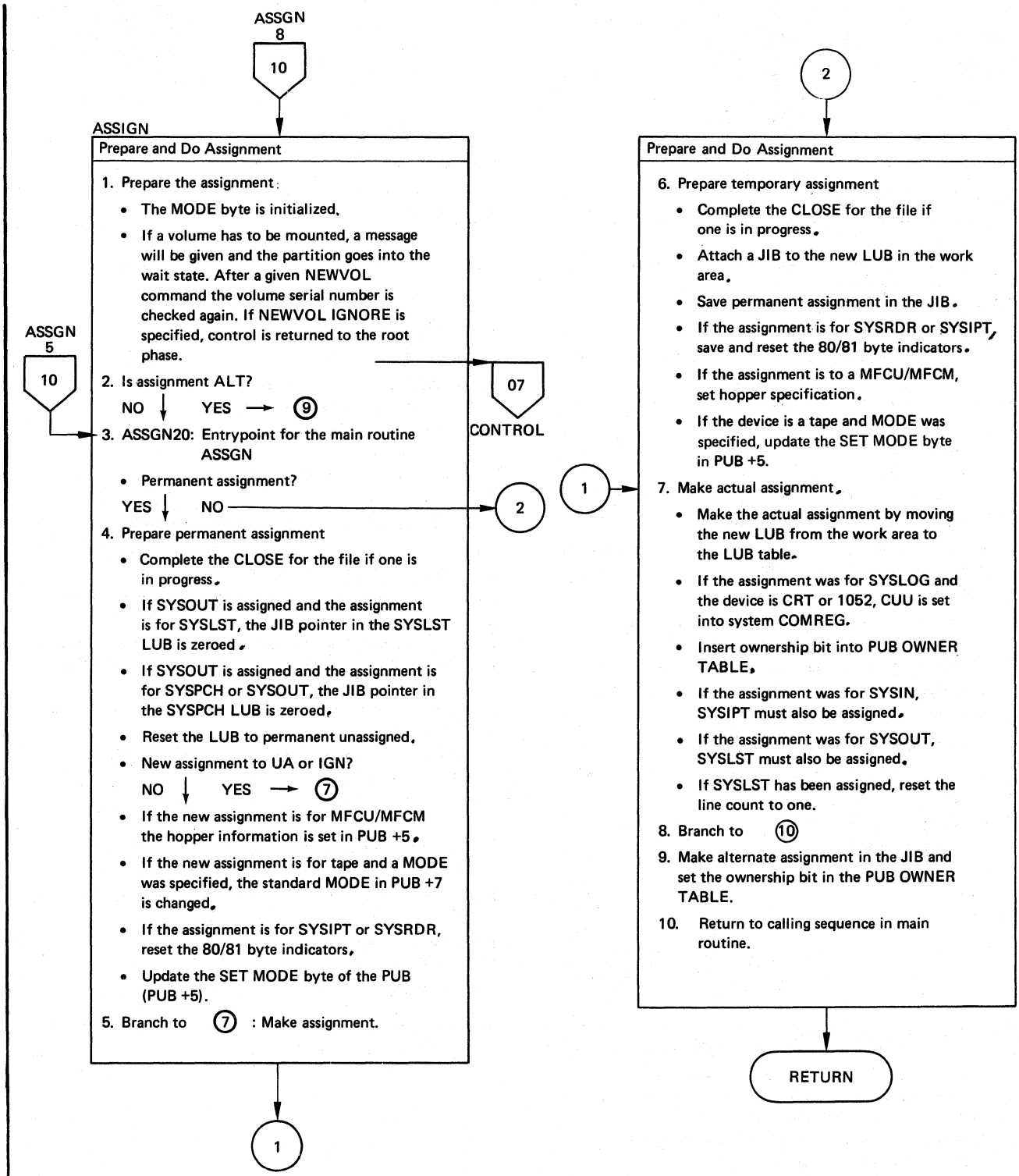


Chart 13. $JOBCTLD - CSECTS OPENSYS and CLOSESYS

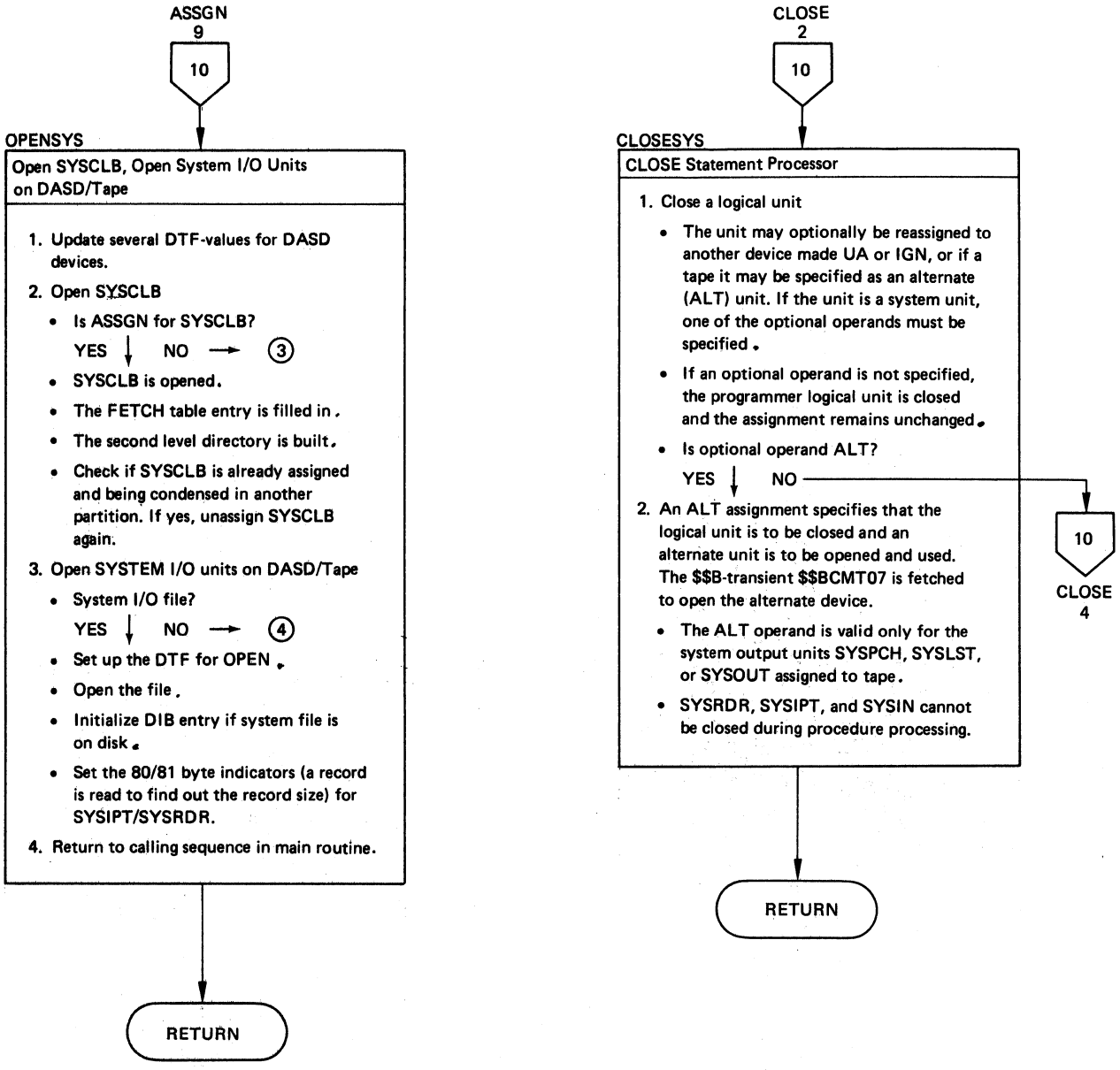


Chart 14. $JOBCTLE - EXEC and CALL Processors (LA-LU)

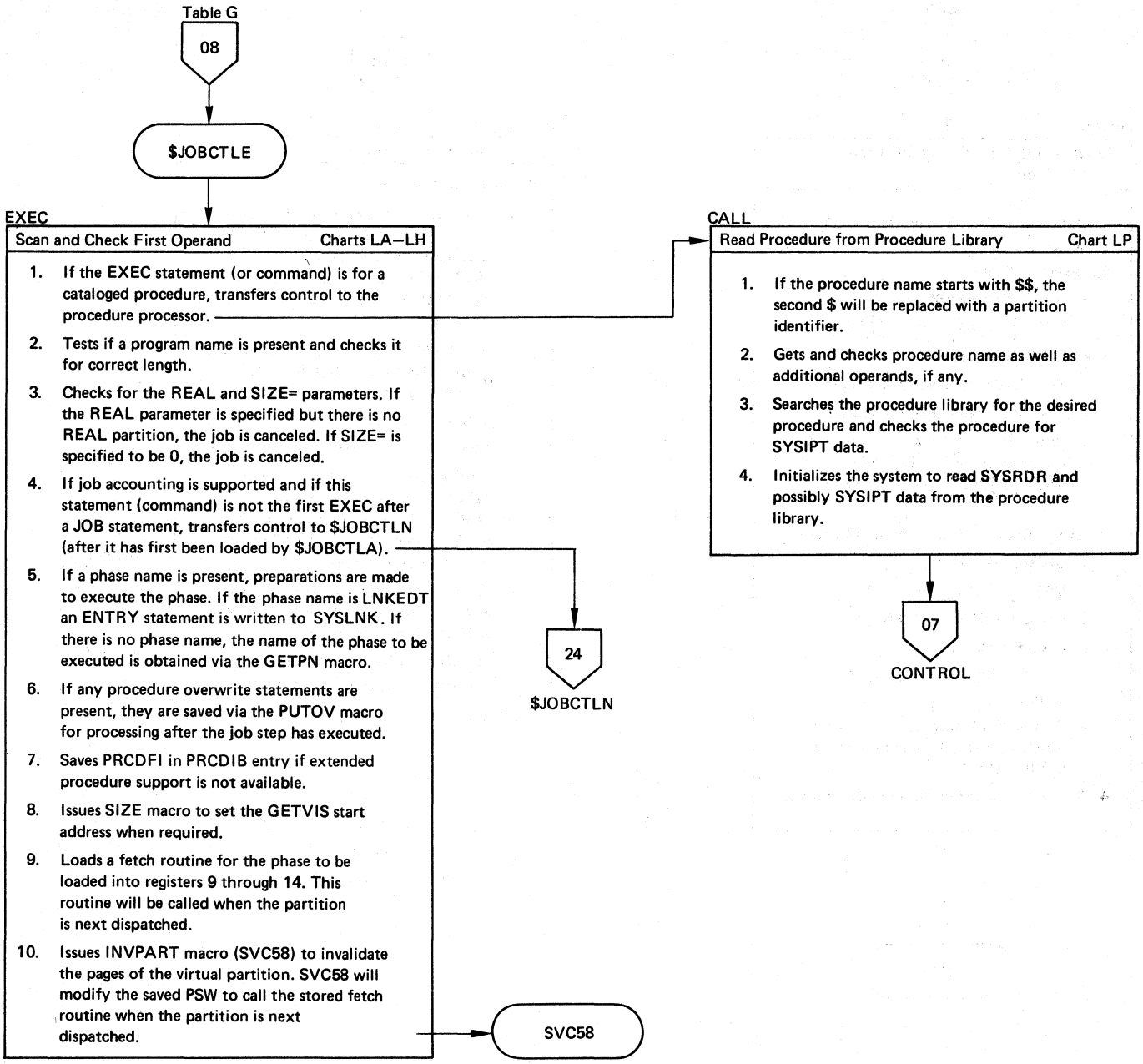
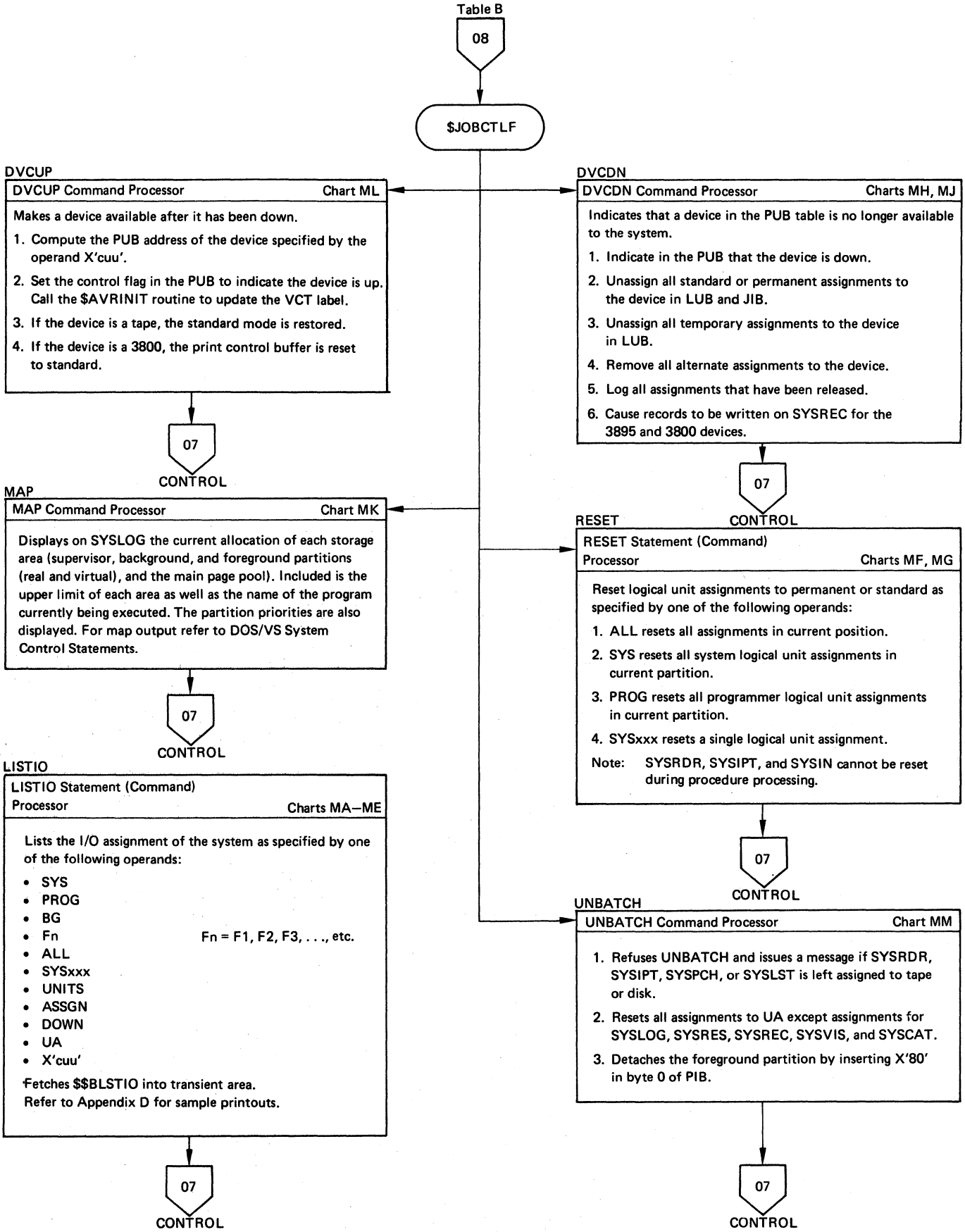


Chart 15. $JOBCTLF - DVCUP, DVCDN, MAP, LISTIO, RESET, UNBATCH Processors (MA-MU)



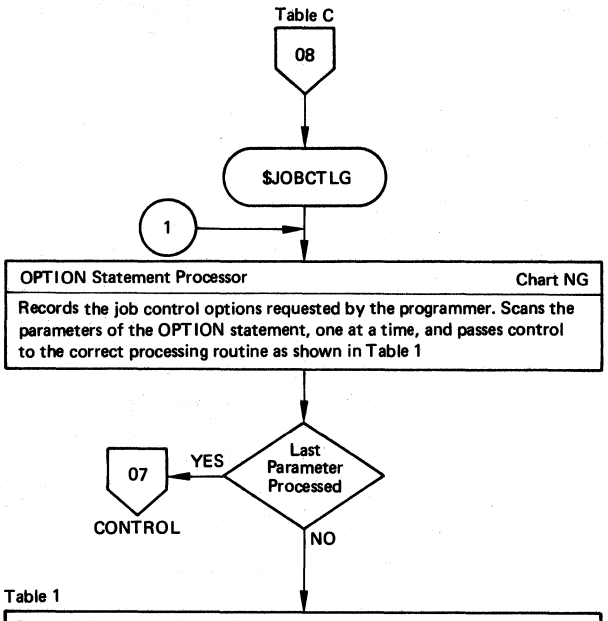


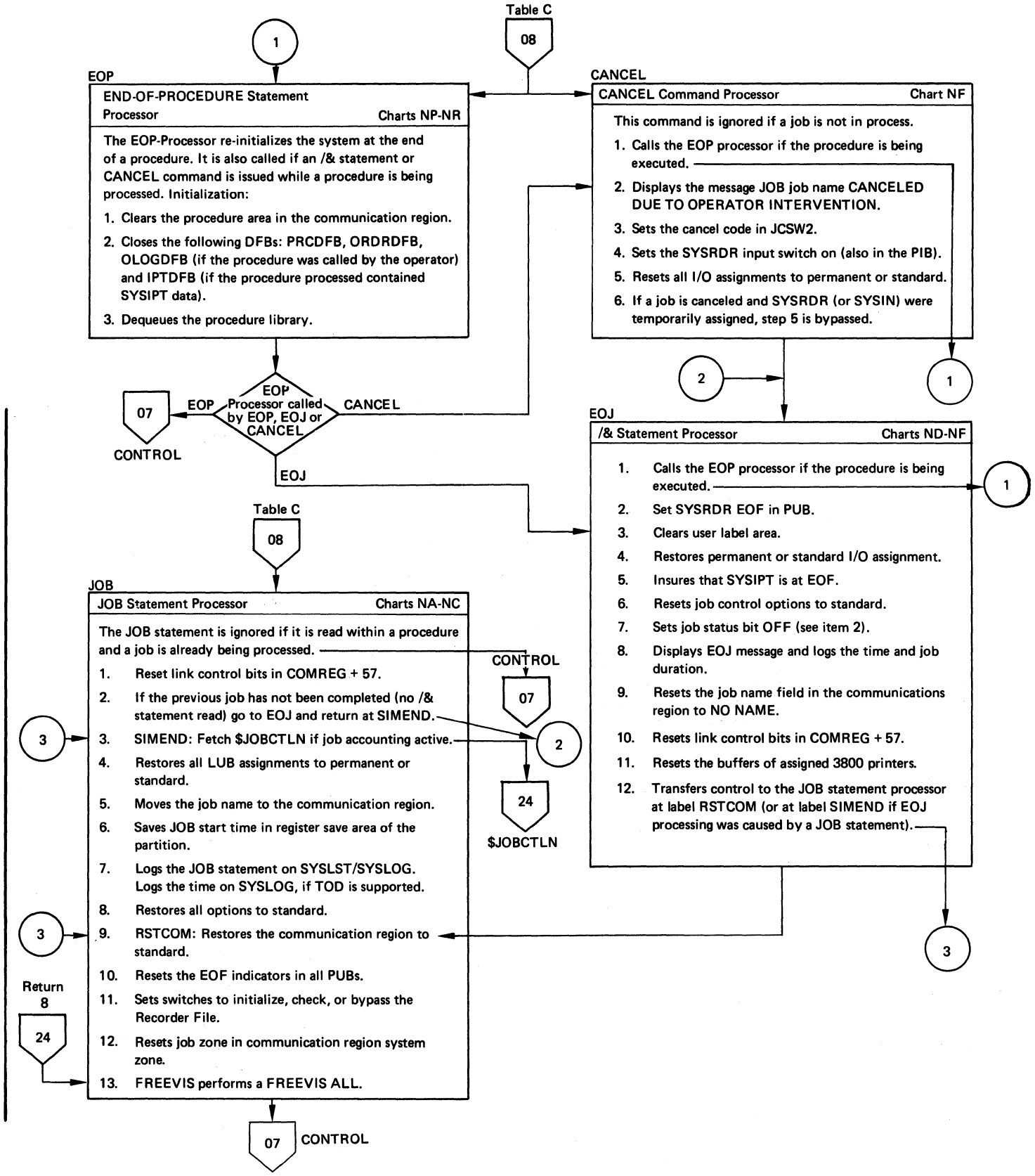
Table 1

Parameter	Routine Label	Chart
DECK	DECK Set DECK option ON, COMREG + 58, bit 0	NH
NODECK	NODECK Set DECK option OFF, COMREG + 58, bit 0	NH
LIST	LIST Set LIST option ON, COMREG + 58, bit 1	NH
NOLIST	NOLIST Set LIST option OFF, COMREG + 58, bit 1	NH
LISTX	LISTX Set LISTX option ON, COMREG+ 58, bit 2	NH
NOLISTX	NLISTX Set LISTX option OFF, COMREG + 58, bit 2	NH
SYM	SYM Set SYM option ON, COMREG + 58, bit 3	NH
NOSYM	NOSYM Set SYM option OFF, COMREG + 58, bit 3	NH
XREF	XREF Set XREF option ON, COMREG + 58, bit 4 Set SXREF option OFF, COMREG + 141, bit 7	NH
SXREF	SXREF Set SXREF option ON, COMREG + 141, bit 7 Set XREF option OFF, COMREG + 58, bit 4	NI
NOXREF	NOXREF Set XREF option OFF, COMREG + 58, bit 4	NH
ERRS	ERRS Set ERRS option ON, COMREG + 58, bit 5	NH
NOERRS	NOERRS Set ERRS option OFF, COMREG + 58, bit 5	NH
48C	C48 Set 48 character set option ON, COMREG + 58, bit 6	NH
60C	C60 Set 60 character set option ON, COMREG + 58, bit 6	NH
LOG	OPTLOG Set the SYSLST-LOG option ON, COMREG + 59, bit 3	NK
NOLOG	OPTNLG Set the SYSLST-LOG option OFF, COMREG + 59, bit 3	NH

Table 1 continued

USRLABEL	USRLBL Initialize job control label processing for USER LABEL processing	NJ
TERM	TERM Set TERM option ON, COMREG + 141, bit 5=1	NI
NOTERM	NOTERM Set TERM option OFF, COMREG + 141, bit 5=0	NI
STDLABEL	STDLBL Initialize job control label processing for STANDARD LABEL processing. Labels available to programs in ANY partition.	NJ
PARSTD	PARSTD Initialize job control label processing for PARTITION STANDARD LABEL processing. Labels available to programs in only one partition.	NJ
DUMP	DUMP Set job control dump option ON, COMREG + 59, bit 1	NK
NODUMP	NODUMP Set job control dump option OFF, COMREG + 59, bit 1	NK
PARTDUMP	PARTDUMP Set job control partition dump option ON, COMREG + 141, bit 3	NK
LINK	LINK Set LNKEDT control bits, COMREG + 57, bits 0, 2. Valid for BG only, unless SYSCLB assigned	NJ
NOLINK	NOLINK Set LINK option OFF, COMREG + 57, bit 0	NK
CATAL	CATAL Set LNKEDT control bits, COMREG + 57, bits 0, 2, 3. Valid for BG only, unless SYSCLB assigned.	NJ
SYSPARM	SYSPRM Save character string	NK
ALIGN	ALIGN Set ALIGN option ON, COMREG + 141, bit 1	NH
NOALIGN	NOALIGN Set ALIGN option OFF, COMREG + 141, bit 1	NH
EDECK	EDECK Set EDECK option ON, COMREG + 141, bit 0	NH
NOEDECK	NOEDECK Set EDECK option OFF, COMREG + 141, bit 0	NH
ACANCEL	Set ACANCEL option ON, COMREG + 141, bit 7	
NOACANCEL	Set ACANCEL option OFF, COMREG + 141, bit 7	
SUBLIB=DF	Set ALTSUBSL option ON, COMREG + 141, bit 6	NK
SUBLIB=AE	Set ALTSUBSL option OFF, COMREG + 141, bit 6	NK
RLD	RLD Set RLD option ON, COMREG + 141, bit 3	NH
NORLD	NORLD Set RLD option OFF, COMREG + 141, bit 3	NH
ONLINE	OPTONL Switch ONLINE SYSGEN ON, COMREG + X'AD', bit 7	NI
NOONLINE	OPTNOONL Switch ONLINE SYSGEN OFF, COMREG + X'AD', bit 7	NI

Chart 17. $JOBCTLG - EOP, EOJ, CANCEL, JOB Processors (NA-NF, NL-NU)



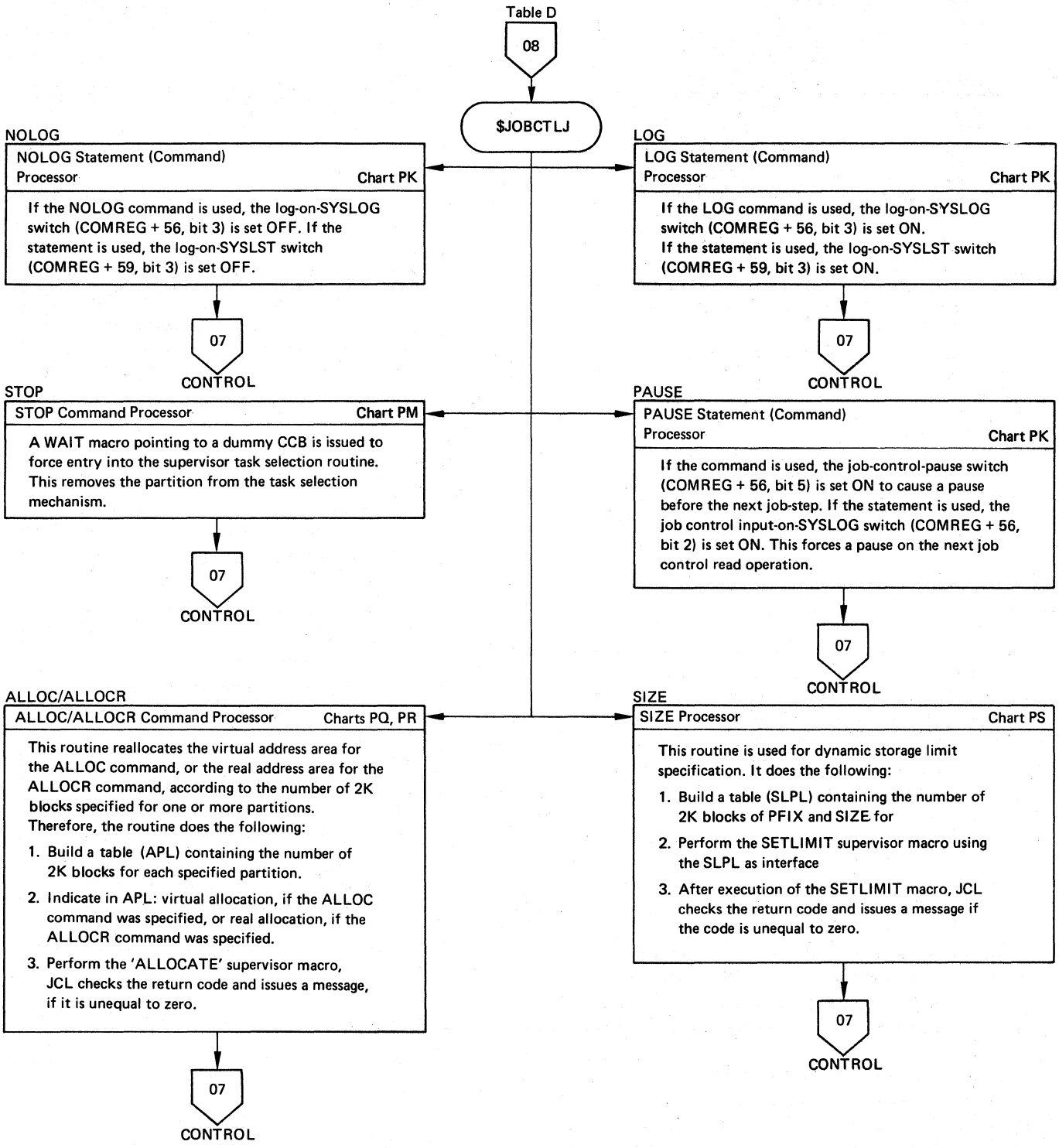
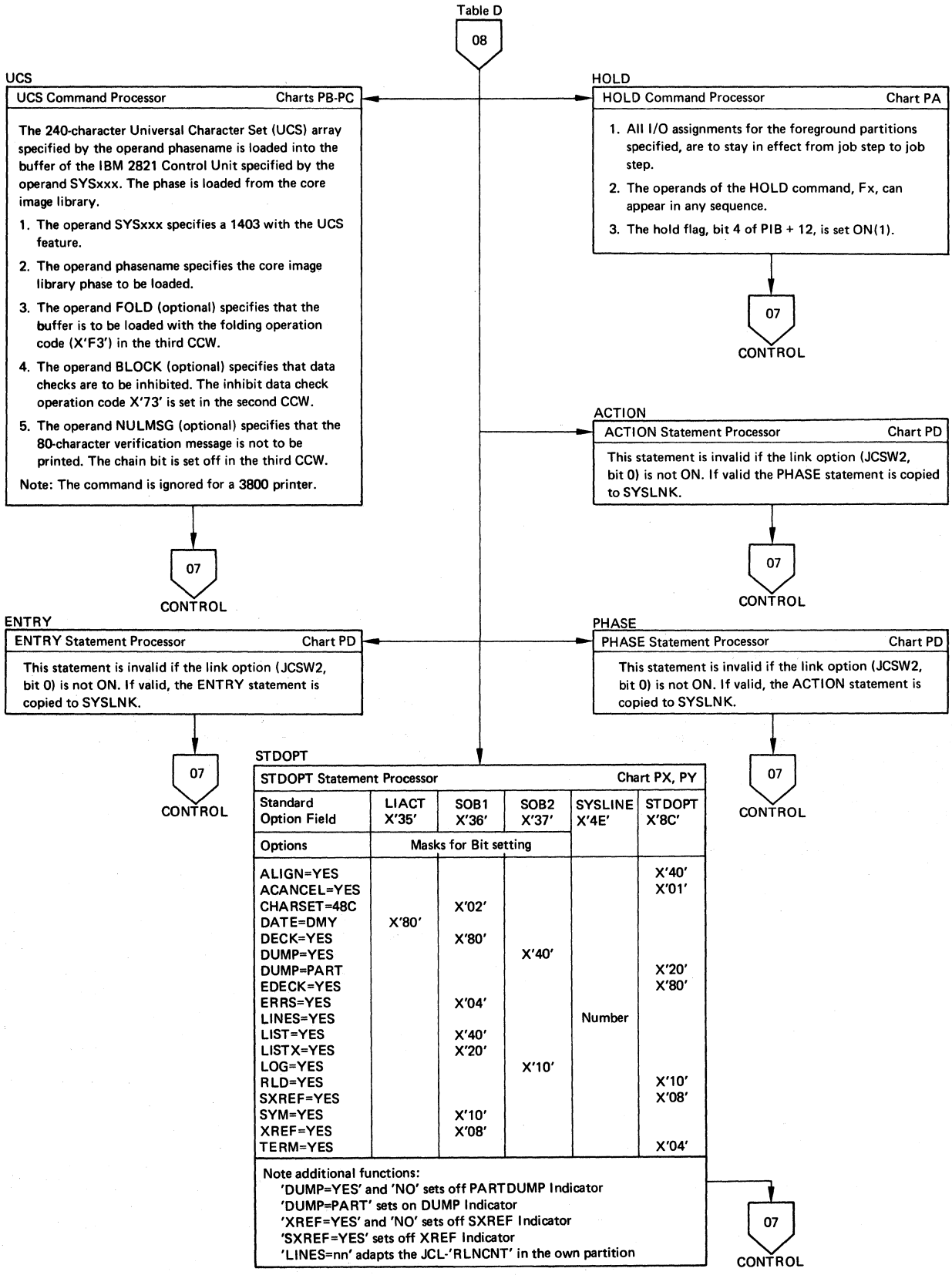


Table D
08
$JOBCTLJ
NOLOG
NOLOG Statement (Command) Processor
Chart PK
If the NOLOG command is used, the log-on-SYSLOG switch (COMREG + 56, bit 3) is set OFF. If the statement is used, the log-on-SYSLST switch (COMREG + 59, bit 3) is set OFF.
07
CONTROL
LOG
LOG Statement (Command) Processor
Chart PK
If the LOG command is used, the log-on-SYSLOG switch (COMREG + 56, bit 3) is set ON.
If the statement is used, the log-on-SYSLST switch (COMREG + 59, bit 3) is set ON.
07
CONTROL
STOP
STOP Command Processor
Chart PM
A WAIT macro pointing to a dummy CCB is issued to force entry into the supervisor task selection routine. This removes the partition from the task selection mechanism.
07
CONTROL
PAUSE
PAUSE Statement (Command) Processor
Chart PK
If the command is used, the job-control-pause switch (COMREG + 56, bit 5) is set ON to cause a pause before the next job-step. If the statement is used, the job control input-on-SYSLOG switch (COMREG + 56, bit 2) is set ON. This forces a pause on the next job control read operation.
07
CONTROL
ALLOC/ALLOCR
ALLOC/ALLOCR Command Processor
Charts PQ, PR
This routine reallocates the virtual address area for the ALLOC command, or the real address area for the ALLOCR command, according to the number of 2K blocks specified for one or more partitions. Therefore, the routine does the following:
1. Build a table (APL) containing the number of 2K blocks for each specified partition.
2. Indicate in APL: virtual allocation, if the ALLOC command was specified, or real allocation, if the ALLOCR command was specified.
3. Perform the 'ALLOCATE' supervisor macro, JCL checks the return code and issues a message, if it is unequal to zero.
07
CONTROL
SIZE
SIZE Processor
Chart PS
This routine is used for dynamic storage limit specification. It does the following:
1. Build a table (SLPL) containing the number of 2K blocks of PFIX and SIZE for
2. Perform the SETLIMIT supervisor macro using the SLPL as interface
3. After execution of the SETLIMIT macro, JCL checks the return code and issues a message if the code is unequal to zero.
07
CONTROL

Chart 19. $JOBCTLJ - UCS, ENTRY, HOLD, ACTION, STDOPT, PHASE Processors



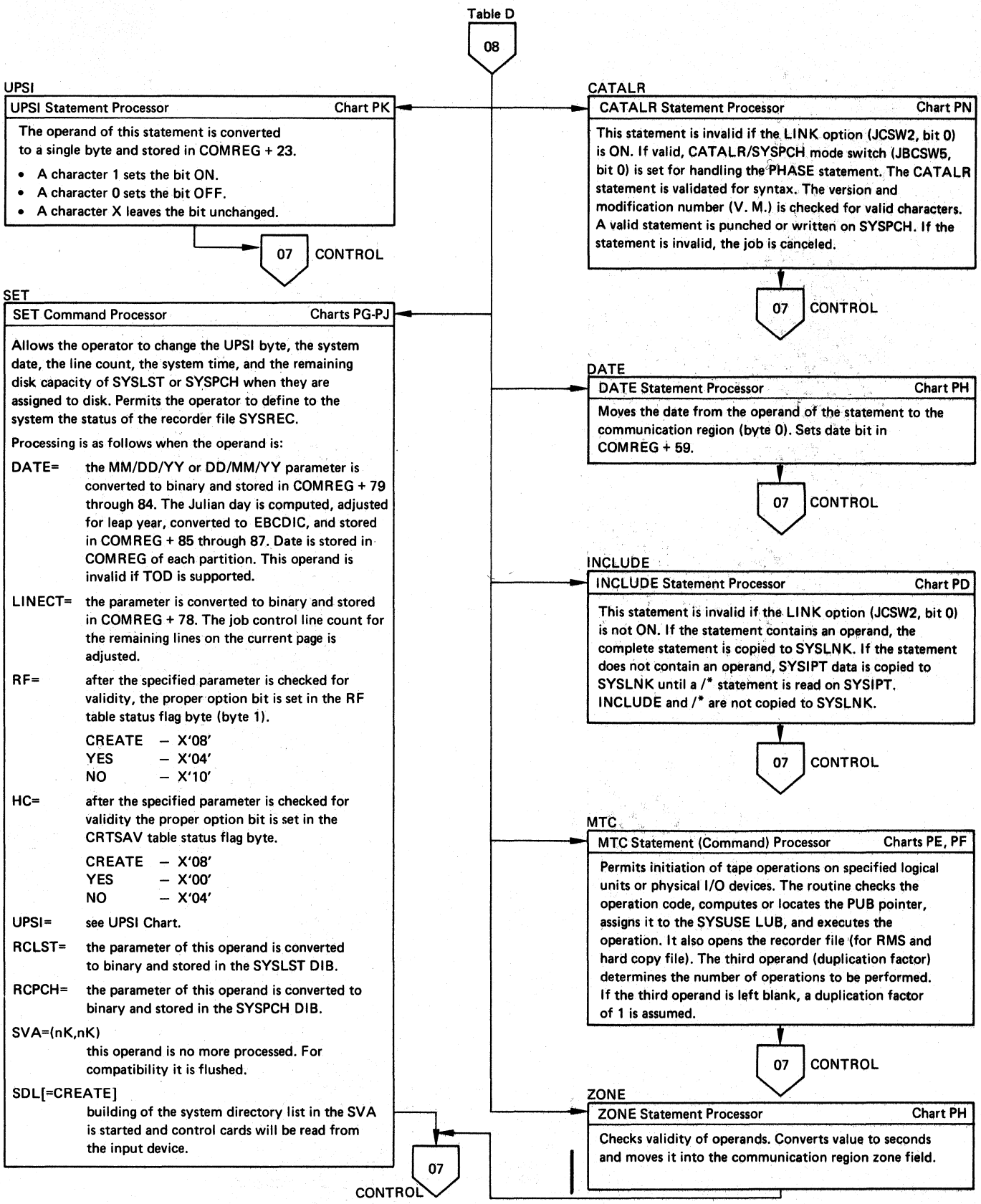
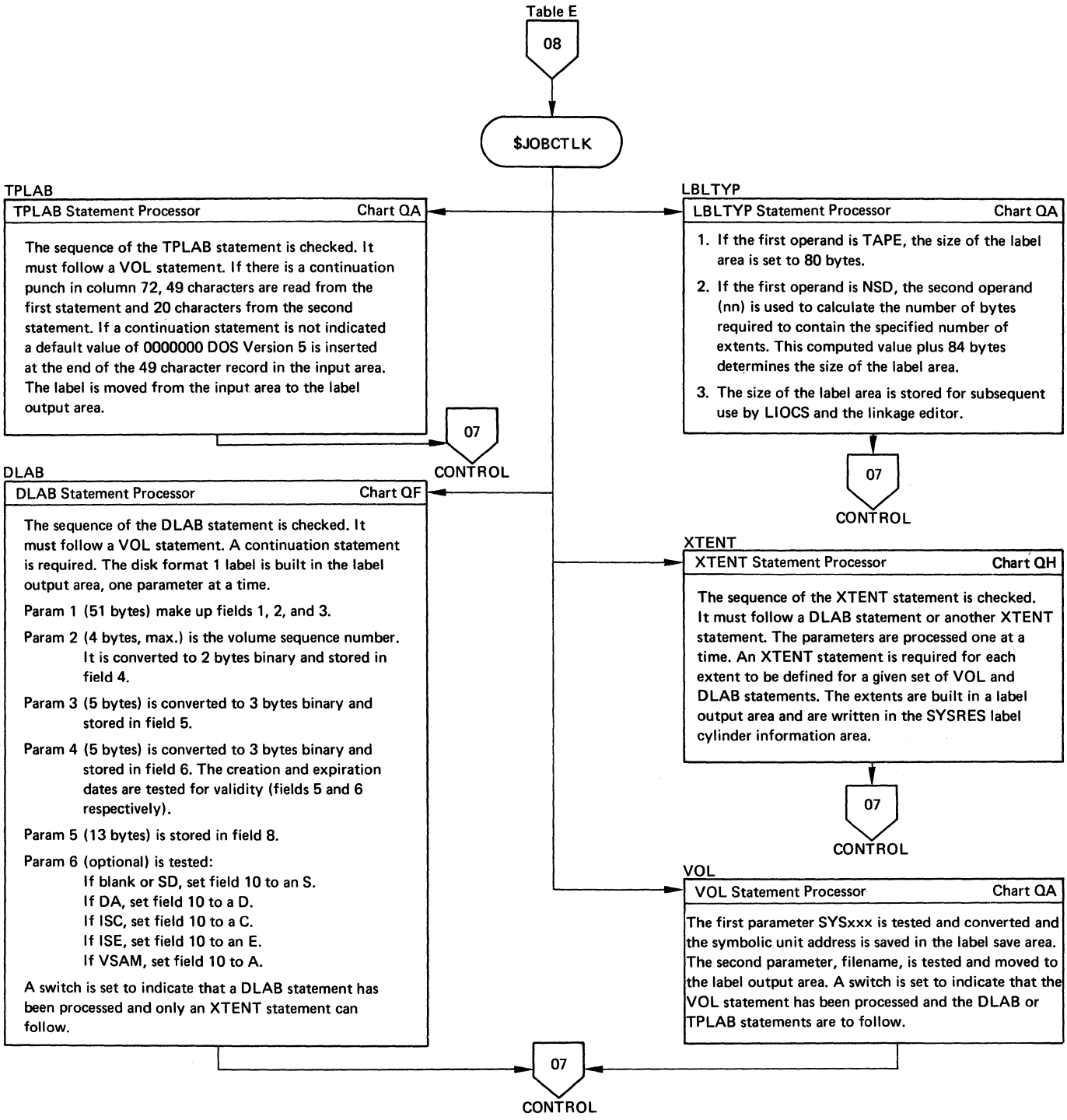


Table D
08
UPSI
UPSI Statement Processor
Chart PK
The operand of this statement is converted to a single byte and stored in COMREG + 23.
• A character 1 sets the bit ON.
• A character 0 sets the bit OFF.
• A character X leaves the bit unchanged.
07
CONTROL
SET
SET Command Processor
Charts PG-PJ
Allows the operator to change the UPSI byte, the system date, the line count, the system time, and the remaining disk capacity of SYSLST or SYSPCH when they are assigned to disk. Permits the operator to define to the system the status of the recorder file SYSREC.
Processing is as follows when the operand is:
DATE= the MM/DD/YY or DD/MM/YY parameter is converted to binary and stored in COMREG + 79 through 84. The Julian day is computed, adjusted for leap year, converted to EBCDIC, and stored in COMREG + 85 through 87. Date is stored in COMREG of each partition. This operand is invalid if TOD is supported.
LINECT= the parameter is converted to binary and stored in COMREG + 78. The job control line count for the remaining lines on the current page is adjusted.
RF= after the specified parameter is checked for validity, the proper option bit is set in the RF table status flag byte (byte 1).
CREATE – X'08'
YES – X'04'
NO – X'10'
HC= after the specified parameter is checked for validity the proper option bit is set in the CRTSAV table status flag byte.
CREATE – X'08'
YES – X'00'
NO – X'04'
UPSI= see UPSI Chart.
RCLST= the parameter of this operand is converted to binary and stored in the SYSLST DIB.
RCPCH= the parameter of this operand is converted to binary and stored in the SYSPCH DIB.
SVA=(nK,nK)
this operand is no more processed. For compatibility it is flushed.
SDL[=CREATE]
building of the system directory list in the SVA is started and control cards will be read from the input device.
07
CONTROL
CATALR
CATALR Statement Processor
Chart PN
This statement is invalid if the LINK option (JCSW2, bit 0) is ON. If valid, CATALR/SYSPCH mode switch (JBCSW5, bit 0) is set for handling the PHASE statement. The CATALR statement is validated for syntax. The version and modification number (V. M.) is checked for valid characters. A valid statement is punched or written on SYSPCH. If the statement is invalid, the job is canceled.
07
CONTROL
DATE
DATE Statement Processor
Chart PH
Moves the date from the operand of the statement to the communication region (byte 0). Sets date bit in COMREG + 59.
07
CONTROL
INCLUDE
INCLUDE Statement Processor
Chart PD
This statement is invalid if the LINK option (JCSW2, bit 0) is not ON. If the statement contains an operand, the complete statement is copied to SYSLNK. If the statement does not contain an operand, SYSIPT data is copied to SYSLNK until a /* statement is read on SYSIPT. INCLUDE and /* are not copied to SYSLNK.
07
CONTROL
MTC
MTC Statement (Command) Processor
Charts PE, PF
Permits initiation of tape operations on specified logical units or physical I/O devices. The routine checks the operation code, computes or locates the PUB pointer, assigns it to the SYSUSE LUB, and executes the operation. It also opens the recorder file (for RMS and hard copy file). The third operand (duplication factor) determines the number of operations to be performed. If the third operand is left blank, a duplication factor of 1 is assumed.
07
CONTROL
ZONE
ZONE Statement Processor
Chart PH
Checks validity of operands. Converts value to seconds and moves it into the communication region zone field.

Chart 21. $JOBCTLK - TPLAB, DLAB, LBLTYP, XTENT, VOL Processors (QA-QW)



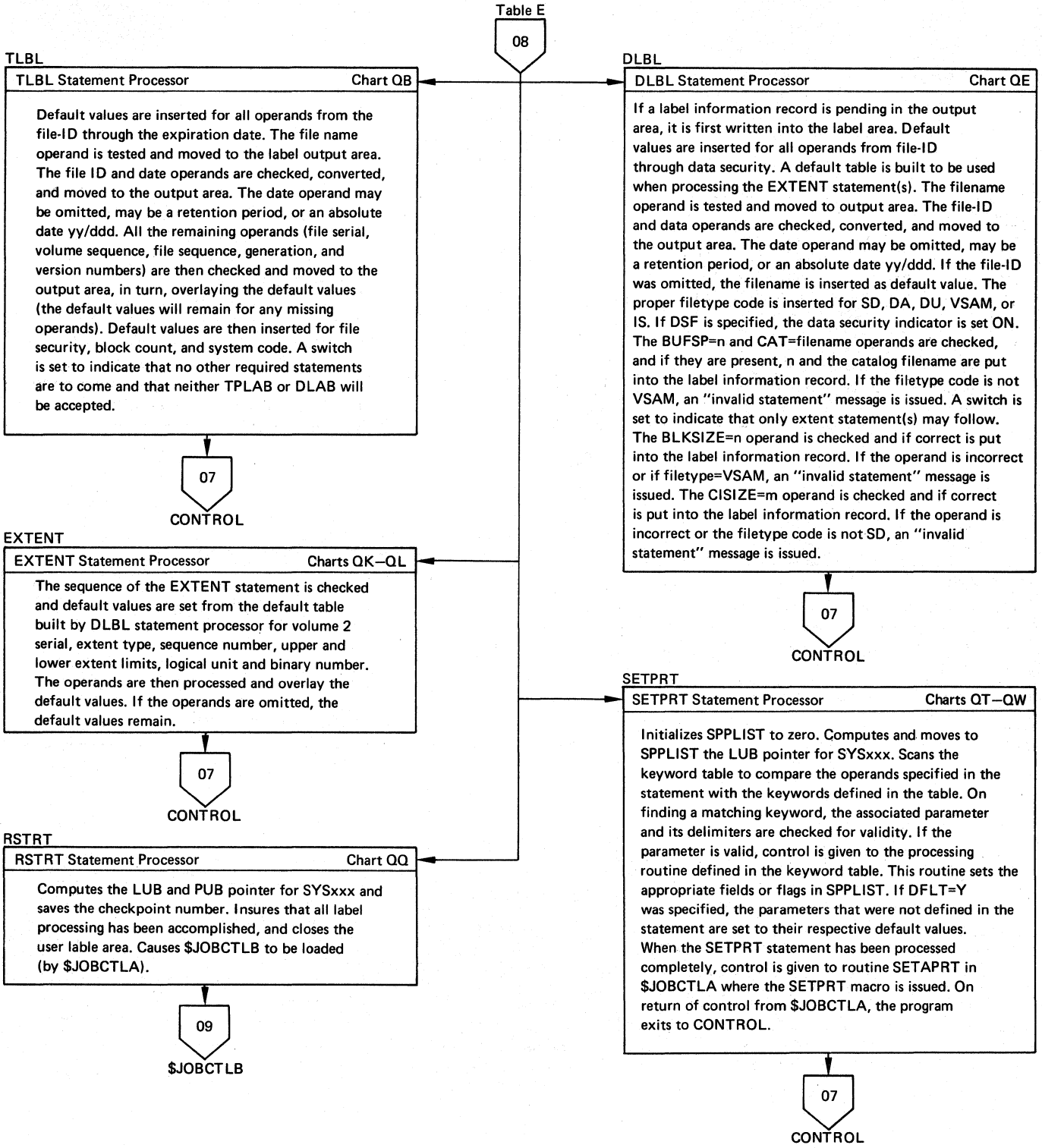


Table E
08
TLBL
TLBL Statement Processor
Chart QB
Default values are inserted for all operands from the file-ID through the expiration date. The file name operand is tested and moved to the label output area. The file ID and date operands are checked, converted, and moved to the output area. The date operand may be omitted, may be a retention period, or an absolute date yy/ddd. All the remaining operands (file serial, volume sequence, file sequence, generation, and version numbers) are then checked and moved to the output area, in turn, overlaying the default values (the default values will remain for any missing operands). Default values are then inserted for file security, block count, and system code. A switch is set to indicate that no other required statements are to come and that neither TPLAB or DLAB will be accepted.
07
CONTROL
DLBL
DLBL Statement Processor
Chart QE
If a label information record is pending in the output area, it is first written into the label area. Default values are inserted for all operands from file-ID through data security. A default table is built to be used when processing the EXTENT statement(s). The filename operand is tested and moved to output area. The file-ID and data operands are checked, converted, and moved to the output area. The date operand may be omitted, may be a retention period, or an absolute date yy/ddd. If the file-ID was omitted, the filename is inserted as default value. The proper filetype code is inserted for SD, DA, DU, VSAM, or IS. If DSF is specified, the data security indicator is set ON. The BUFSP=n and CAT=filename operands are checked, and if they are present, n and the catalog filename are put into the label information record. If the filetype code is not VSAM, an "invalid statement" message is issued. A switch is set to indicate that only extent statement(s) may follow. The BLKSIZE=n operand is checked and if correct is put into the label information record. If the operand is incorrect or if filetype=VSAM, an "invalid statement" message is issued. The CISIZE=m operand is checked and if correct is put into the label information record. If the operand is incorrect or the filetype code is not SD, an "invalid statement" message is issued.
07
CONTROL
EXTENT
EXTENT Statement Processor
Charts QK–QL
The sequence of the EXTENT statement is checked and default values are set from the default table built by DLBL statement processor for volume 2 serial, extent type, sequence number, upper and lower extent limits, logical unit and binary number. The operands are then processed and overlay the default values. If the operands are omitted, the default values remain.
07
CONTROL
SETPRT
SETPRT Statement Processor
Charts QT–QW
Initializes SPPLIST to zero. Computes and moves to SPPLIST the LUB pointer for SYSxxx. Scans the keyword table to compare the operands specified in the statement with the keywords defined in the table. On finding a matching keyword, the associated parameter and its delimiters are checked for validity. If the parameter is valid, control is given to the processing routine defined in the keyword table. This routine sets the appropriate fields or flags in SPPLIST. If DFLT=Y was specified, the parameters that were not defined in the statement are set to their respective default values. When the SETPRT statement has been processed completely, control is given to routine SETAPRT in $JOBCTLA where the SETPRT macro is issued. On return of control from $JOBCTLA, the program exits to CONTROL.
07
CONTROL
RSTRT
RSTRT Statement Processor
Chart QQ
Computes the LUB and PUB pointer for SYSxxx and saves the checkpoint number. Insures that all label processing has been accomplished, and closes the user lable area. Causes $JOBCTLB to be loaded (by $JOBCTLA).
09
$JOBCTLB

Chart 23. $JOBCTLM - RF and ROD Processors (RA-RT)

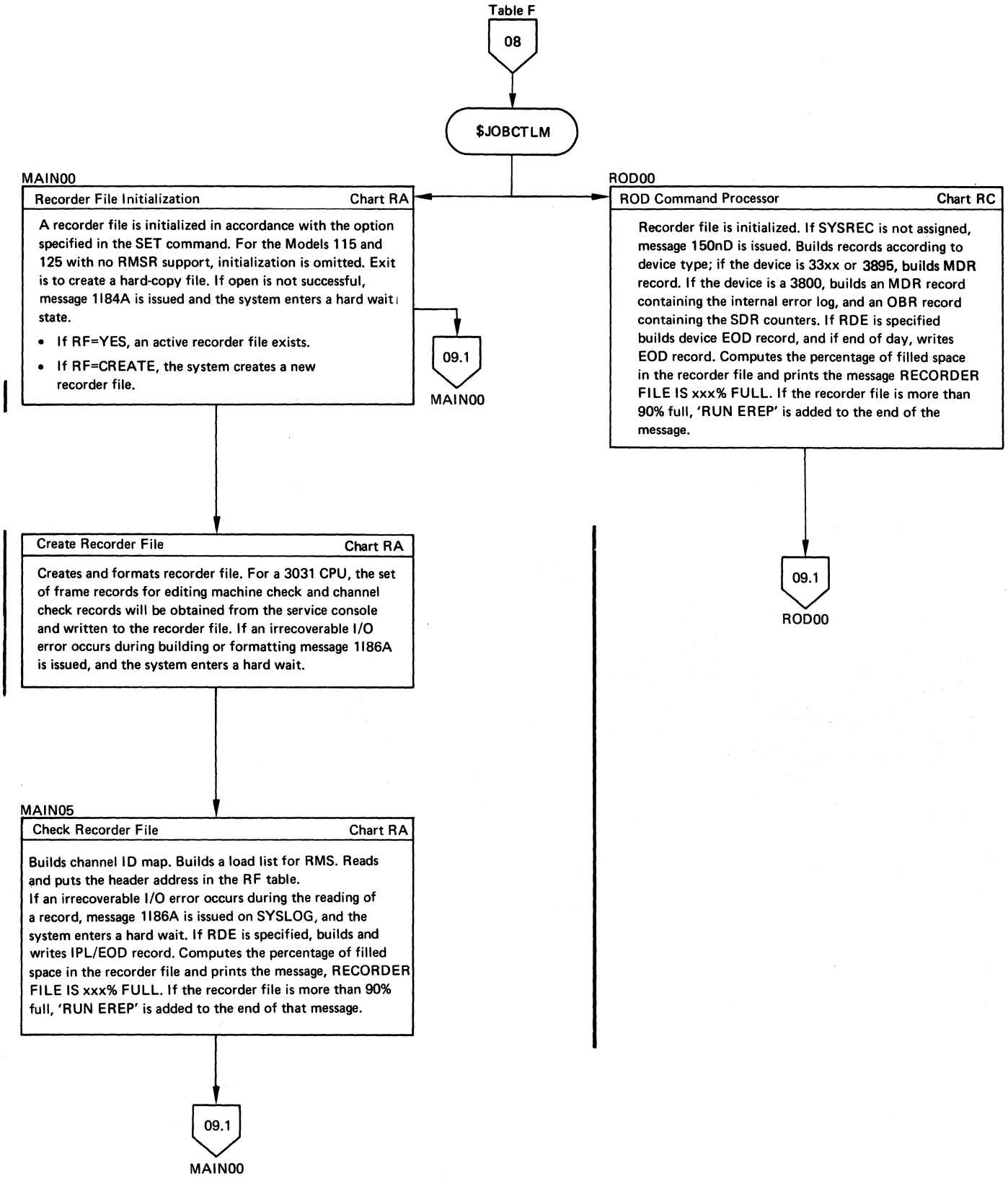


Chart 24. $JOBCTLN - Job Accounting Routines (SA-SB)

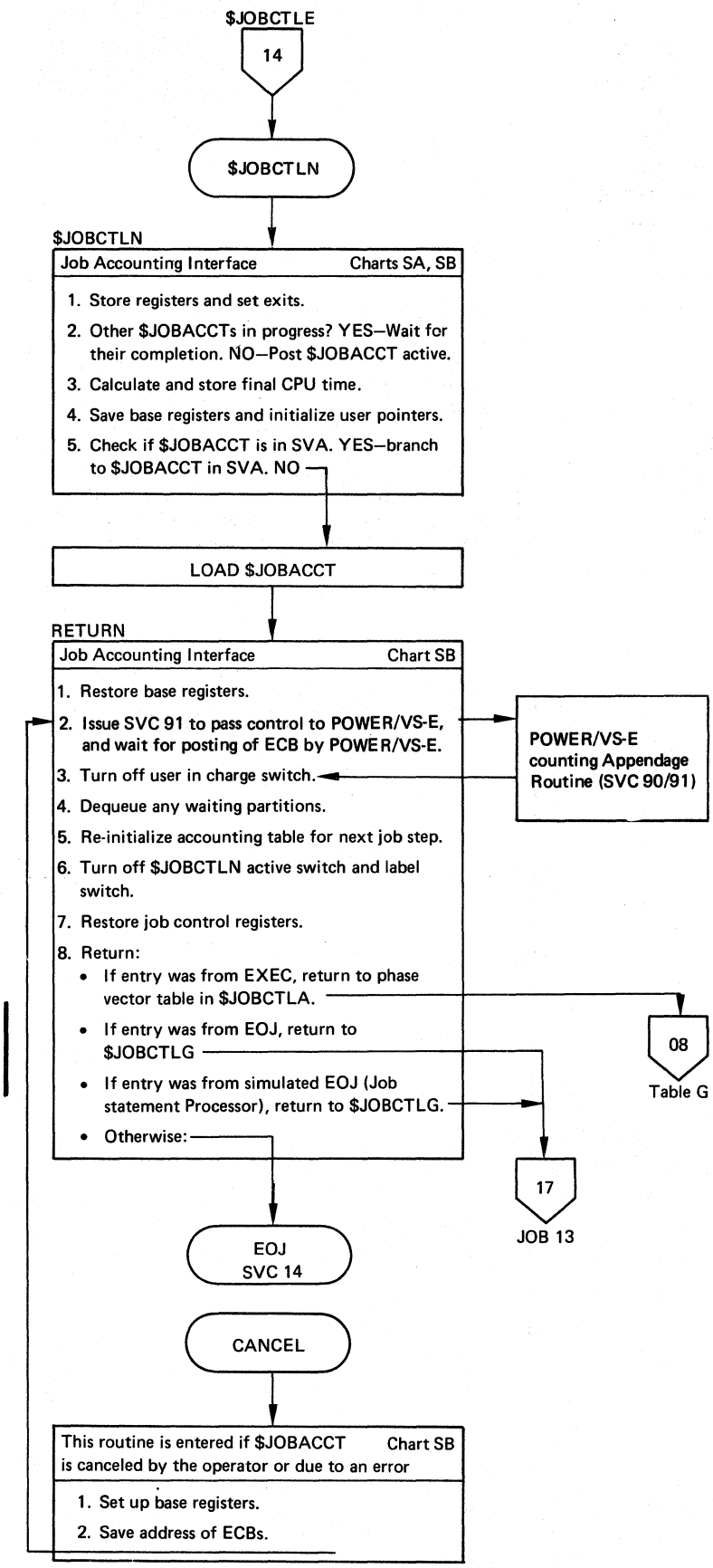


Chart 25. $IJBSLA - SLA Root

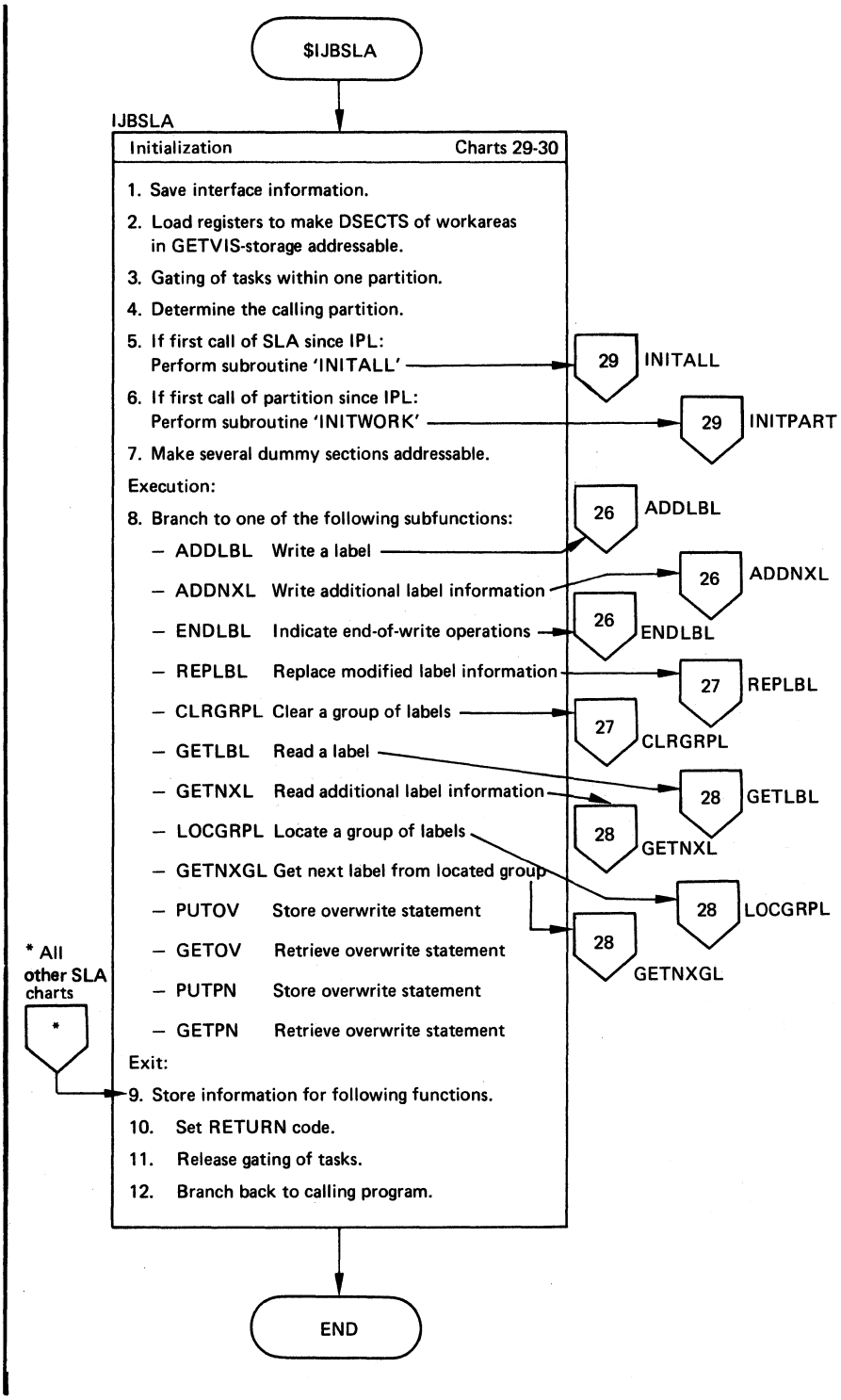


Chart 26. $IJBSLA - ADDLBL, ADDNXL, ENDLBL Processors

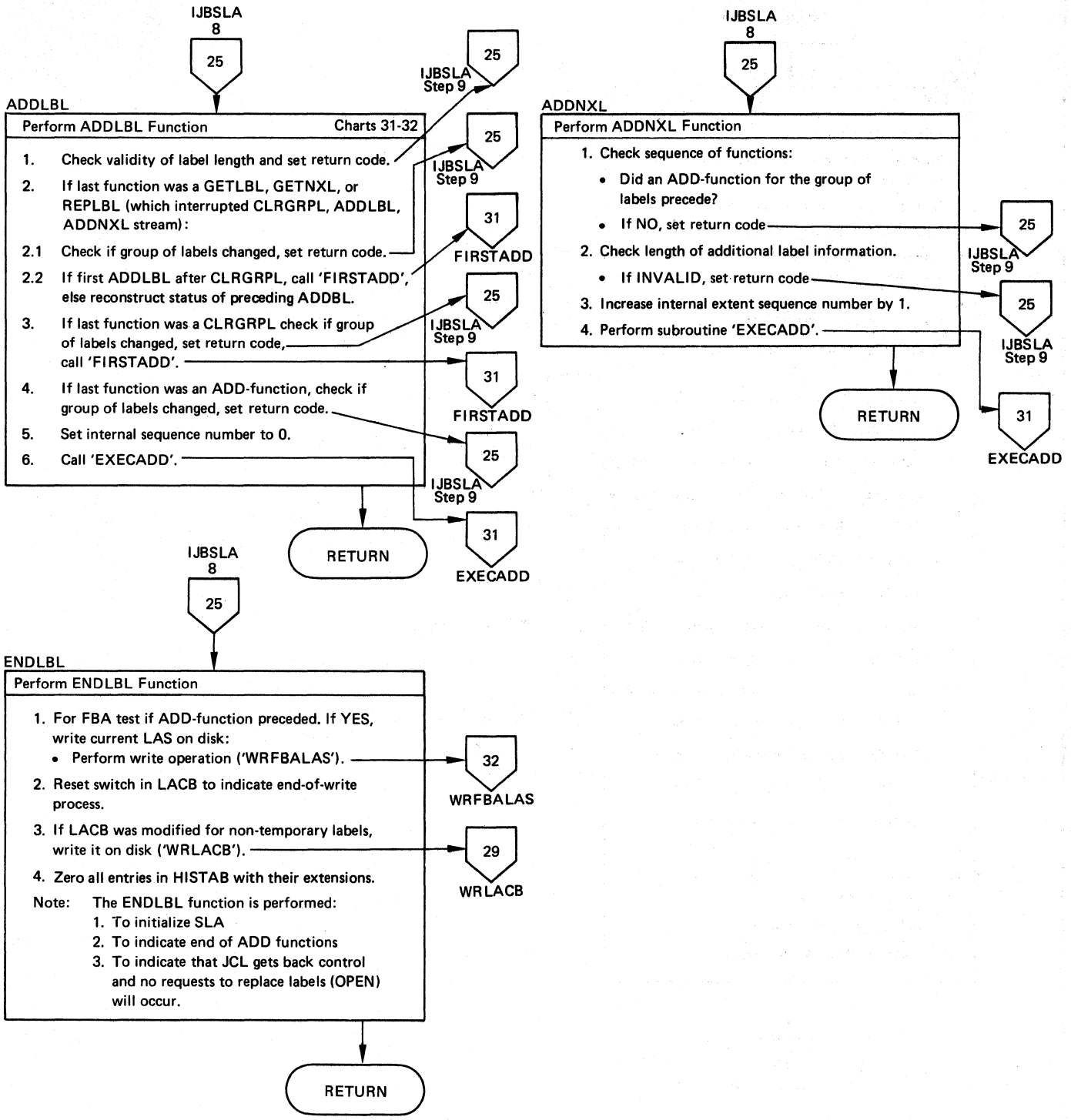


Chart 27. $IJBSLA - REPLBL, CLRGRPL Processors

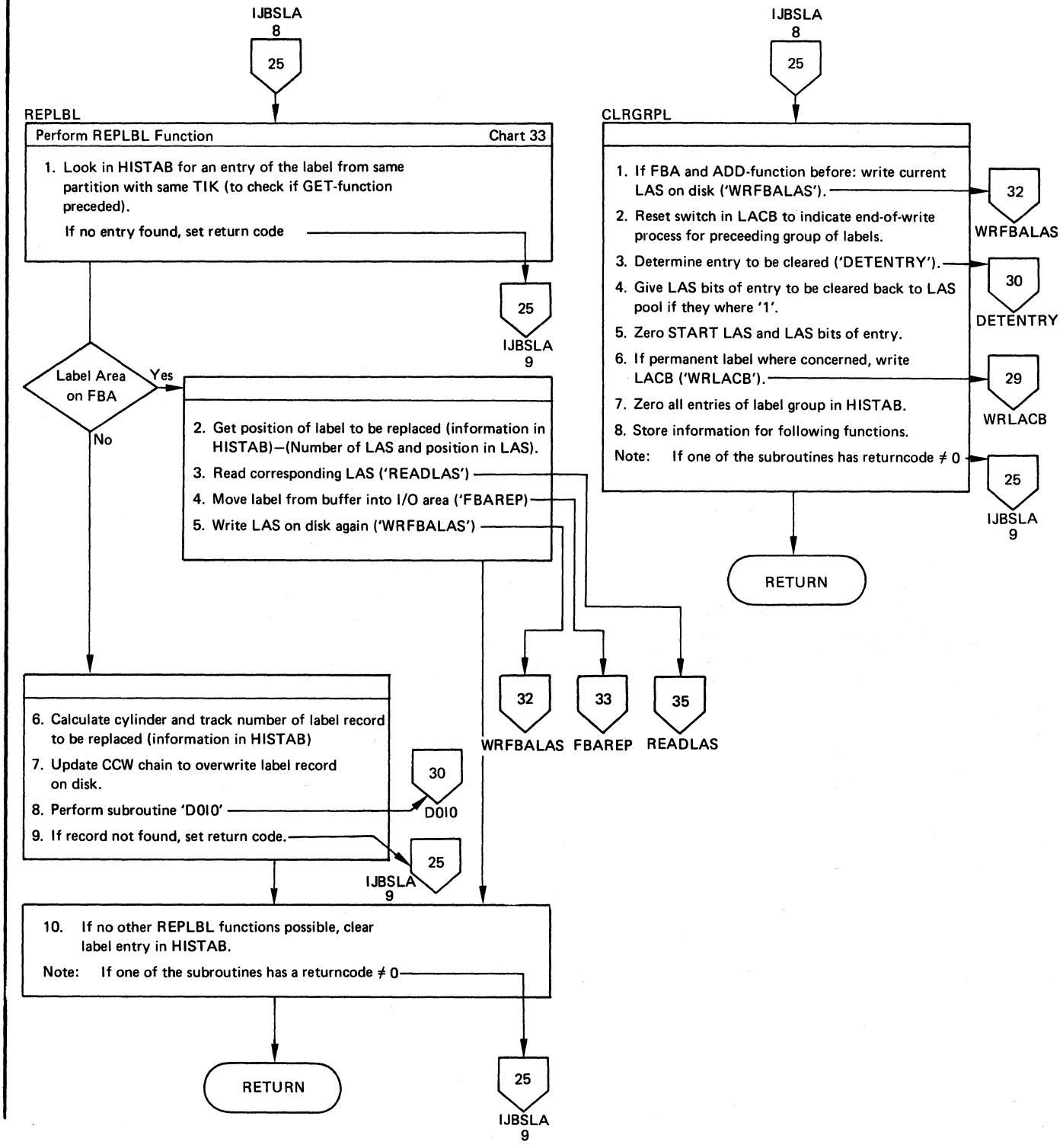


Chart 28. $IJBSLA - GETLBL, GETNXL, LOCGRPL, GETNXGL Processors

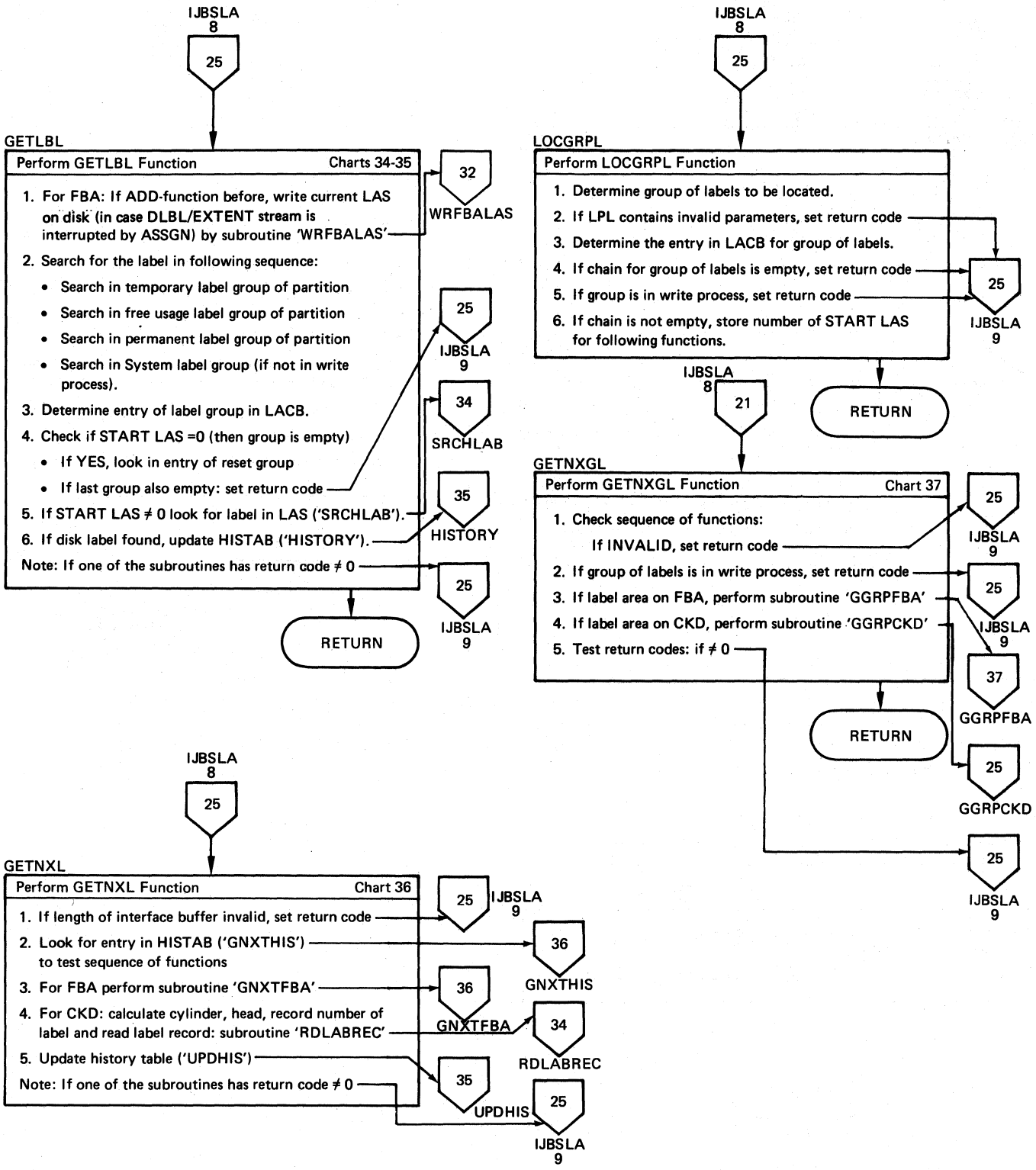


Chart 29. $IJBSLA - SLA Initialization Routines

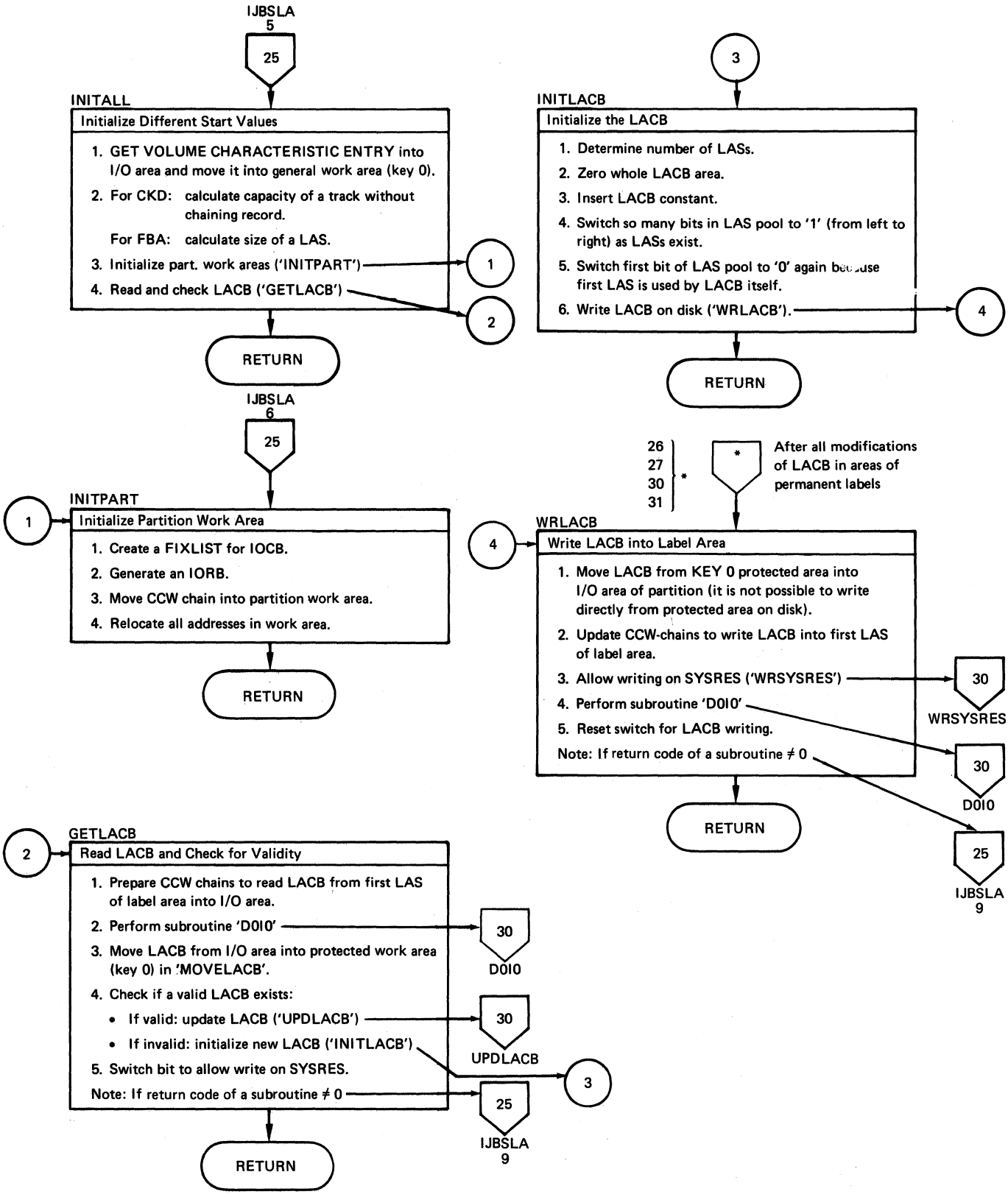


Chart 30. $IJBSLA - Routines

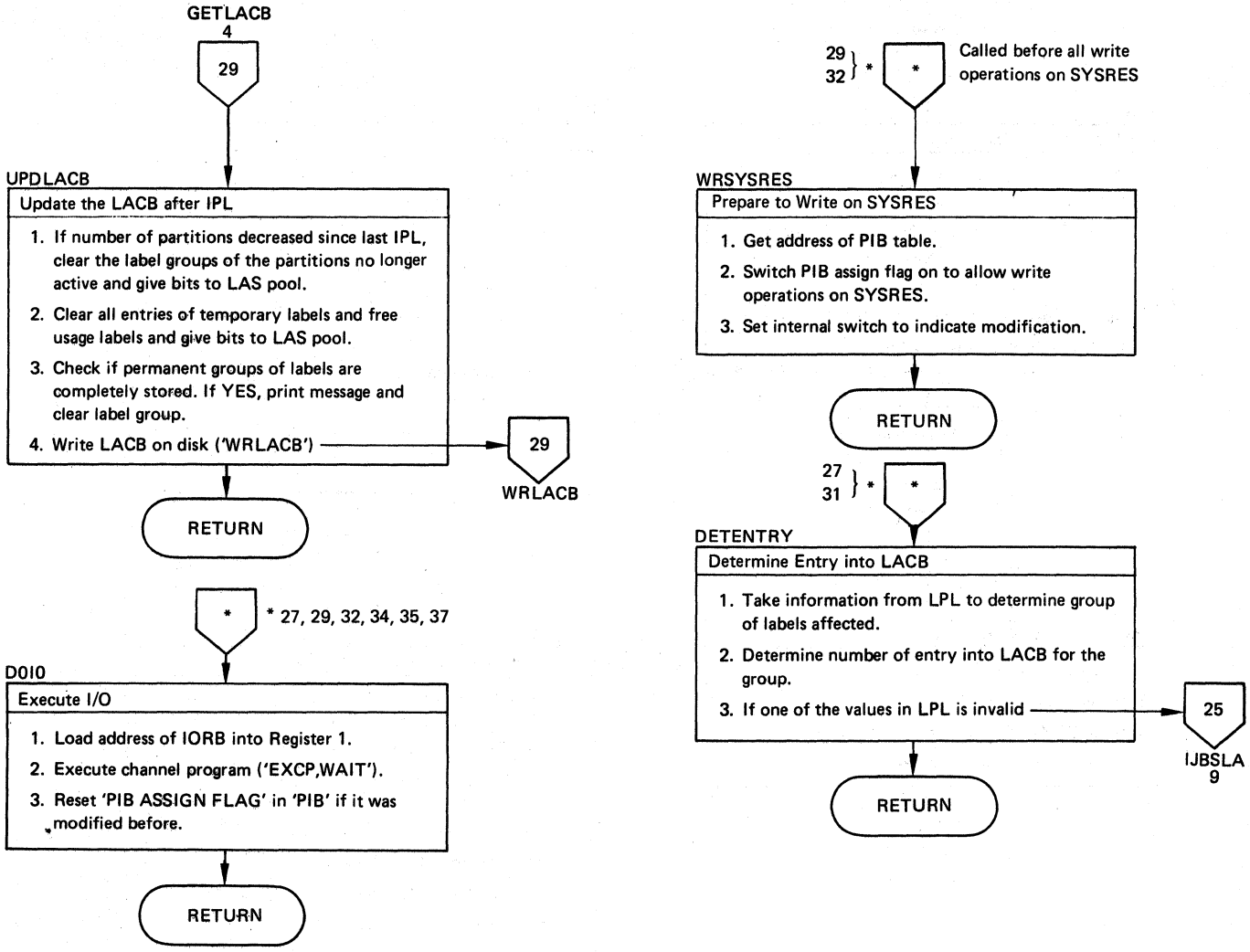


Chart 31. $IJBSLA - Routines

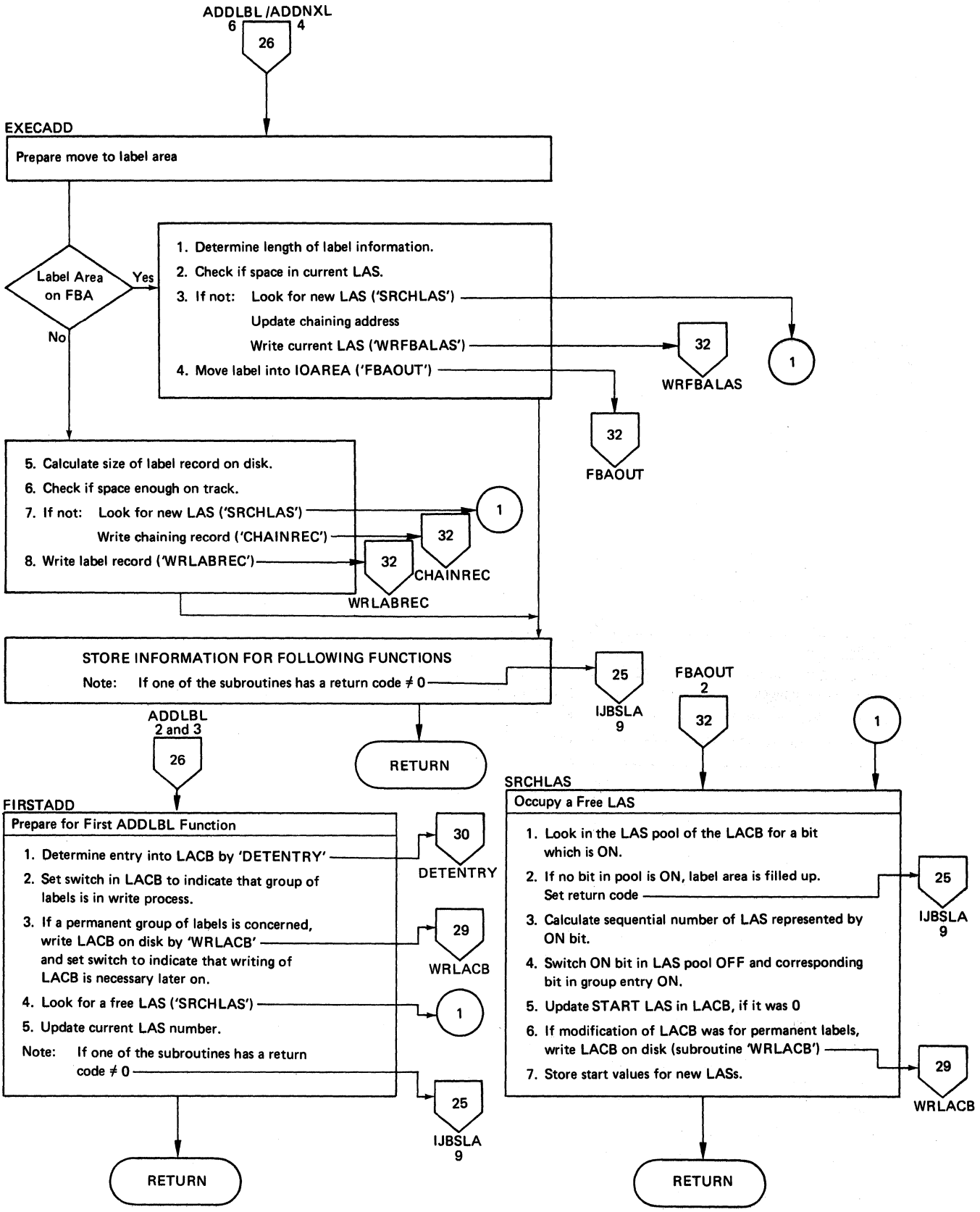


Chart 32. $IJBSLA - Routines

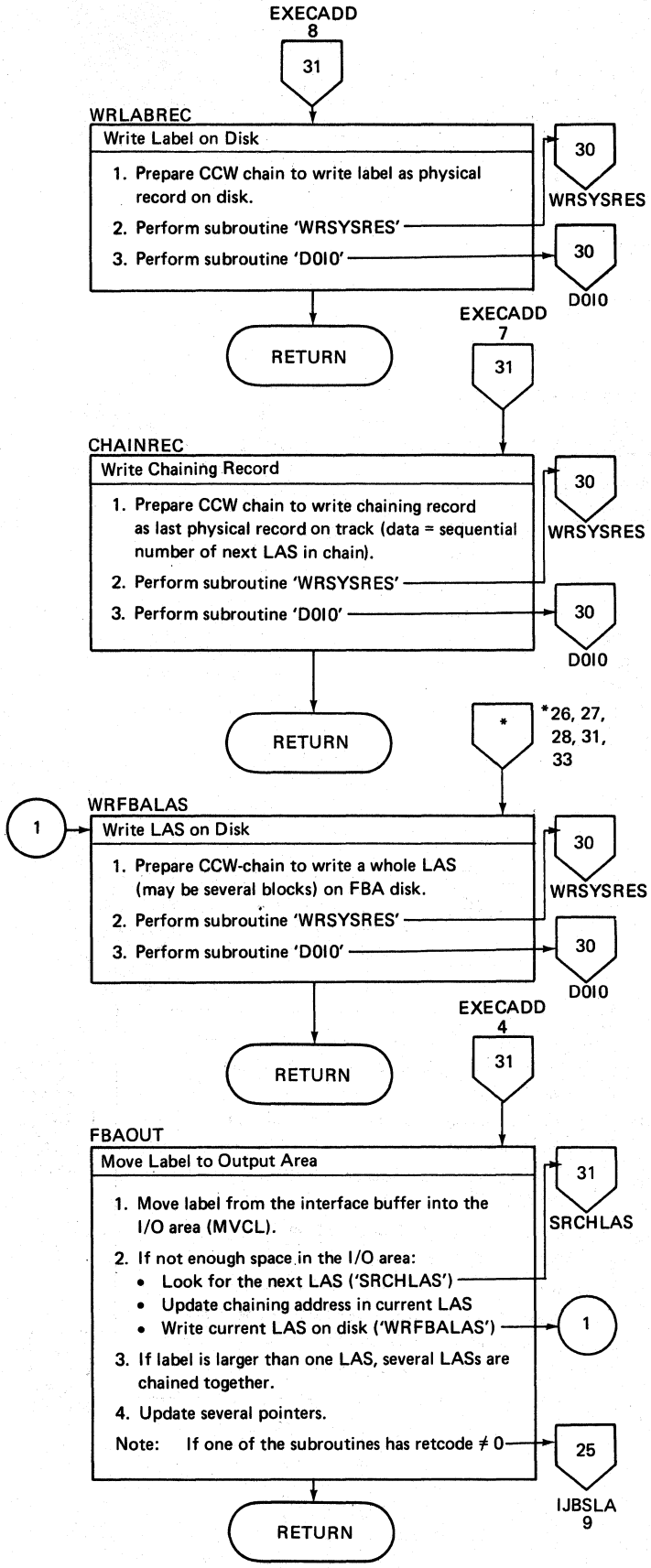


Chart 33. $IJBSLA - Routines

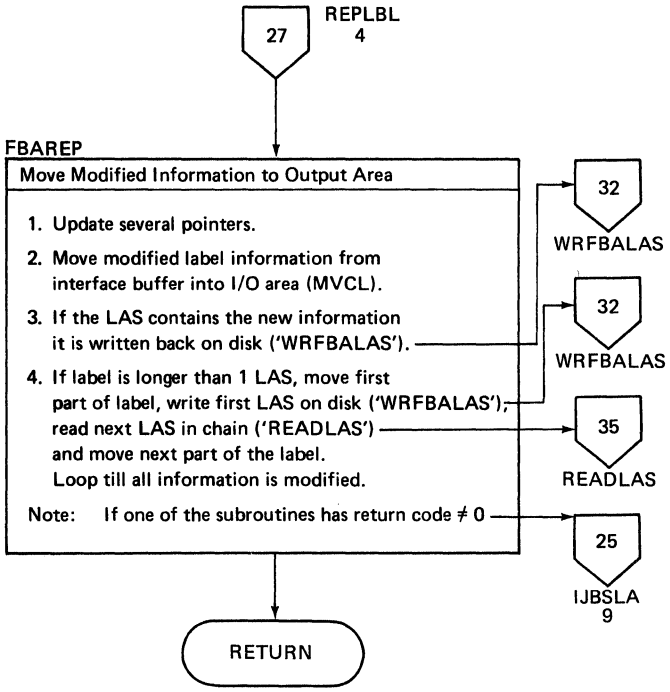


Chart 34. $IJBSLA - Routines

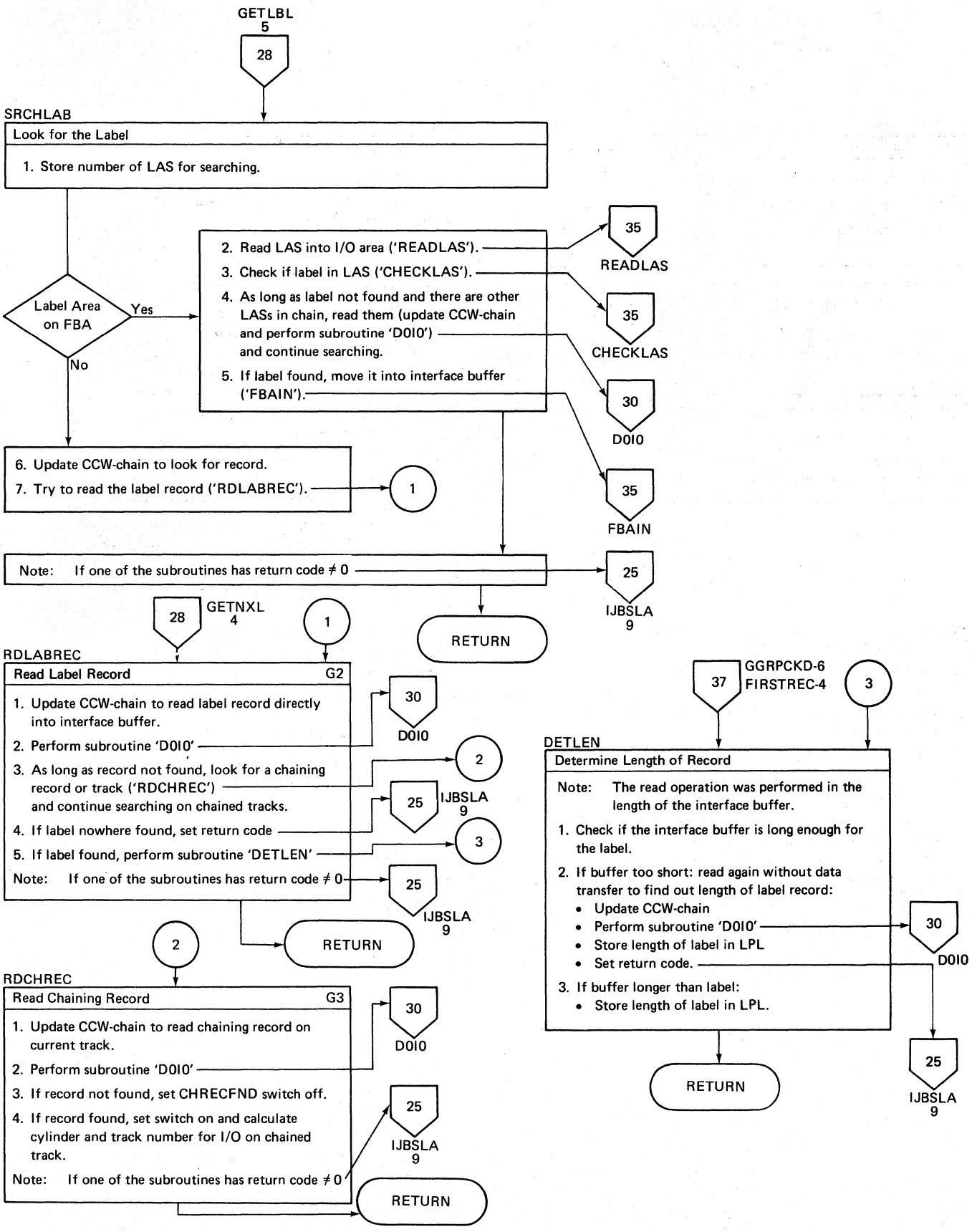


Chart 35. $IJBSLA - Routines

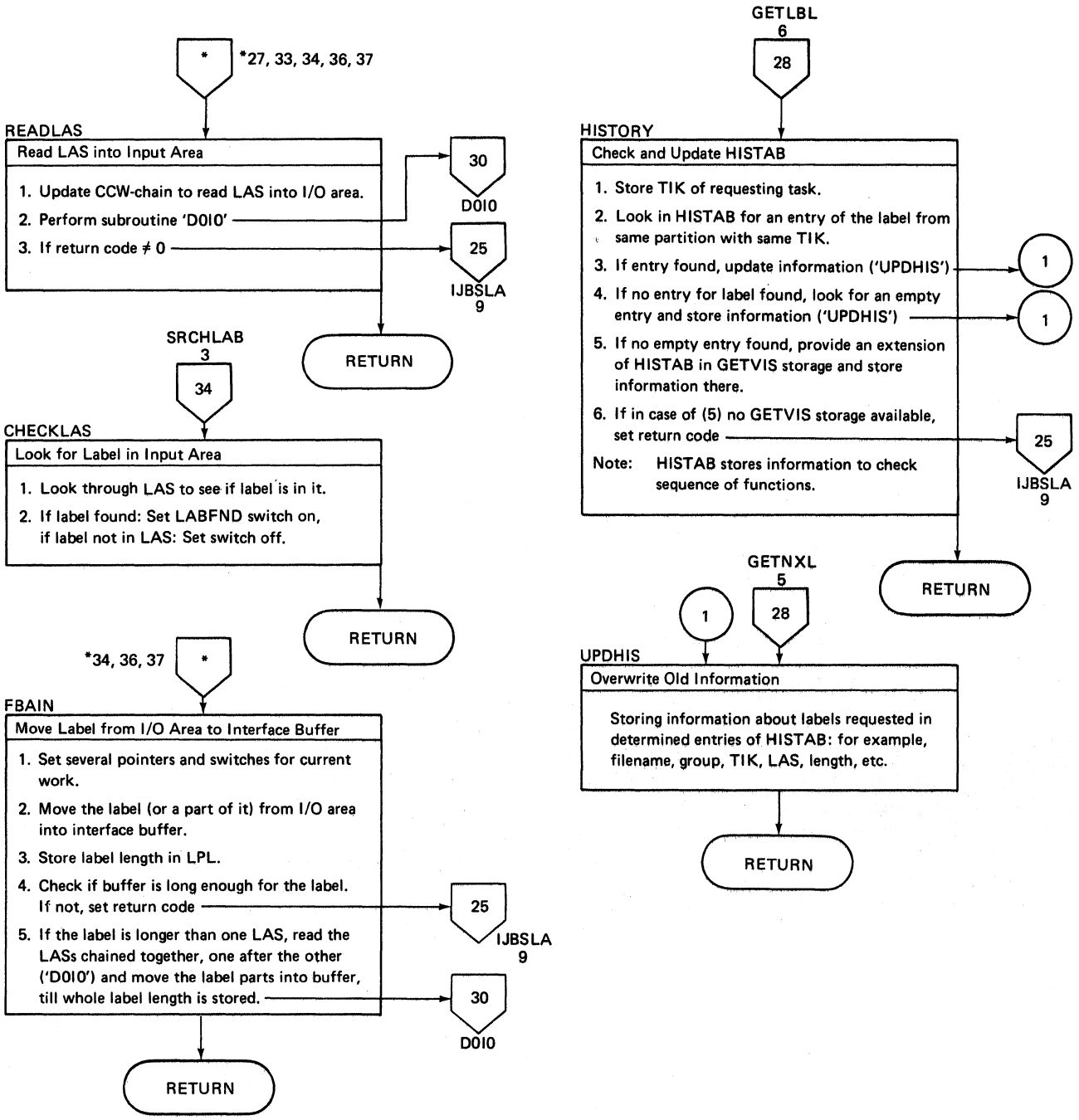


Chart 36. $IJBSLA - Routines

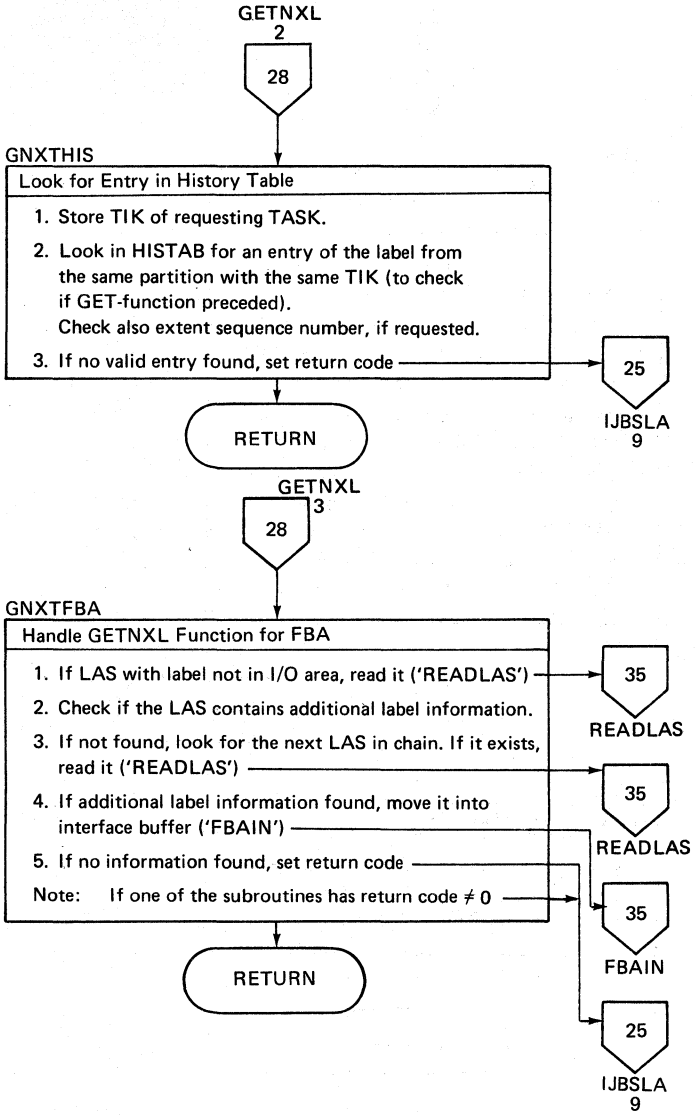
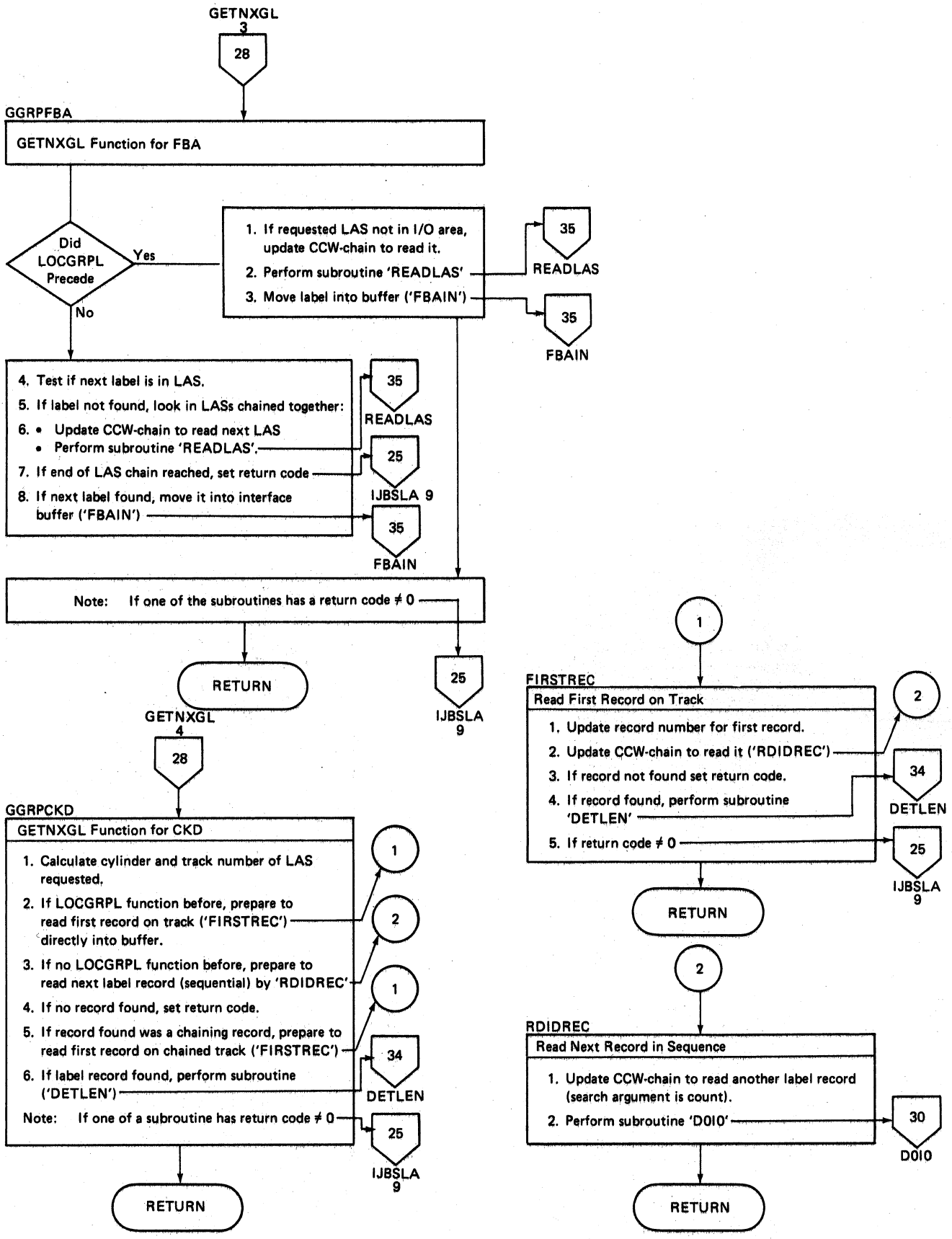
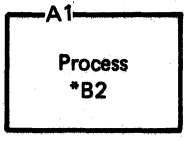


Chart 37. $IJBSLA - Routines



Explanation of Flowchart Symbols

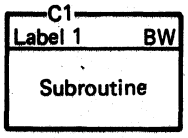


DESCRIPTION

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

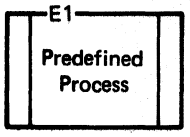
*B2

If any additional explanation is required, its location on the chart is identified by an asterisk and the block ID.

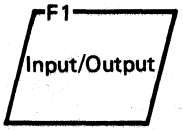


Description of a subroutine. The starting label of the routine appears above the stripe. If the subroutine is documented in detail on another flowchart, the ID of this flowchart is also shown.

An instruction, or group of instructions, that changes portions of a routine or initializes a routine for given conditions.

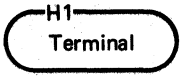


A group of operations not detailed in the flowcharts in this manual, such as user's routines.



Any function of an input/output device or program, usually branching to an I/O routine to perform the function stated in the block.

Points where the program branches to alternate processing, based upon variable conditions such as program switch settings and test results.



The beginning, end or point of interruption in a program.

On-page connector. An entry from or an exit to another function on the same flowchart. The number in the connector identifies the corresponding entry or exit on the chart.

Off-page connector, an entry from, or an exit to, a given point on another flowchart. The characters in the connector identify the chart and block. The corresponding label, if any, is placed outside the connector. For multiple entries and exits, an asterisk appears in the connector and the characters are listed nearby.

EXAMPLE

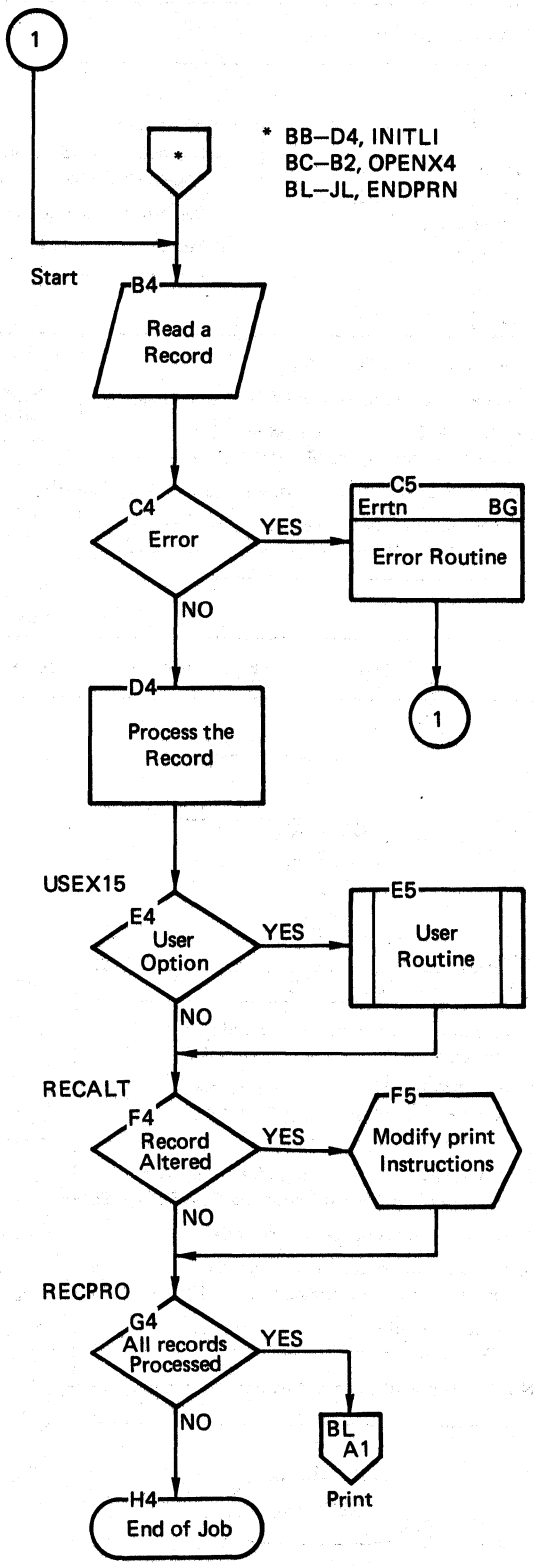


Chart AA. $$A$IPL1 - IPL Bootstrap Loader, CKD SYSRES
Refer to Chart 01.

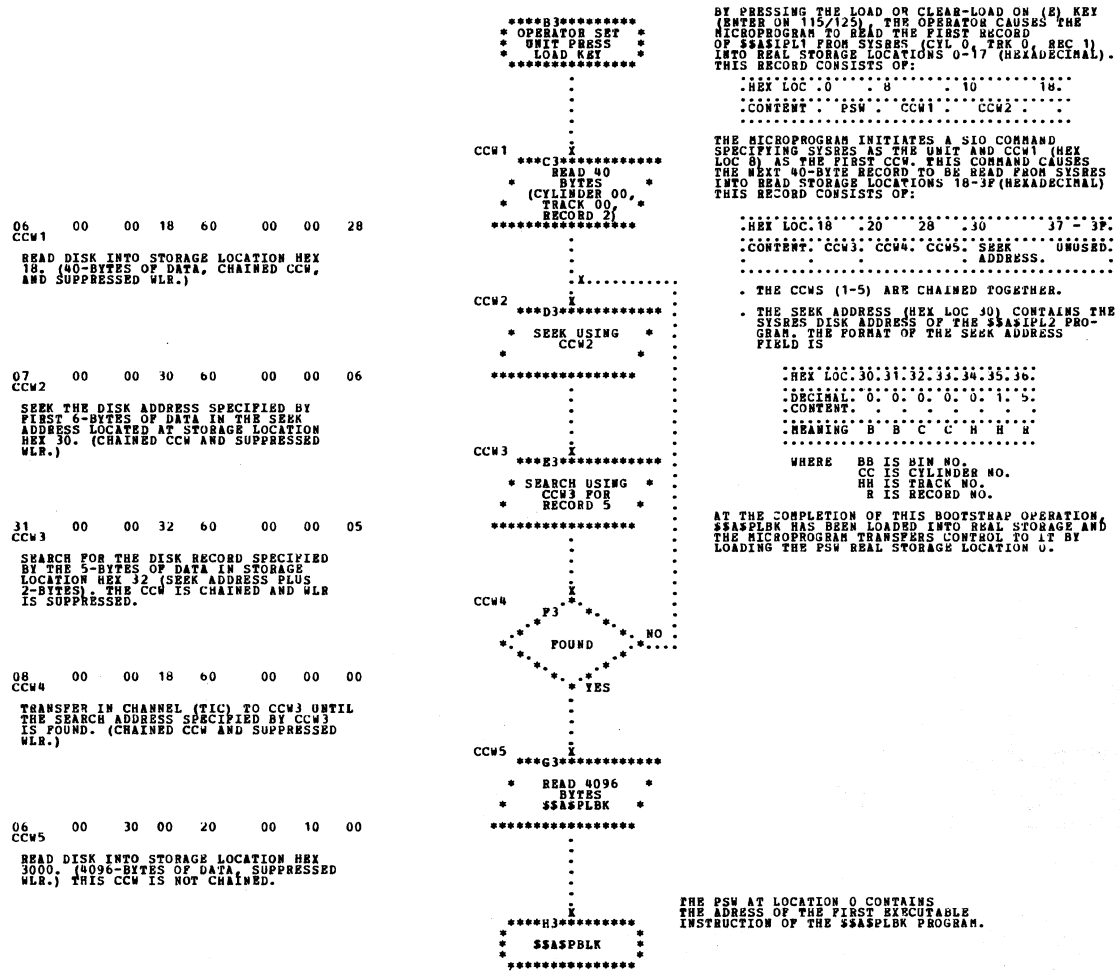


Chart AB. $$A$IPL0 - IPL Bootstrap Loader, Fixed Block SYSRES
Refer to Chart 01.

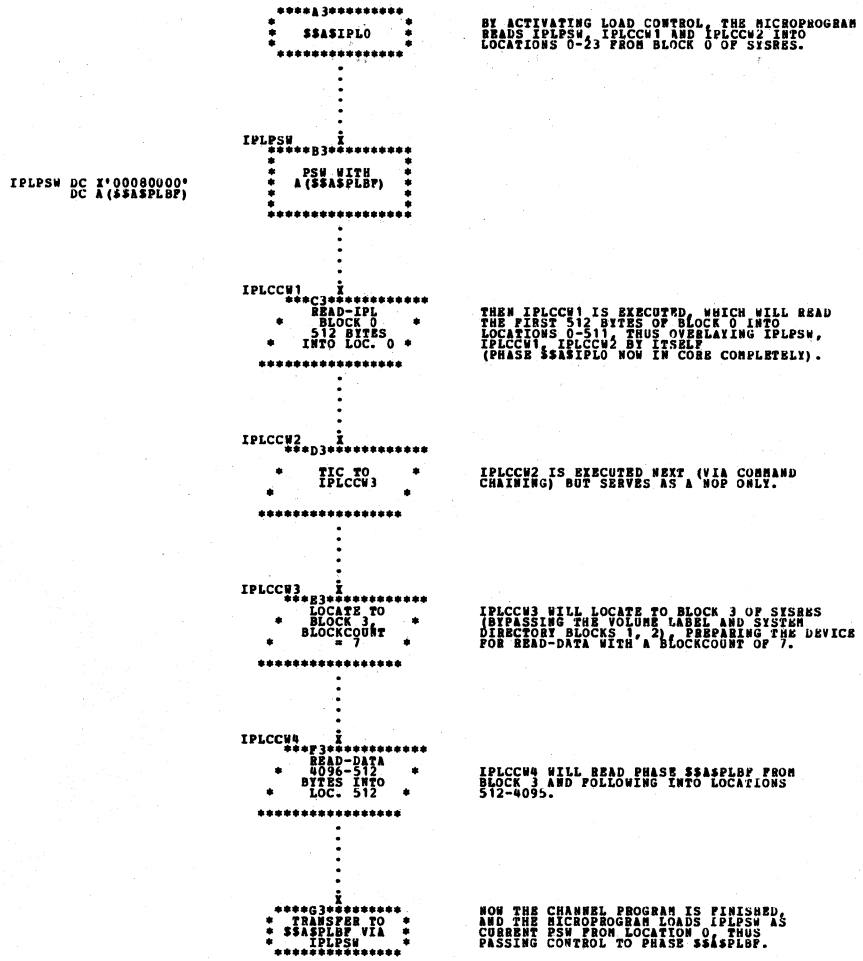


Chart AC. $$A$PLBK - Bootstrap Program, CKD SYSRES
Refer to Chart 01.

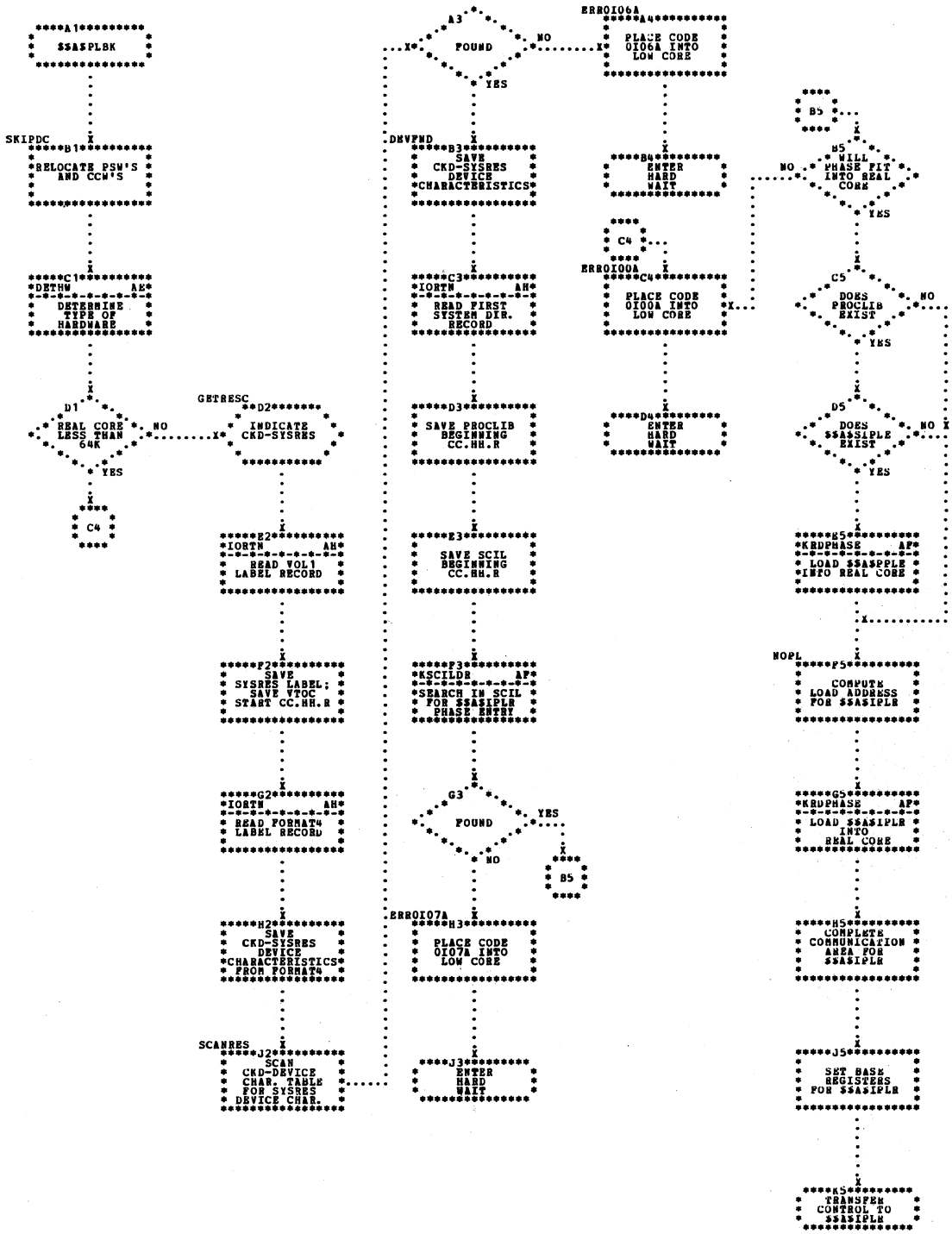


Chart AD. $$A$PLBF - Bootstrap Program, Fixed Block SYSRES
Refer to Chart 01.

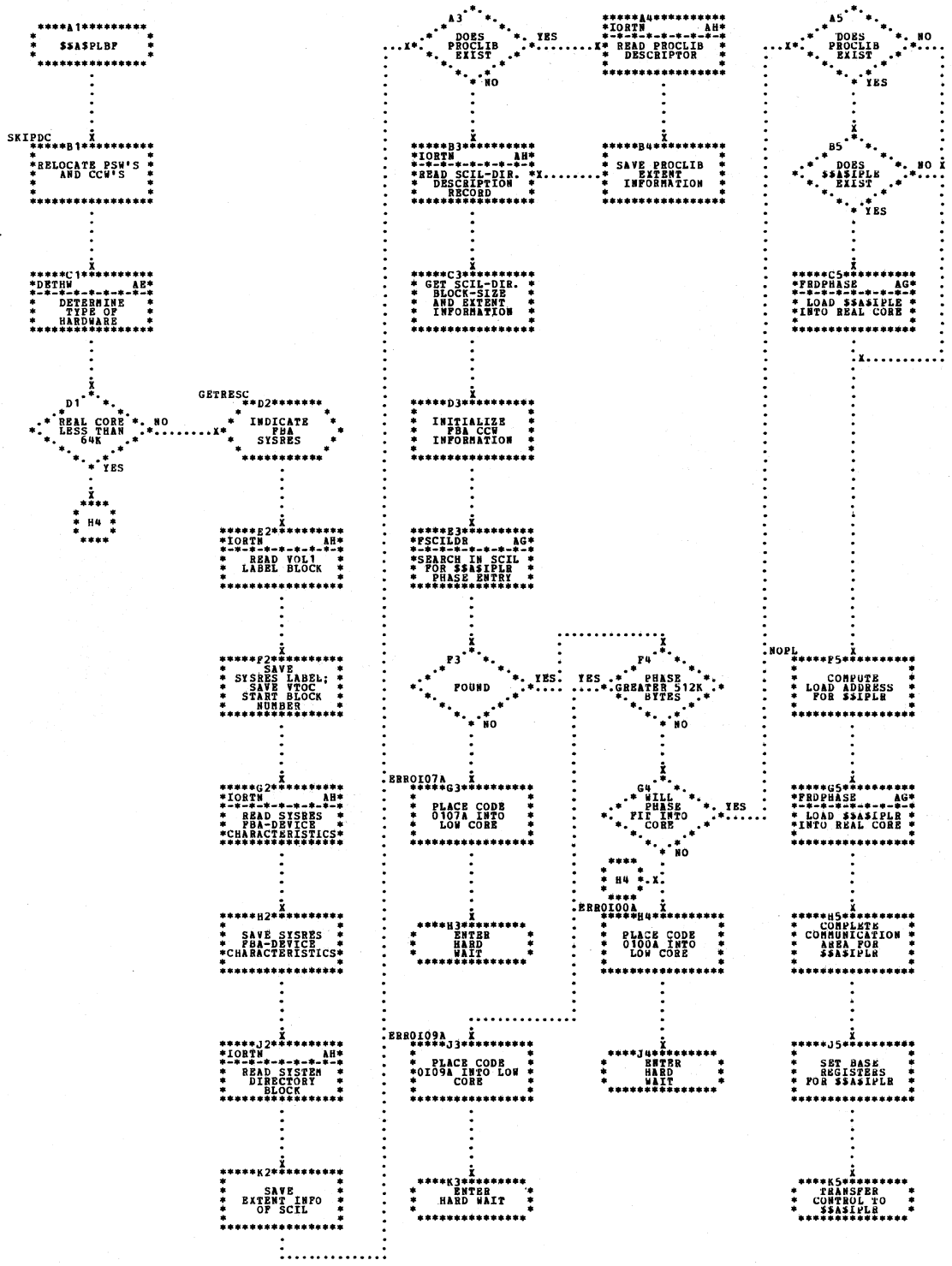


Chart AE. DETHW (Subroutine of Macro IPLBMAC) - Determine Type of Hardware

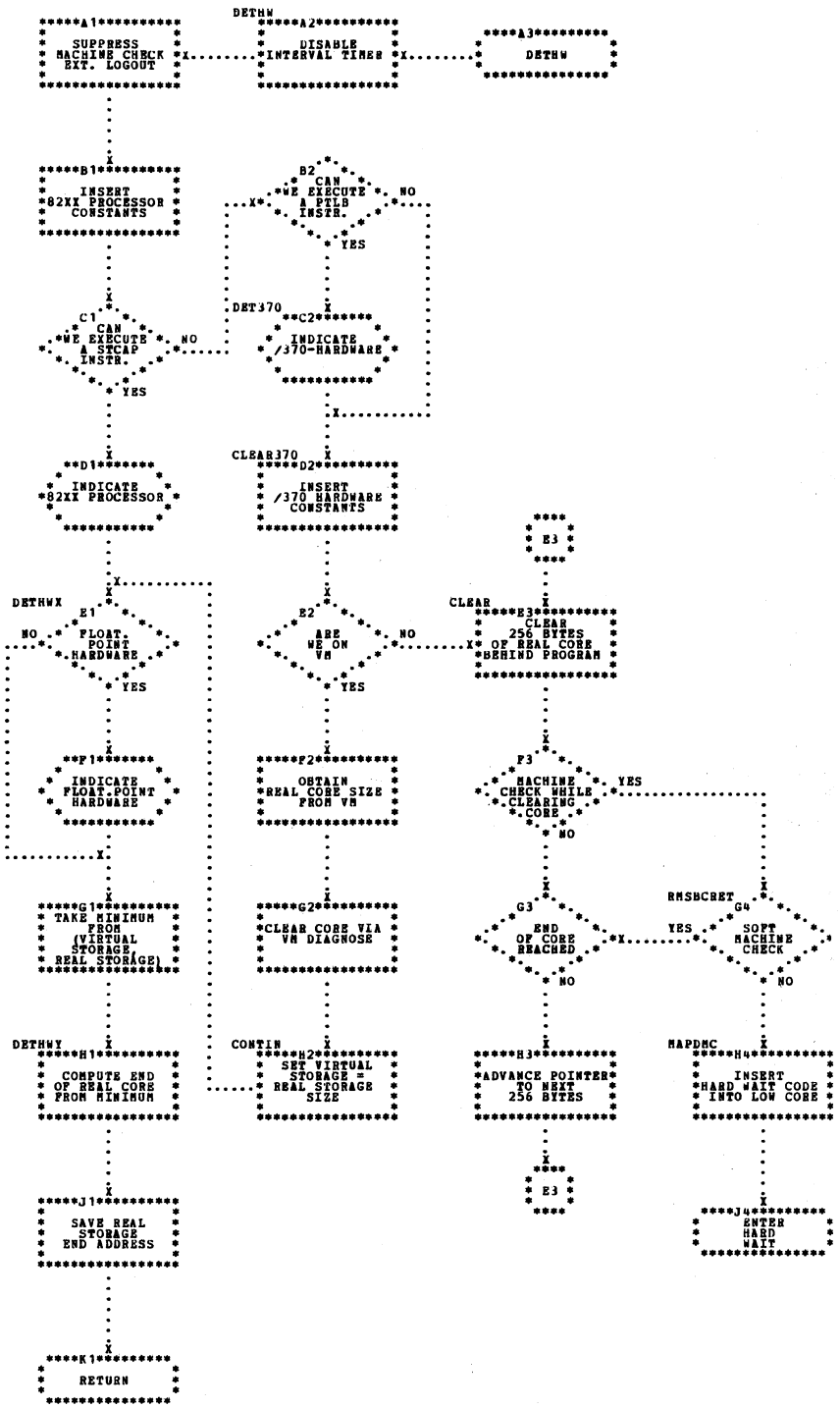


Chart AF. IPLBMAC - CKD Routines

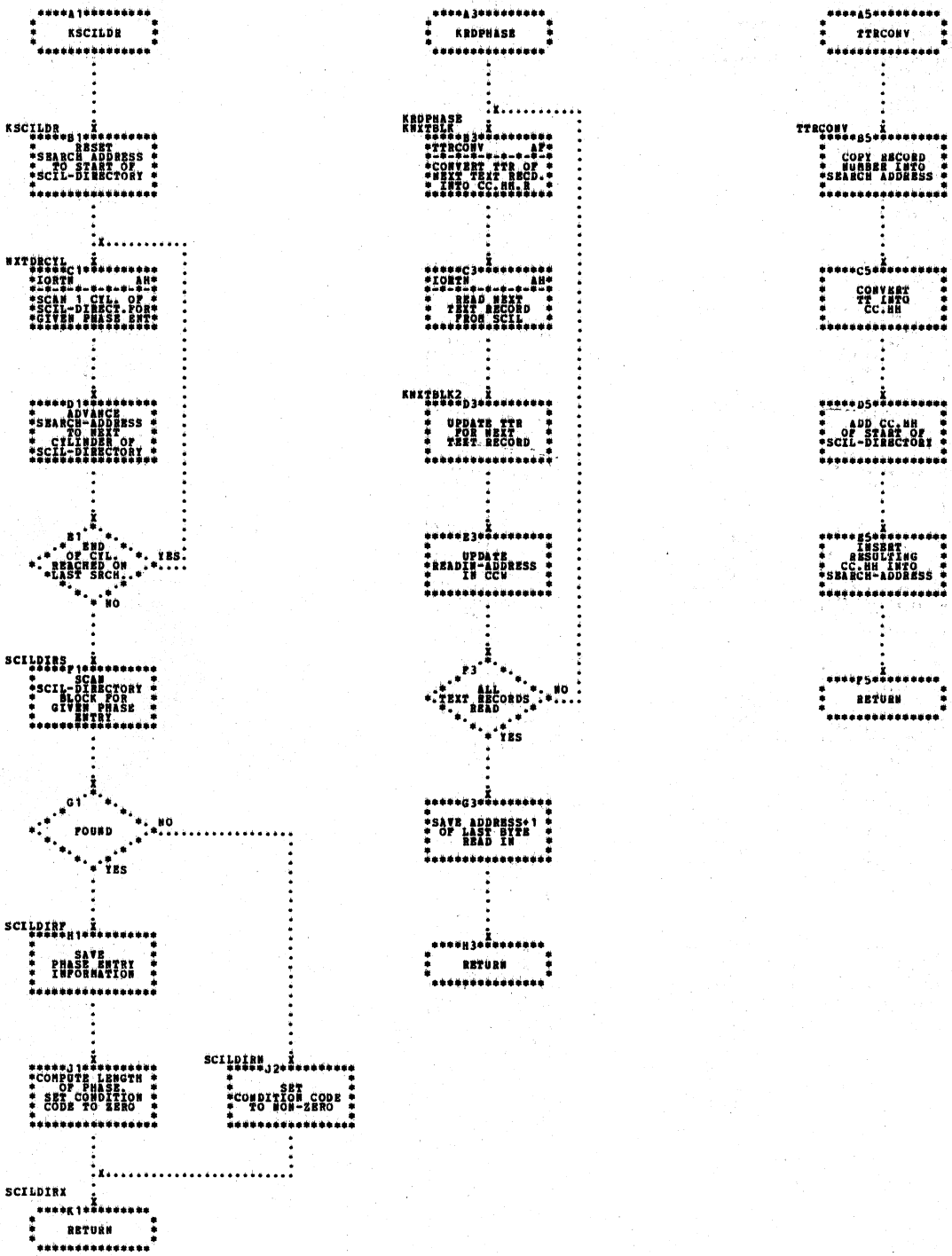


Chart AG. IPLBMAC - Fixed Block Routines

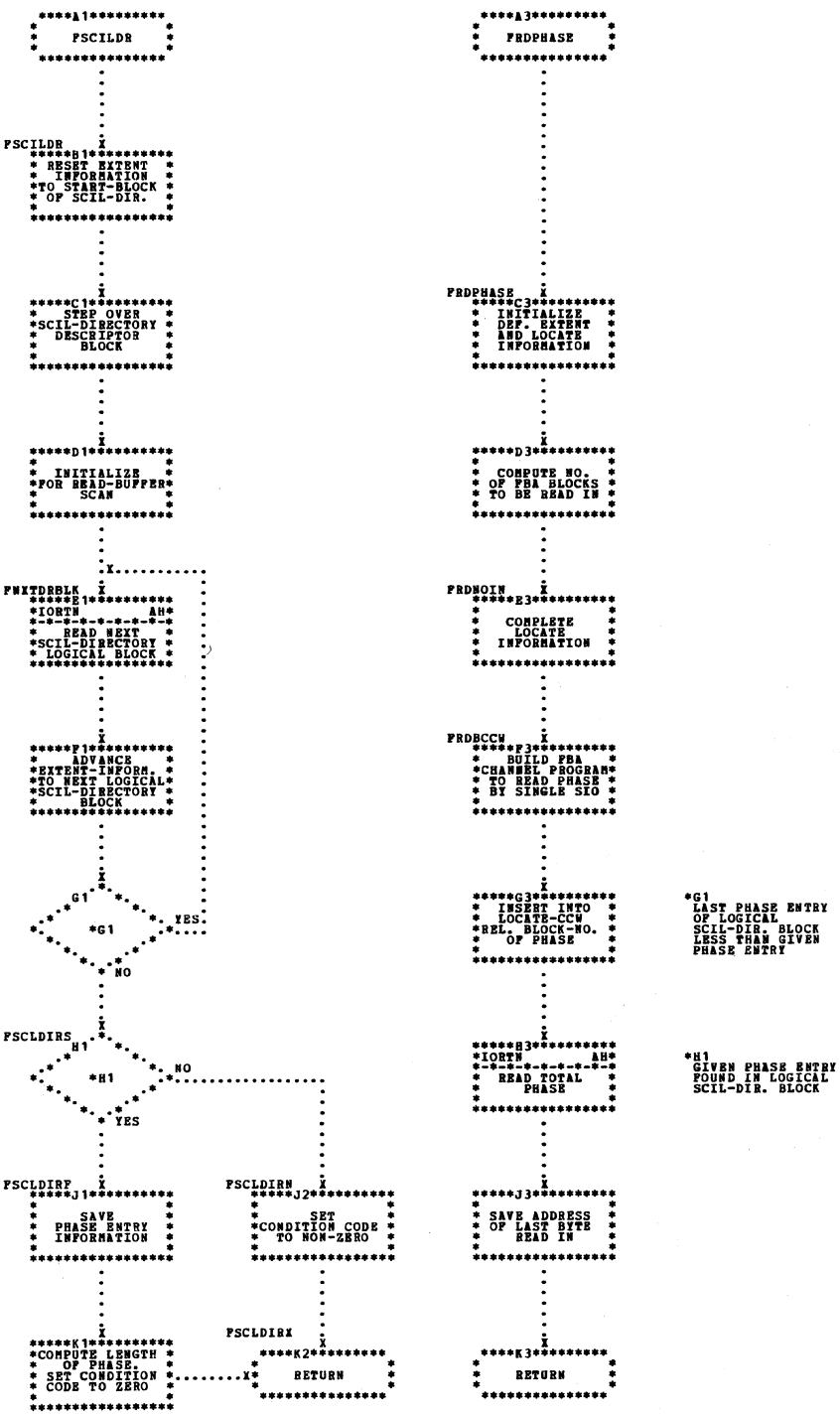


Chart AH. IPLBMAC - Common I/O Subroutine

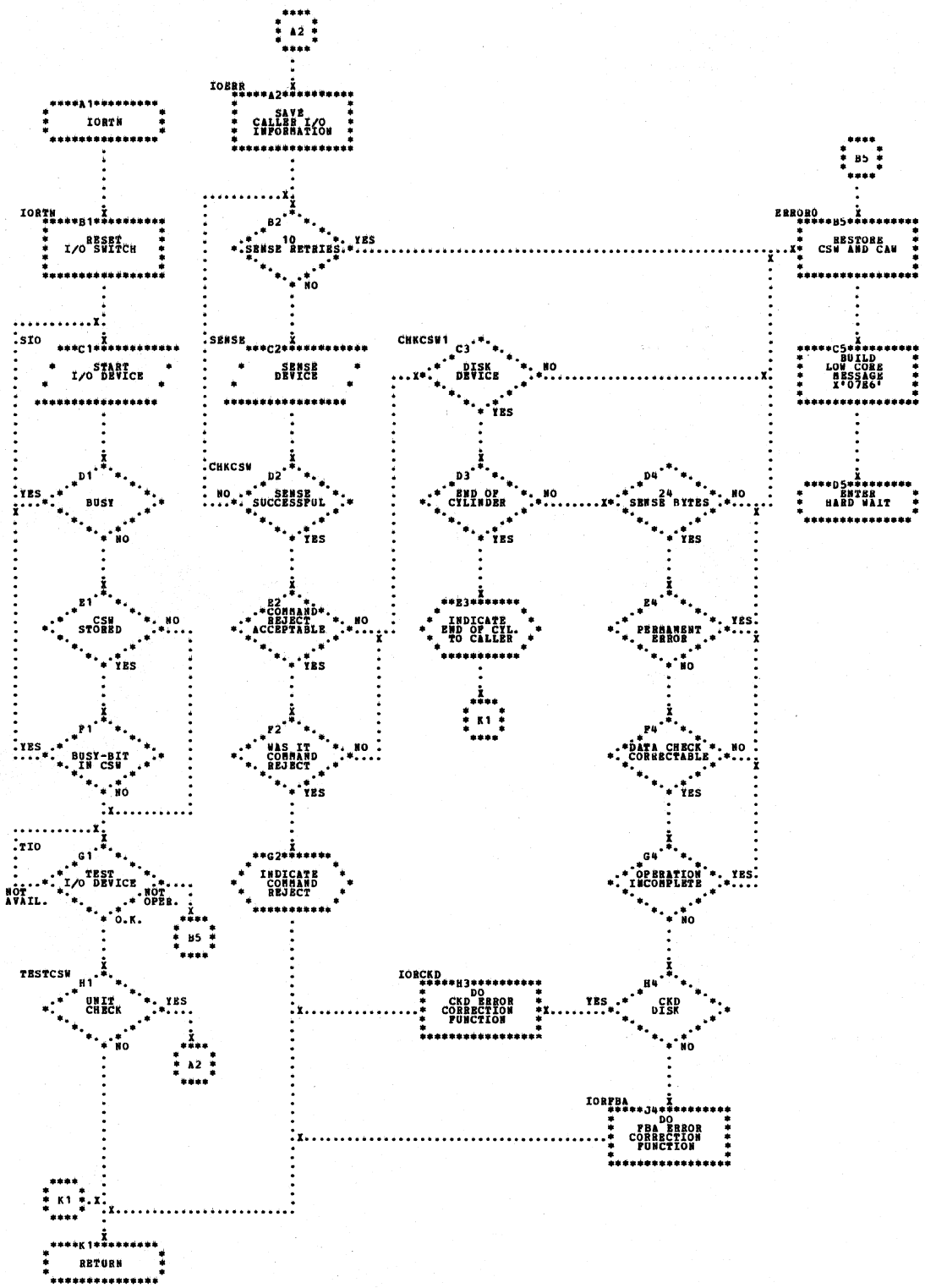


Chart BA. $$A$IPLR - Initialization
Refer to Chart 01.

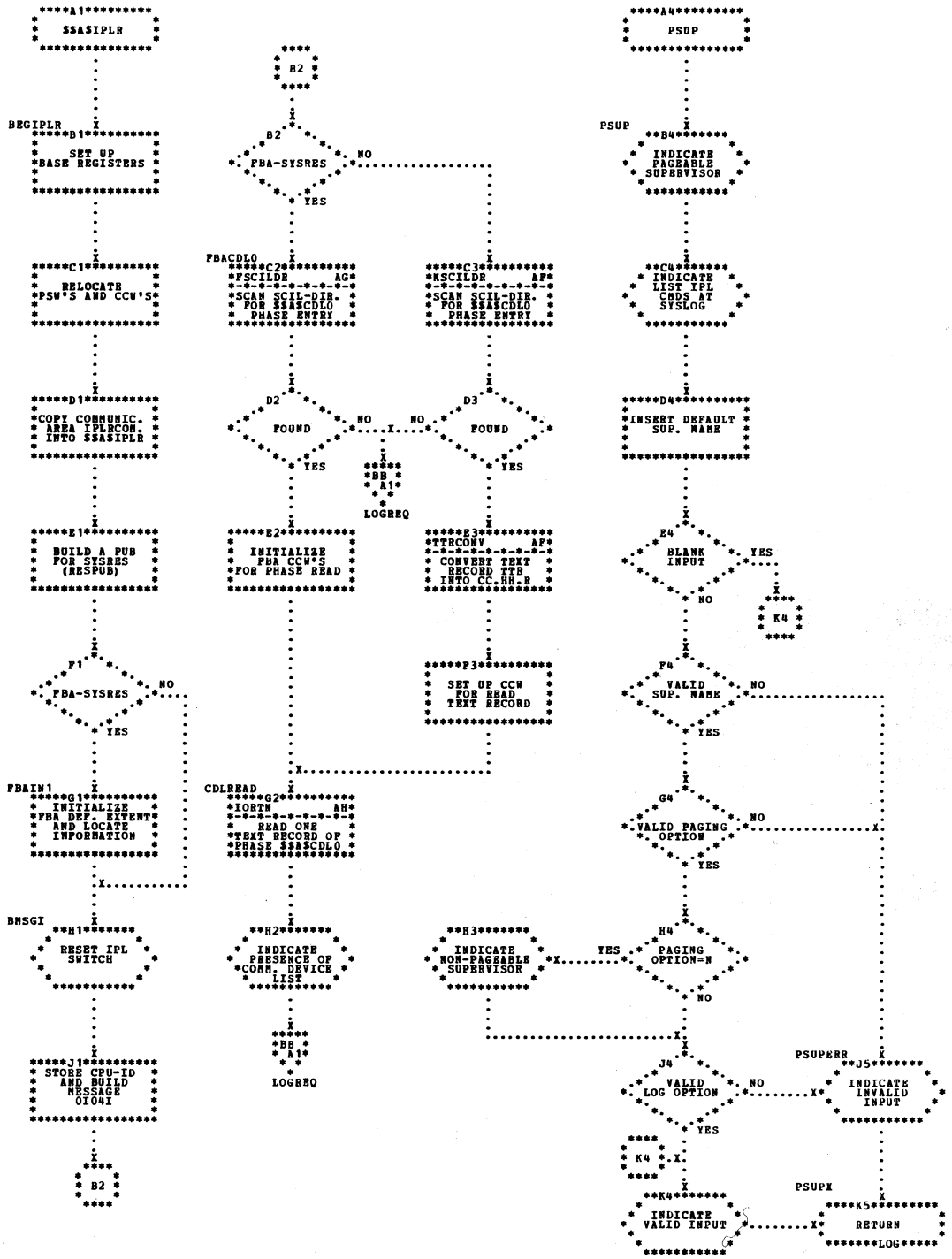


Chart BB. $$A$IPLR - Load Selected Supervisor
Refer to Chart 01.

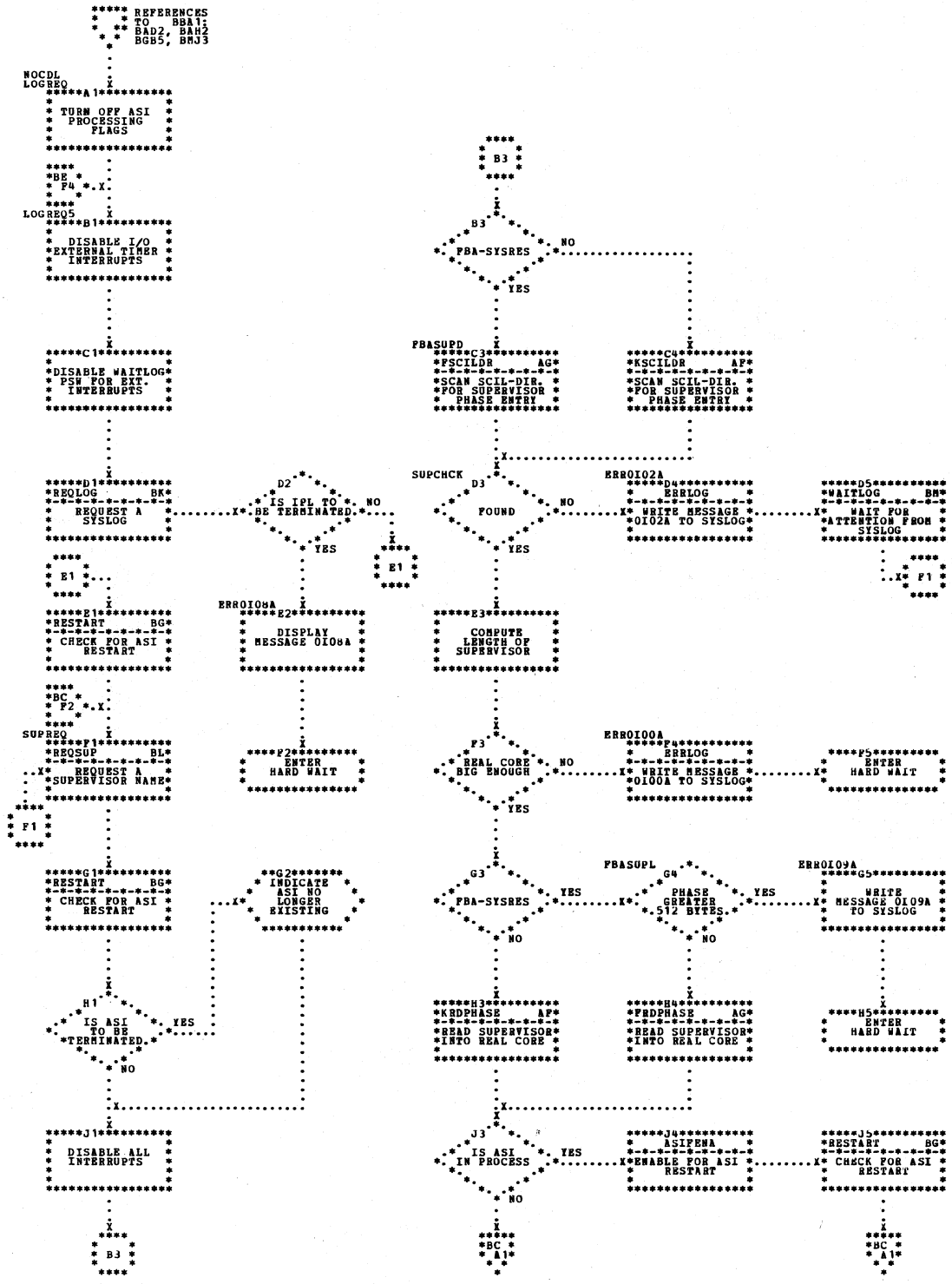


Chart BC. $$A$IPLR - Check Supervisor Generation Options. Initiate System.
Refer to Chart 01.

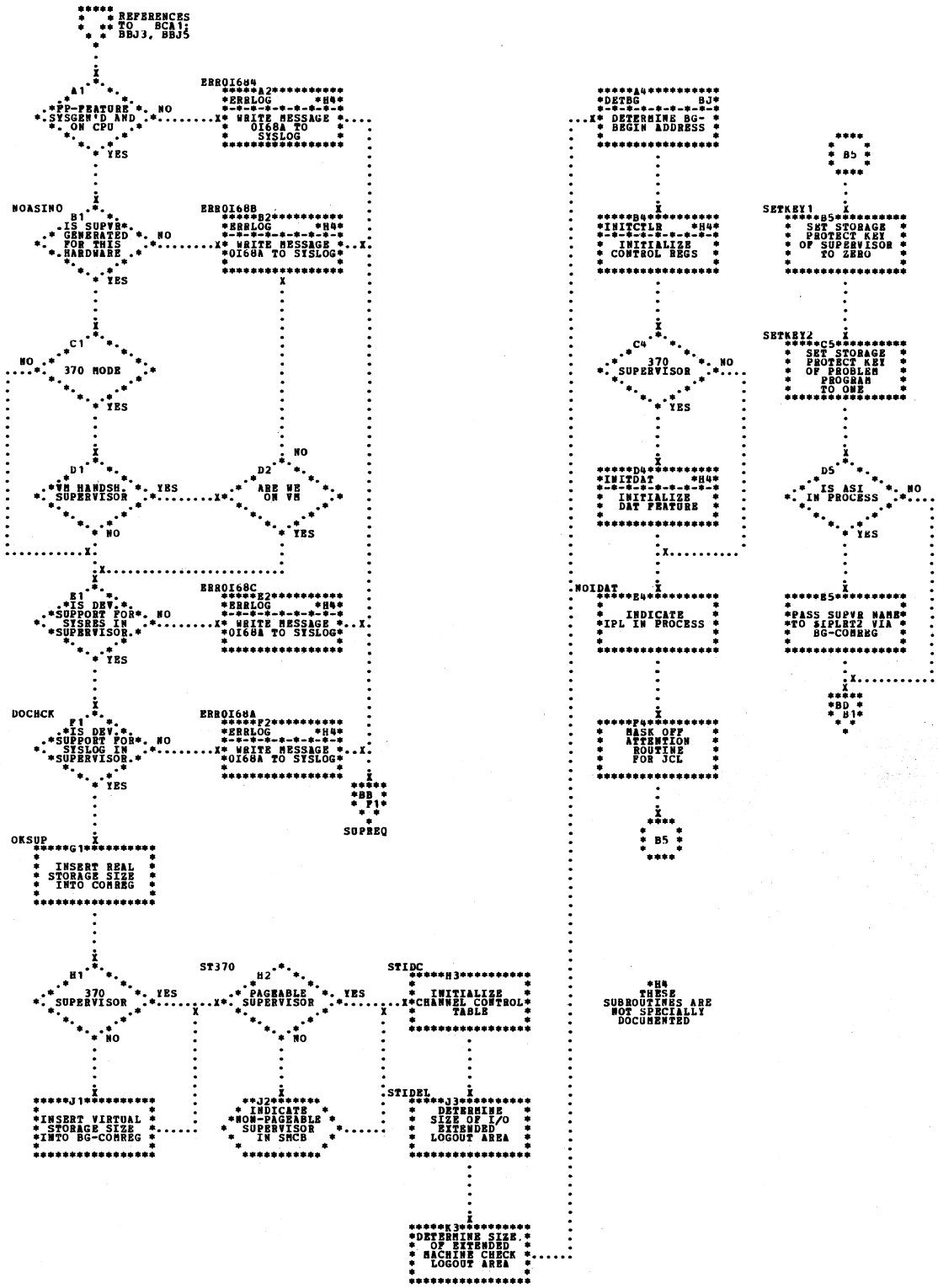


Chart BD. $$A$IPLR - Build Communication Area and 3-Device System
Refer to Chart 01.

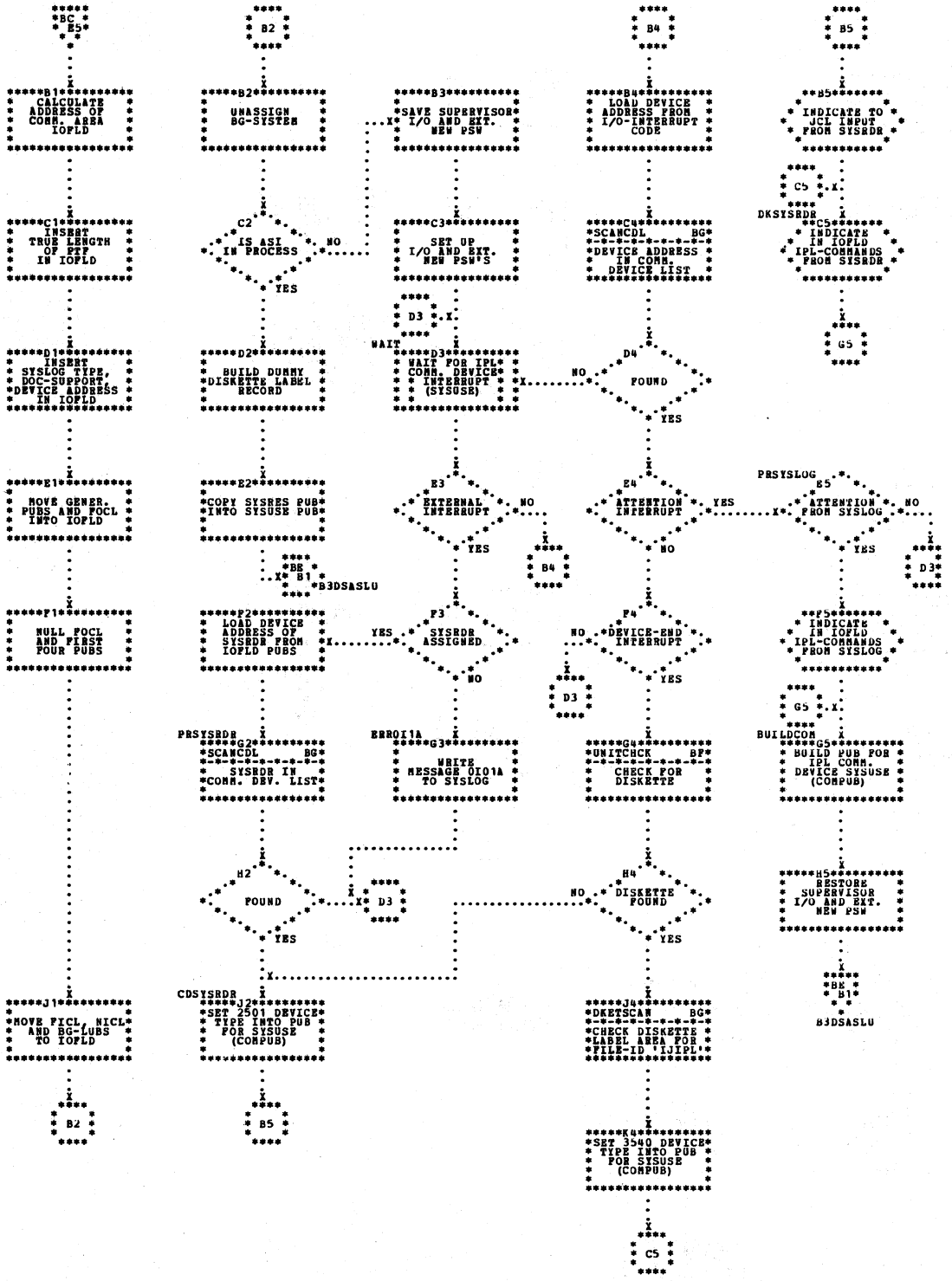


Chart BE. $$A$IPLR - Initiate Supervisor Components.
Pass Control to Command Processing in Supervisor State.

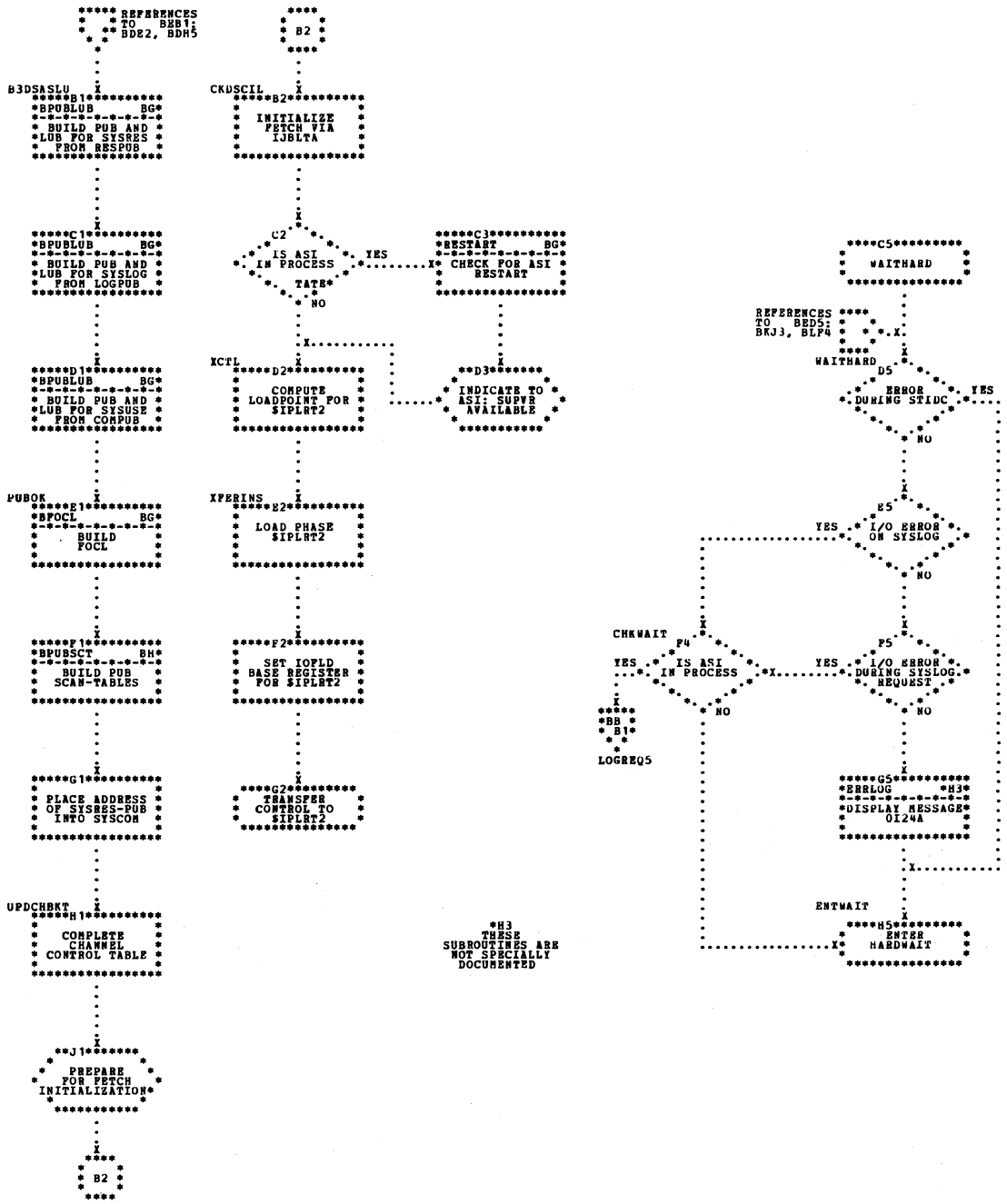


Chart BF. $$A$IPLR - Subroutines
Refer to Chart 01.

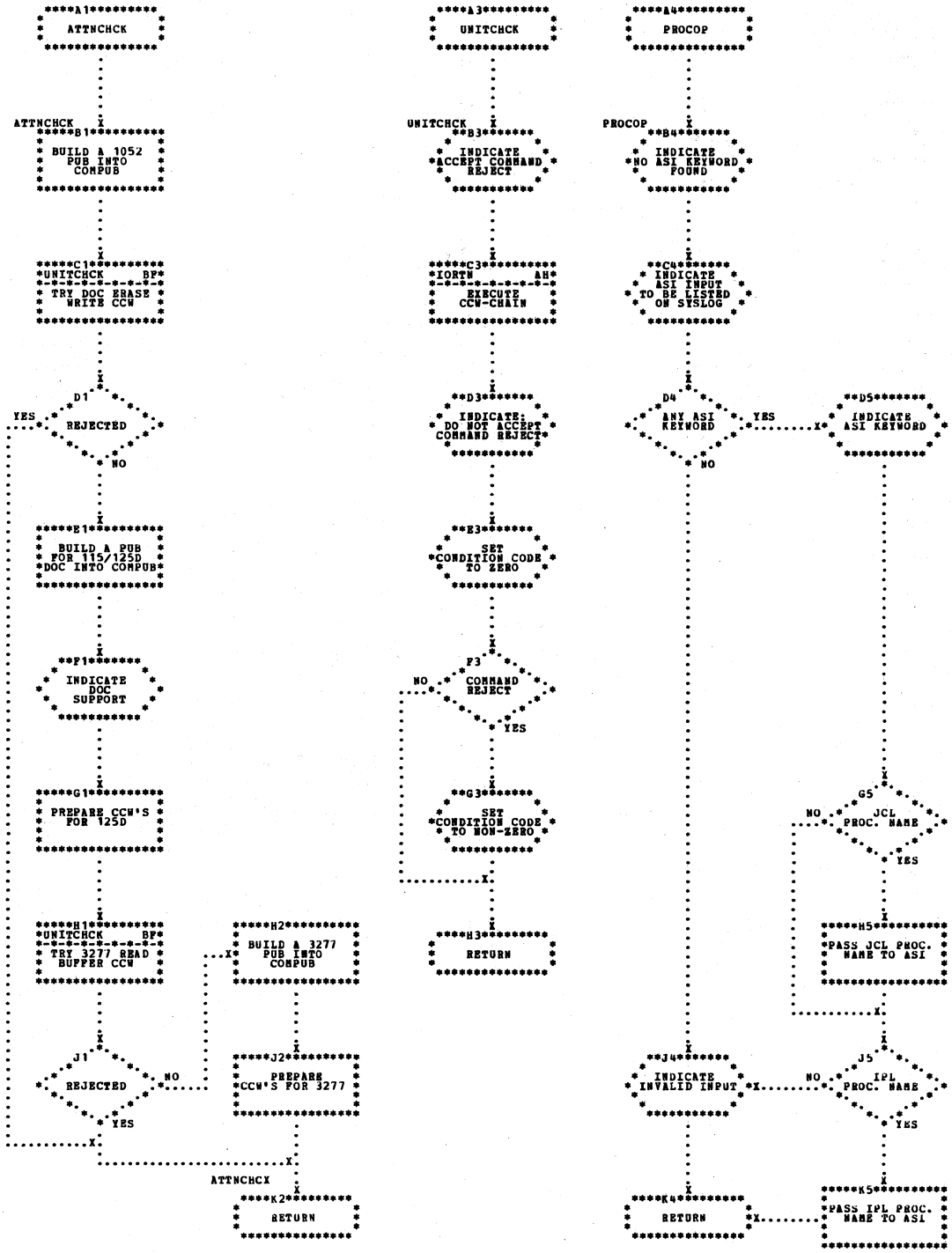


Chart BG. $$A$IPLR - Subroutines
Refer to Chart 01.

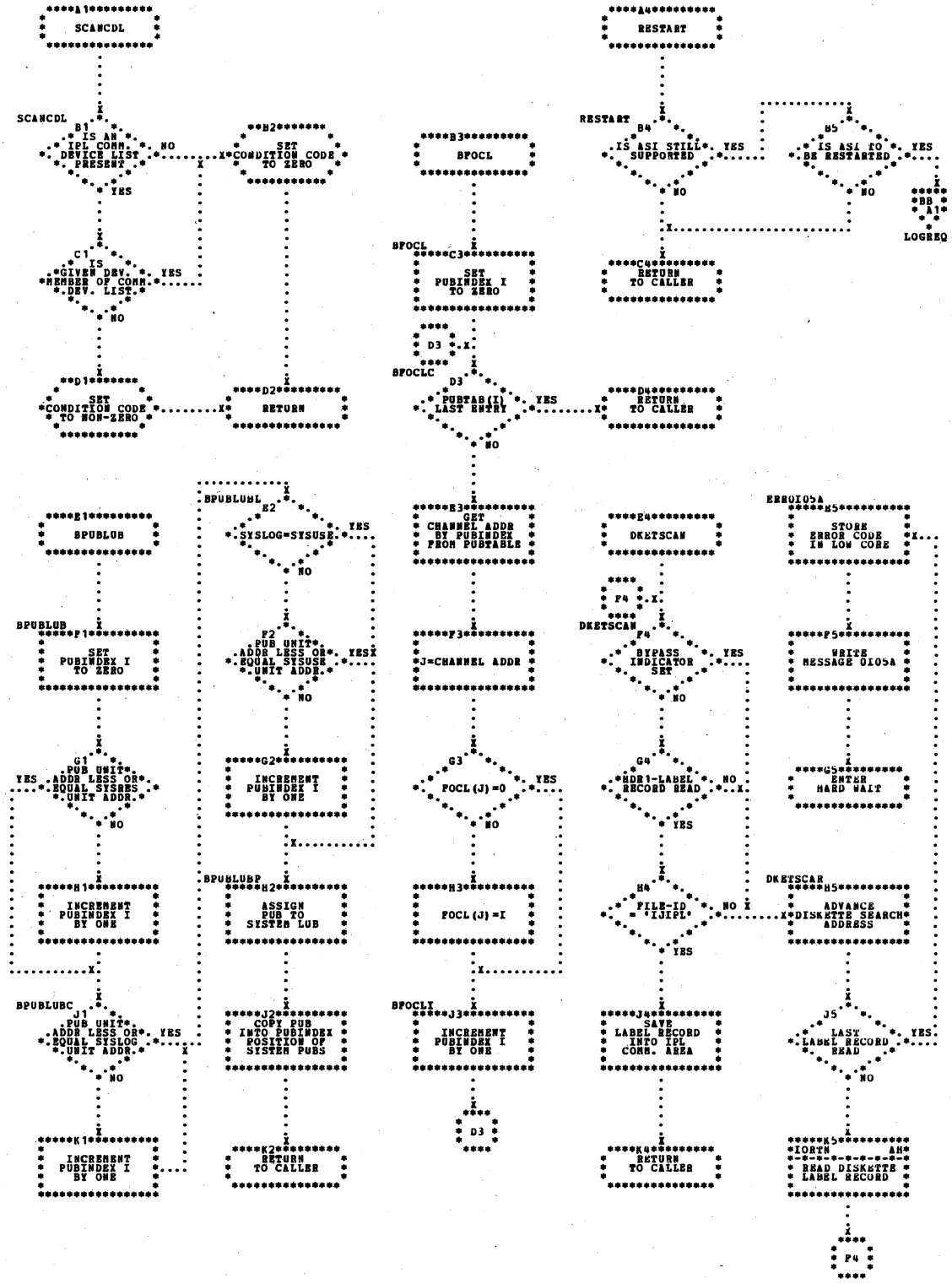


Chart BH. $$A$IPLR - Build PUB Scan Tables
Refer to Chart 01.

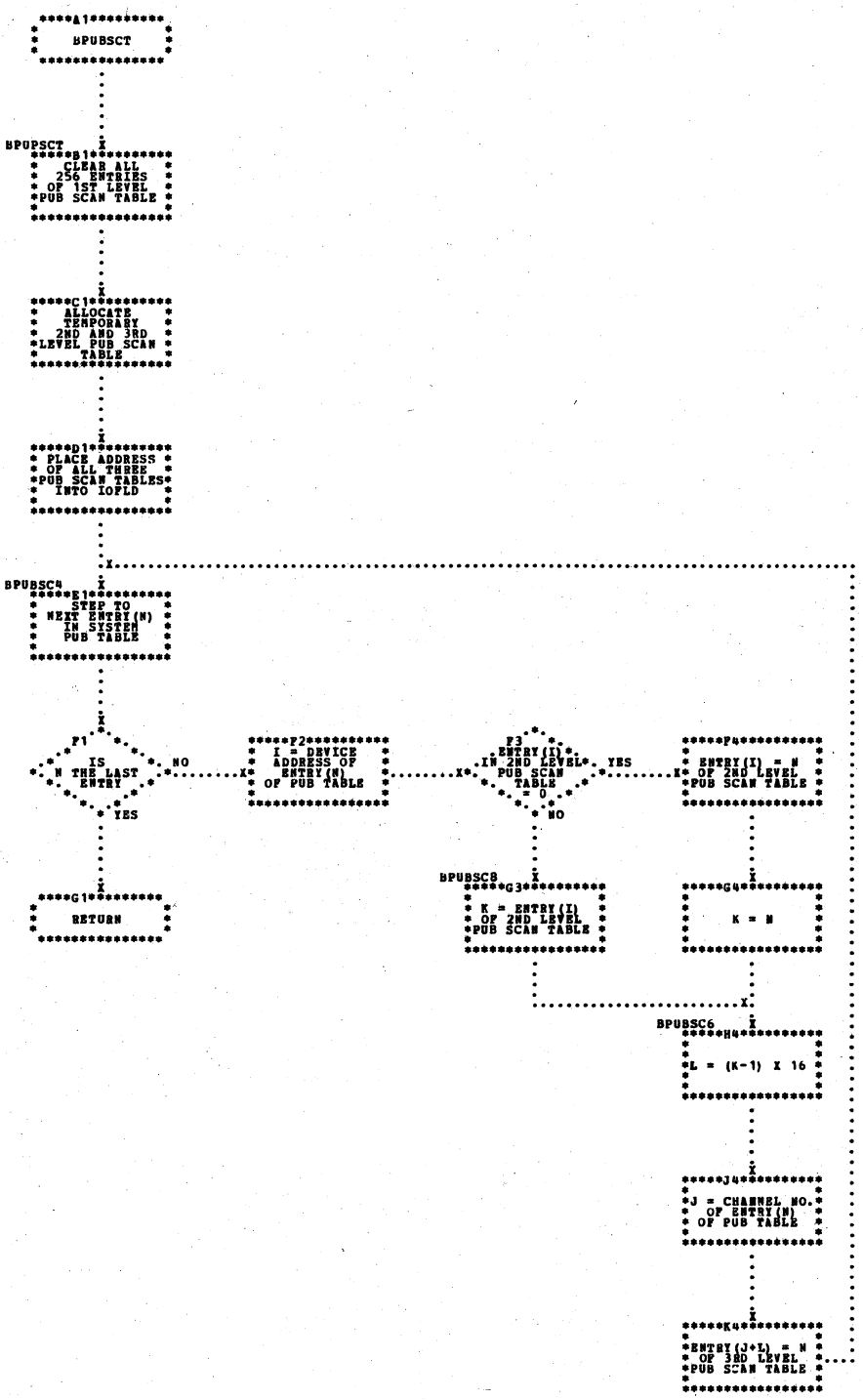


Chart BJ. $$A$IPLR - Determining Background Start
Refer to Chart 01.

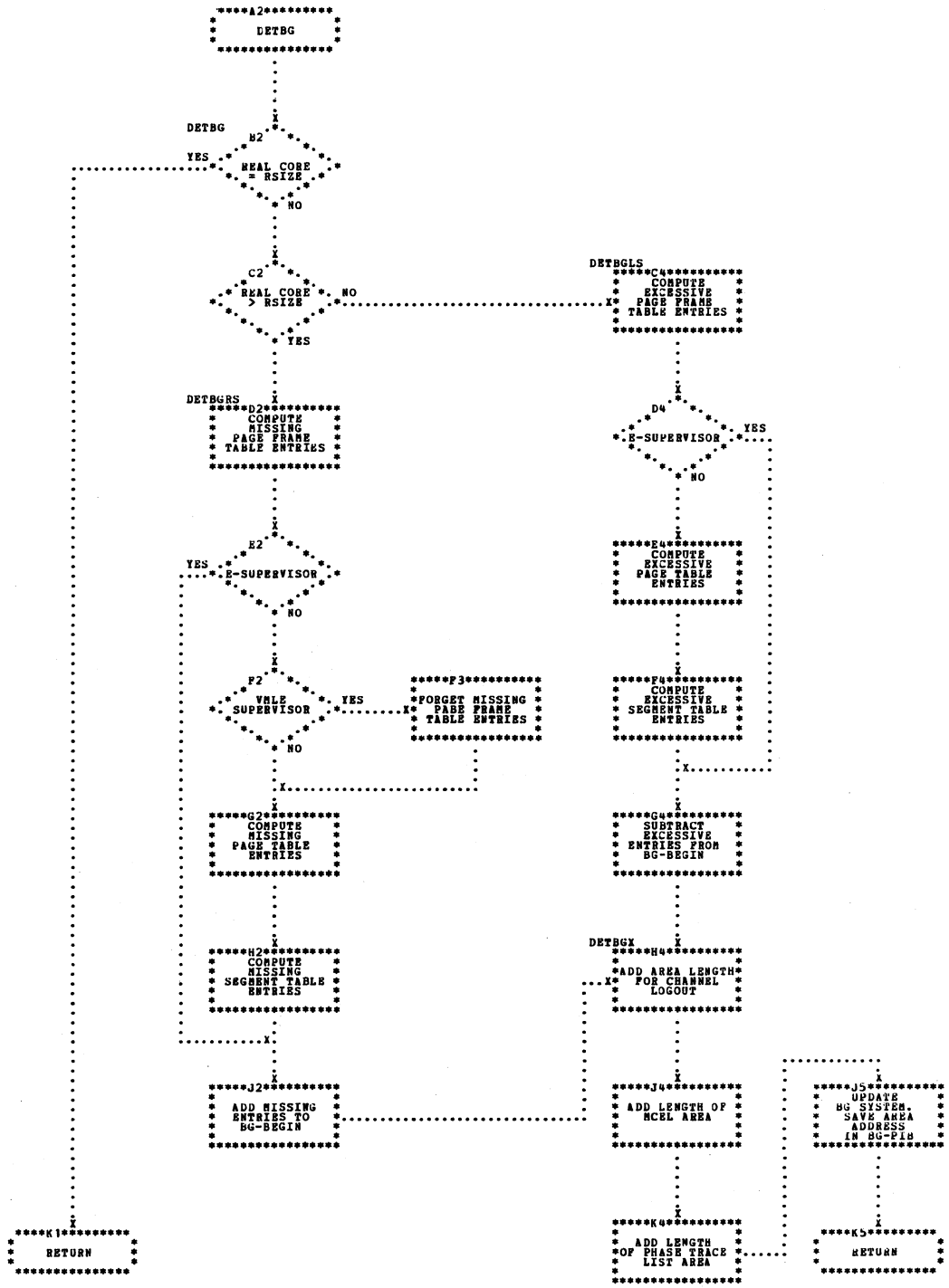


Chart BK. $$A$IPLR - Request SYSLOG

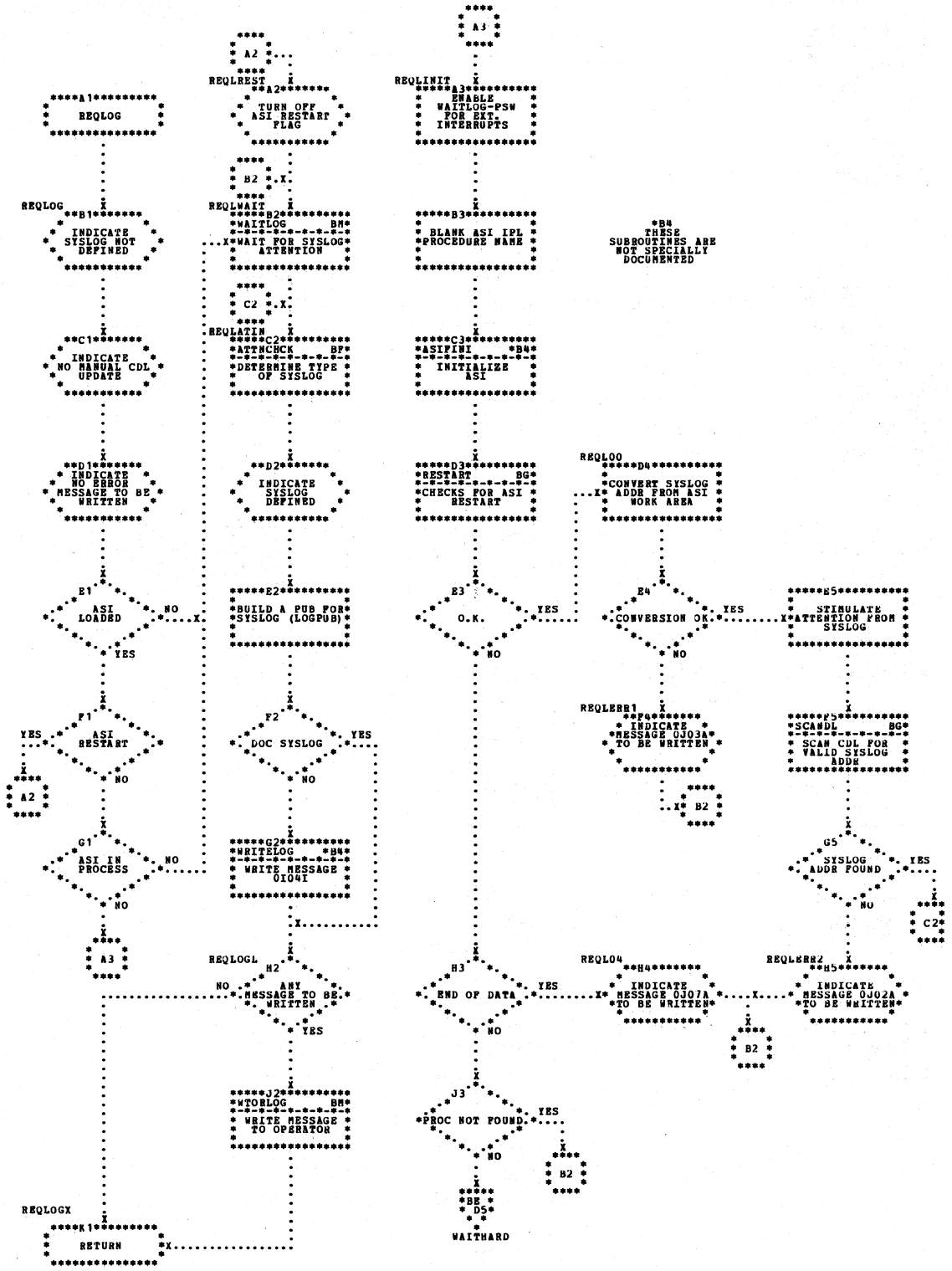


Chart BL. $$A$IPLR - Request Supervisor

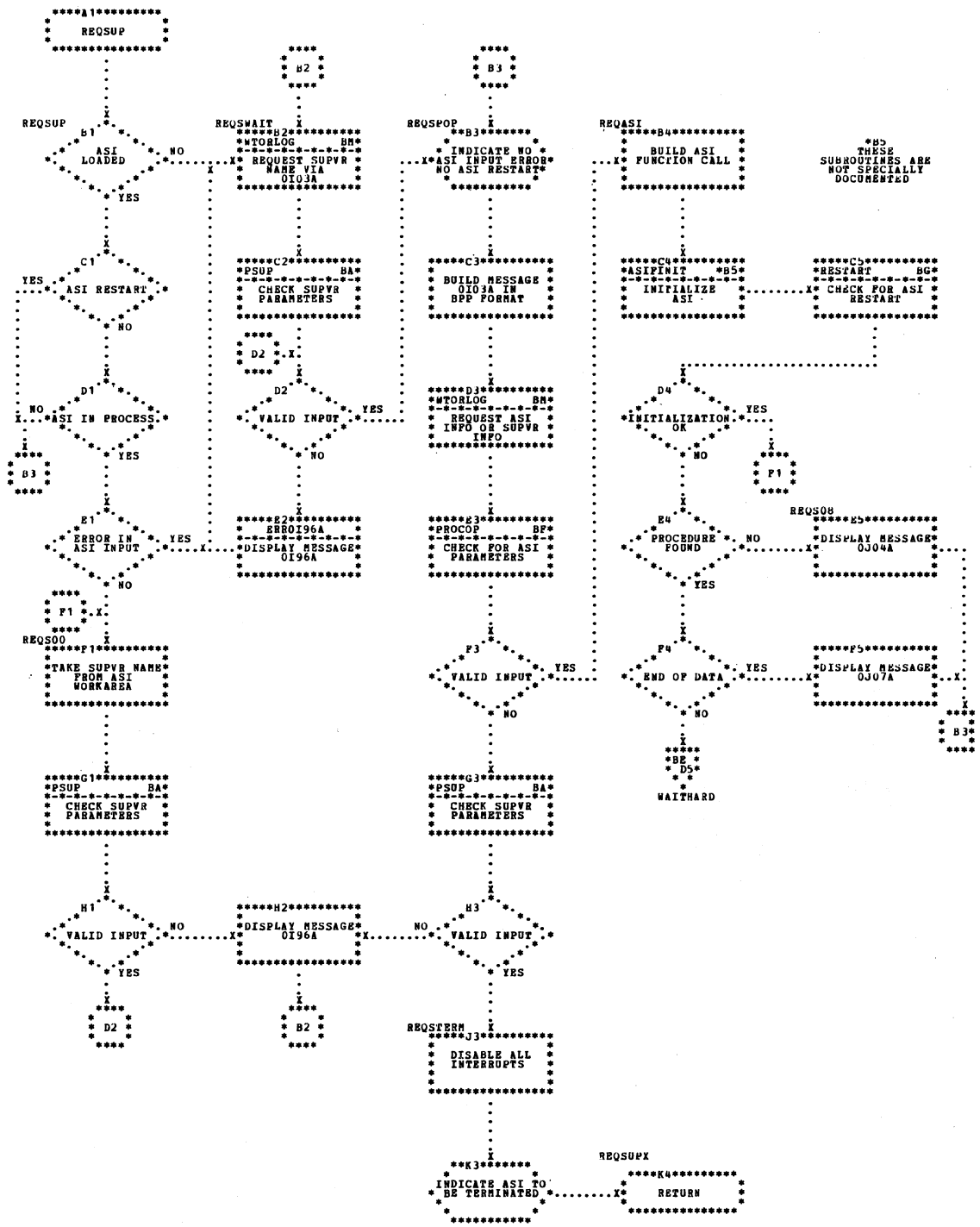


Chart BM. $$A$IPLR - Write to Operator with Reply

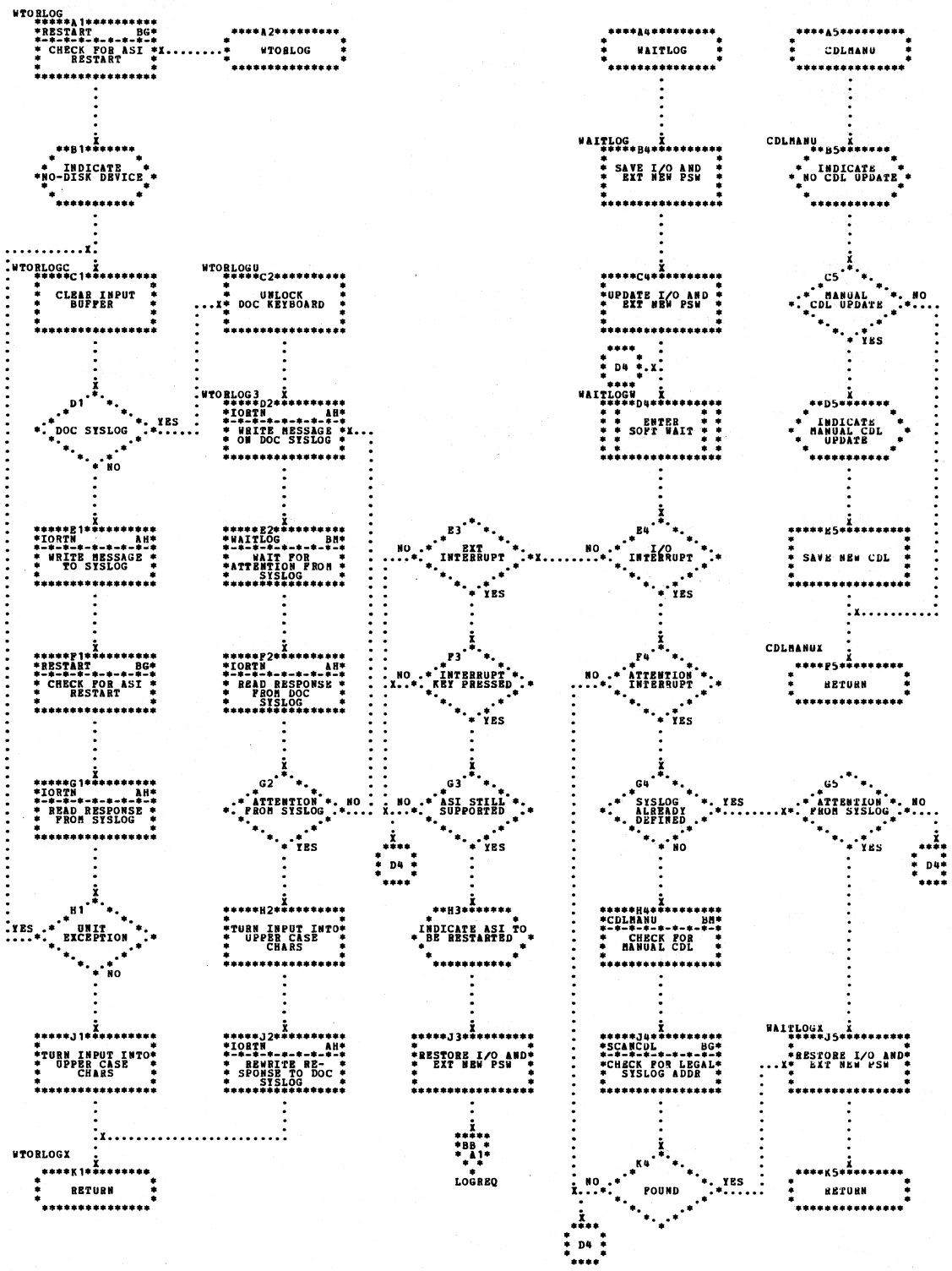


Chart CA. $IPLRT2 - Initialization Routine (Part 1 of 3)
Refer to Chart 02.

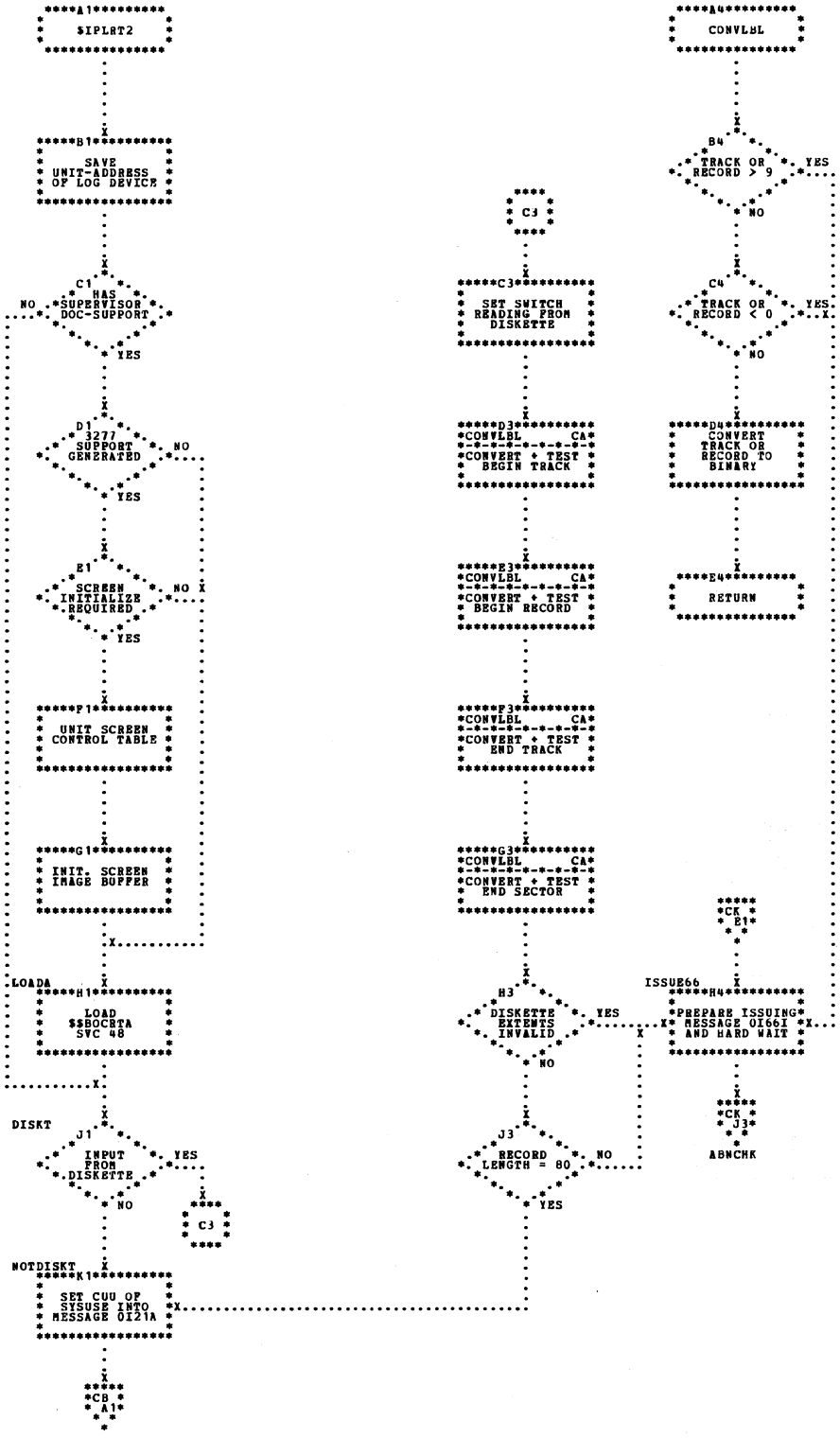


Chart CB. $IPLRT2 - Initialization Routine (Part 2 of 3)
Refer to Chart 02.

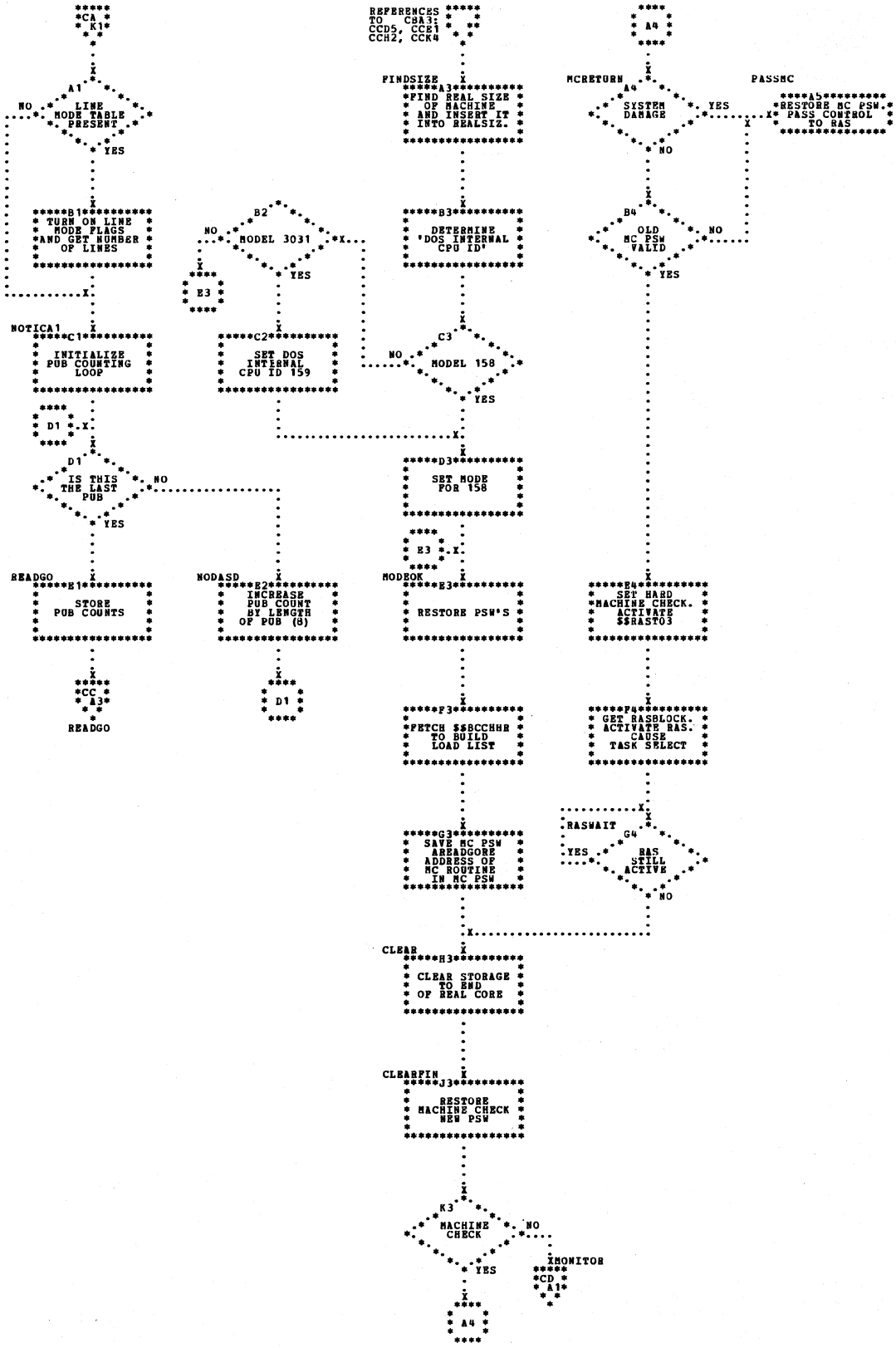


Chart CC. $IPLRT2 - Initialization Routine (Part 3 of 3)
Refer to Chart 02.

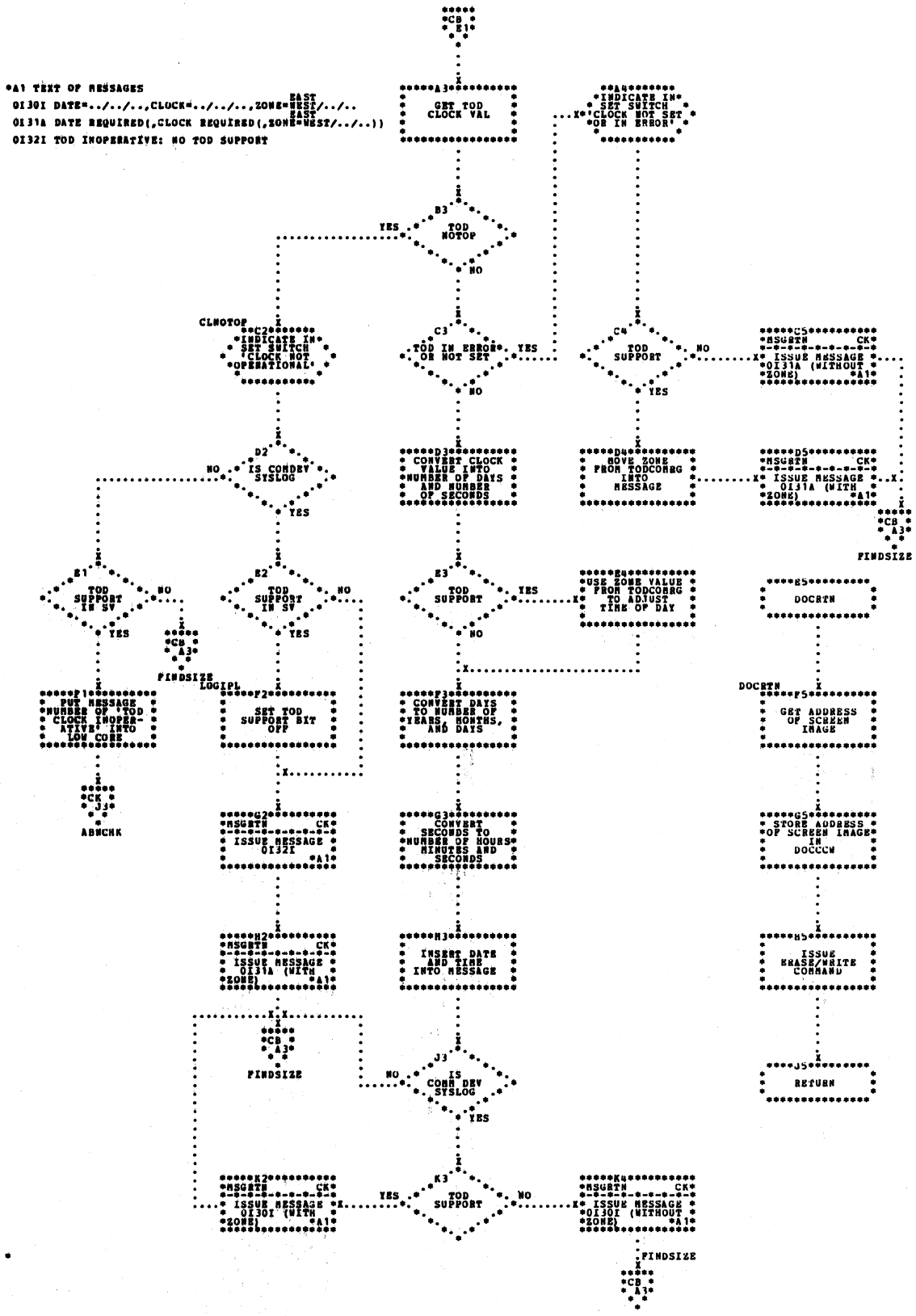


Chart CD. $IPLRT2 - Monitor and Read Control Statement
Refer to Chart 02.

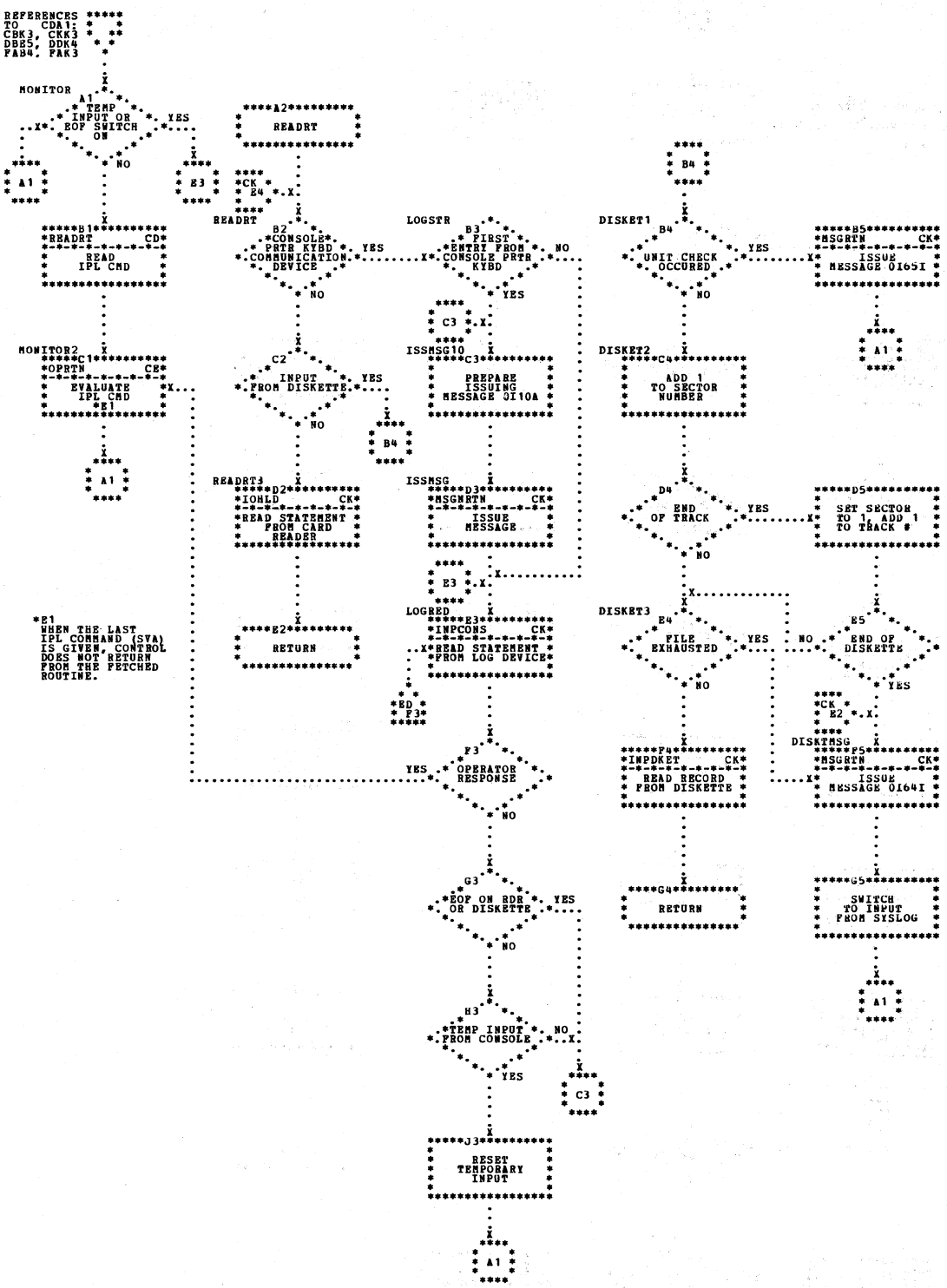


Chart CE. $IPLRT2 - Operation Scan Routines
Refer to Chart 02.

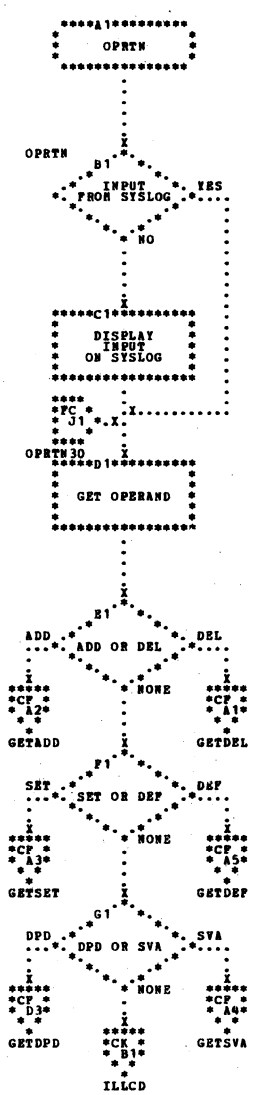


Chart CF. $IPLRT2 - Fetch Appropriate IPL Phase
Refer to Chart 02.

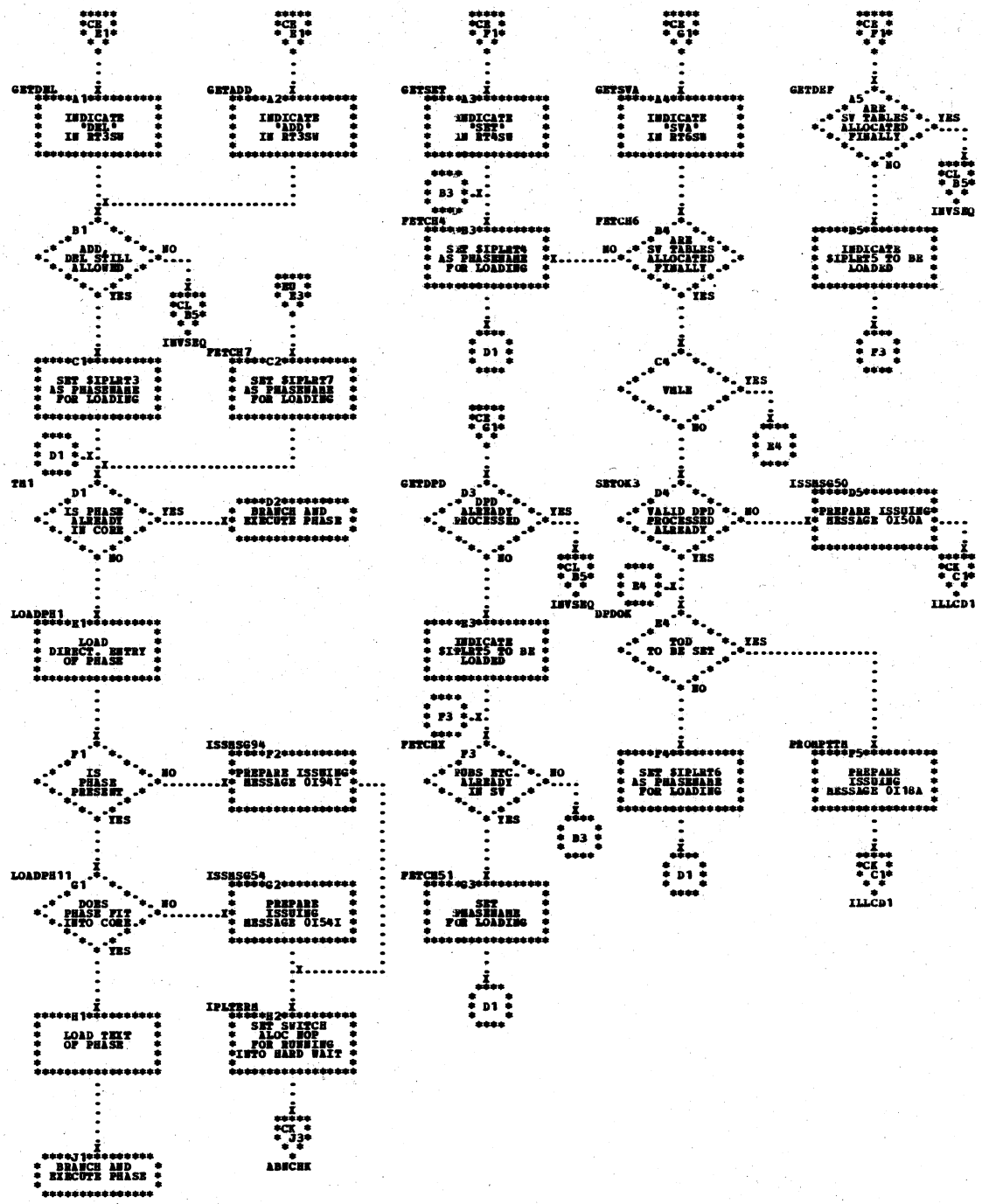


Chart CH. $IPLRT2 - Move Routine
Refer to Chart 02.

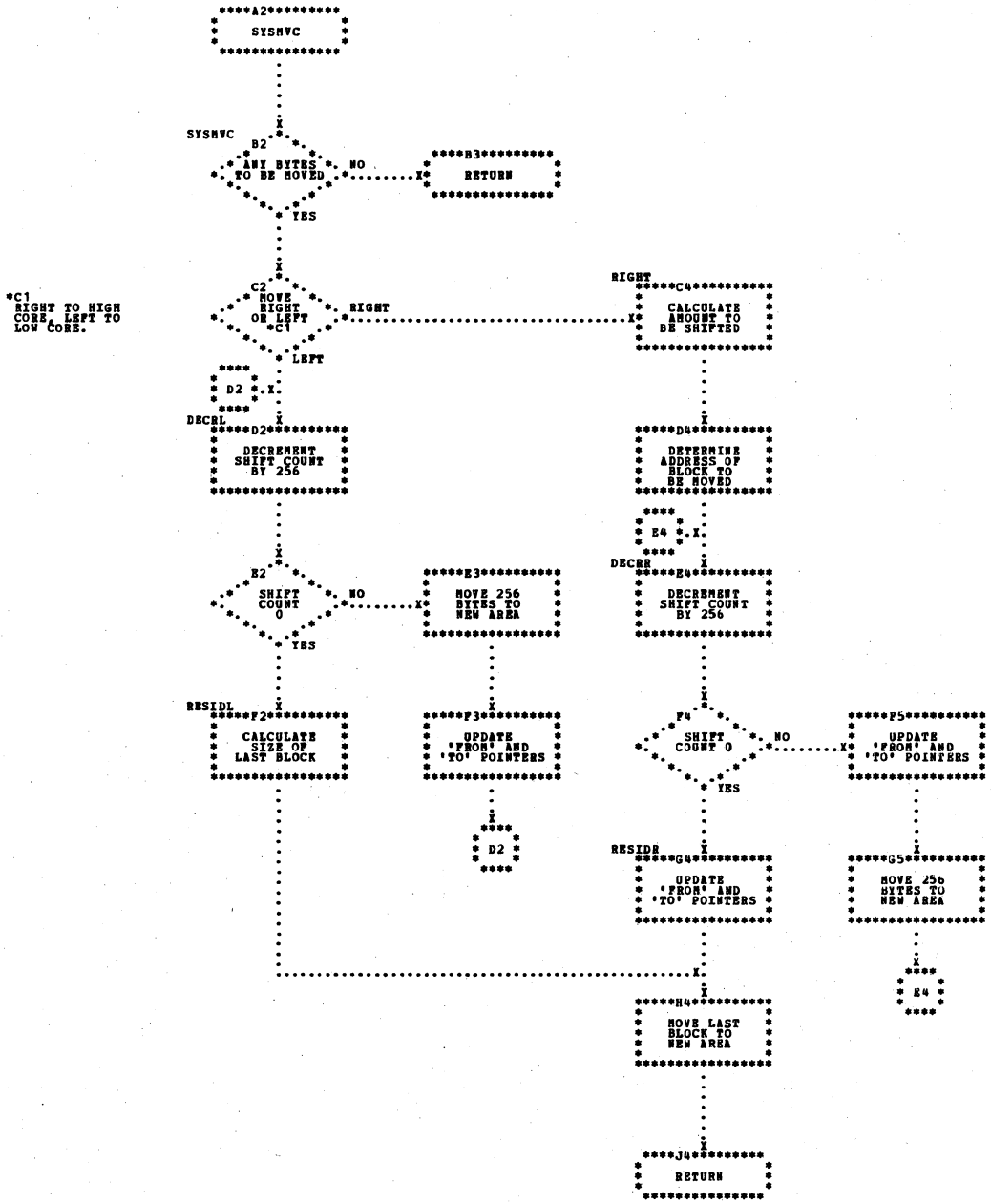


Chart CJ. $IPLRT2 - Get Operand and Conversion Subroutines
Refer to Chart 02.

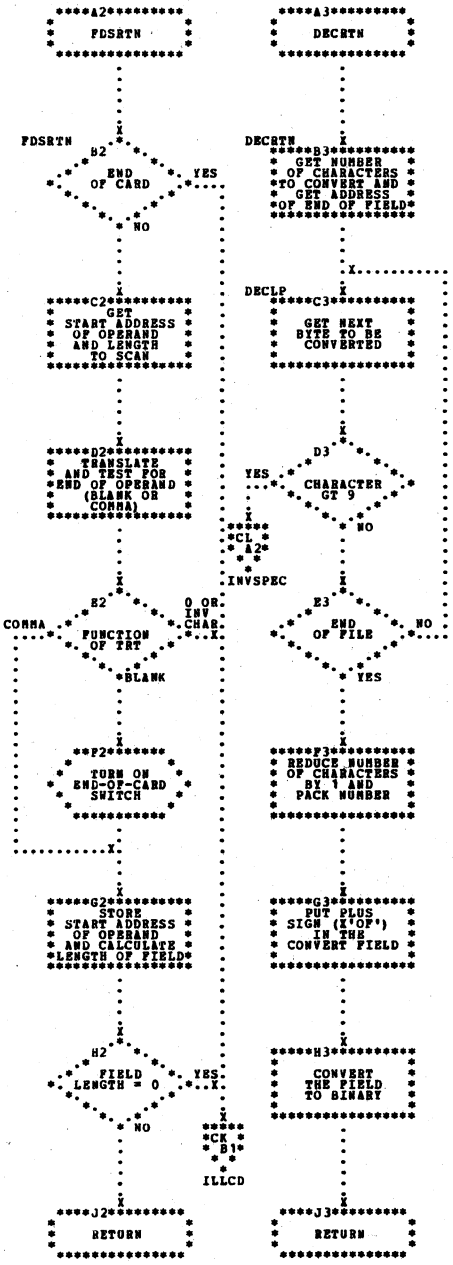


Chart CK. $IPLRT2 - I/O Subroutines
Refer to Chart 02.

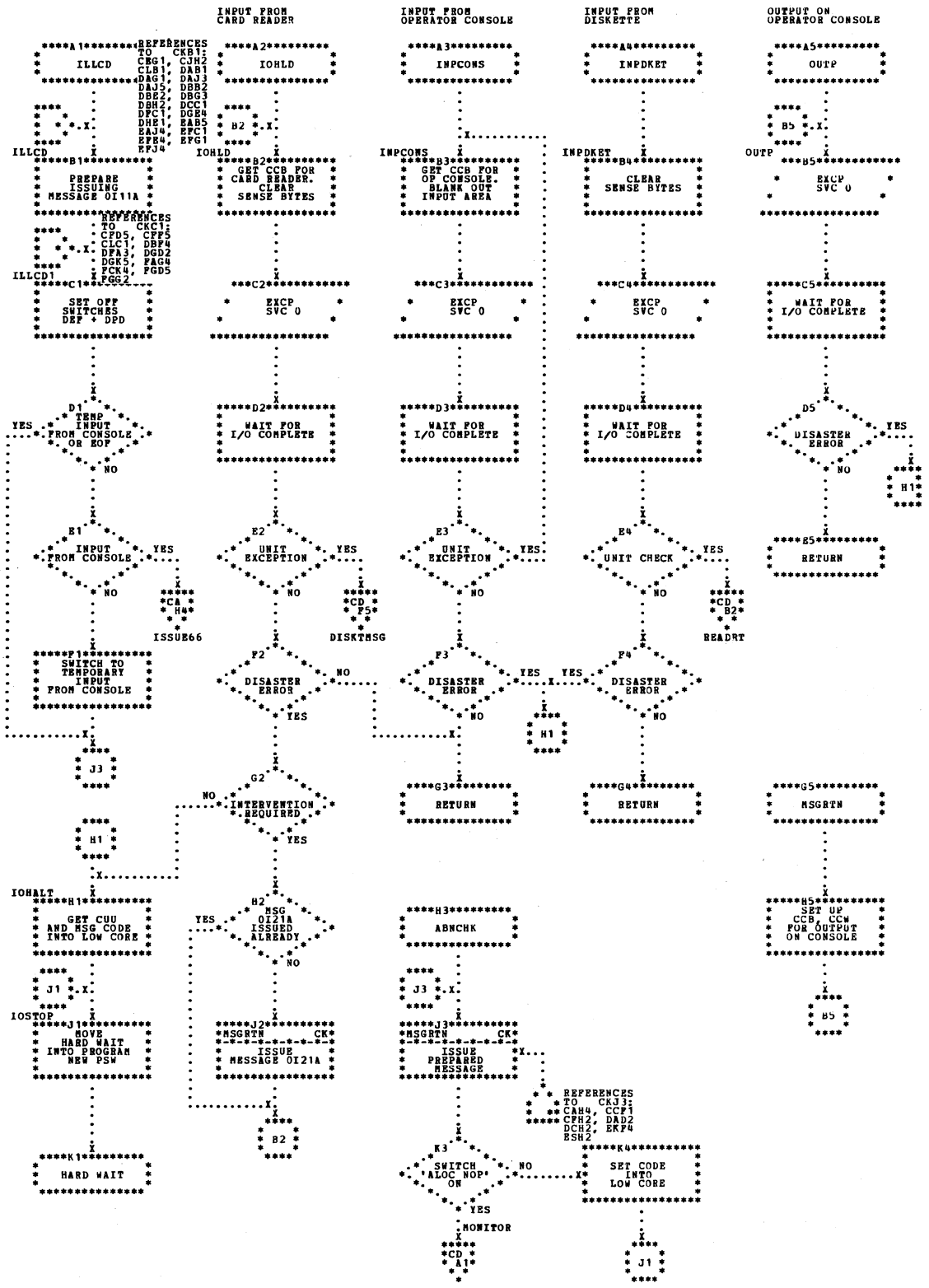


Chart CL. $IPLRT2 - Message Subroutines
Refer to Chart 02.

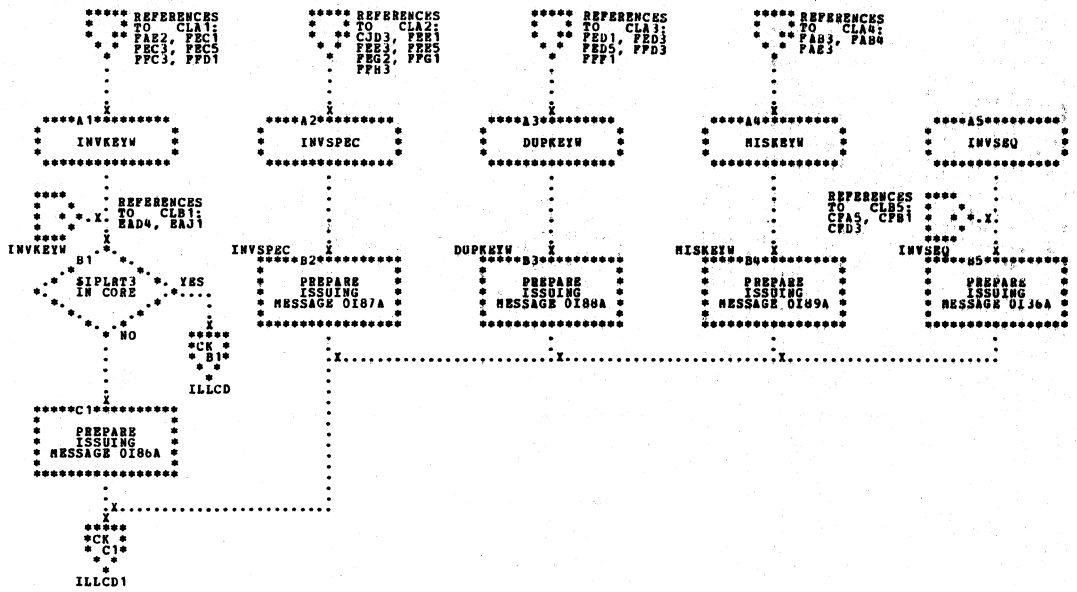


Chart DA. $IPLRT3 - ADD a Device (Part 1 of 2)
Refer to Chart 03.

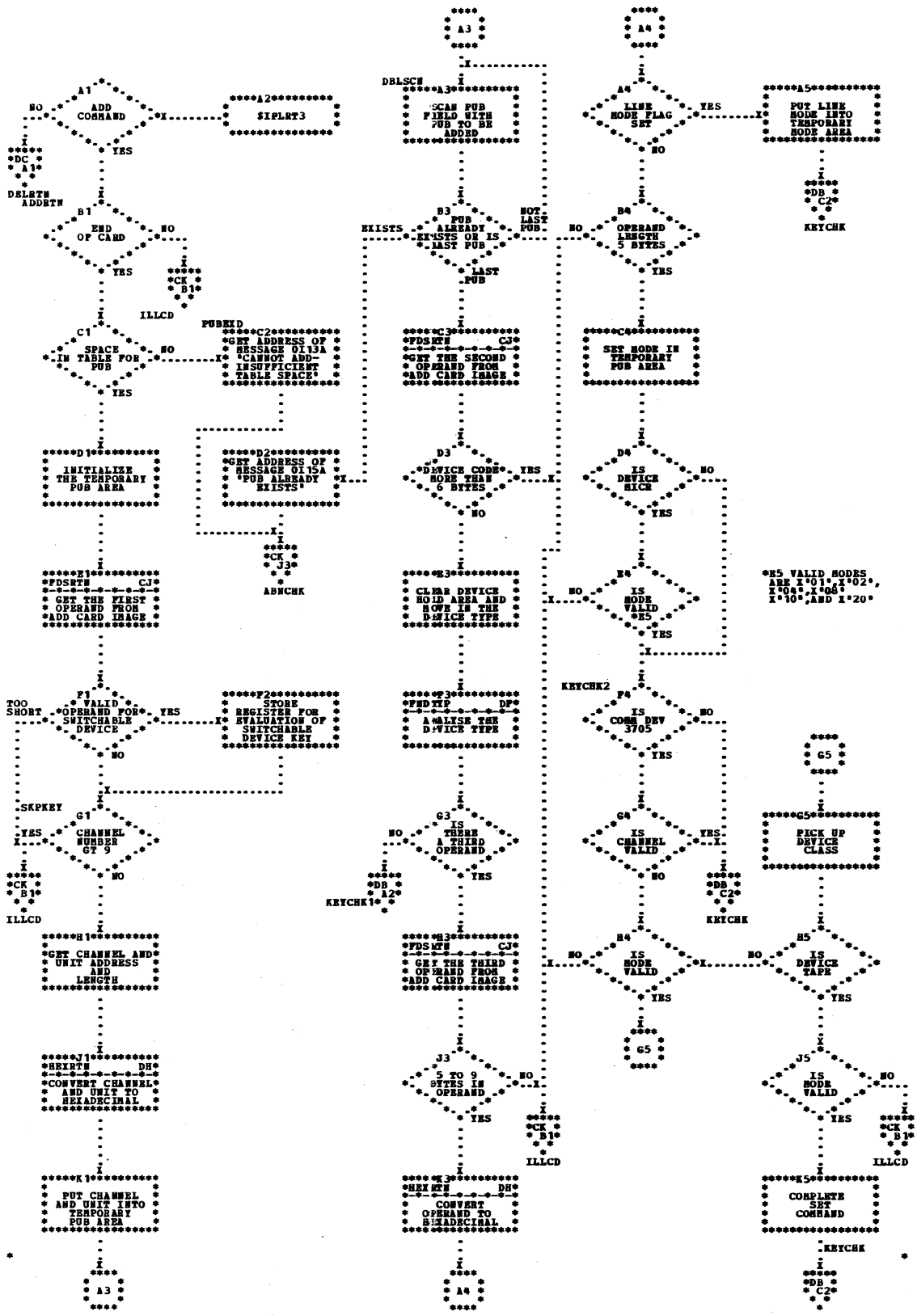


Chart DB. $IPLRT3 - ADD a Device (Part 2 of 2)
Refer to Chart 03.

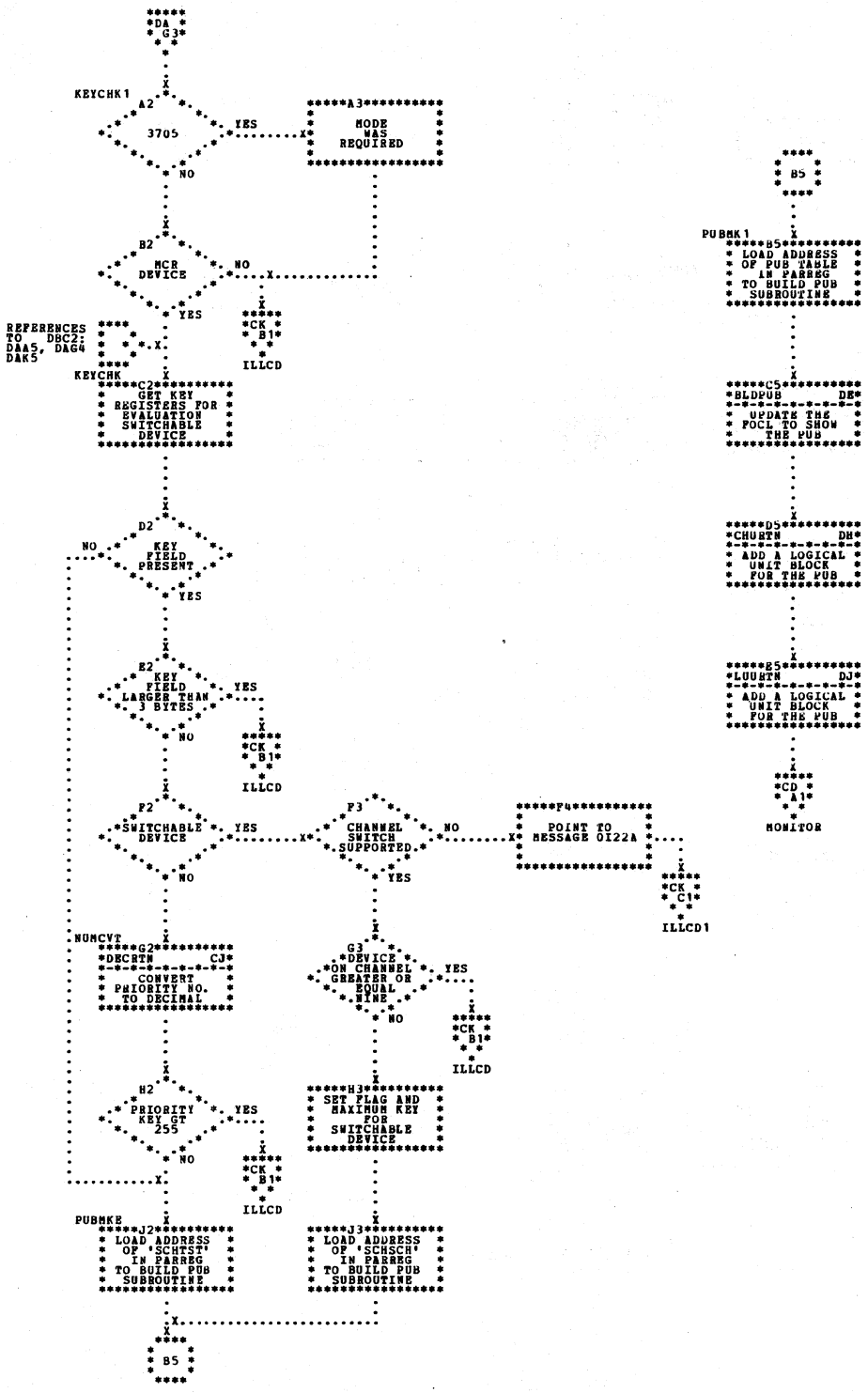


Chart DC. $IPLRT3 - Delete a PUB (Part 1 of 2)
Refer to Chart 03.

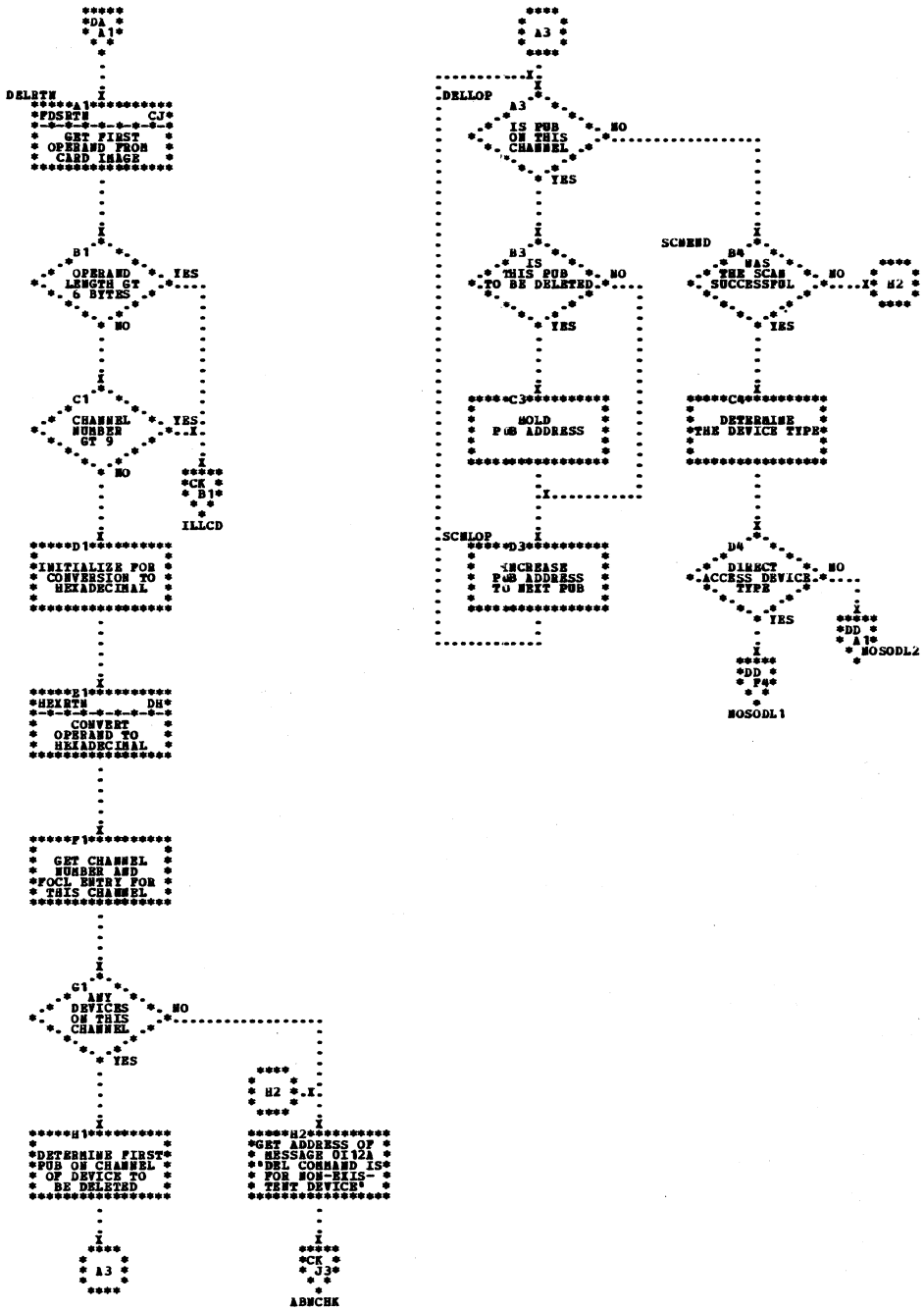


Chart DD. $IPLRT3 - Delete a PUB (Part 2 of 2)
Refer to Chart 03.

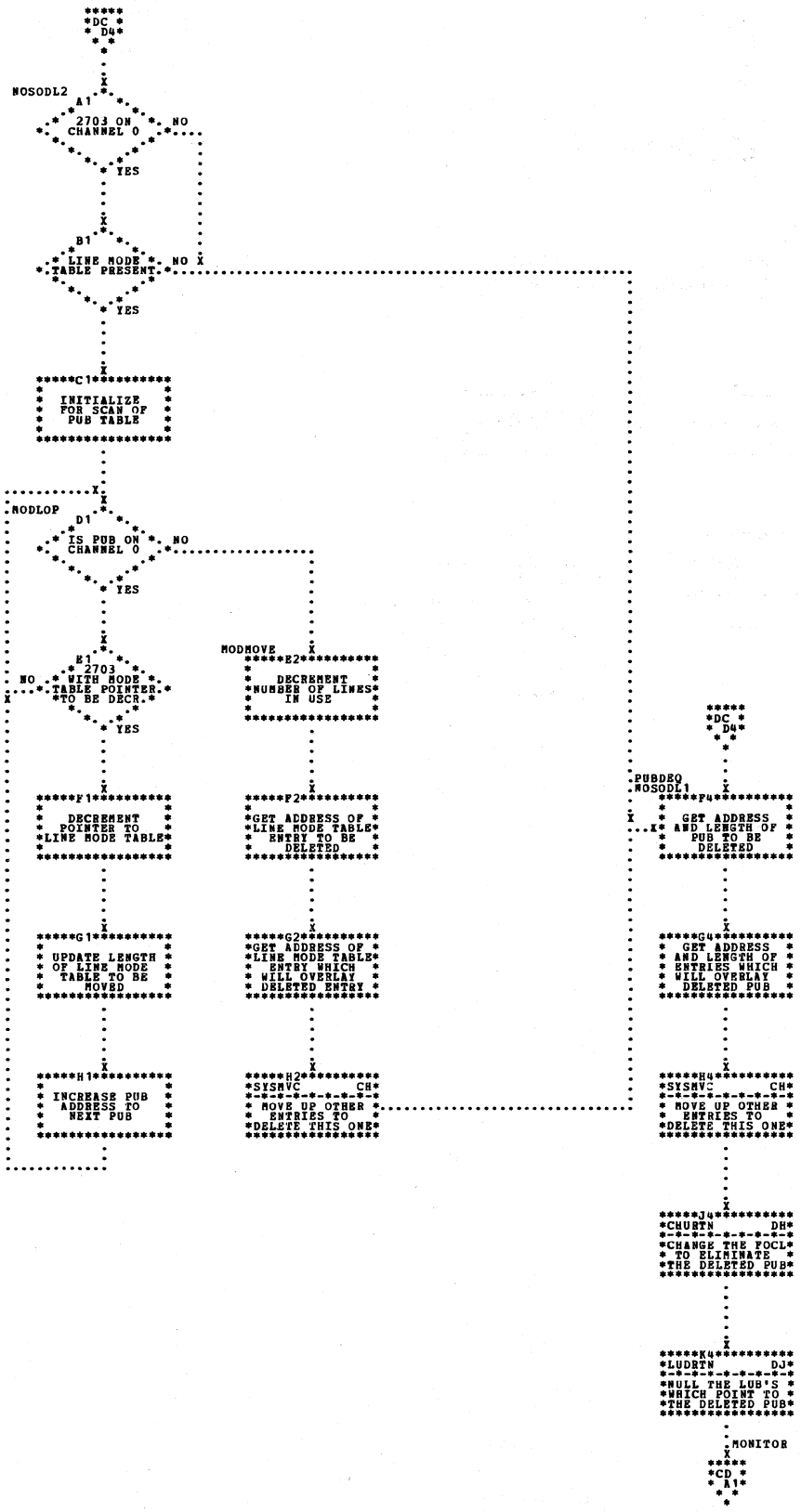


Chart DE. $IPLRT3 - Build PUB Table Subroutine
Refer to Chart 03.

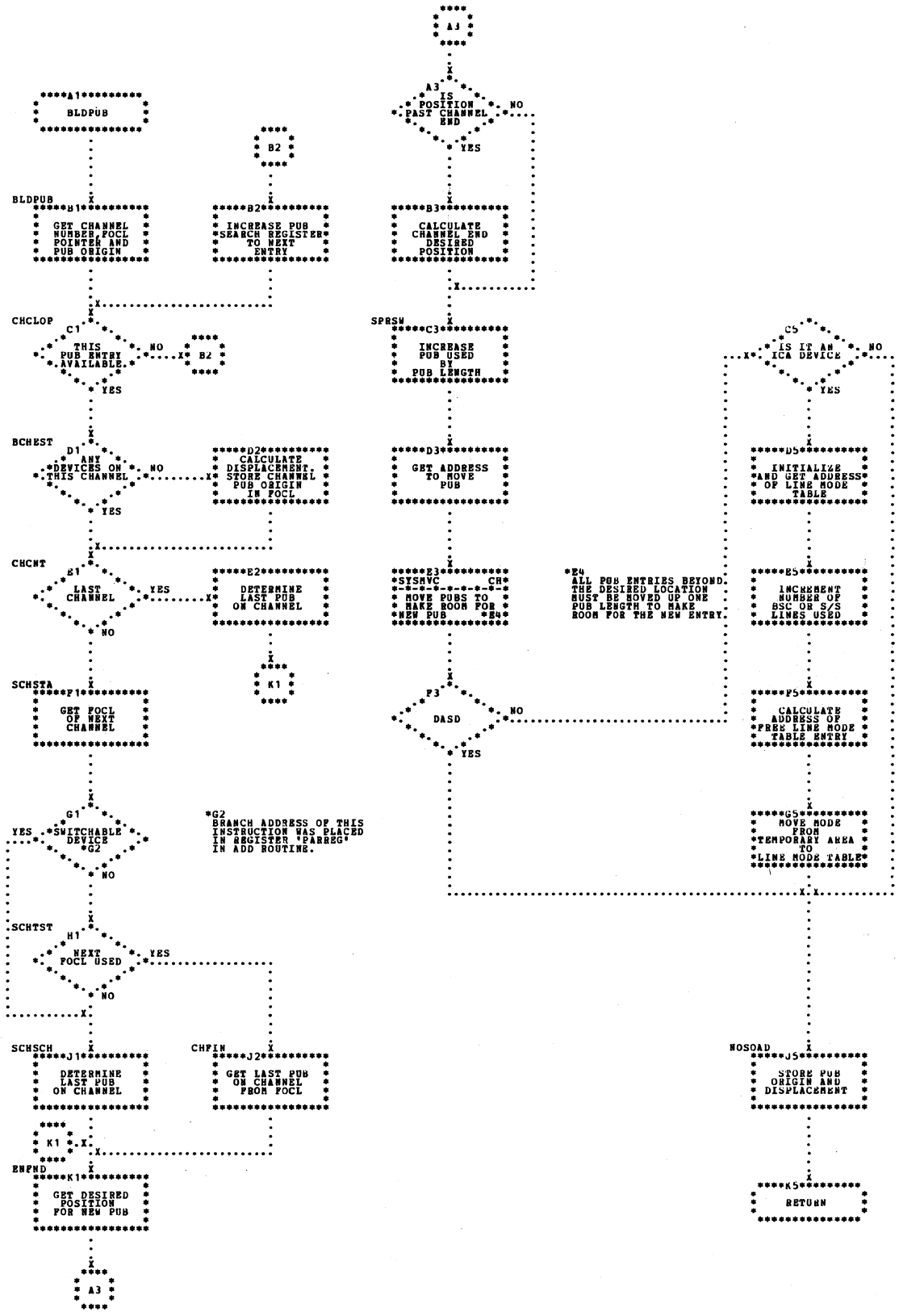


Chart DF. $IPLRT3 - Device Type Conversion Subroutine (Part 1 of 2)
Refer to Chart 03.

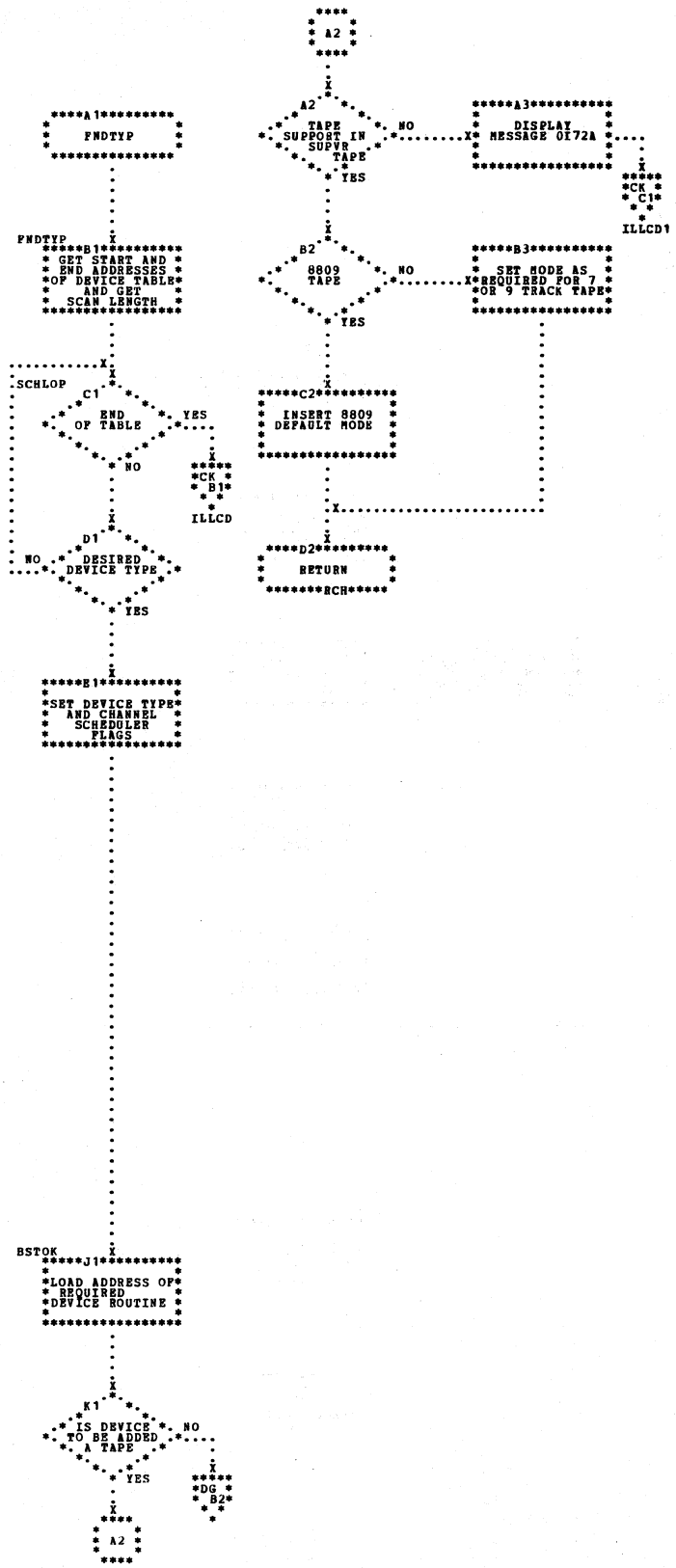


Chart DG. $IPLRT3 - Device Type Conversion Subroutine (Part 2 of 2)
Refer to Chart 03

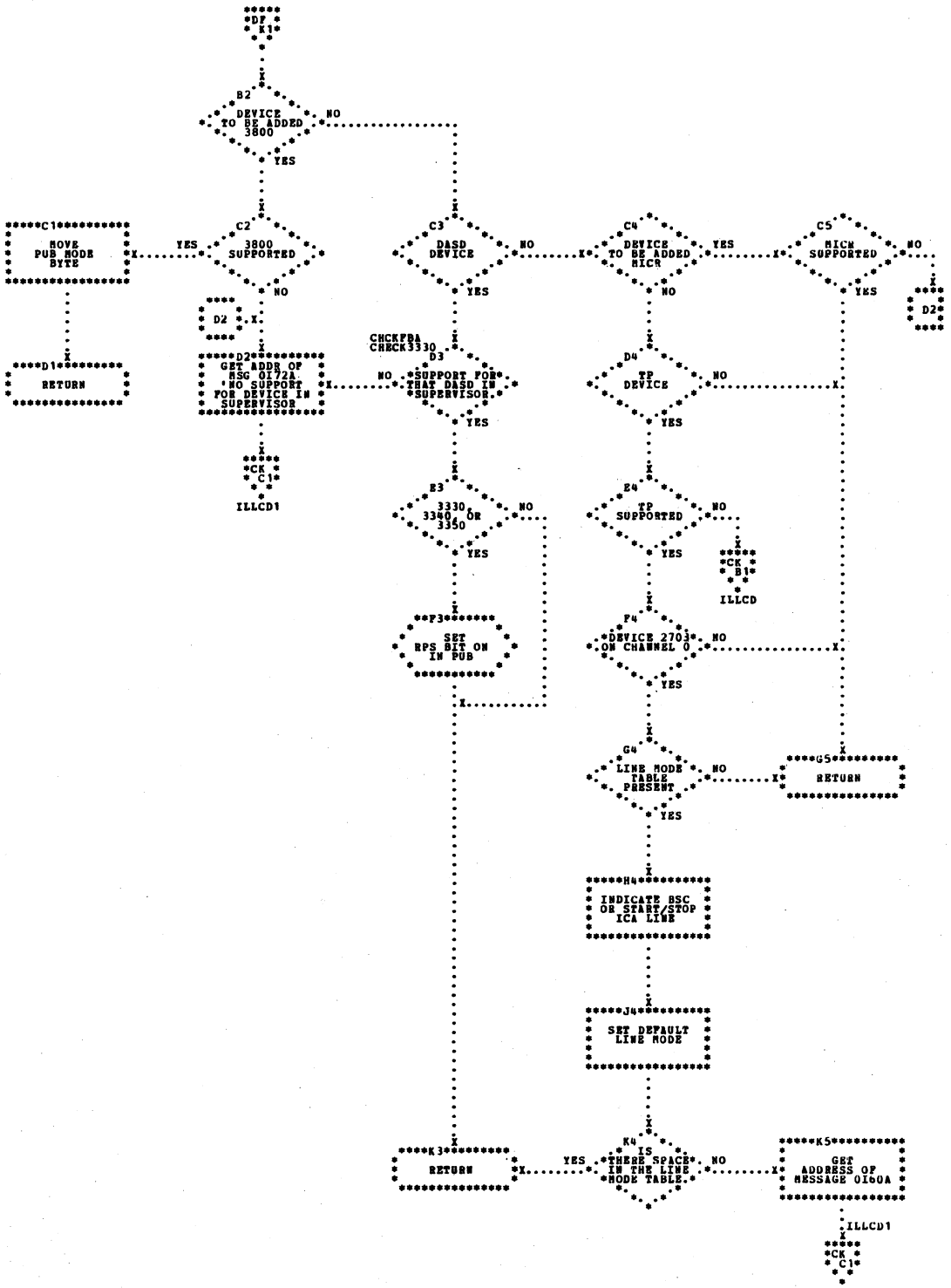


Chart DH. $IPLRT3 - Conversion and Update FOCL Subroutines
Refer to Chart 03.

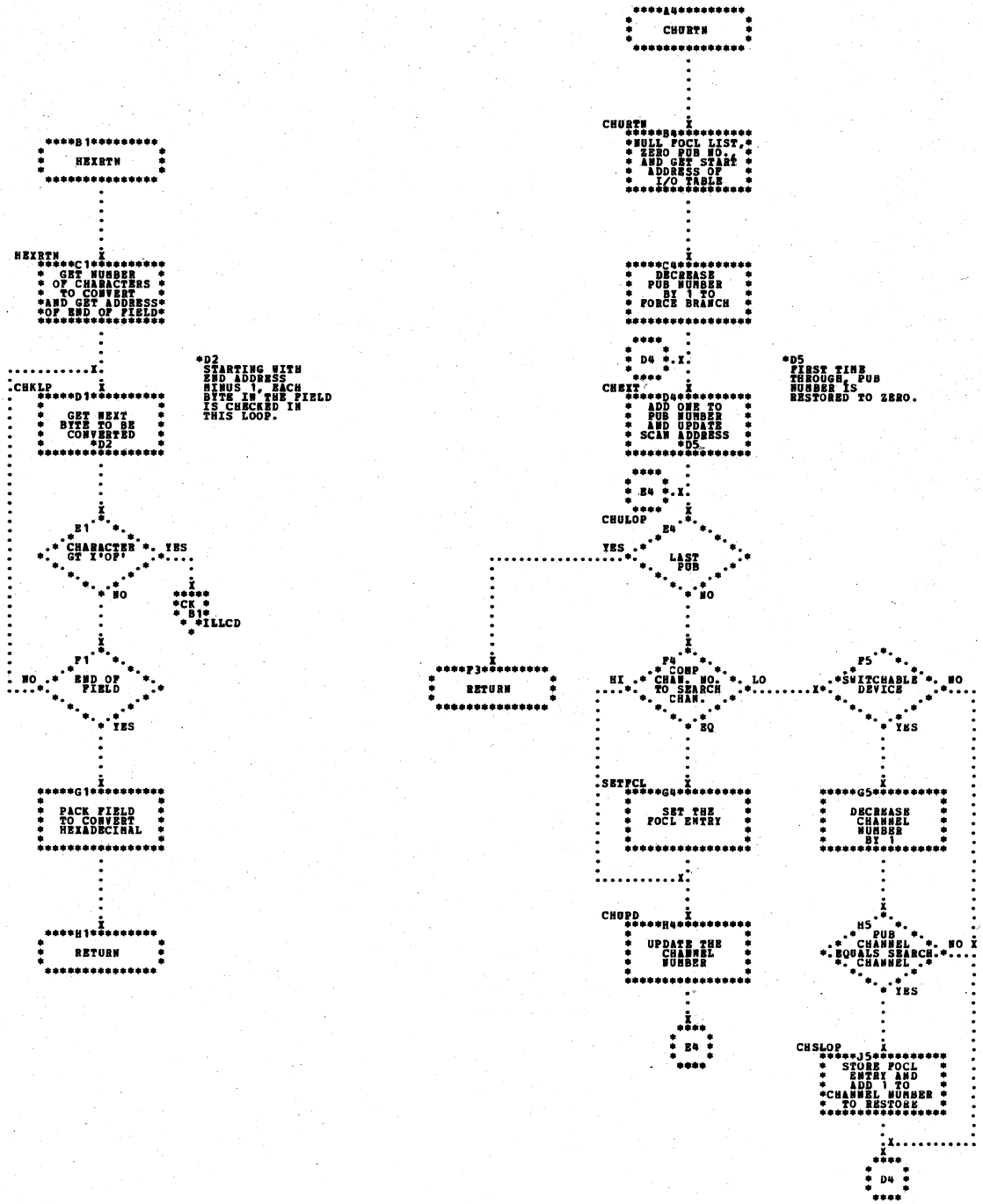


Chart DJ. $IPLRT3 - Update LUBs
Refer to Chart 03.

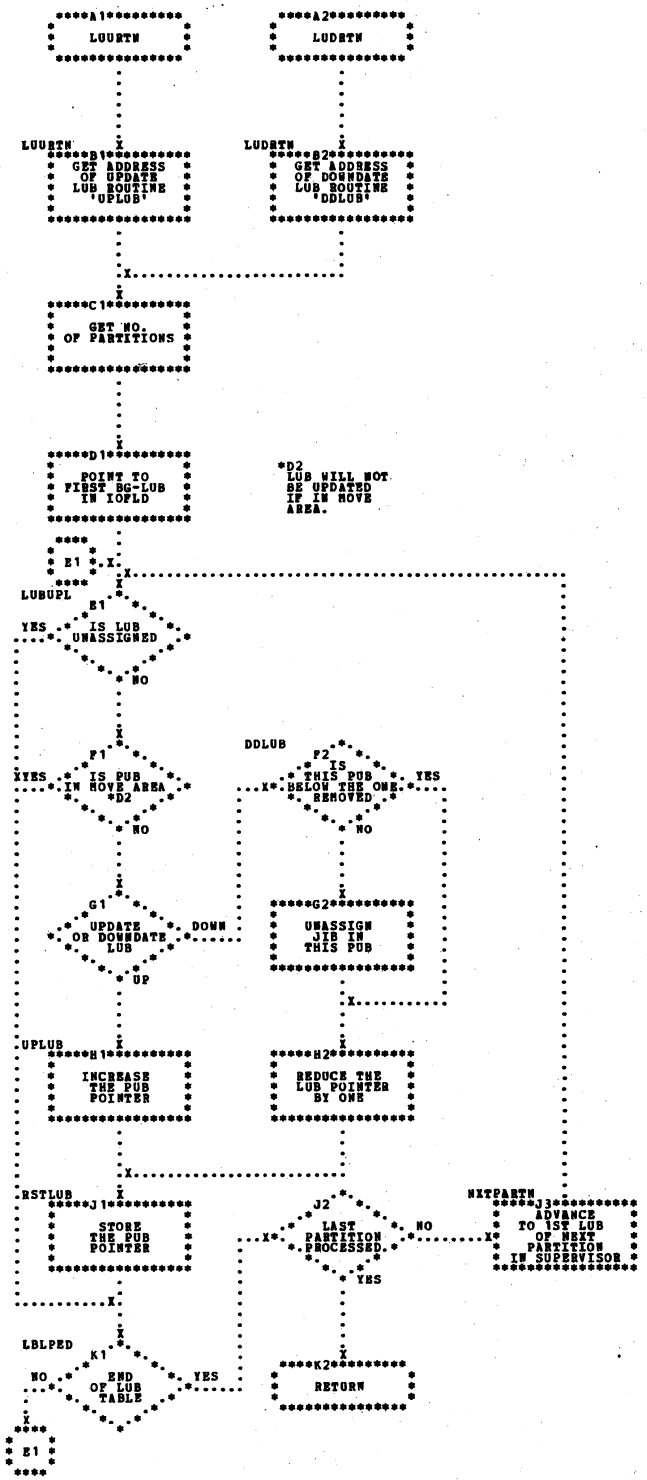


Chart EA. $IPLRT4 - SET Statement Processor
Refer to Chart 03.

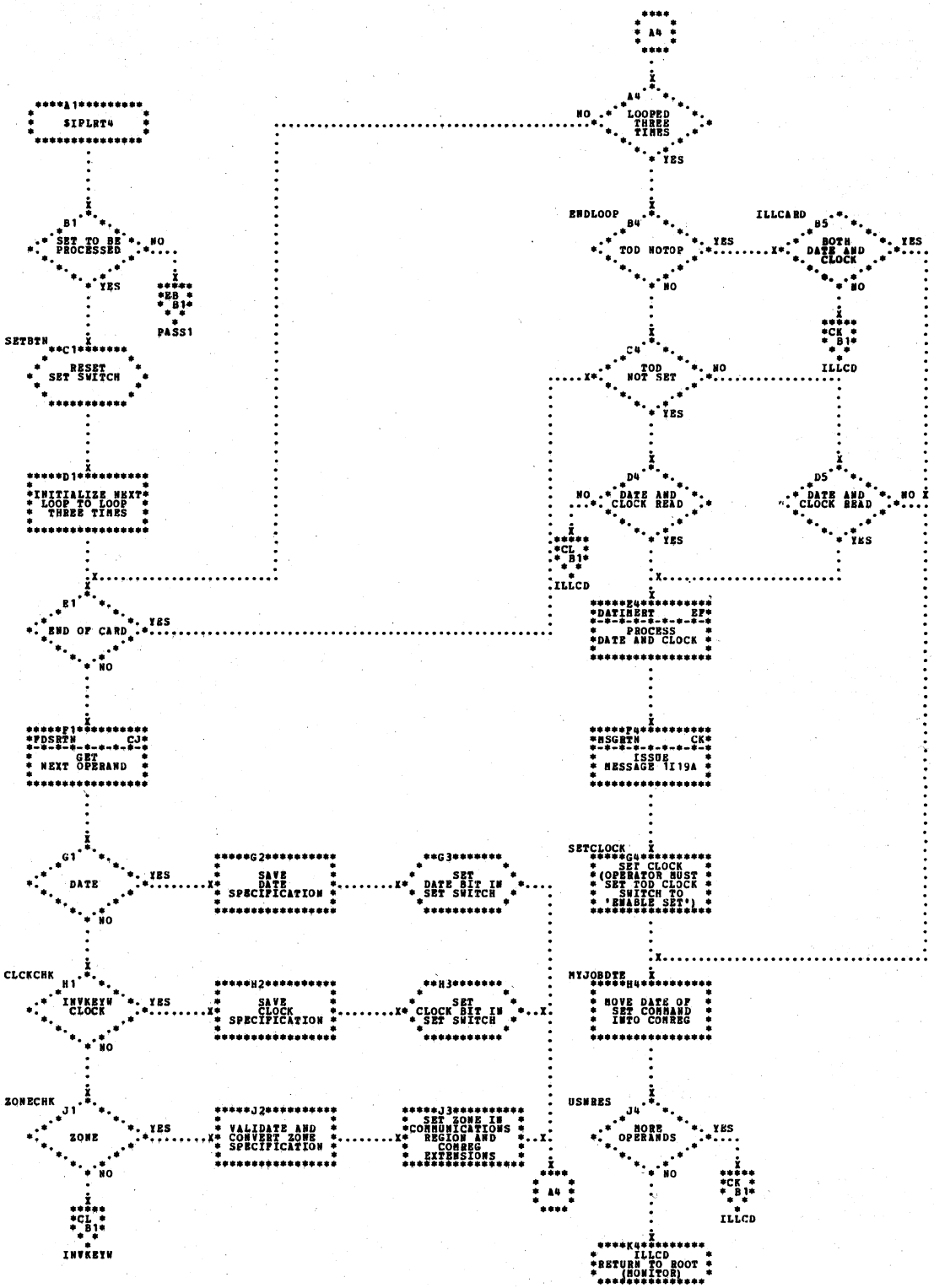


Chart EB. $IPLRT4 - Complete SV Tables. Calculate Addresses of Dynamic SV Areas
Refer to Chart 03.

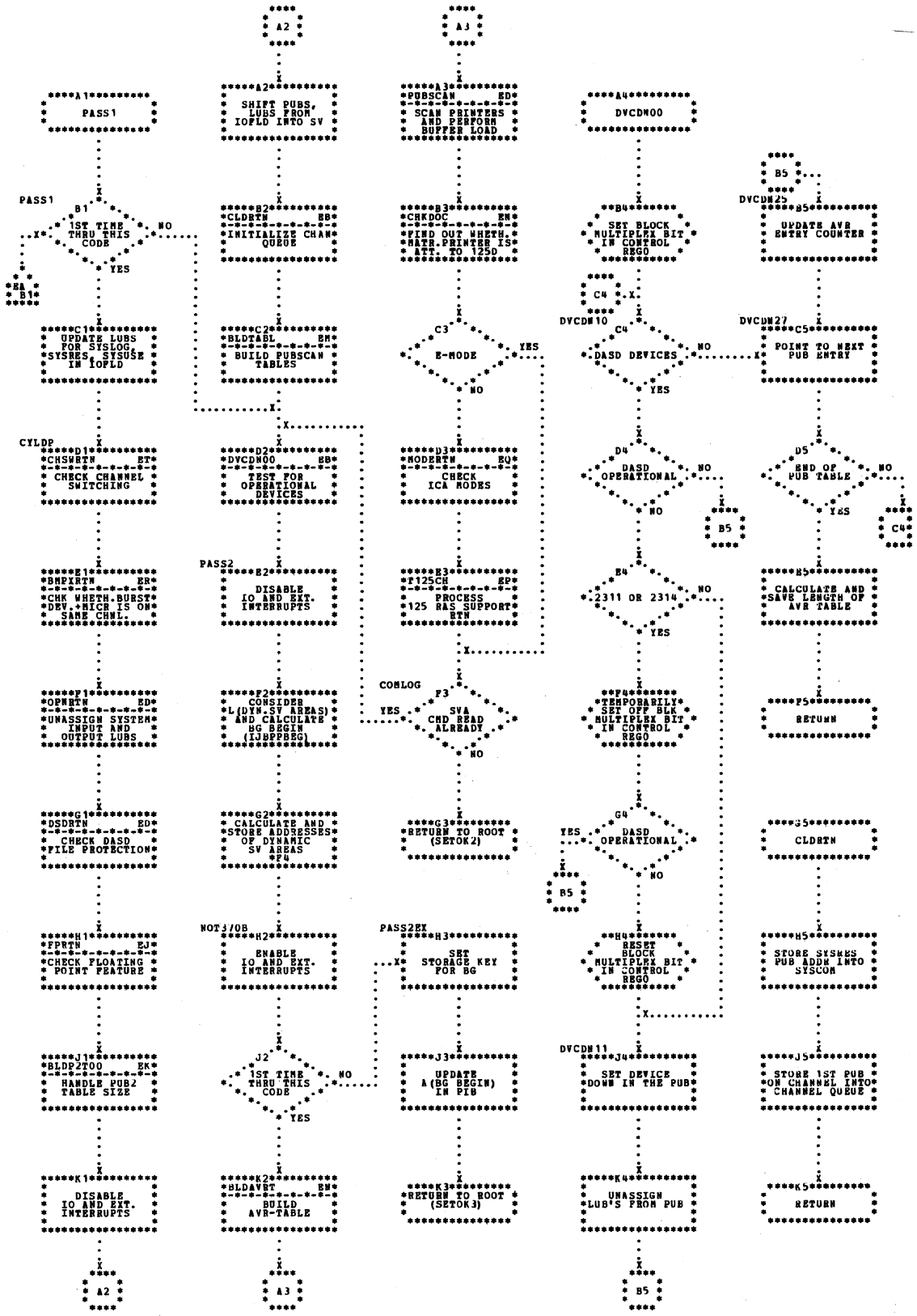


Chart ED. $IPLRT4 - I/O and Check Device Type Subroutines
Refer to Chart 03.

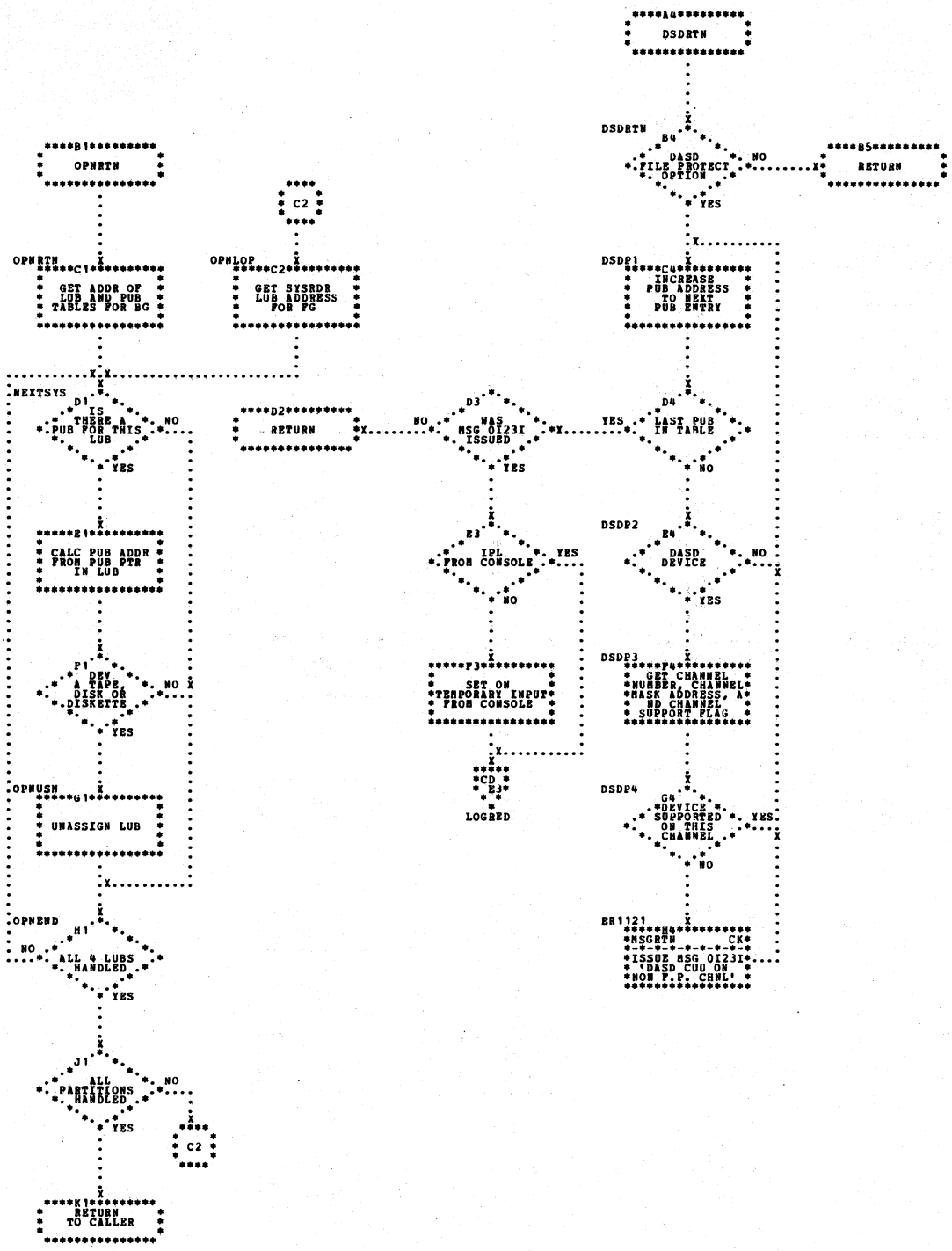


Chart EE. $IPLRT4 - Find PUB and I/O Subroutines
Refer to Chart 03.

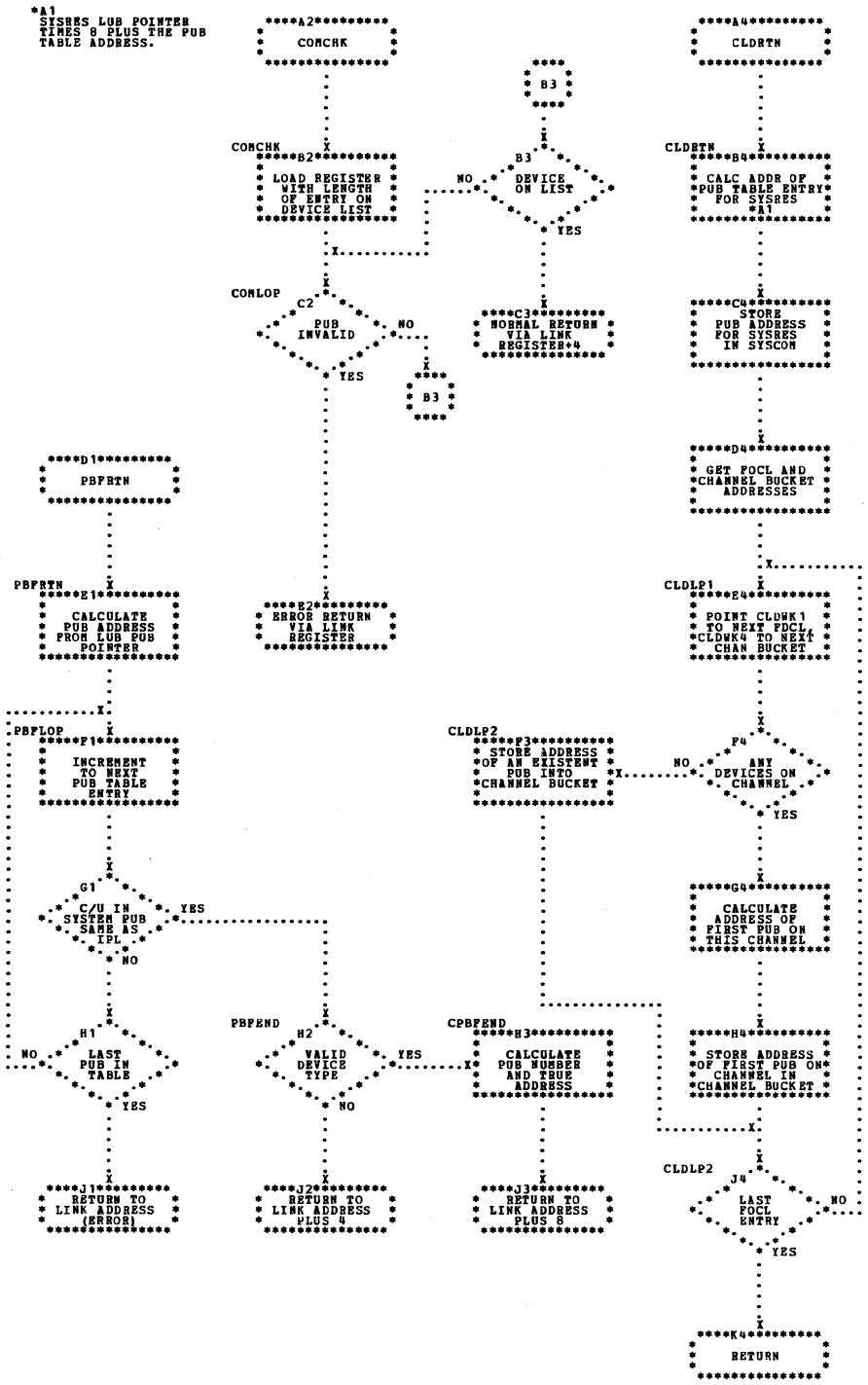


Chart EF. $IPLRT4 - Date and Time Subroutines
Refer to Chart 03.

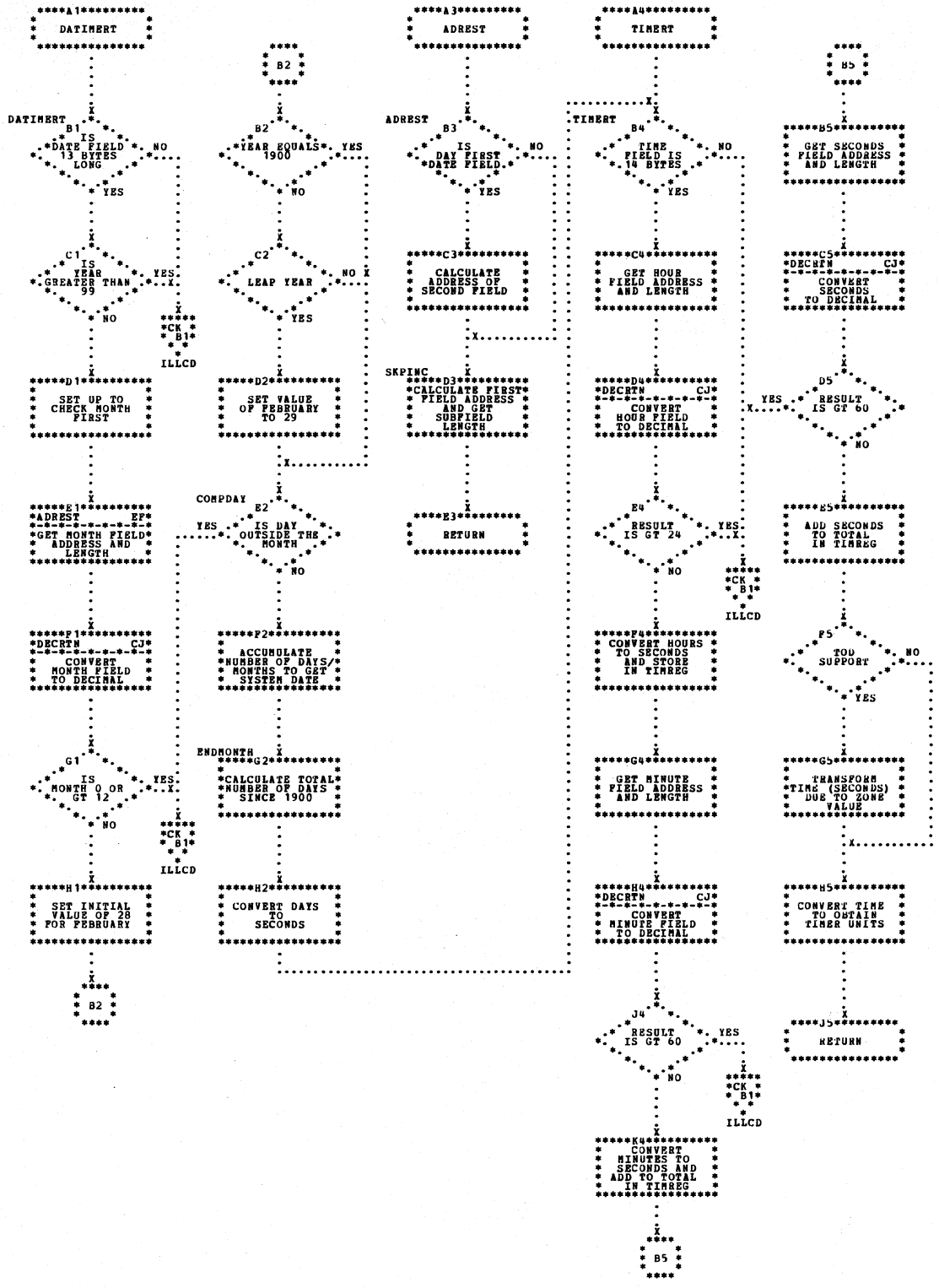


Chart EJ. $IPLRT4 - Reorder MPX Channel LUBs and PUBs
Refer to Chart 03.

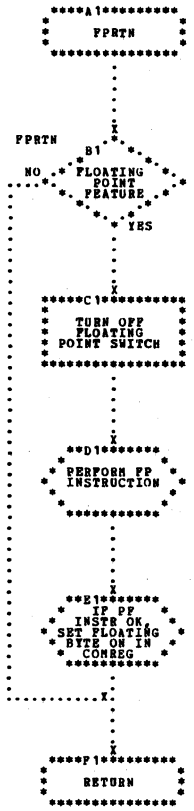


Chart EK. $IPLRT4 - Build-PUB2-Table Subroutine
Refer to Chart 03.

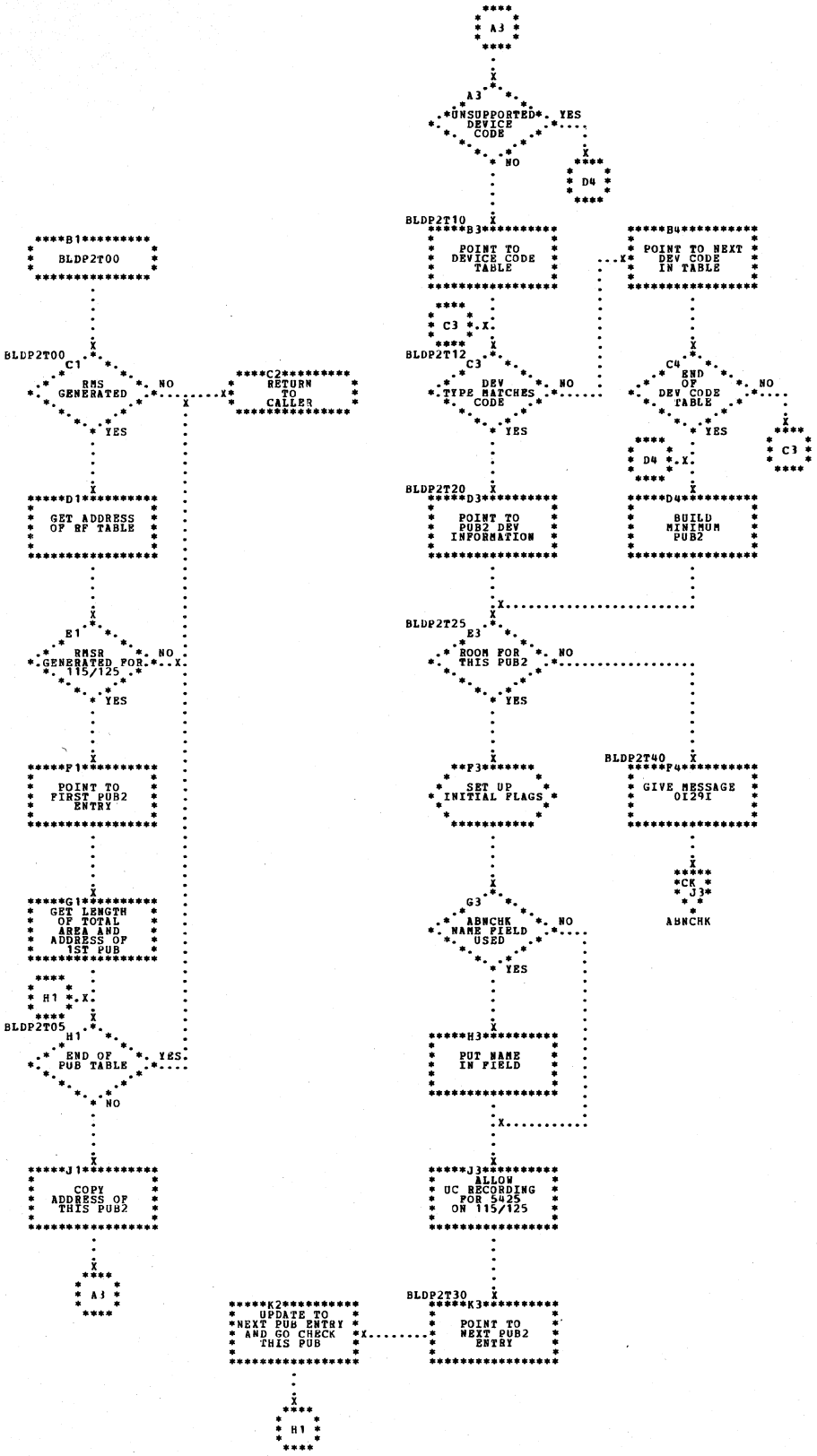


Chart EL. $IPLRT4 - Test if Devices are Operational
Refer to Chart 03.

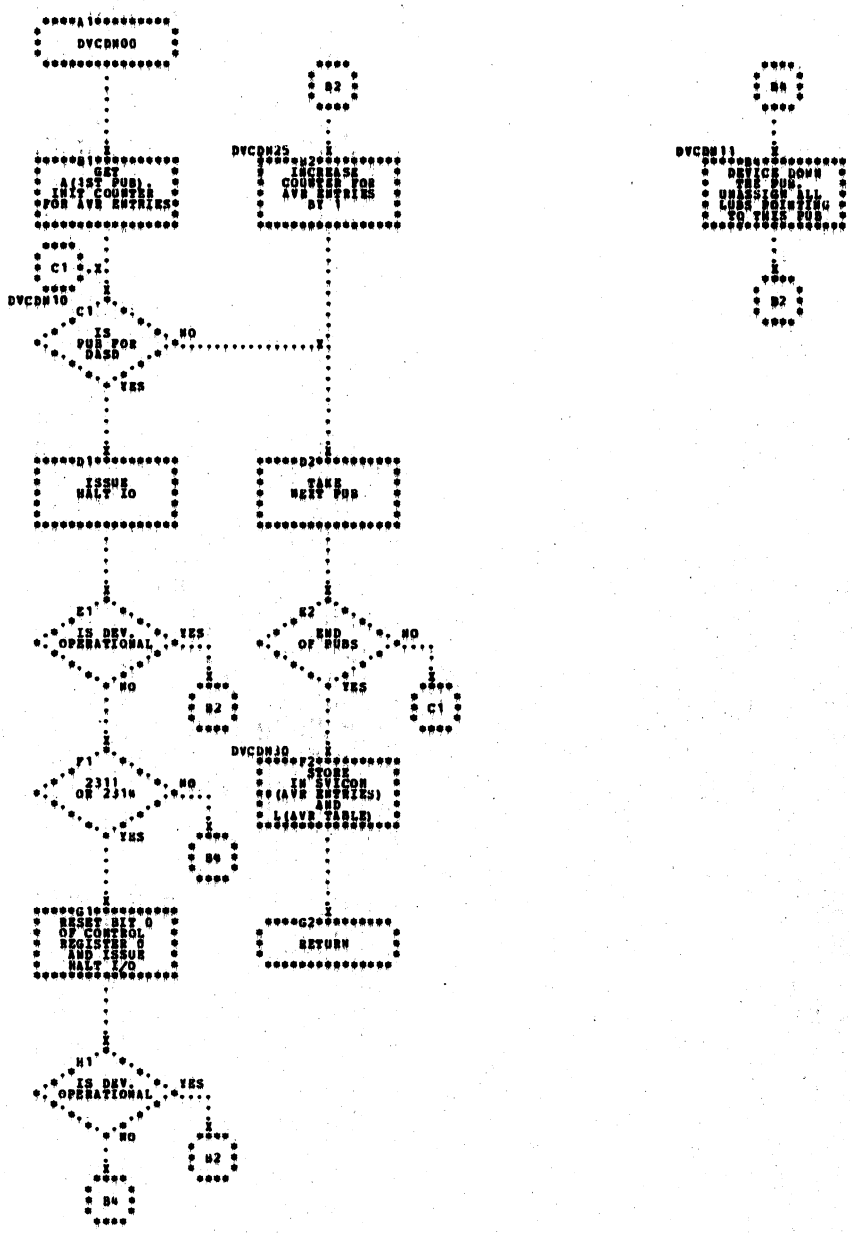


Chart EM. $IPLRT4 - Build PUB Scan Tables
Refer to Chart 03.

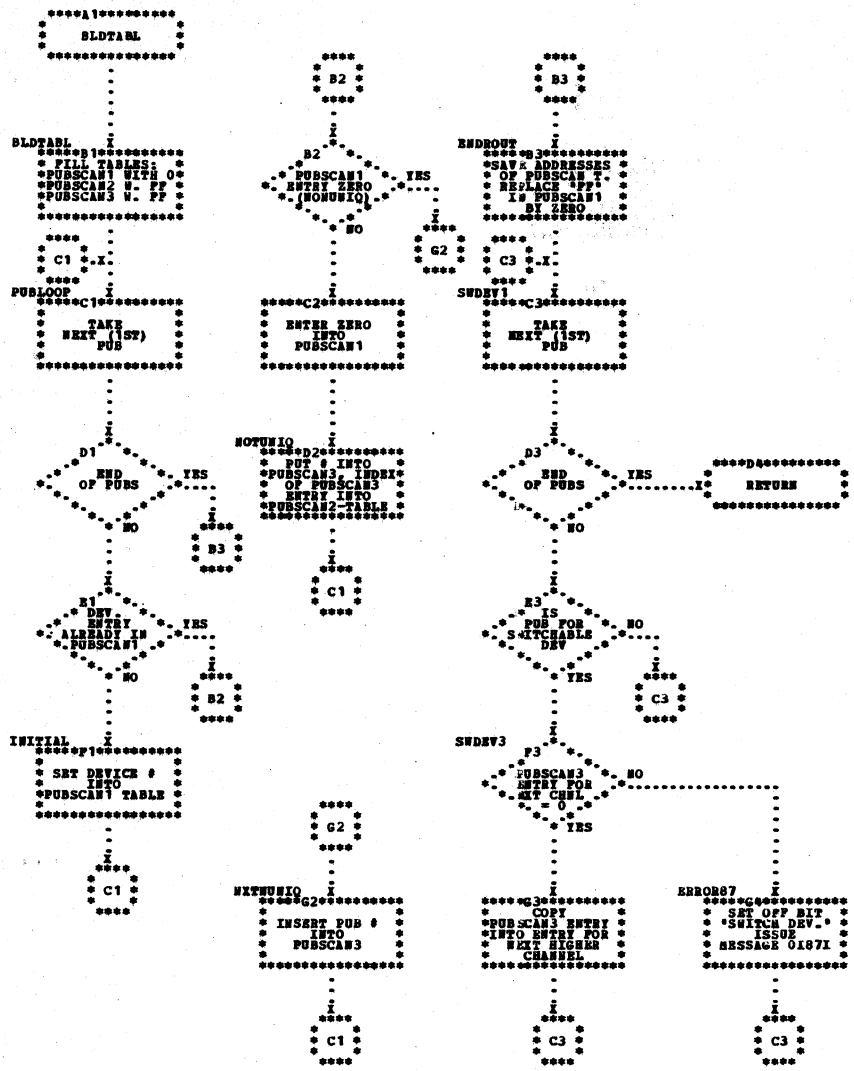


Chart EN. $IPLRT4 - Subroutines
Refer to Chart 03.

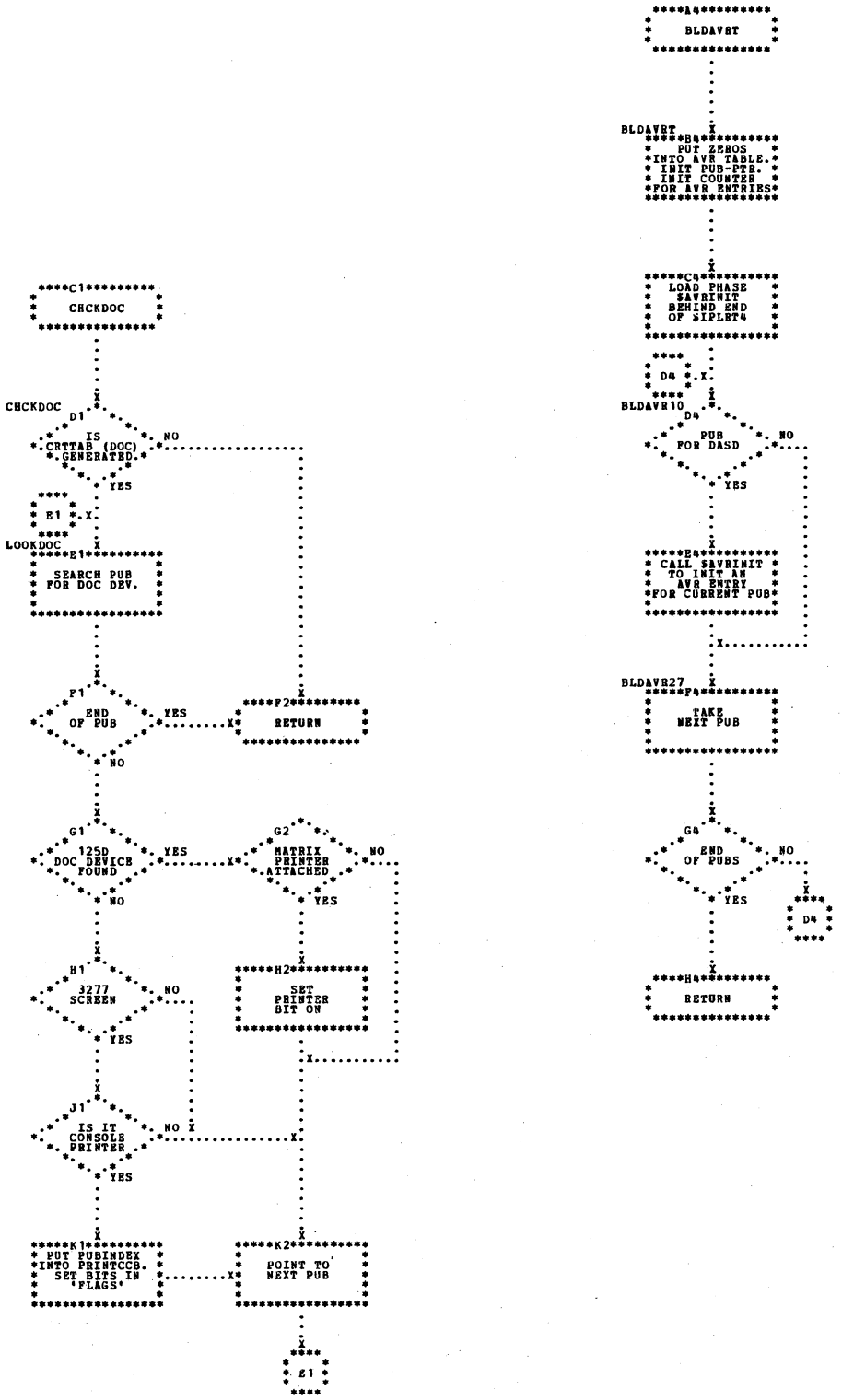


Chart EP. $IPLRT4 - Subroutines
Refer to Chart 03.

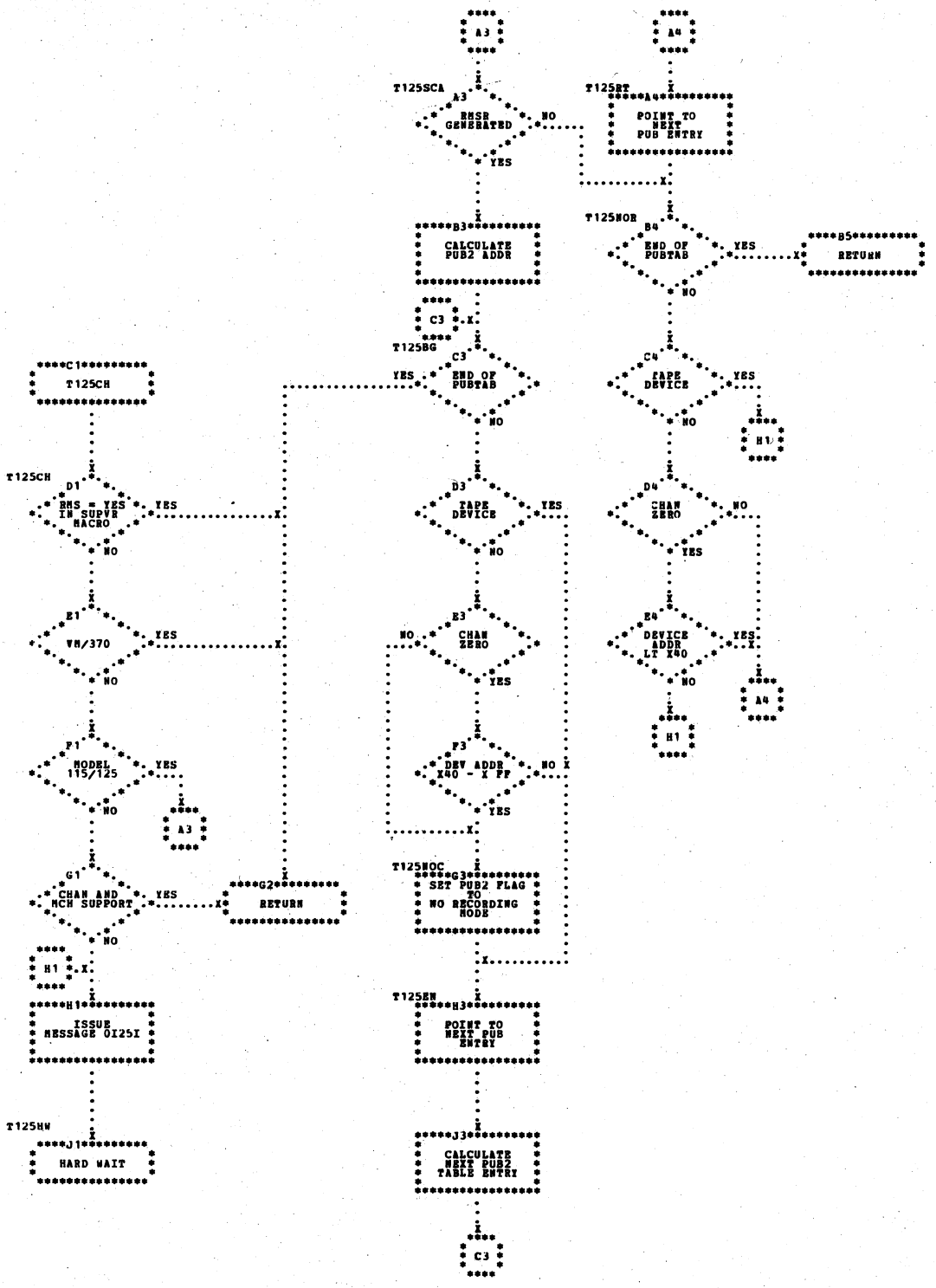


Chart EQ. $IPLRT4 - Subroutines
Refer to Chart 03.

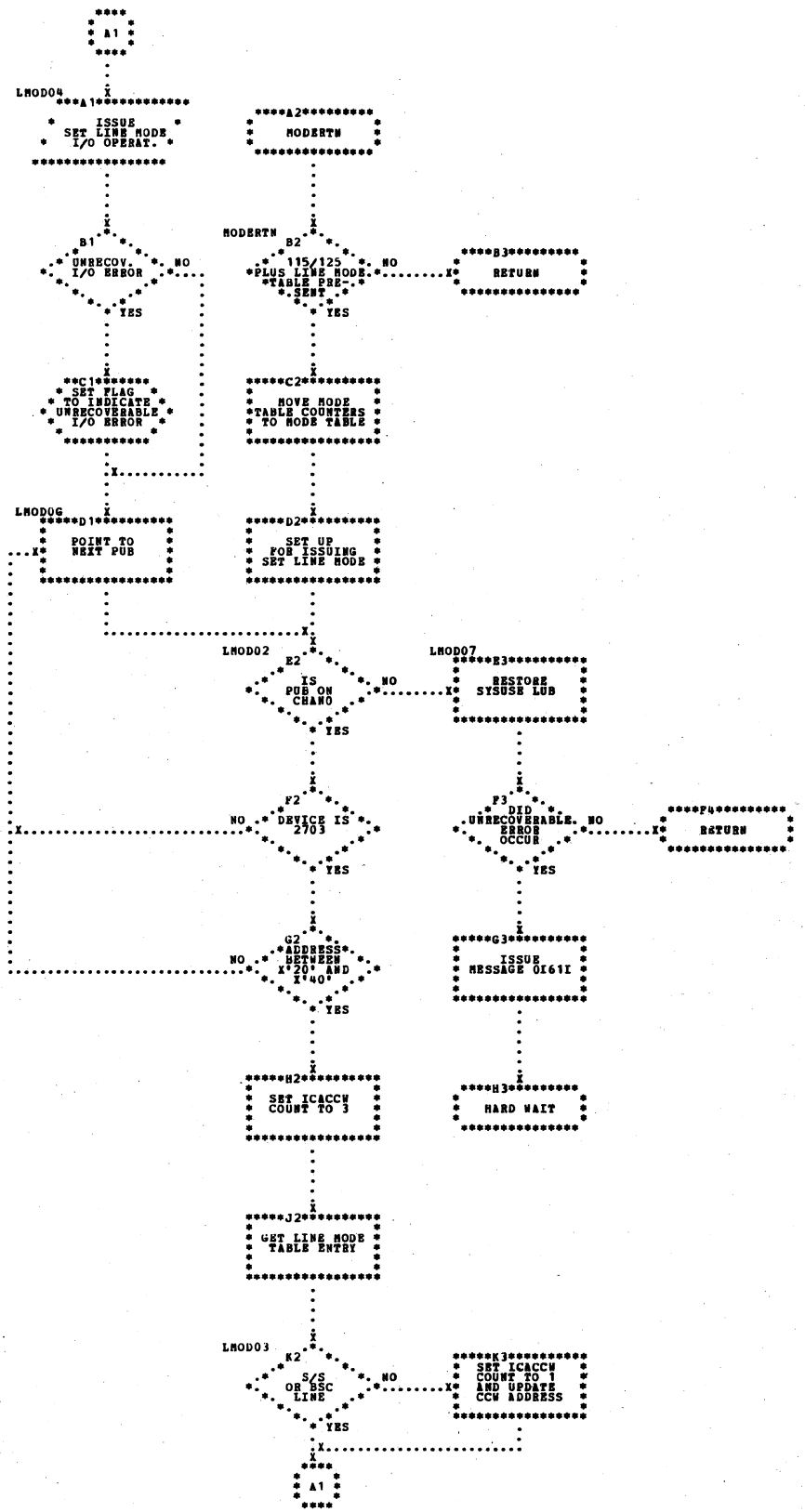


Chart ER. $IPLRT4 - Subroutines
Refer to Chart 03.

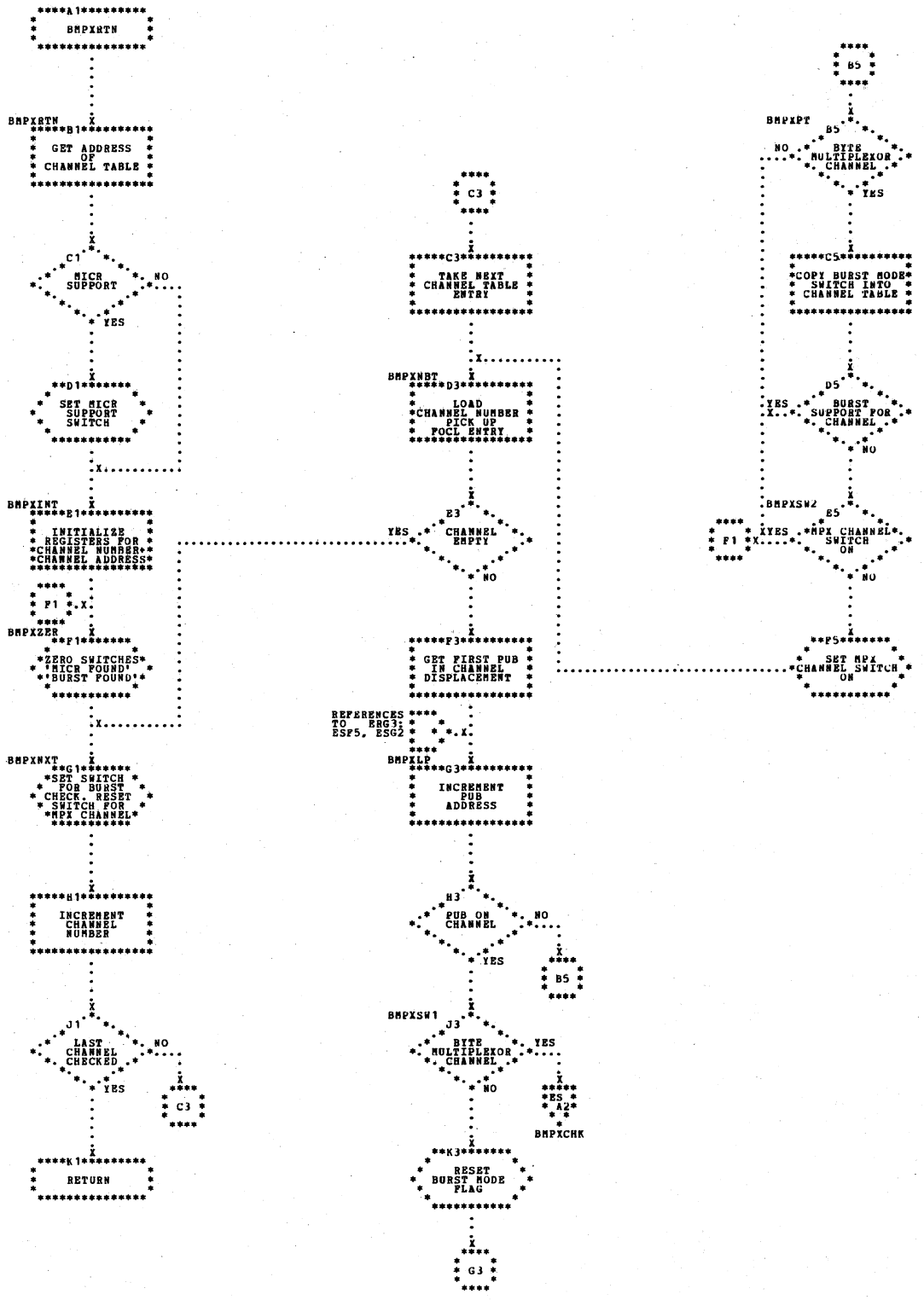


Chart ES. $IPLRT4 - Subroutines
Refer to Chart 03.

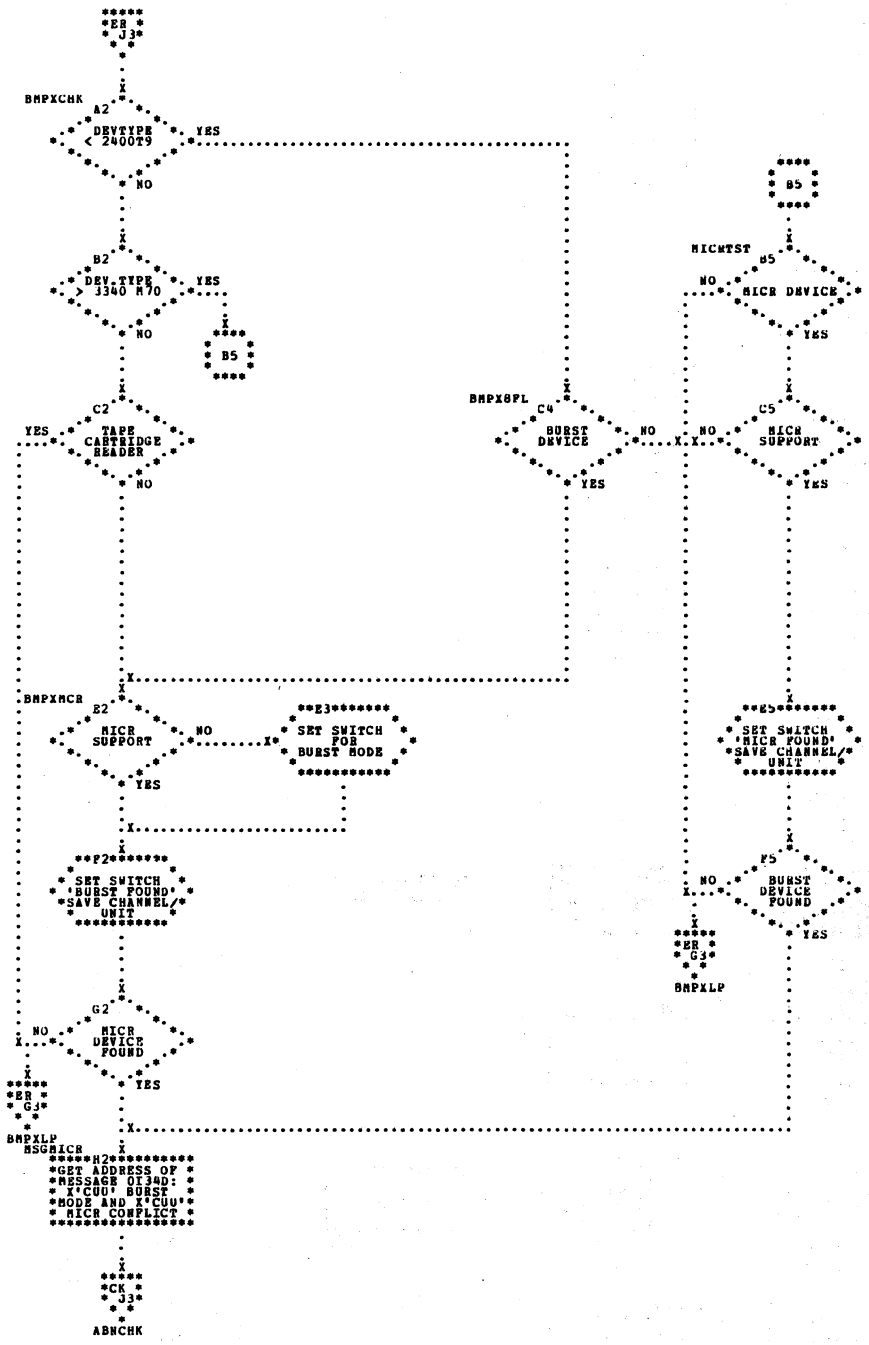


Chart ET. $IPLRT4 - Subroutines
Refer to Chart 03.

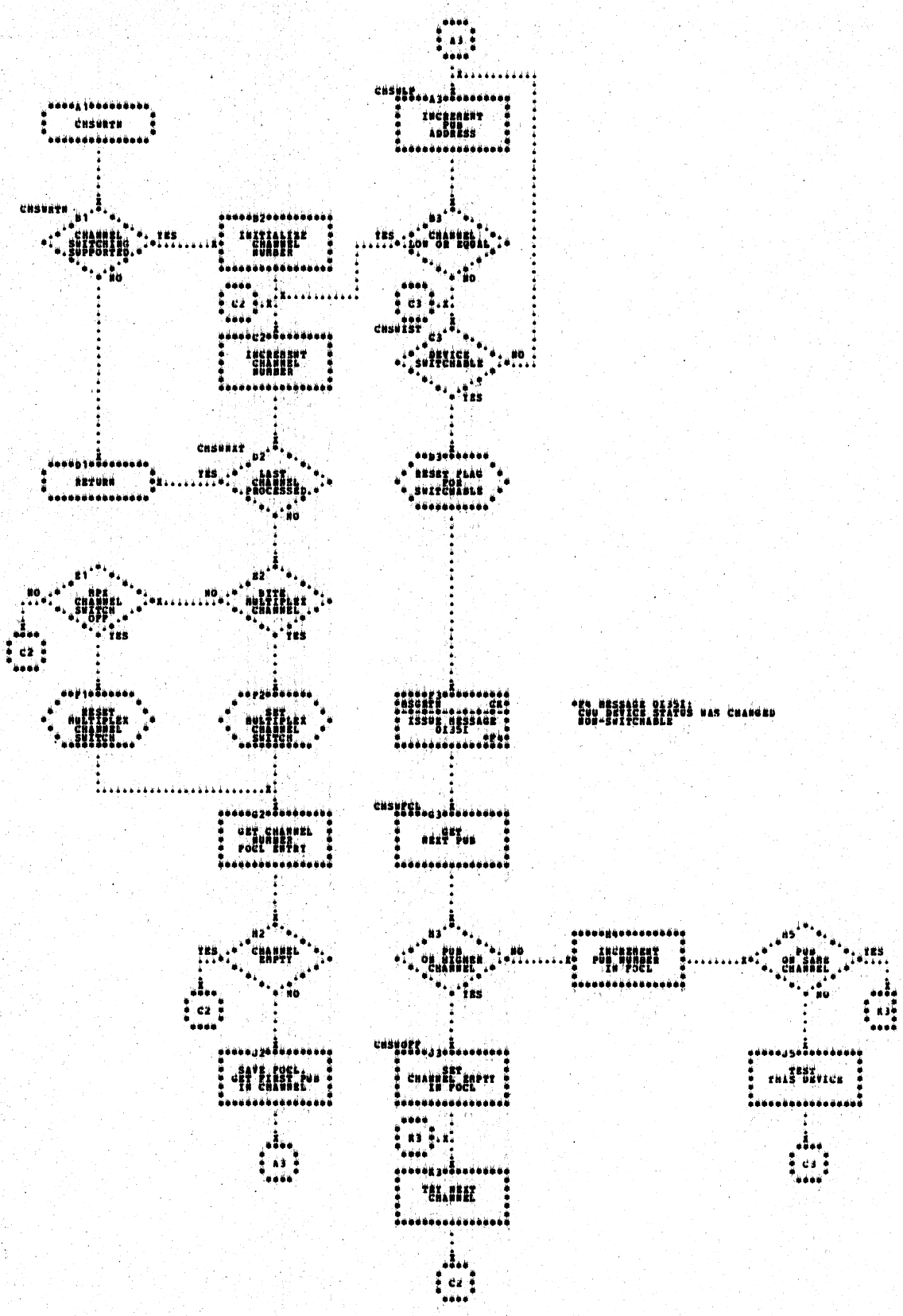


Chart EU. $IPLRT6 - Process SVA Command
Refer to Chart 04.

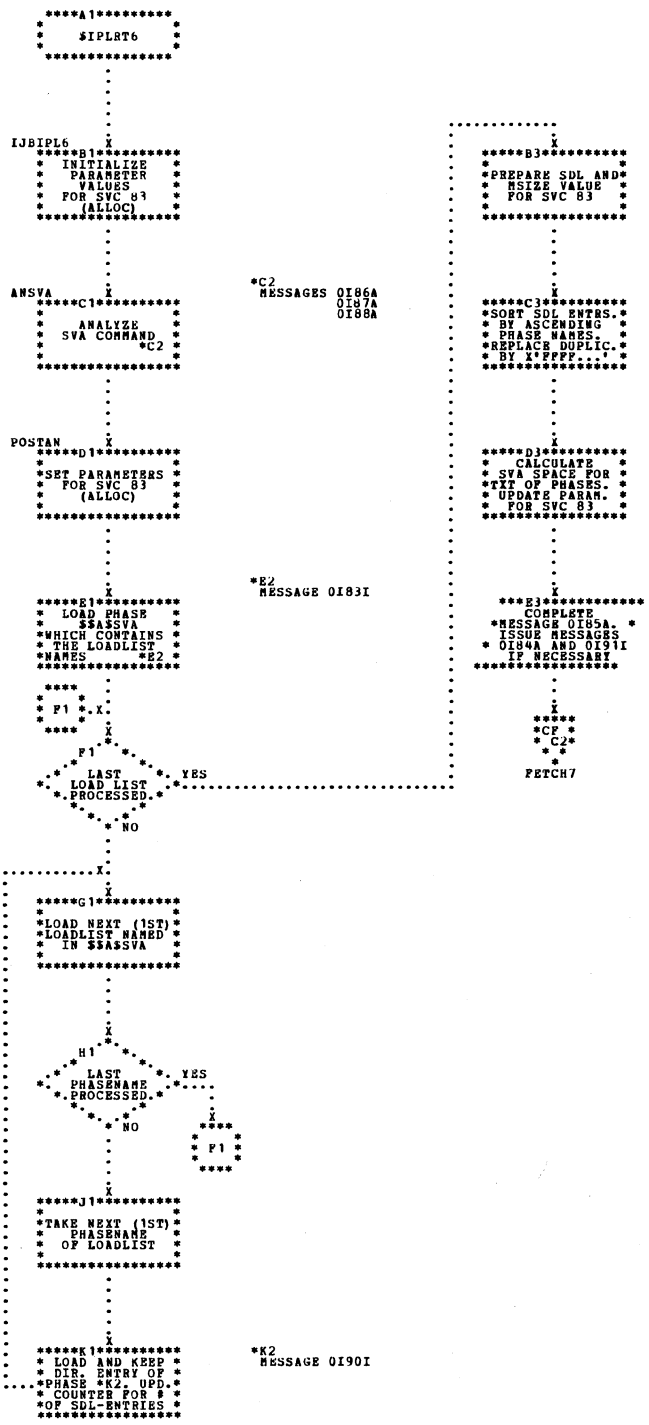


Chart EV. $IPLRT7 - Load SVA and Final IPL Processing
Refer to Chart 04.

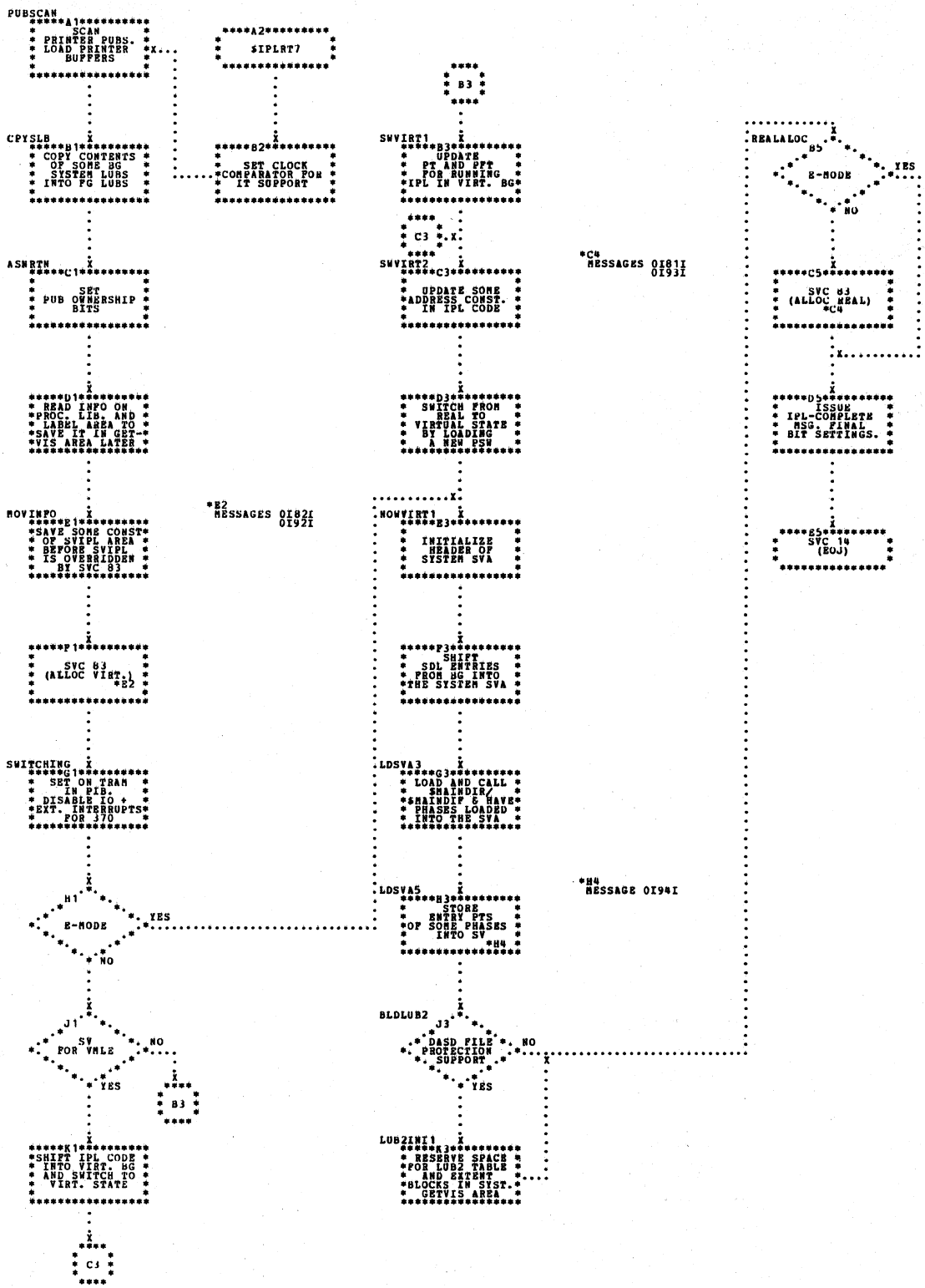


Chart FA. $IPLRT5 - DEF and DPD Processors
Refer to Chart 04.

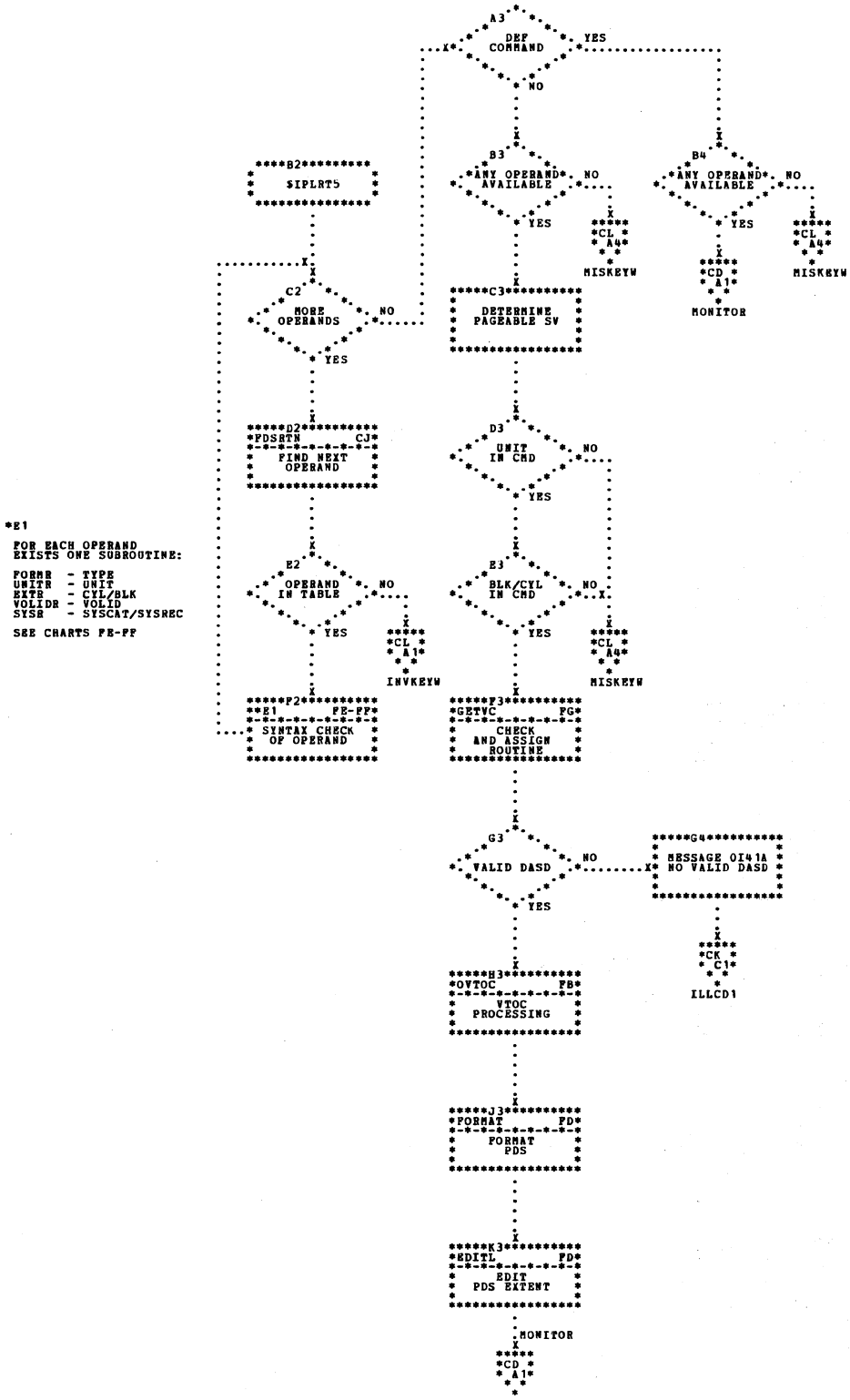


Chart FB. $IPLRT5 - DPD Processor, OPEN and Overlay Check
Refer to Chart 04.

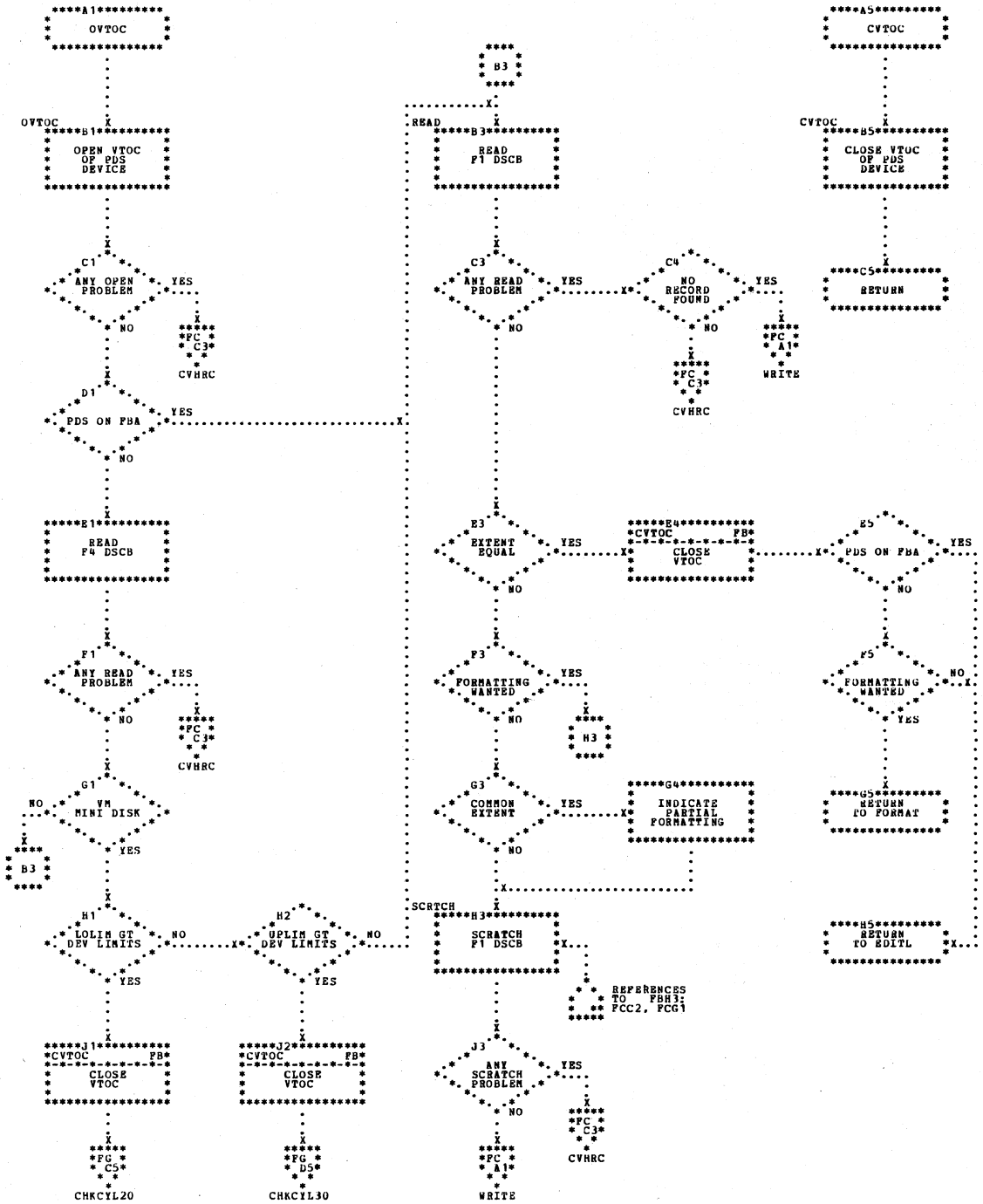


Chart FC. $IPLRT5 - OPEN and Overlay Check
Refer to Chart 04.

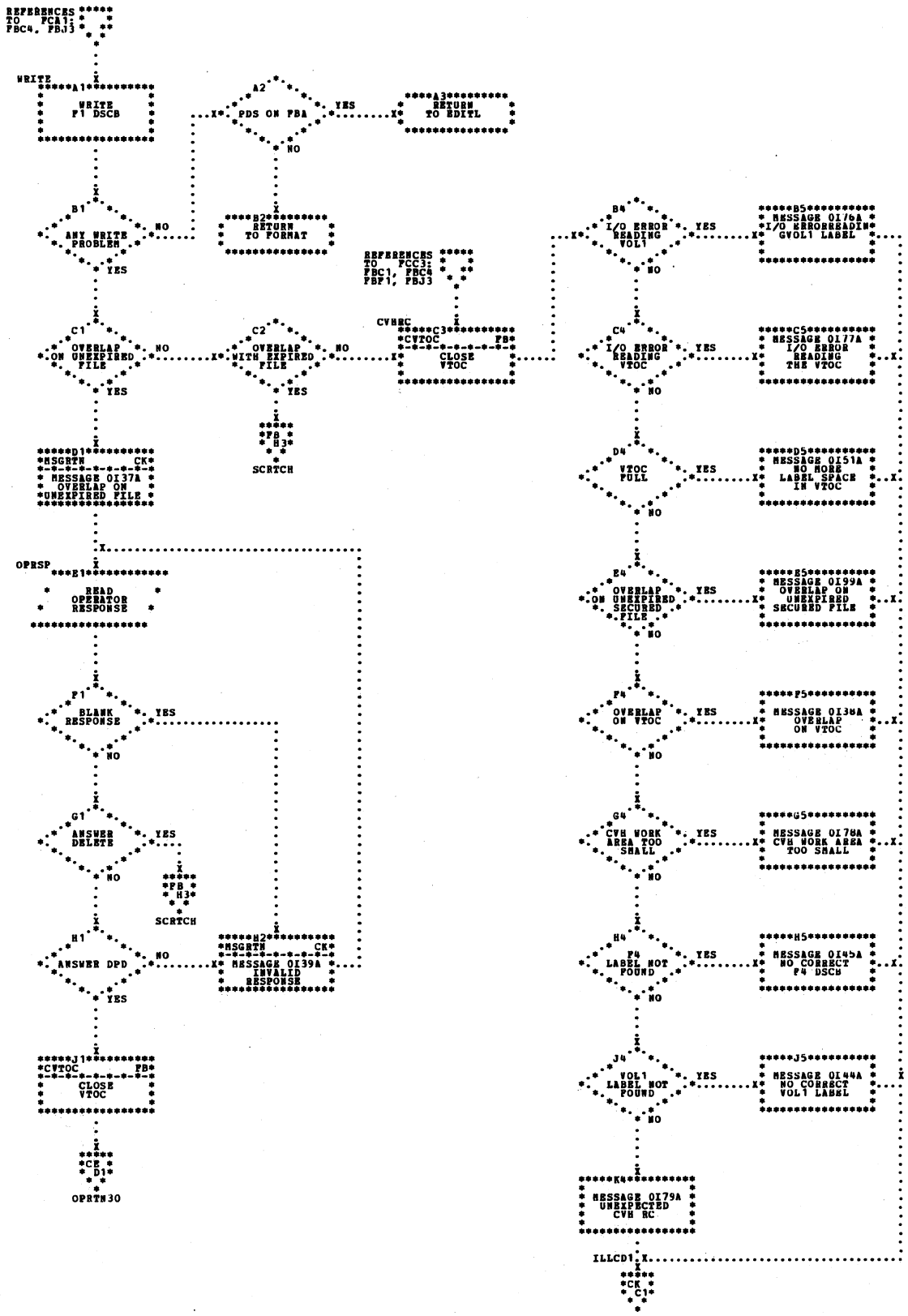


Chart FD. $IPLRT5 - DPD Processing. Formatting
Refer to Chart 04.

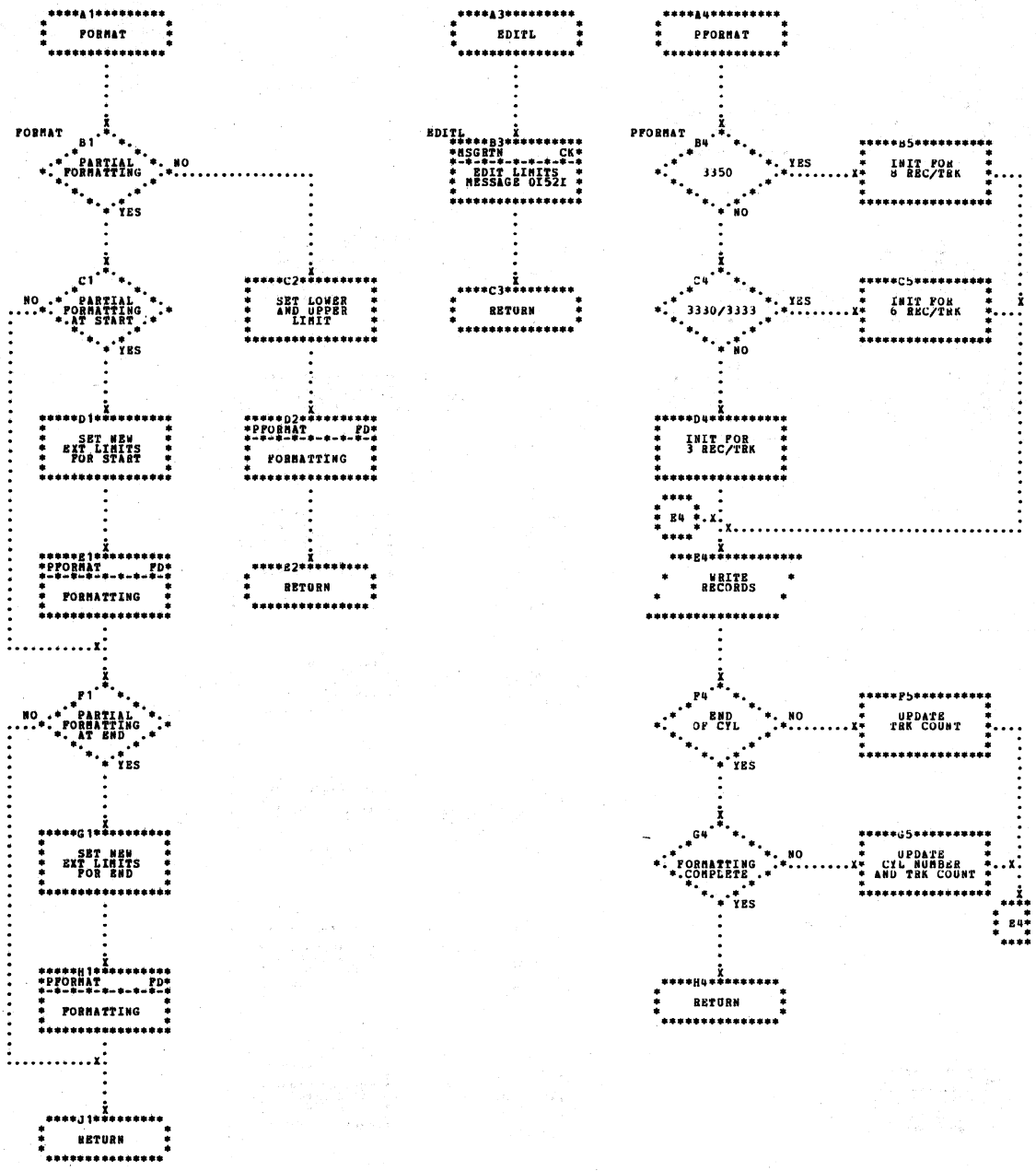


Chart FE. $IPLRT5 - DEF and DPD Operand Processing (Part 1 of 2)
Refer to Chart 04.

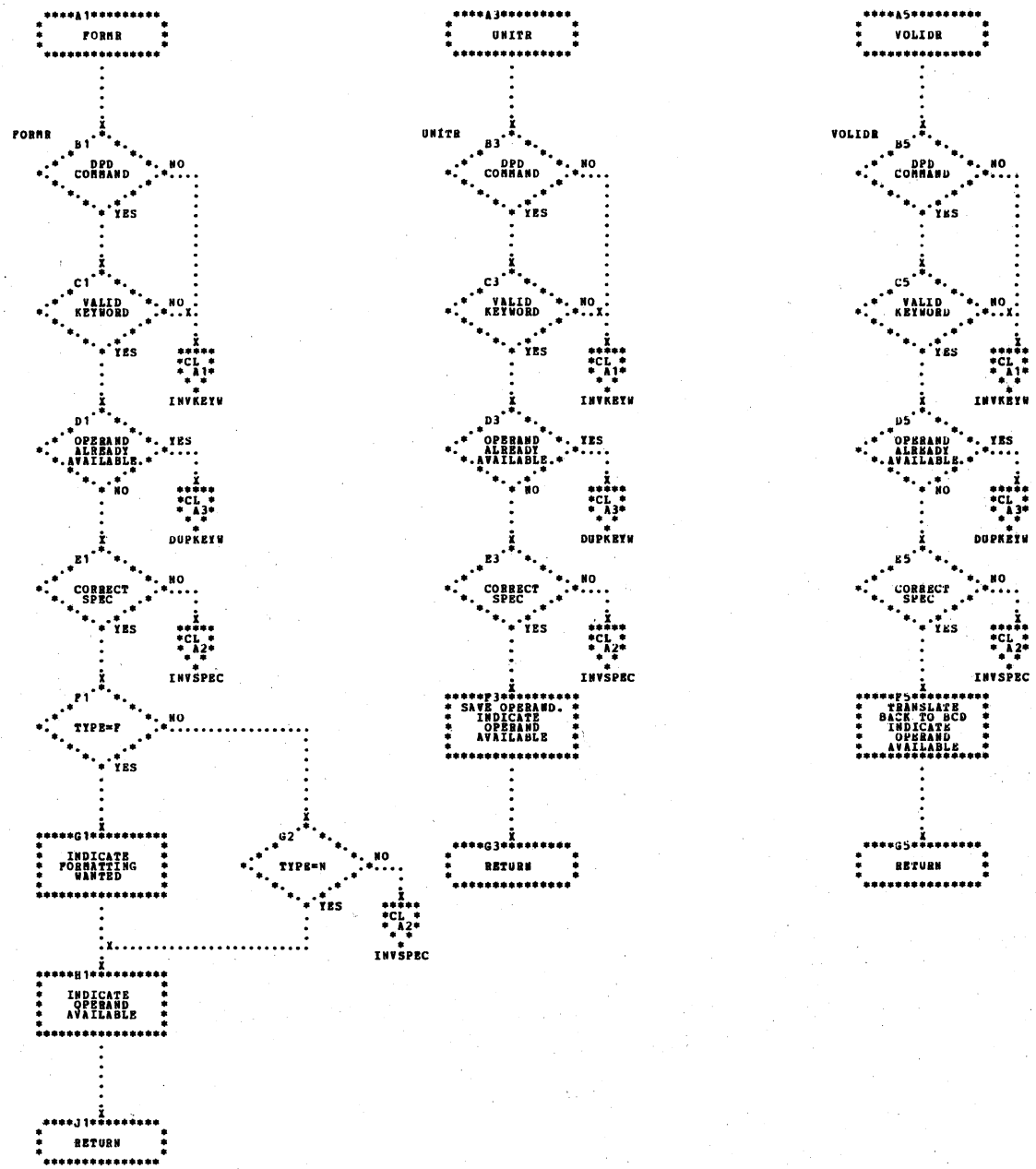


Chart FF. $IPLRT5 - DEF and DPD Operand Processing (Part 2 of 2)
Refer to Chart 04.

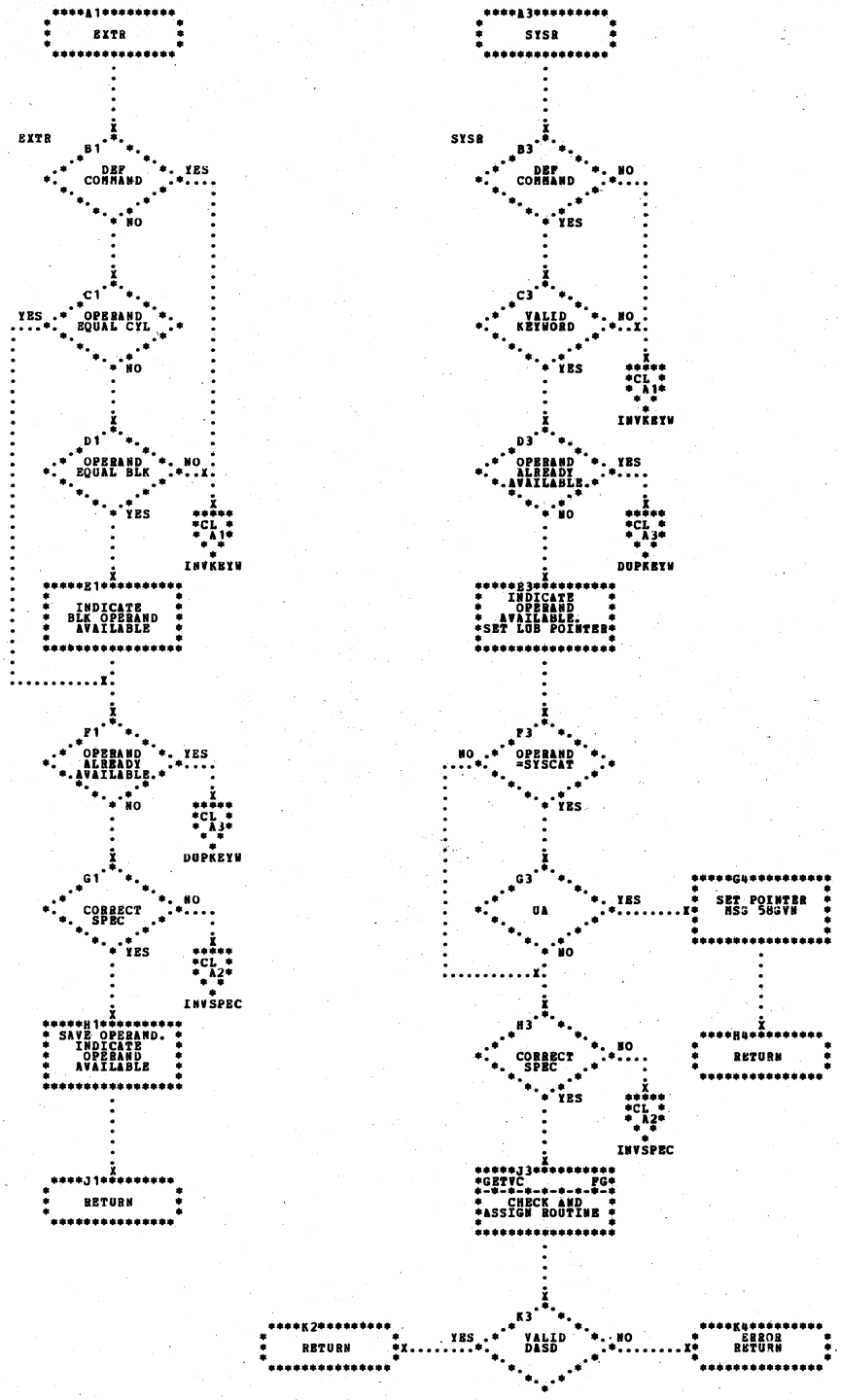


Chart FG. $IPLRT5 - DEF and DPD Check and Assign Routines
Refer to Chart 04.

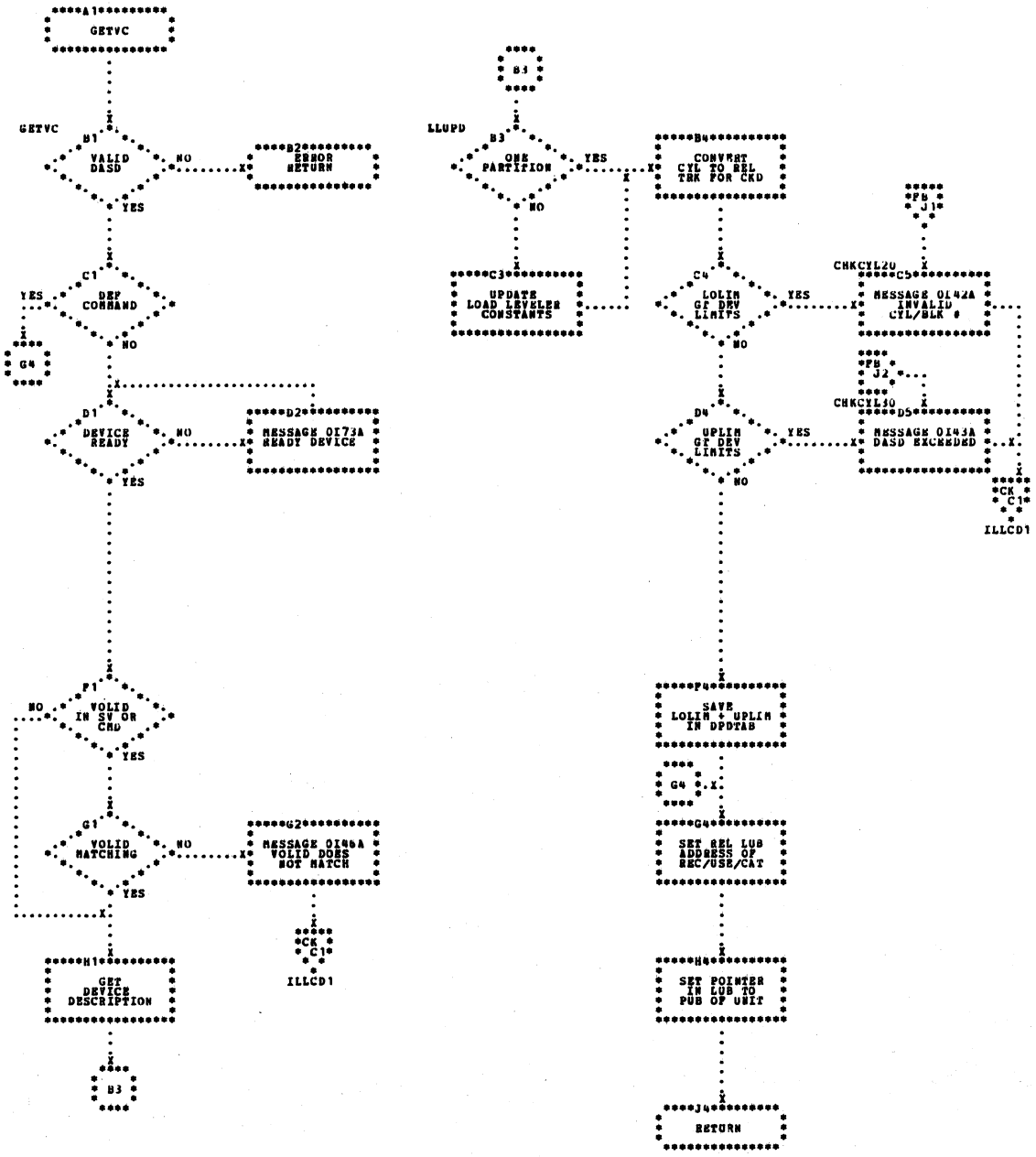


Chart FH. $$BUFLDR - UCB Loading for 3203, 3800, PRT1, 1403U, and 5203 Printers
Refer to Chart 05.

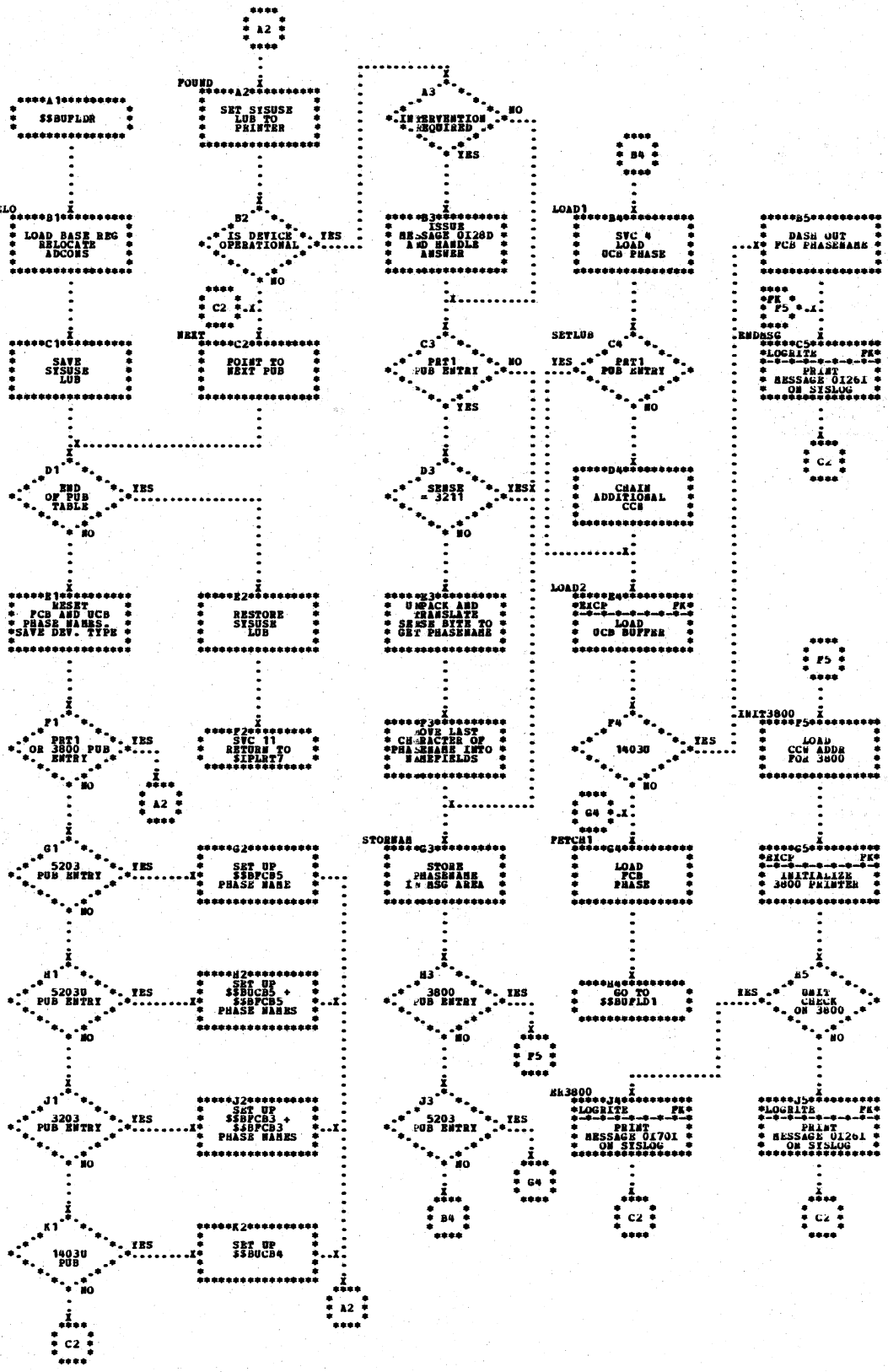


Chart FK. $$BUFLDR - Print Messages on SYSLOG, FCB Loading, and Error Recovery
Refer to Chart 05.

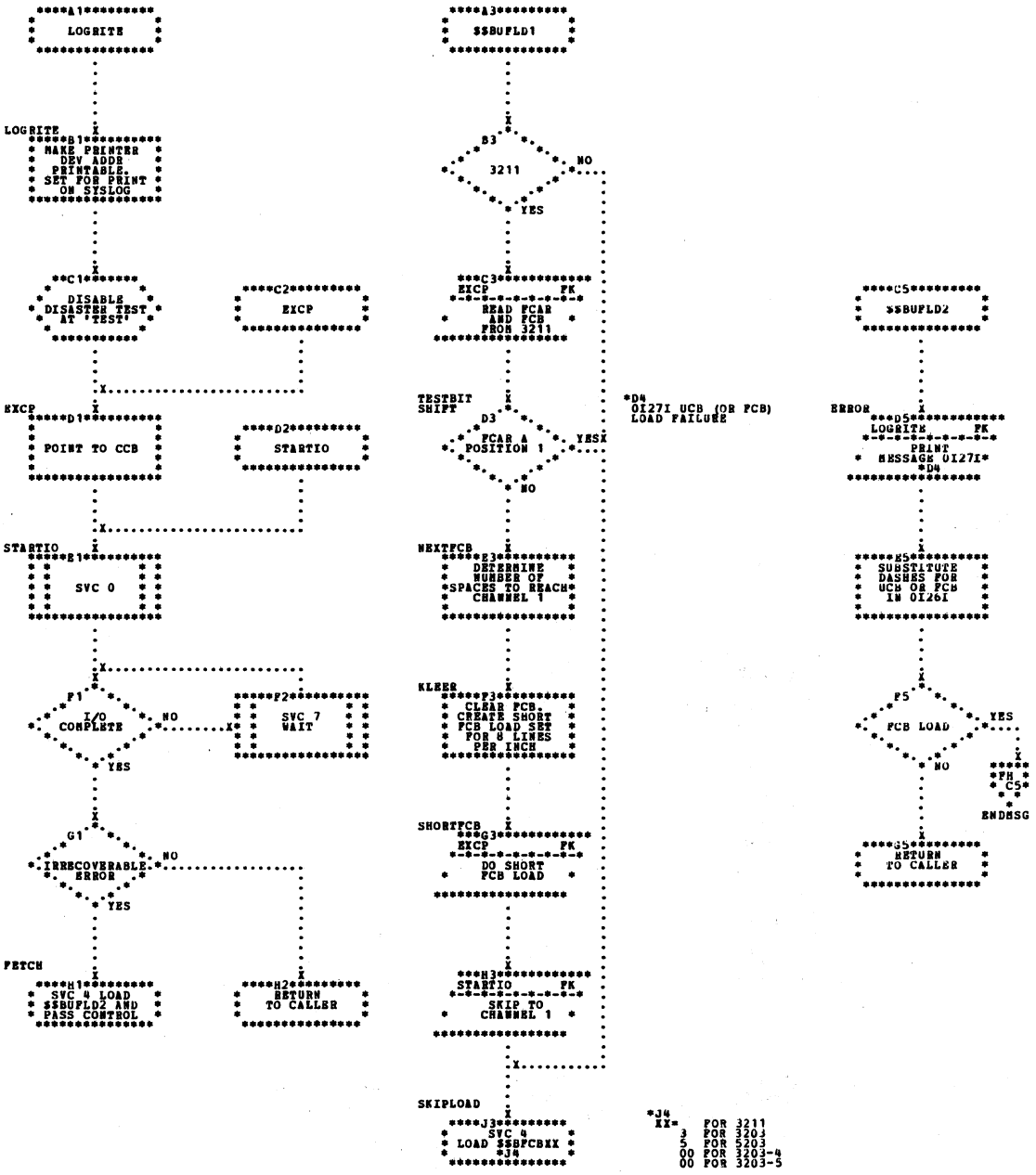


Chart FM. SYSBUFLD - Read Statement and Print on SYSLOG and SYSLST
Refer to Chart 06.

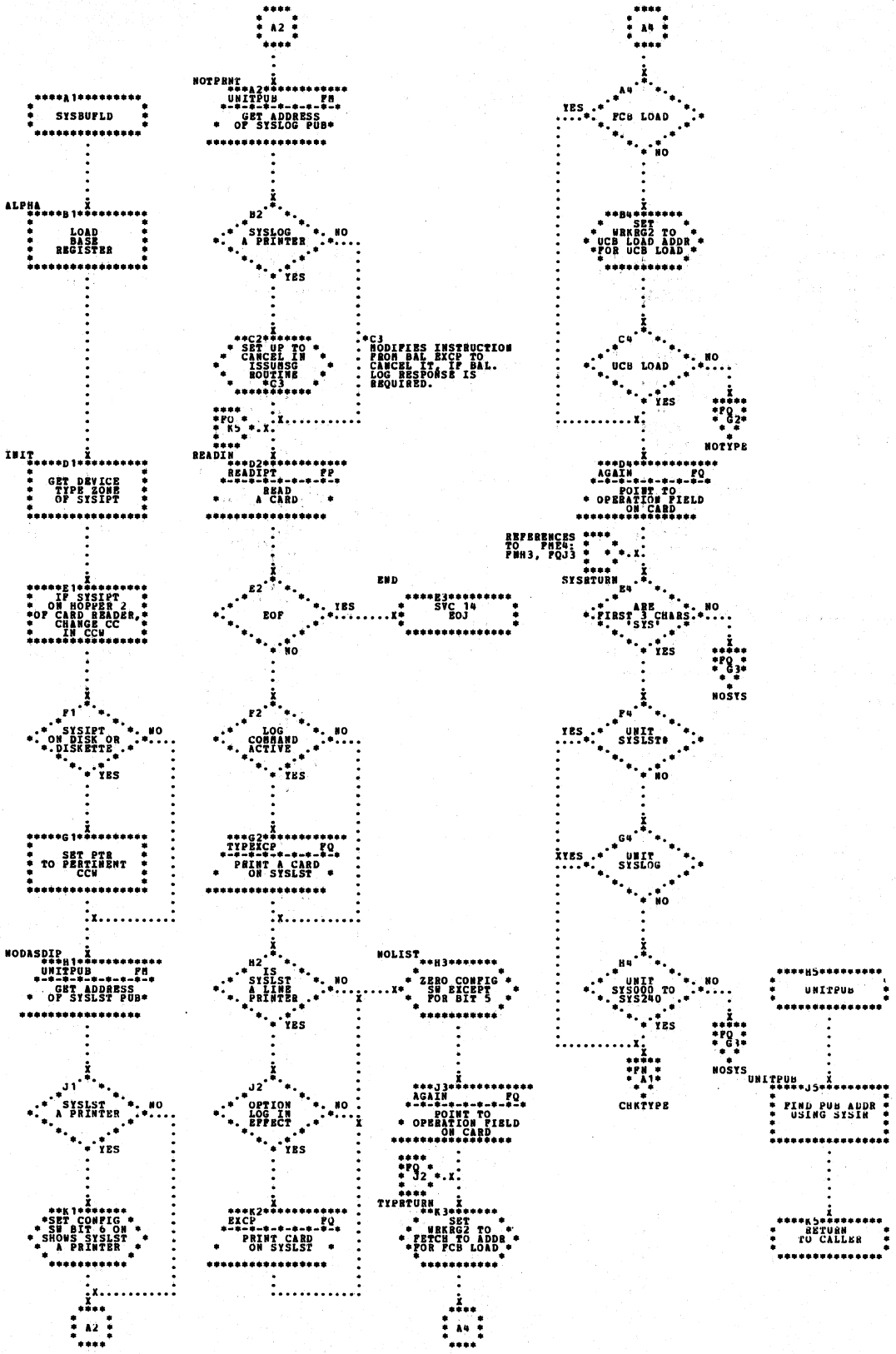


Chart FN. SYSBUFLD - Check for Valid Device and Command
Refer to Chart 06.

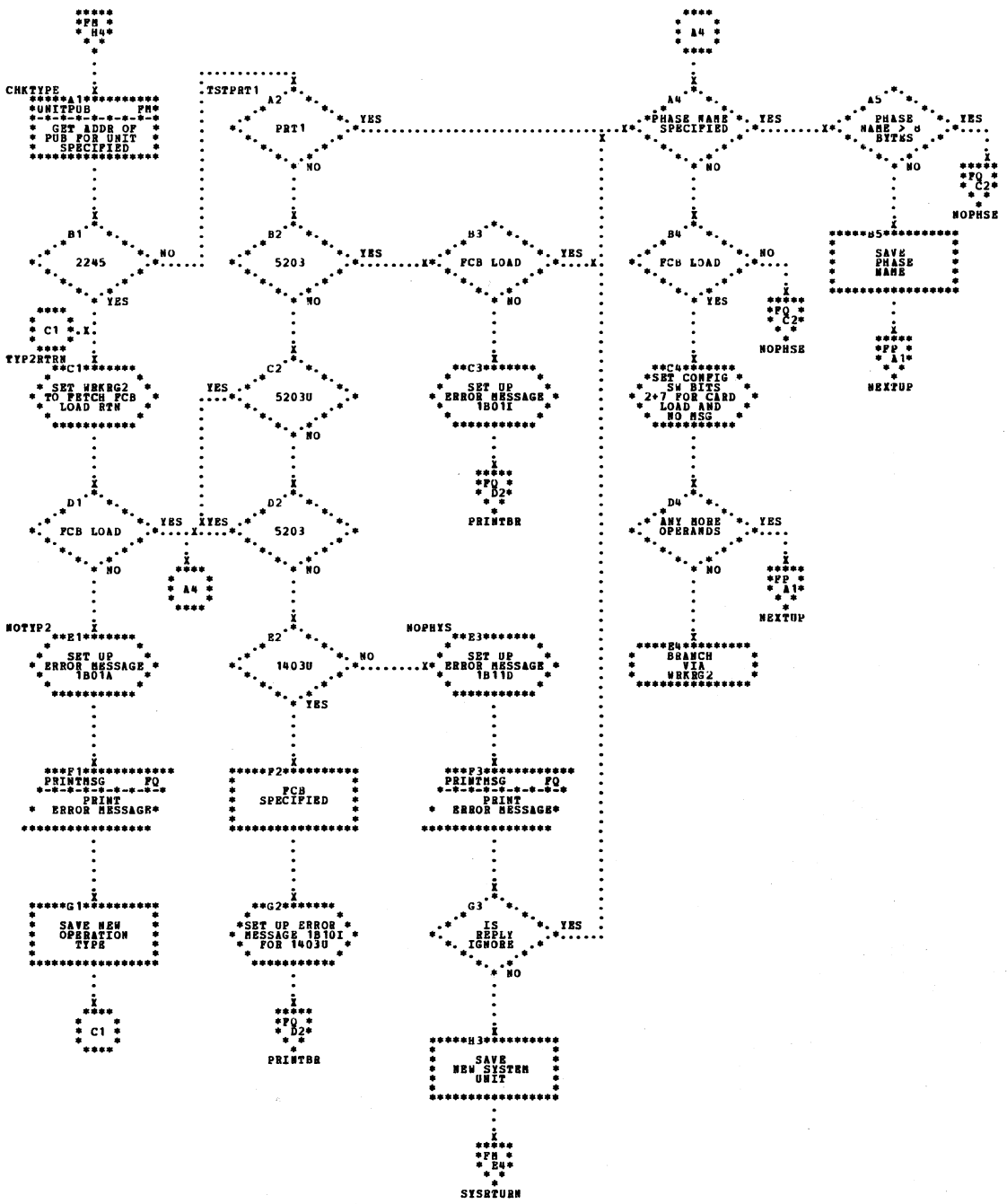


Chart FO. SYSBUFLD - Load UCB
Refer to Chart 06.

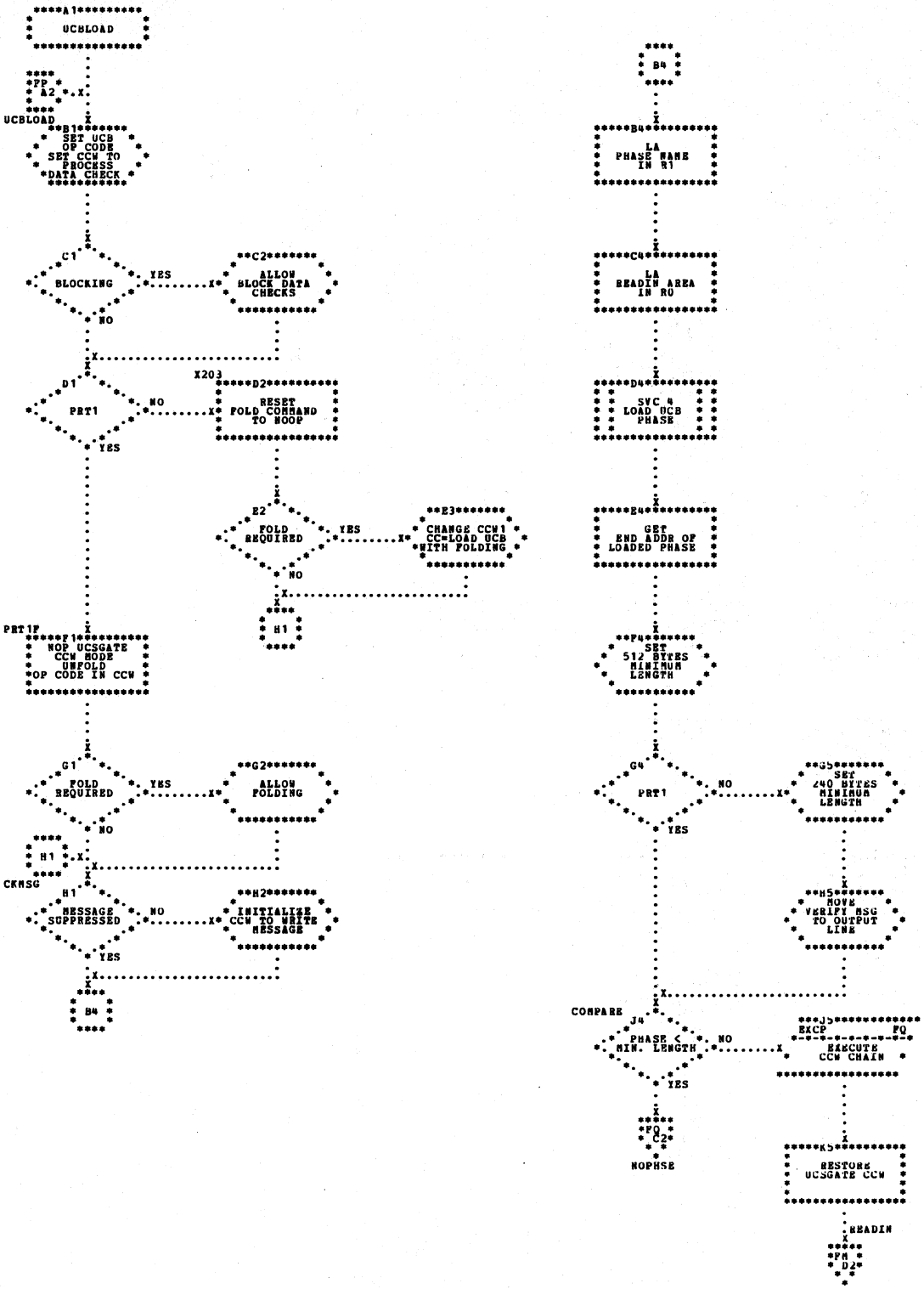


Chart FP. SYSBUFLD - Check Optional Operands
Refer to Chart 06.

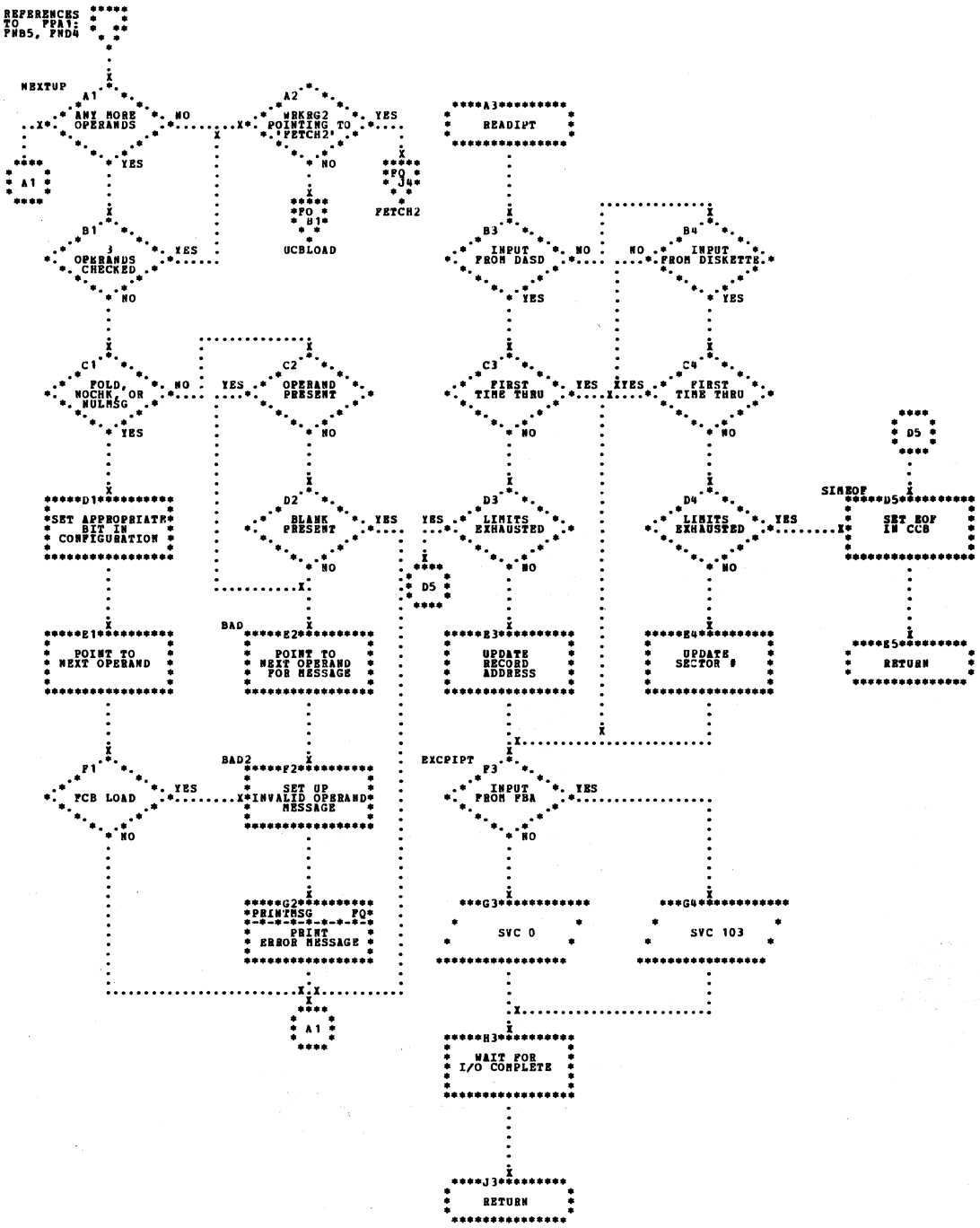


Chart FQ. SYSBUFLD - Subroutines
Refer to Chart 06.

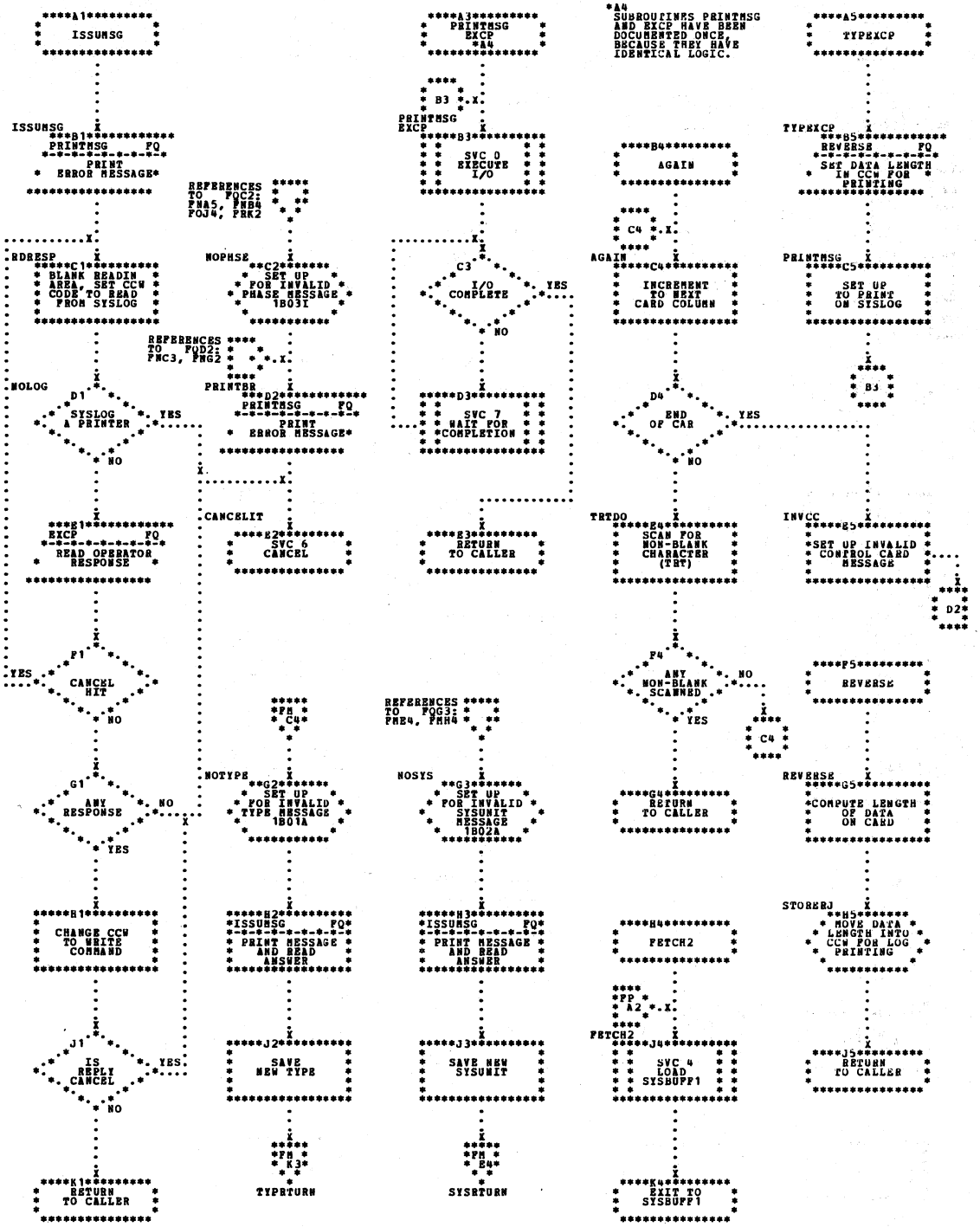


Chart FR. SYSBUFF1 - Load FCB (Part 1 of 2)
Refer to Chart 06.

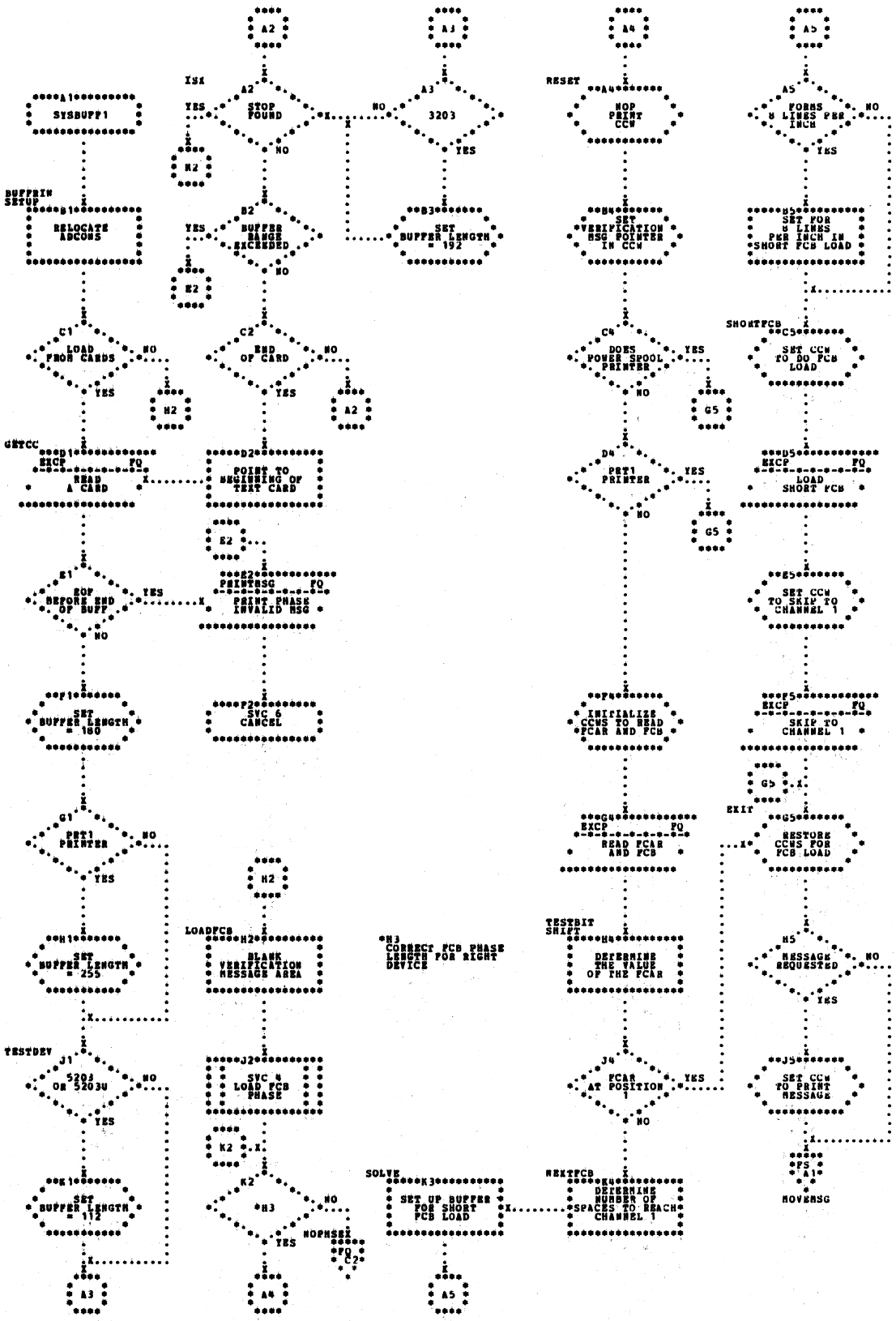


Chart FS. SYSBUFF1 - Load FCB (Part 2 of 2)
Refer to Chart 06.

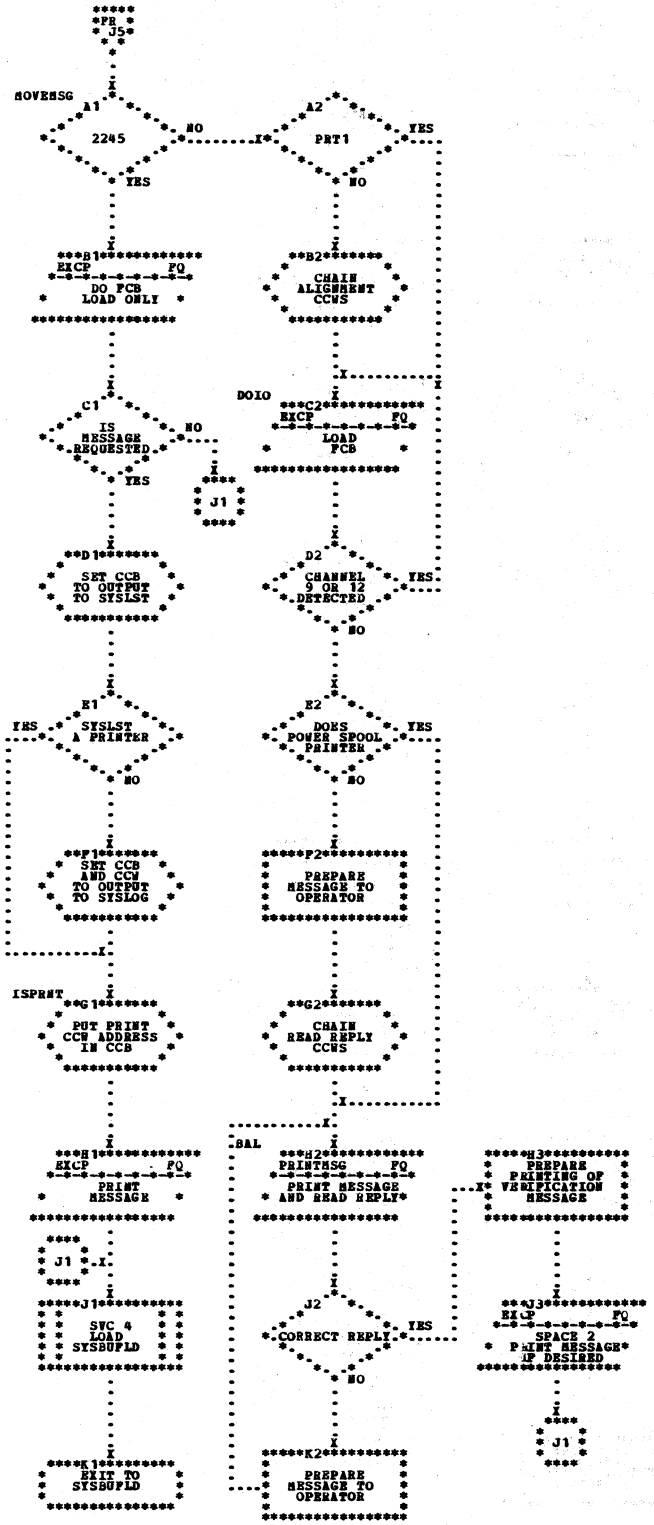


Chart GA. $JOBCTLA - Initialization (Part 1 of 2)
Refer to Charts 07 and 08.

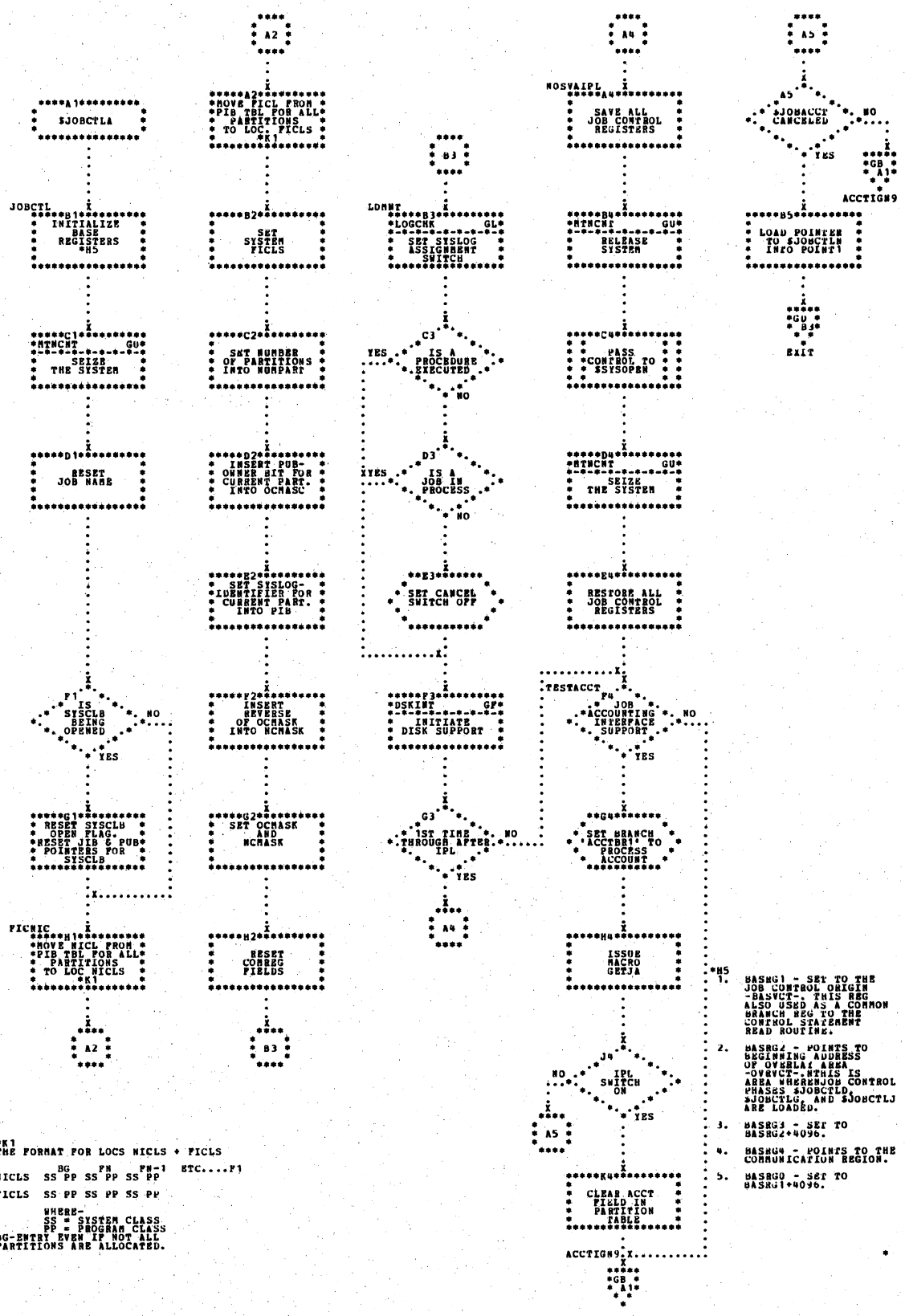


Chart GB. $JOBCTLA - Initialization (Part 2 of 2)
Refer to Charts 07 and 08.

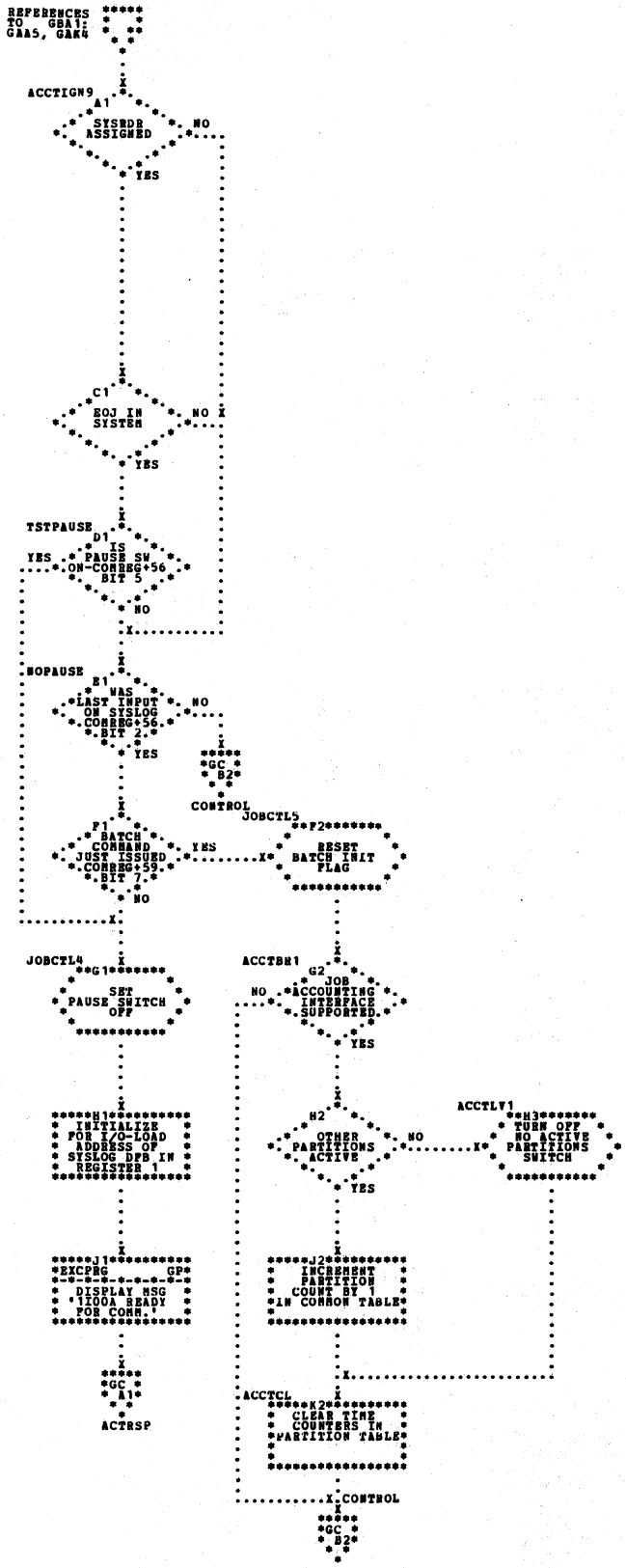


Chart GC. $JOBCTLA - Read Control Statement
Refer to Charts 07 and 08.

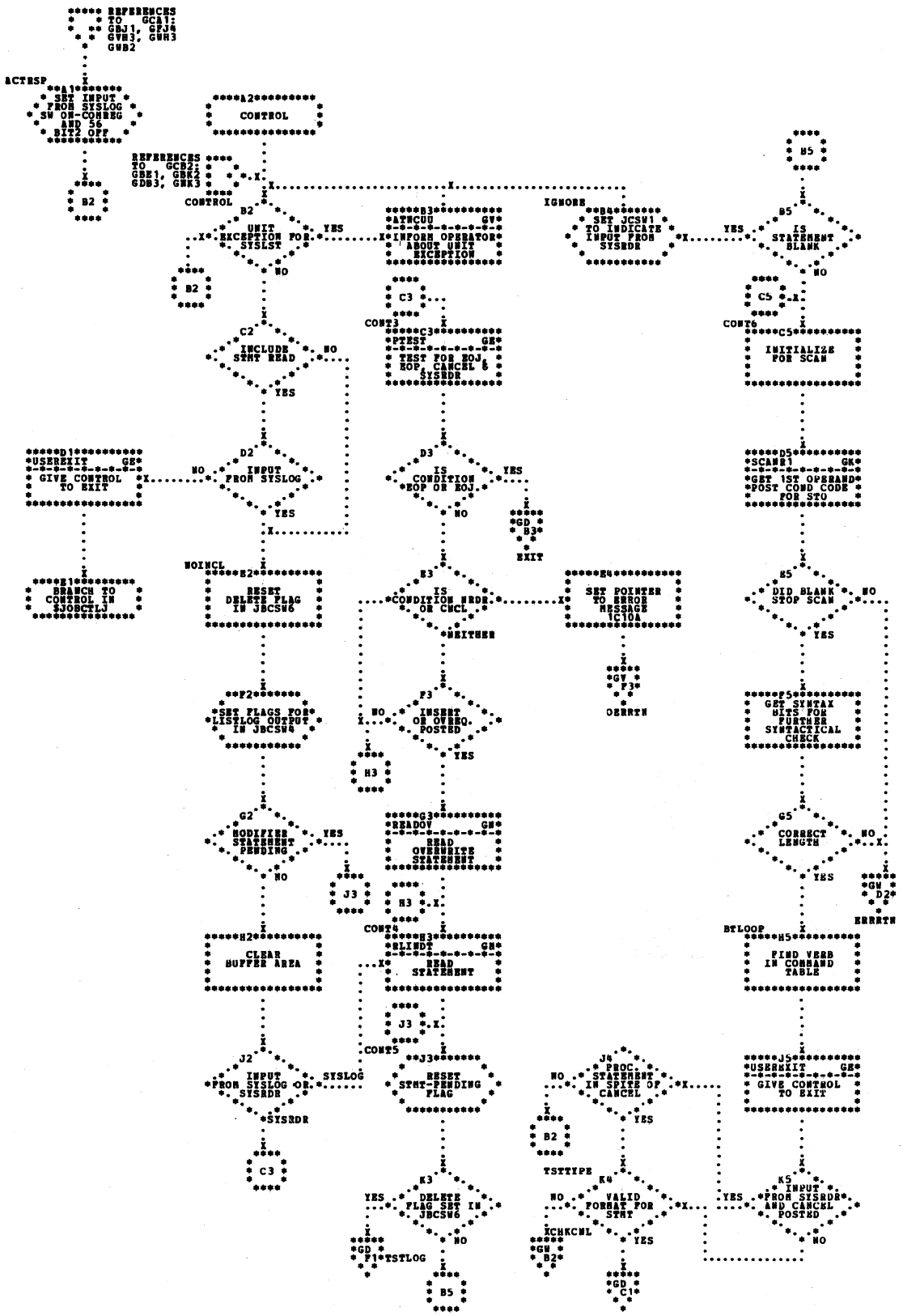


Chart GD. $JOBCTLA - Read Control Statement and Phase Vector Table Lookup
Refer to Charts 07 and 08.

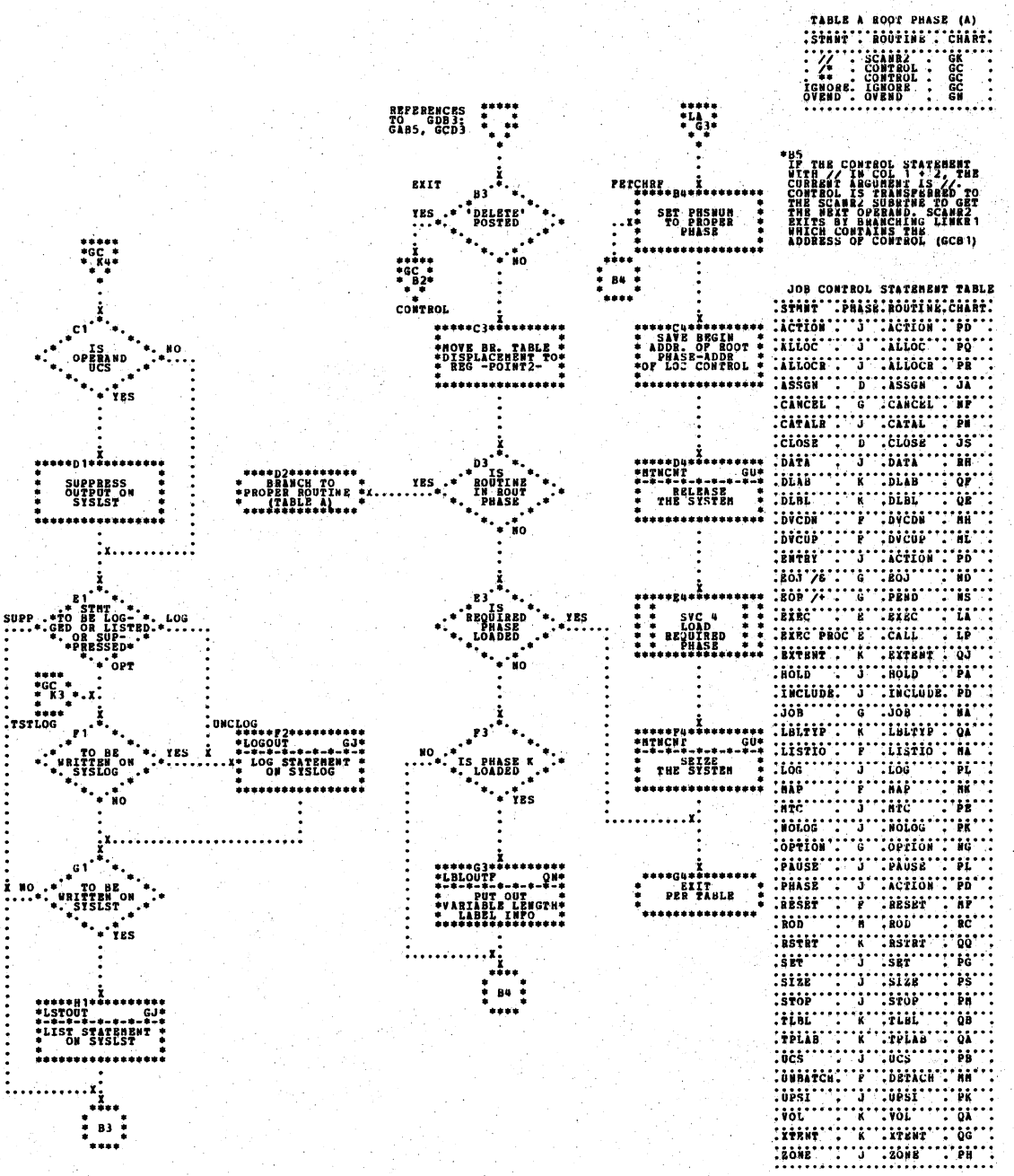


TABLE A ROOT PHASE (A)

STMNT	ROUTINE	CHART
//	SCANR2	GK
/*	CONTROL	GC
**	CONTROL	GC
IGNORE	IGNORE	GC
OVEND	OVEND	GN

*B5
IF THE CONTROL STATEMENT WITH // IN COL 1 + 2, THE CURRENT ARGUMENT IS */, CONTROL IS TRANSFERRED TO THE SCANR2 SUBRTNE TO GET THE NEXT OPERAND. SCANR2 EXITS BY BRANCHING LINKE1 WHICH CONTAINS THE ADDRESS OF CONTROL (GC81)

JOB CONTROL STATEMENT TABLE

STMNT	PHASE	ROUTINE	CHART
ACTION	J	ACTION	PD
ALLOC	J	ALLOC	PQ
ALLOCR	J	ALLOCR	PR
ASSGN	D	ASSGN	JA
CANCEL	G	CANCEL	NF
CATALR	J	CATAL	PN
CLOSE	D	CLOSE	JS
DATA	J	DATA	NH
DLAB	K	DLAB	QP
DLBL	K	DLBL	QE
DVCDN	F	DVCDN	NH
DVCUP	F	DVCUP	NL
ENTRY	J	ACTION	PD
EOJ /&	G	EOJ	ND
EOP /+	G	PEND	NS
EXEC	E	EXEC	LA
EXEC PROC	E	CALL	LP
EXTENT	K	EXTENT	QJ
HOLD	J	HOLD	PA
INCLUDE	J	INCLUDE	PD
JOB	G	JOB	NA
LBLTYP	K	LBLTYP	QA
LISTIO	F	LISTIO	NA
LOG	J	LOG	PL
MAP	F	MAP	NK
MTC	J	MTC	PE
NOLOG	J	NOLOG	PK
OPTION	G	OPTION	NG
PAUSE	J	PAUSE	PL
PHASE	J	ACTION	PD
RESET	F	RESET	NP
ROD	H	ROD	RC
RSTRT	K	RSTRT	QQ
SET	J	SET	PG
SIZE	J	SIZE	PS
STOP	J	STOP	PH
TLBL	K	TLBL	QB
TPLAB	K	TPLAB	QA
UCS	J	UCS	PB
UNBATCH	F	DETACH	NH
UPSI	J	UPSI	PK
VOL	K	VOL	QA
XTENT	K	XTENT	QG
ZONE	J	ZONE	PH

Chart GE. $JOBCTLA - PTEST and USEREXIT Routines
Refer to Charts 07 and 08.

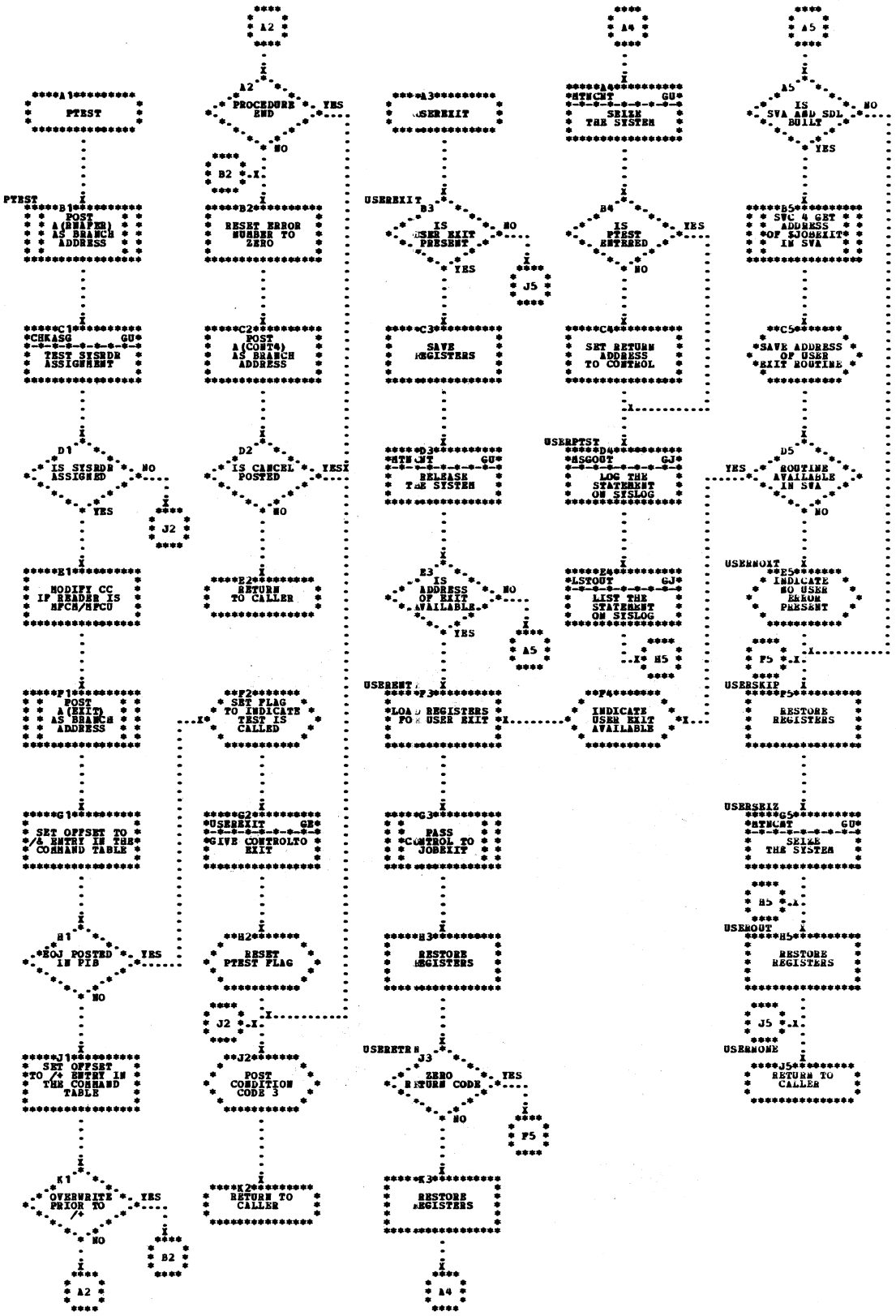


Chart GF. $JOBCTLA - DSKINT Subroutine
Refer to Charts 07 and 08.

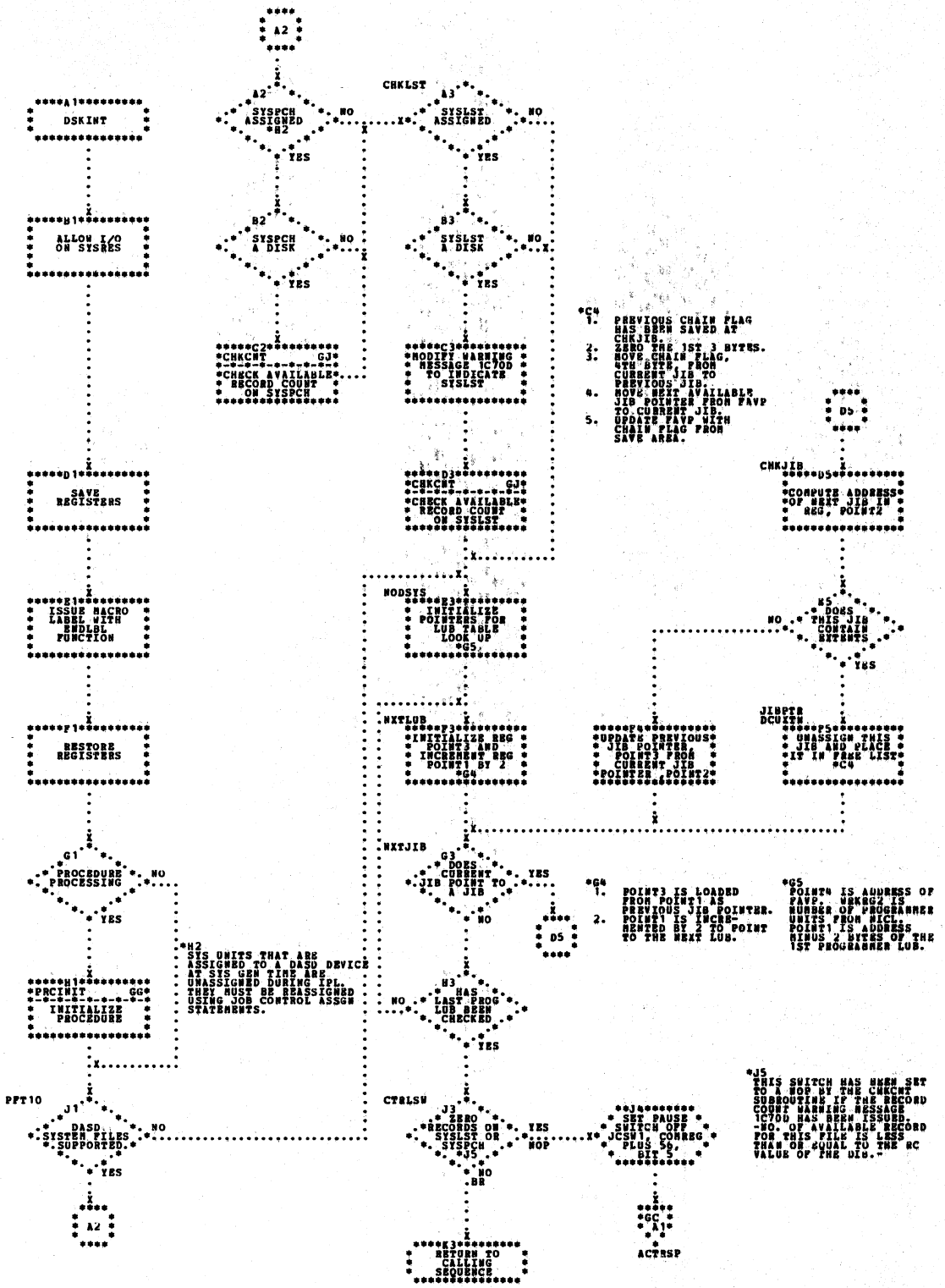


Chart GG. $JOBCTLA - Initialization of Procedure Processing
Refer to Charts 07 and 08.

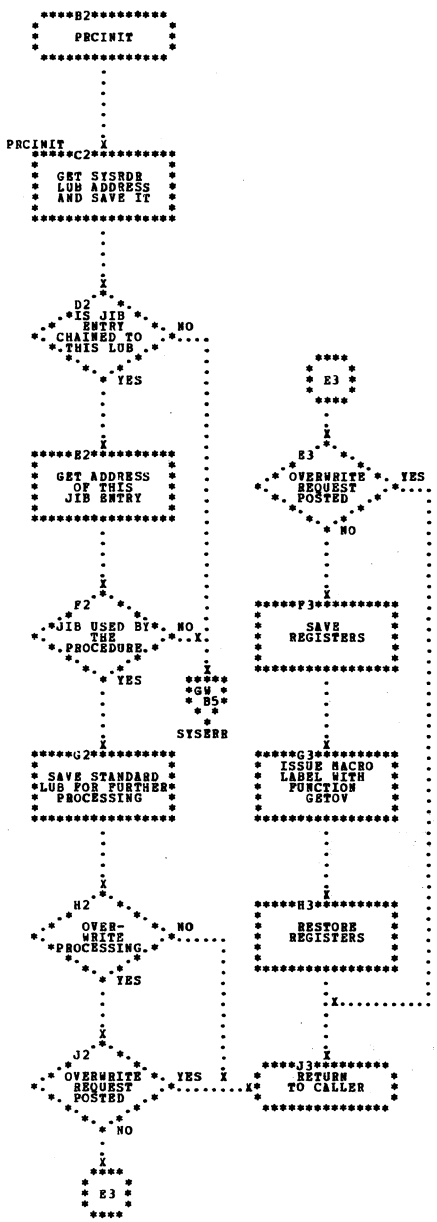


Chart GJ. $JOBCTLA - Message Subroutines
Refer to Charts 07 and 08.

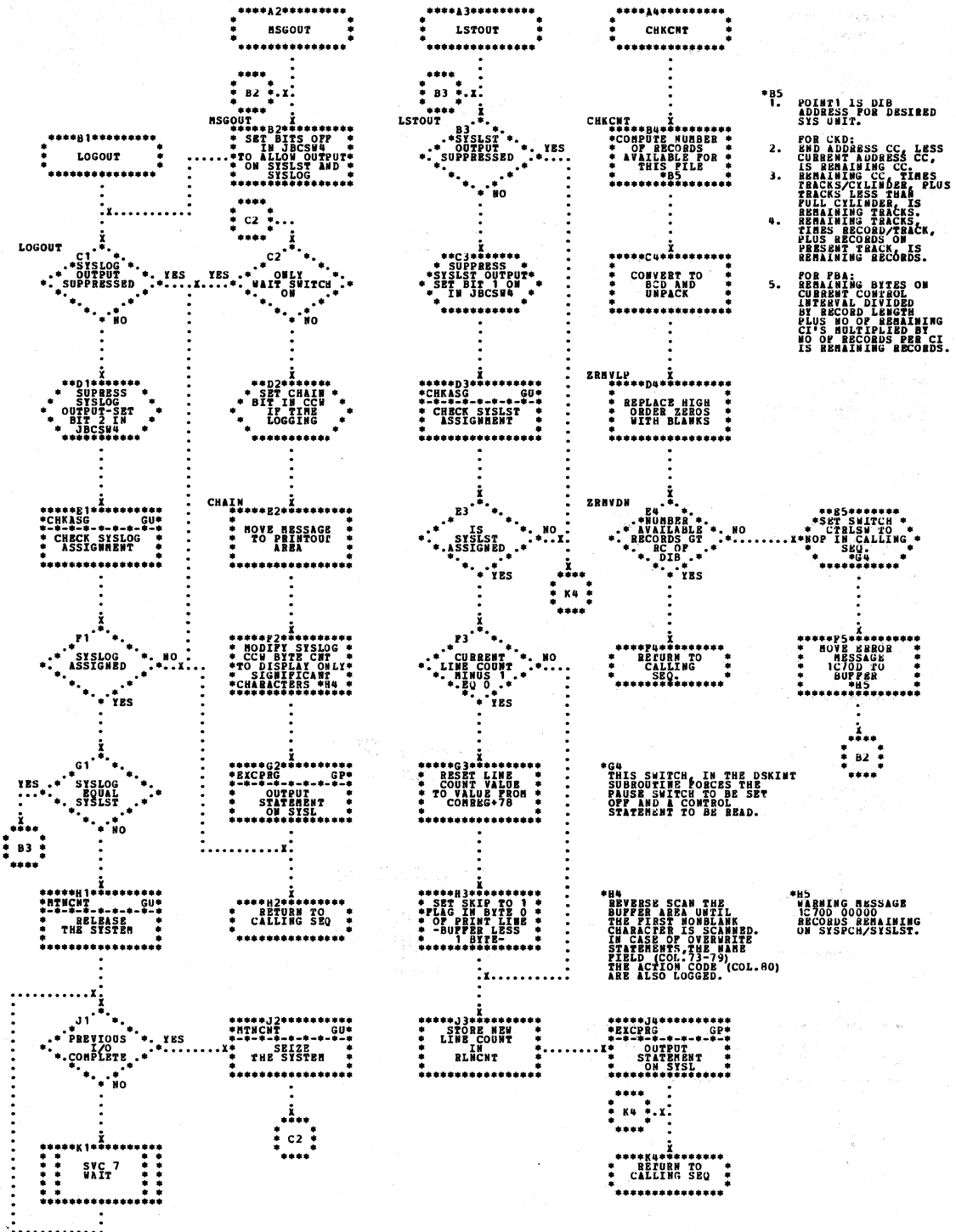


Chart GK. $JOBCTLA - Operand Scan Subroutines
Refer to Charts 07 and 08.

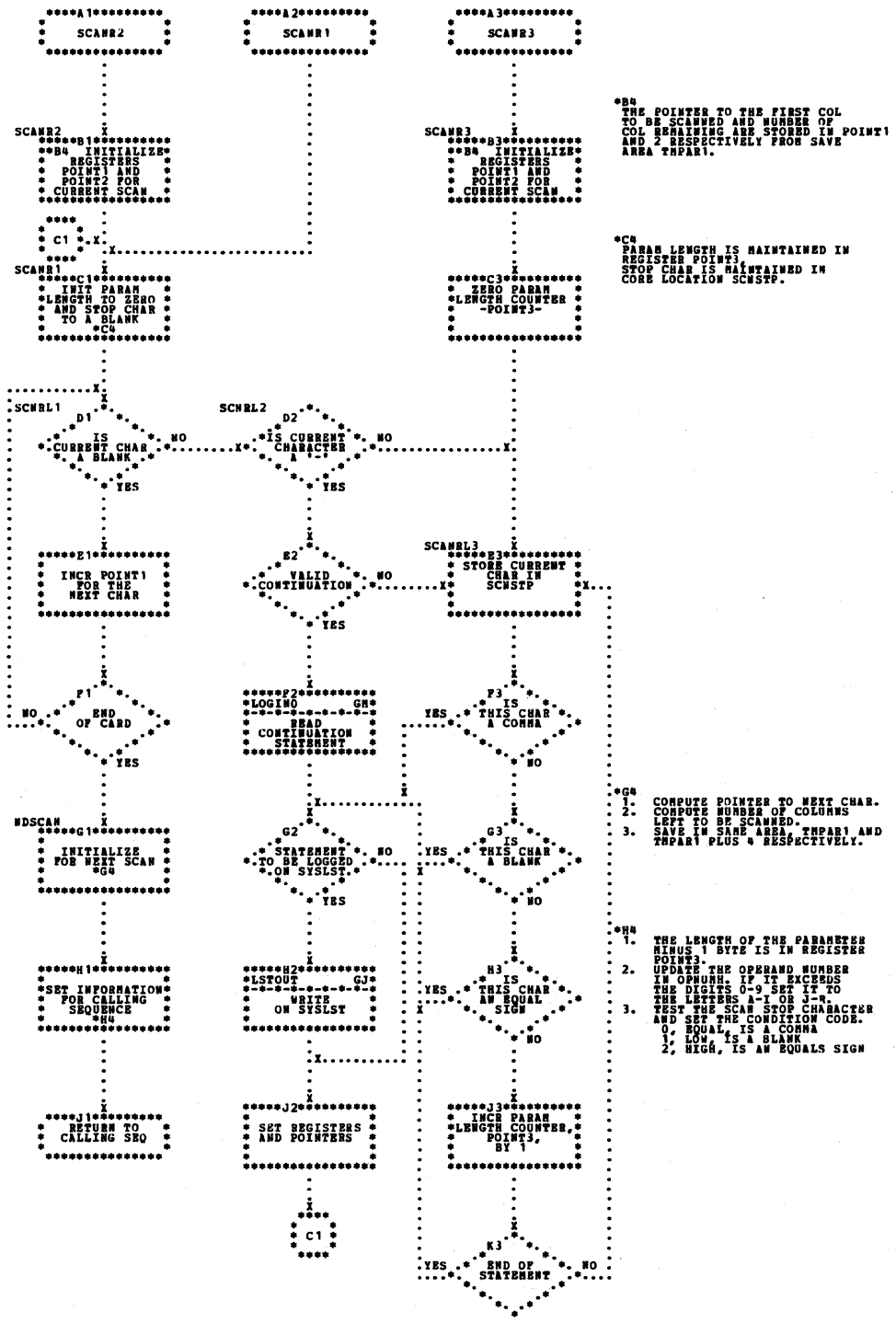


Chart GL. $JOBCTLA - Subroutines
Refer to Charts 07 and 08.

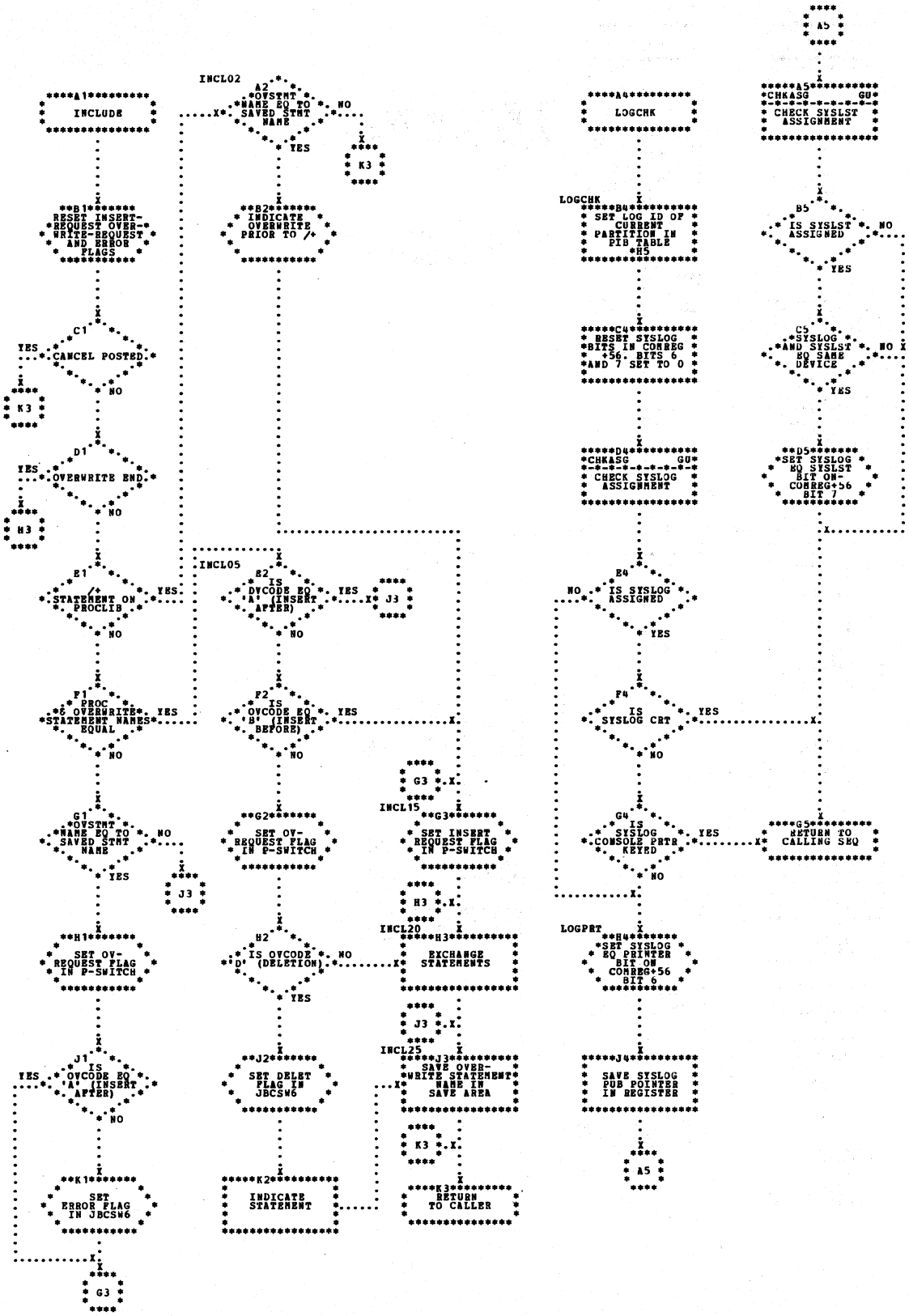


Chart GM. $JOBCTLA - Read Control Statement from SYSLOG or SYSRDR
Refer to Charts 07 and 08.

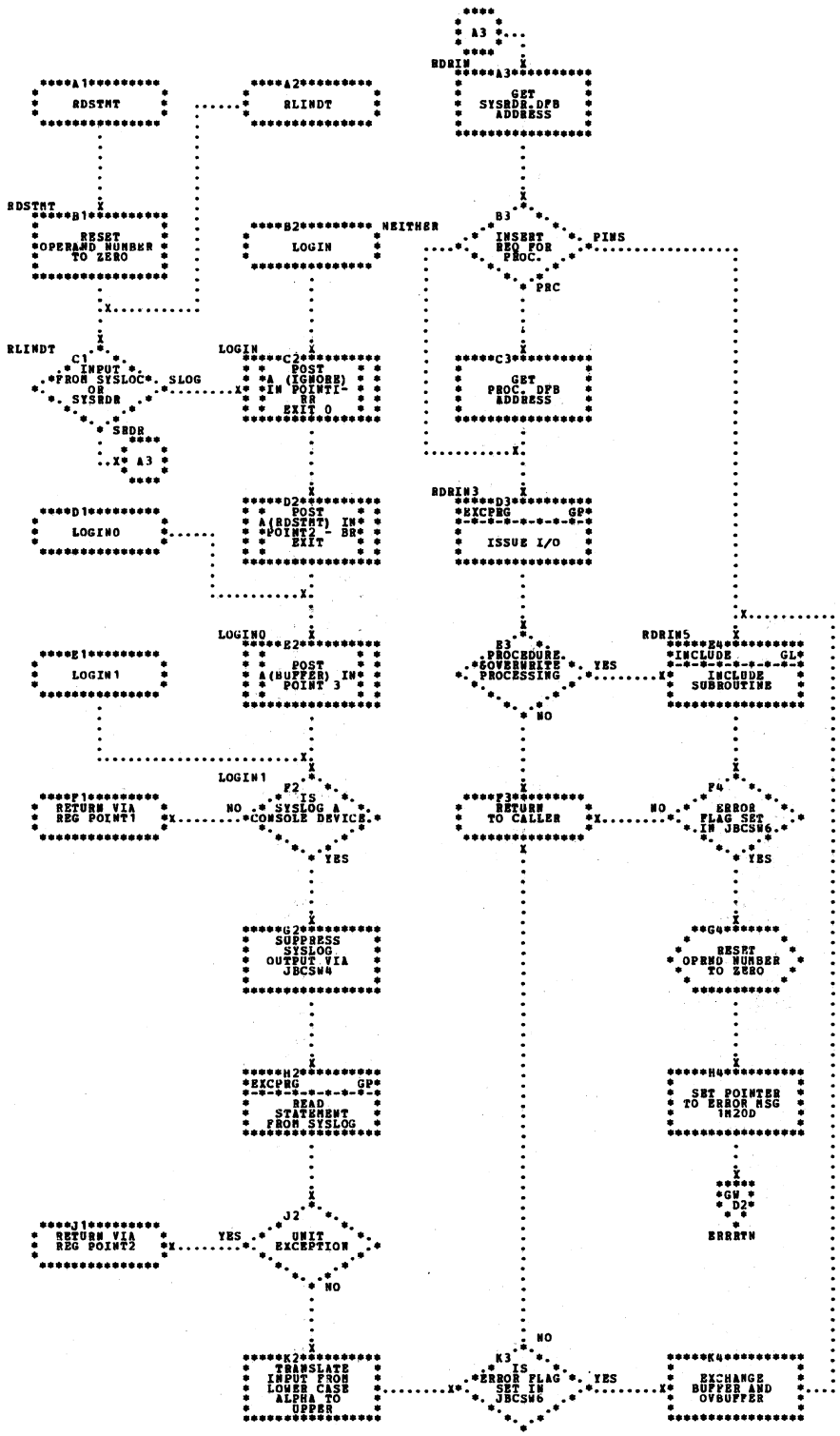


Chart GN. $JOBCTLA - Subroutines for Cataloged Procedure Overwrite Handling
Refer to Charts 07 and 08.

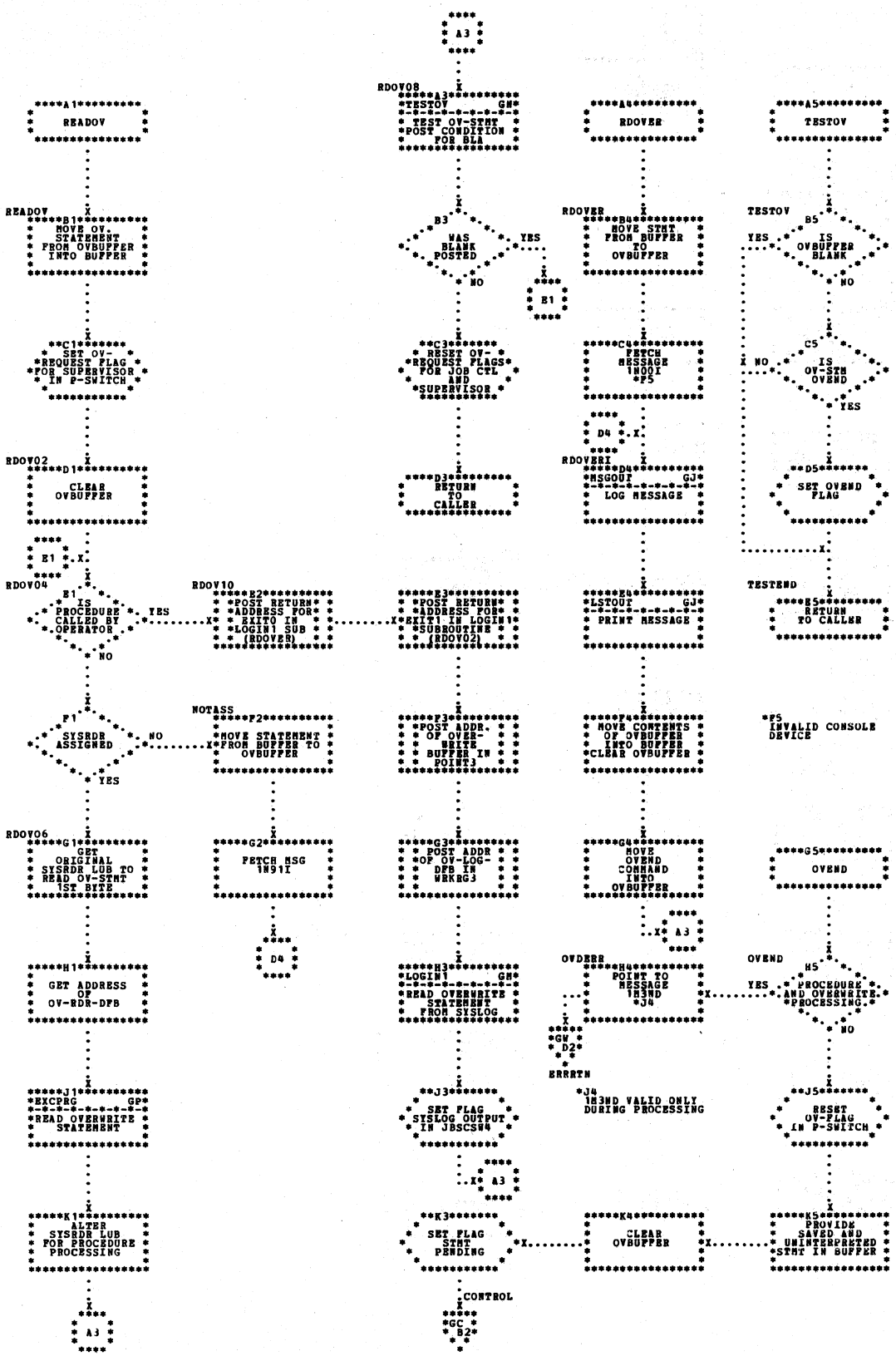


Chart GP. $JOBCTLA - EXCP Subroutines
Refer to Charts 07 and 08.

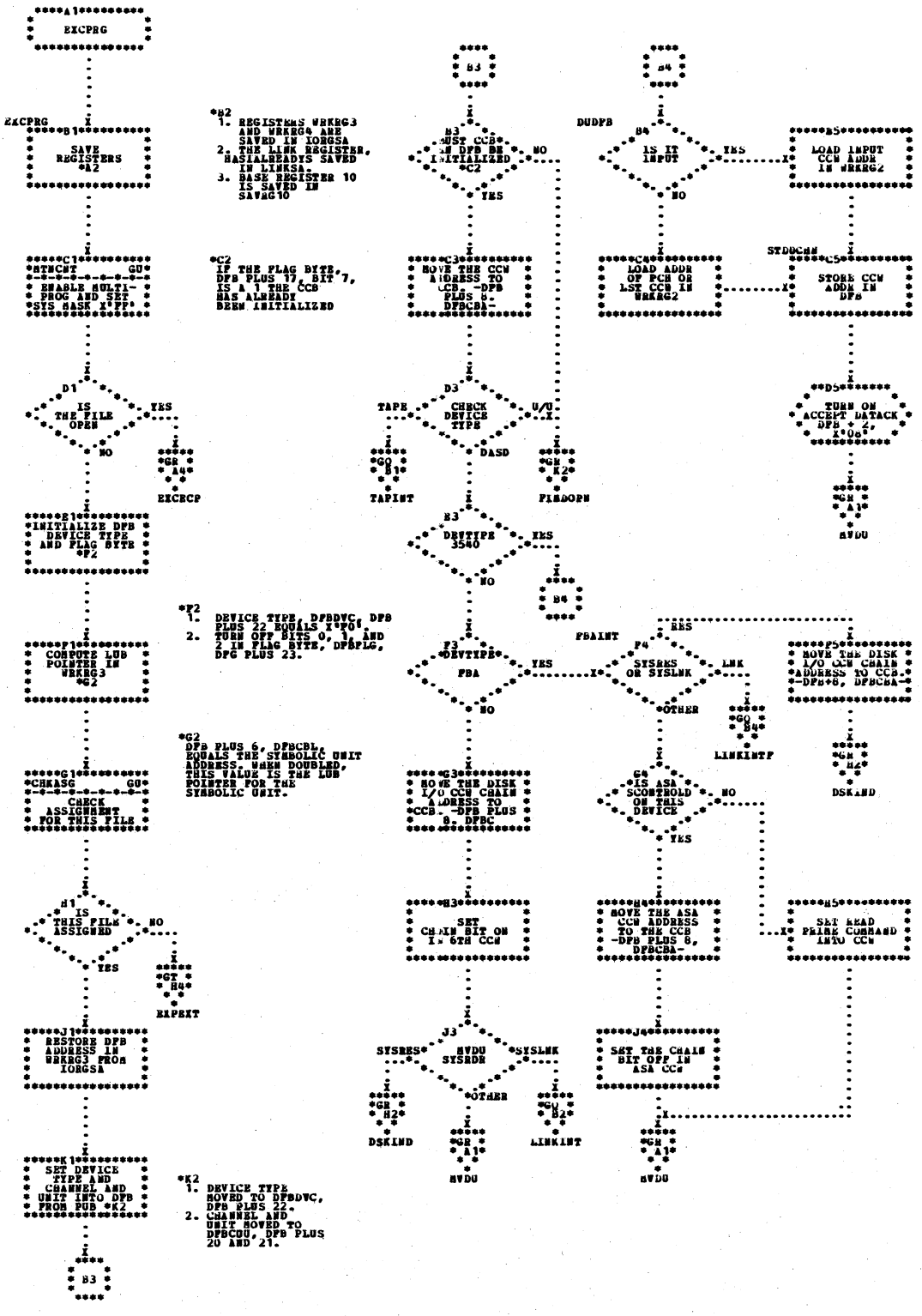
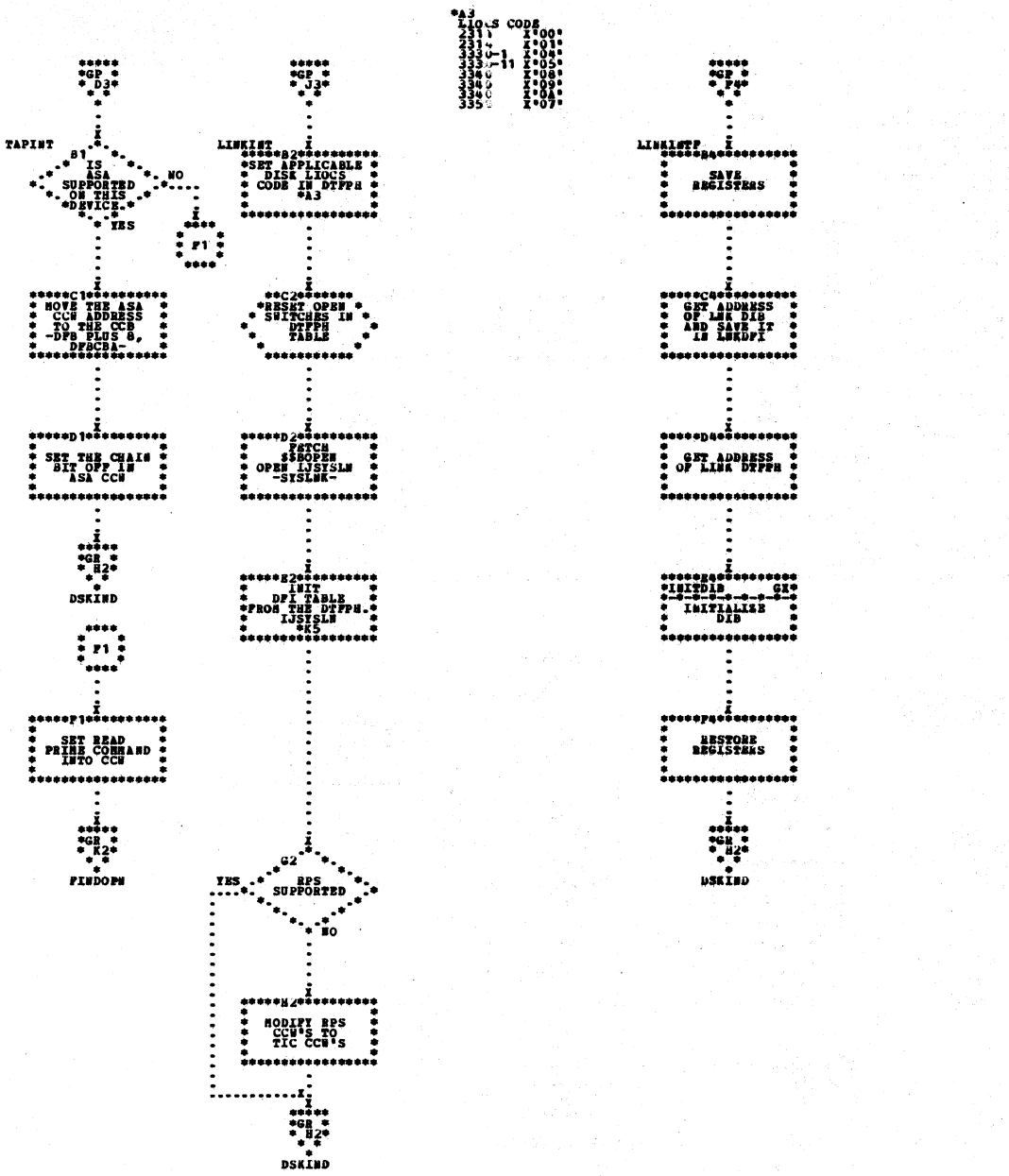


Chart GQ. $JOBCTLA - EXCP Subroutines
Refer to Charts 07 and 08.



*K5
SET CORE ADDR
SET HK SRCH ADDR
SET UPPER LIMIT
SET LOWER HEAD LIMIT
SET UPPER HEAD LIMIT

Chart GR. $JOBCTLA - EXCP Subroutines
Refer to Charts 07 and 08.

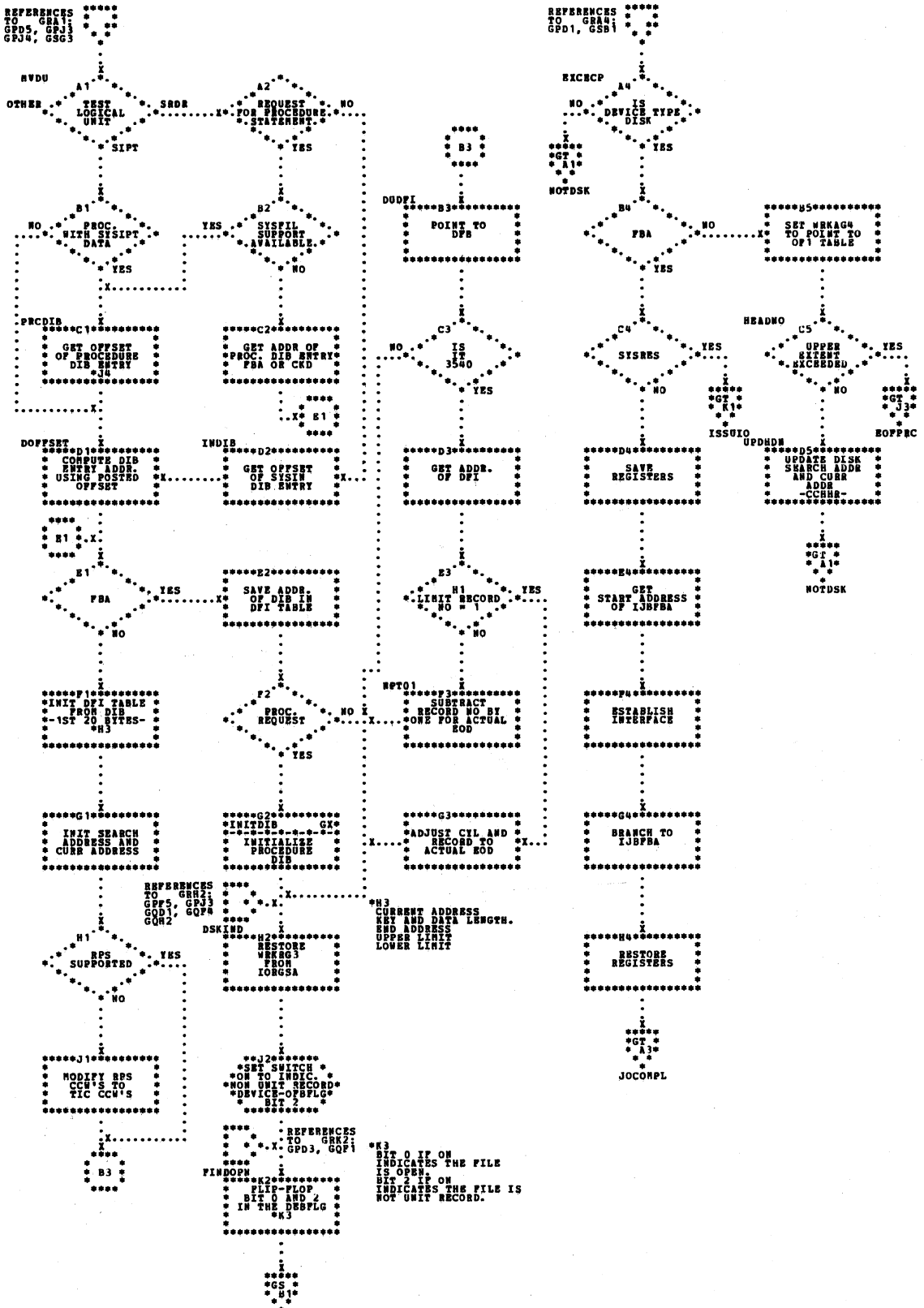


Chart GS. $JOBCTLA - EXCP Subroutines
Refer to Charts 07 and 08.

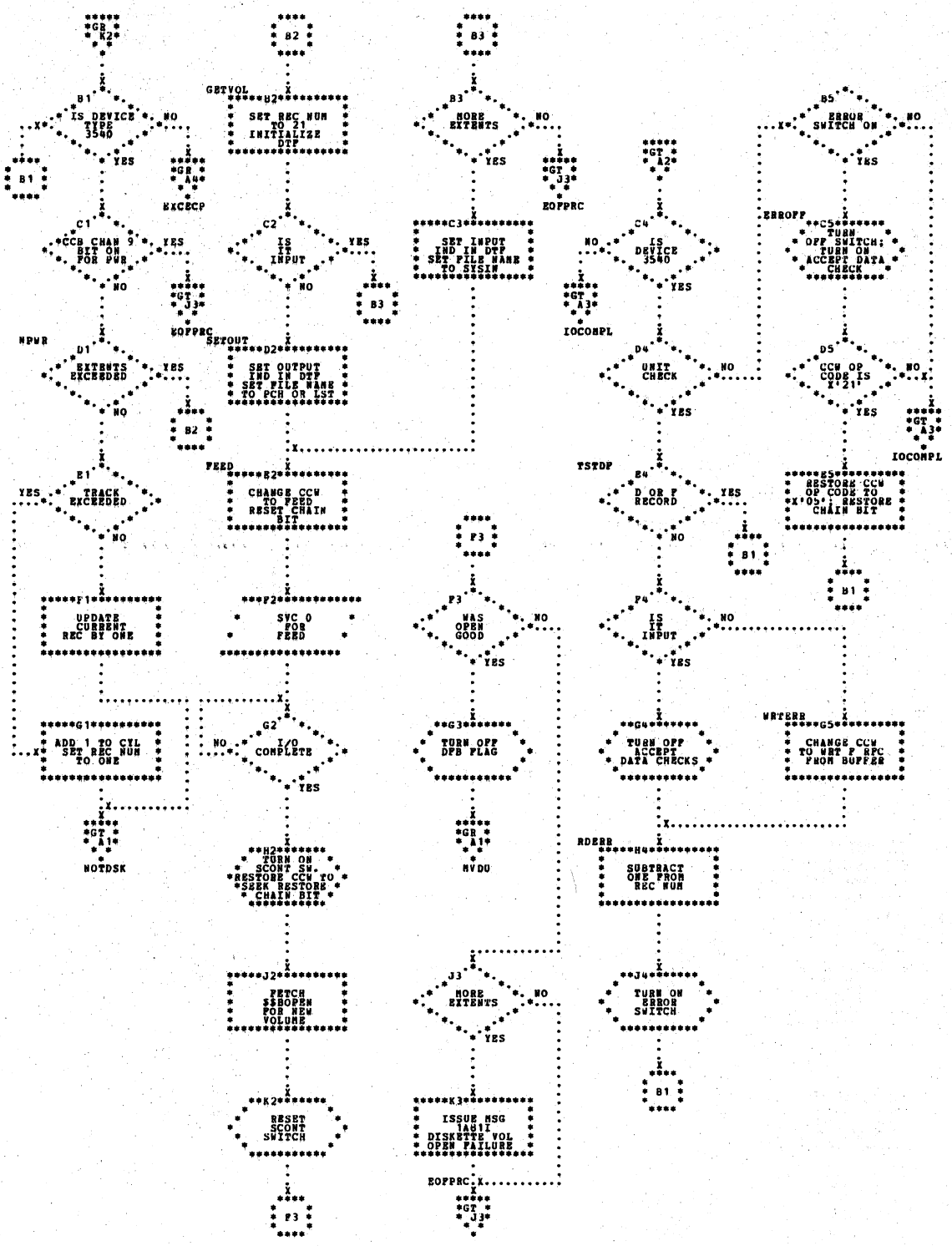


Chart GT. $JOBCTLA - EXCP Subroutines
Refer to Charts 07 and 08

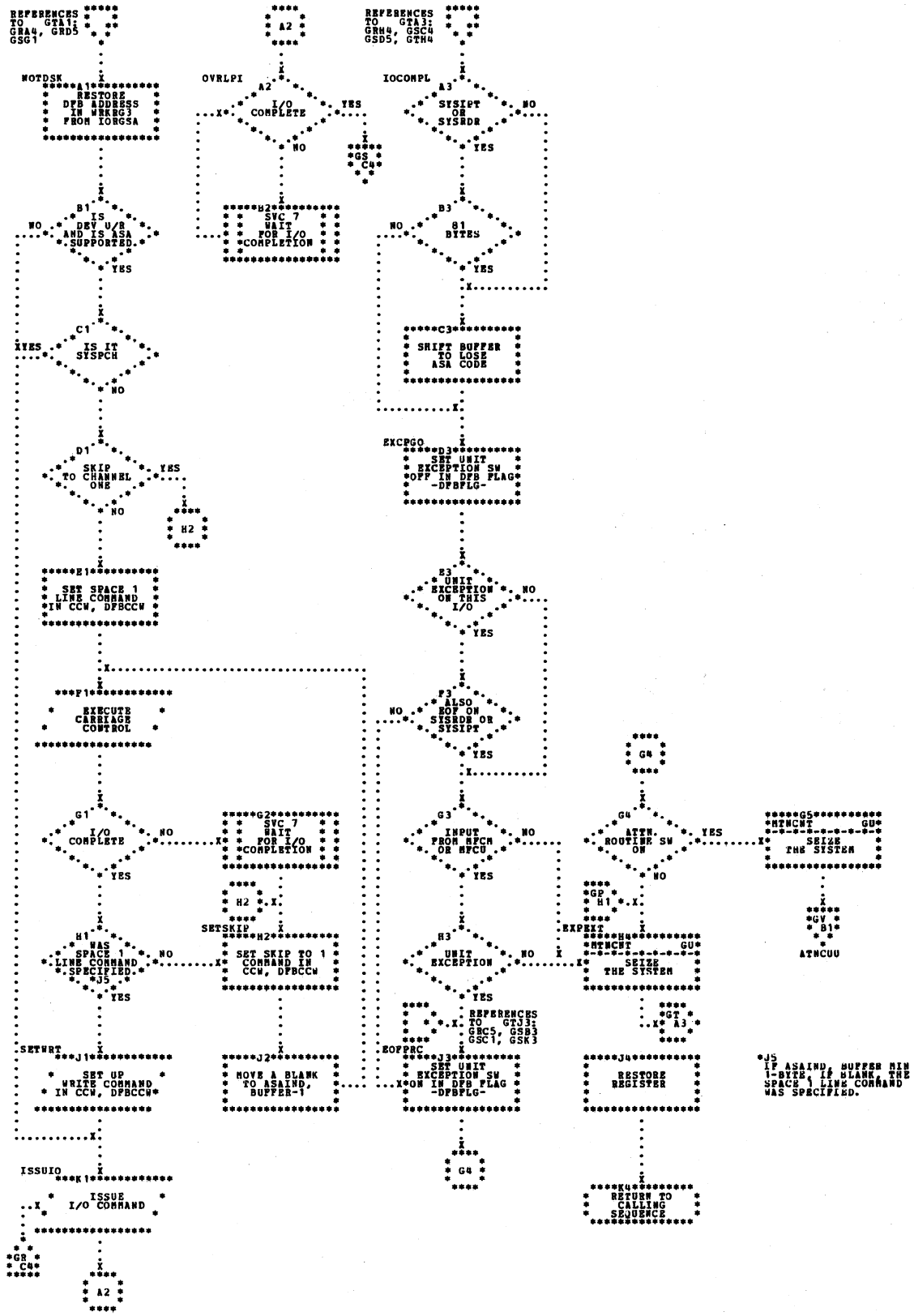


Chart GU. $JOBCTLA - Subroutines
Refer to Charts 07 and 08.

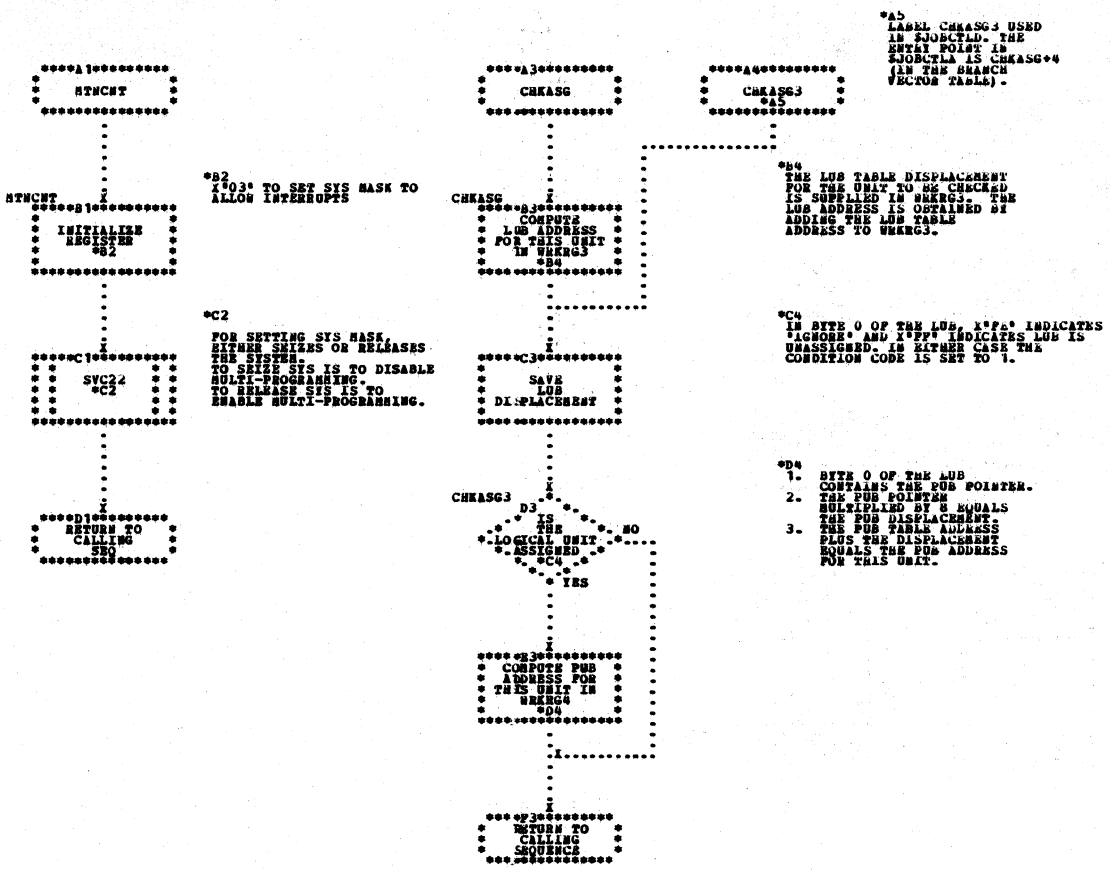


Chart GV. $JOBCTLA - Error Subroutines
Refer to Charts 07 and 08.

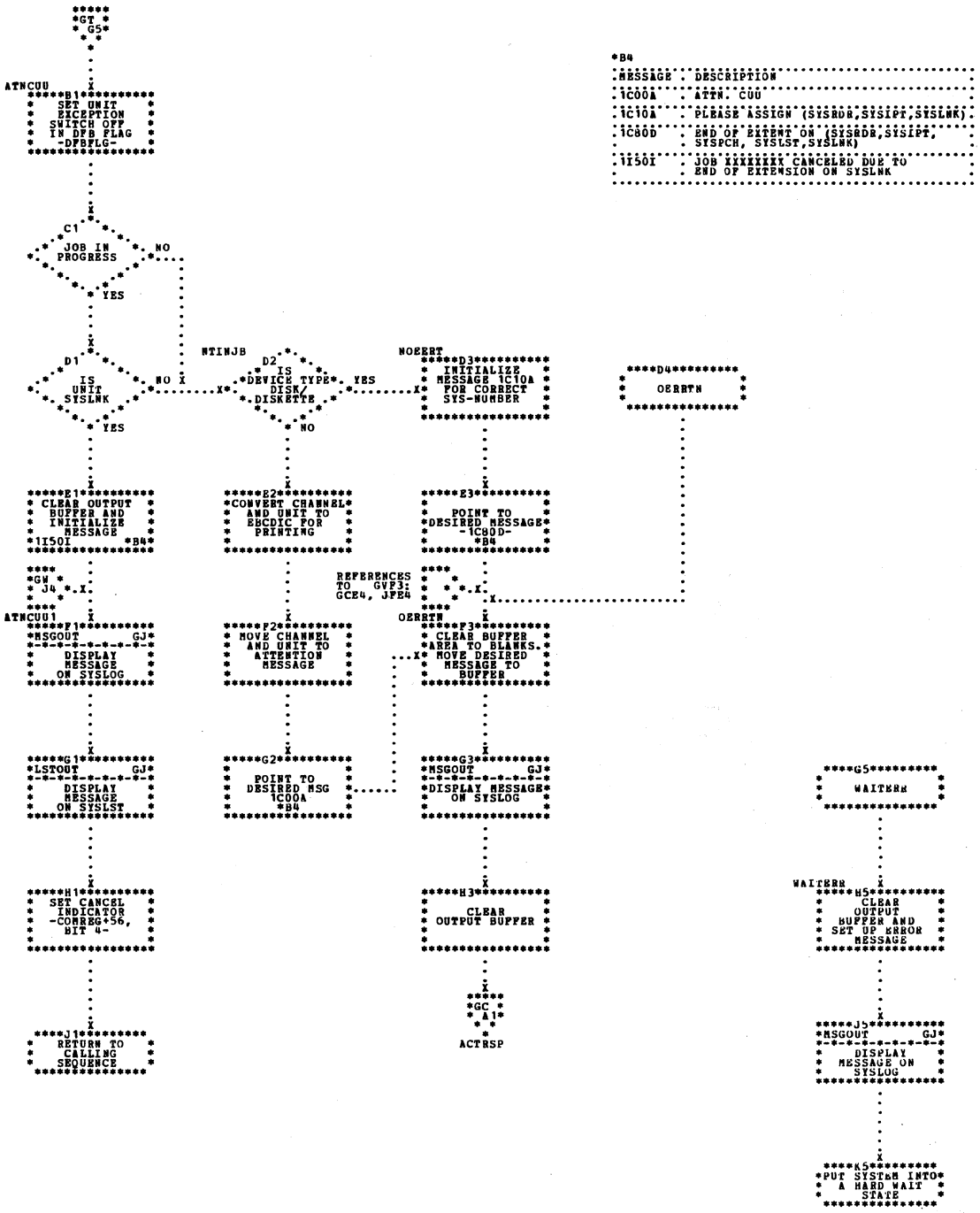


Chart GW. $JOBCTLA - Error Subroutines
Refer to Charts 07 and 08.

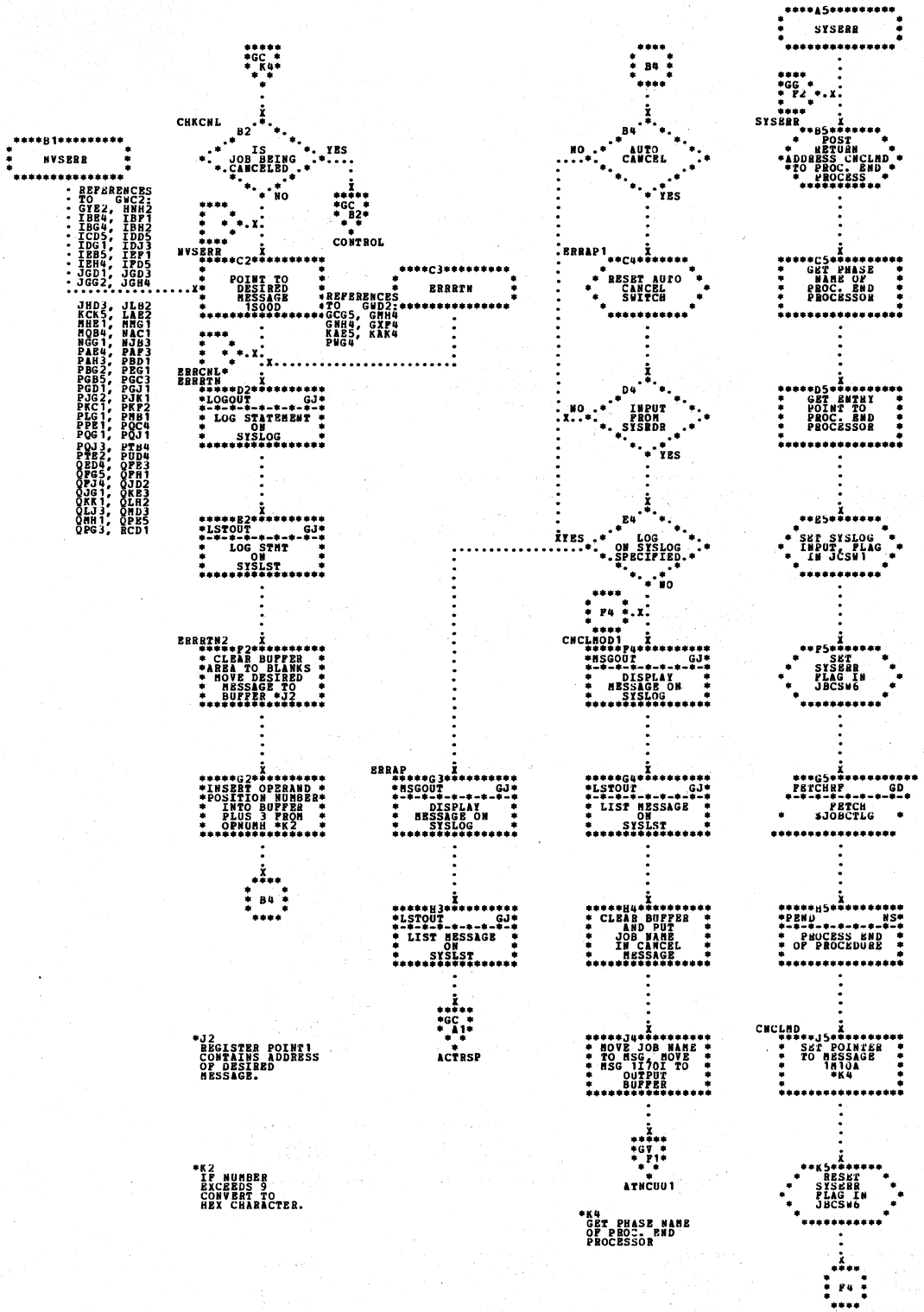


Chart GX. $JOBCTLA - Initialize Fixed Block DIB
Refer to Charts 07 and 08.

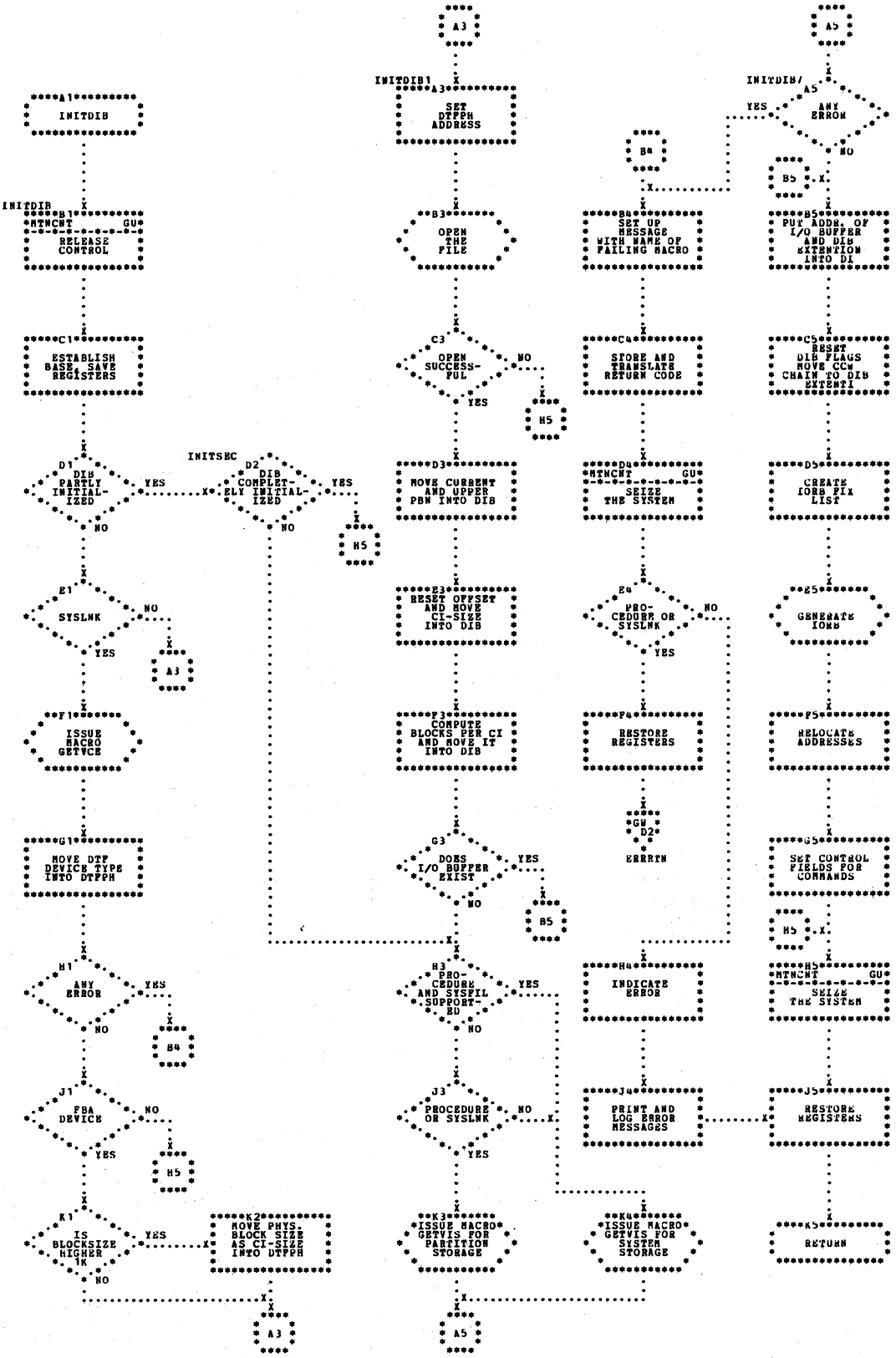


Chart GY. $JOBCTLA - Set Buffer Values for 3800 Printer
Refer to Charts 07 and 08.

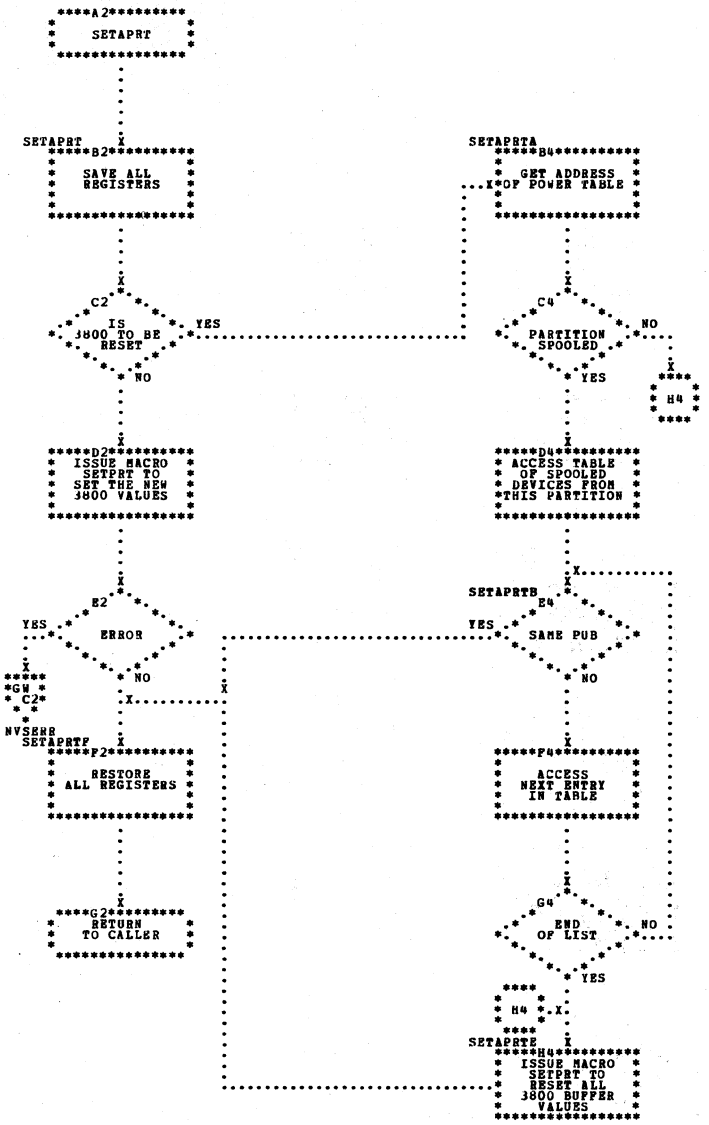


Chart HA. $JOBCTLB - Restart Processor (Part 1 of 3)
Refer to Chart 09.

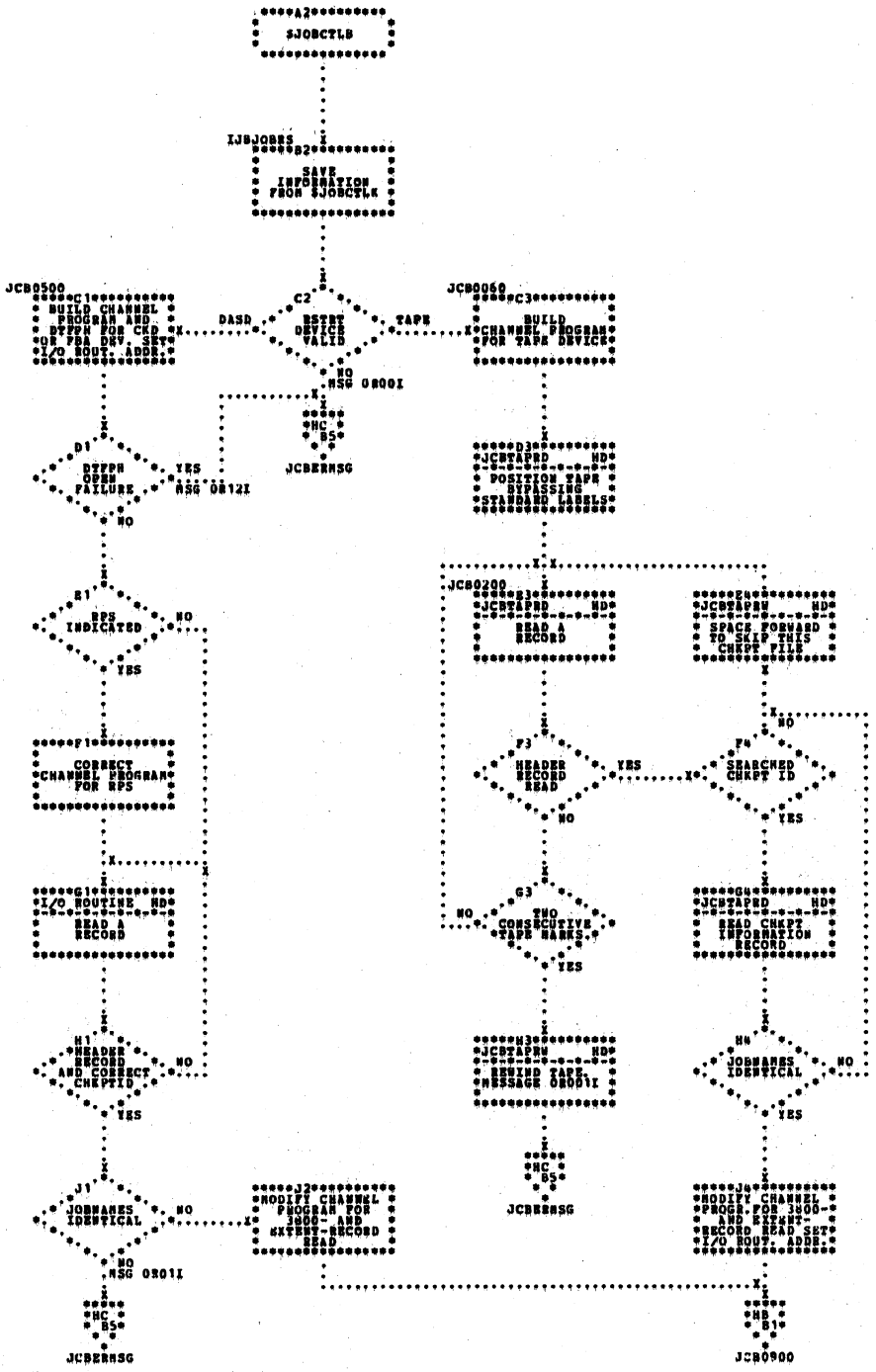


Chart HB. $JOBCTLB - Restart Processor (Part 2 of 3)
Refer to Chart 09.

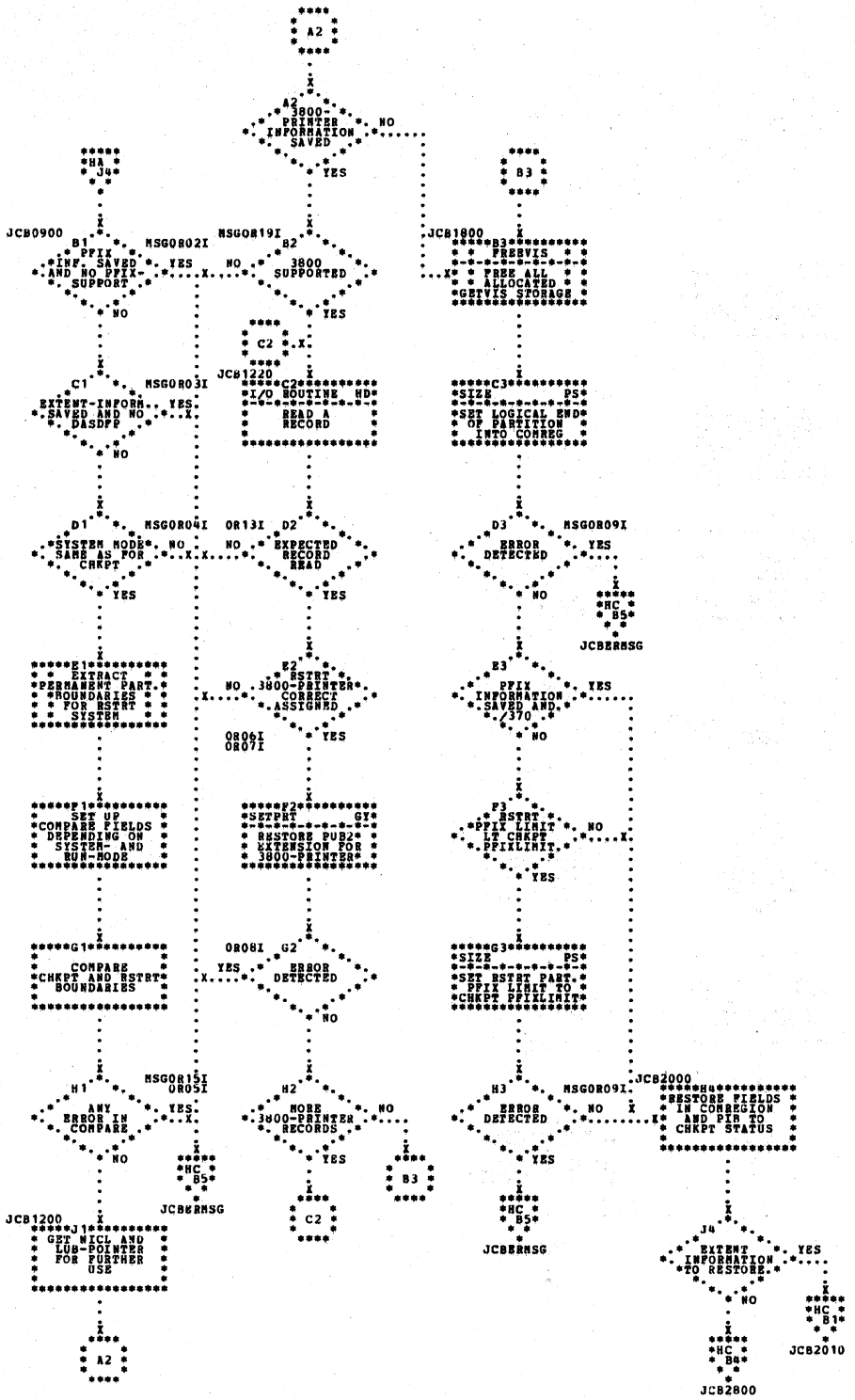


Chart HC. $JOBCTLB - Restart Processor (Part 3 of 3)
Refer to Chart 09.

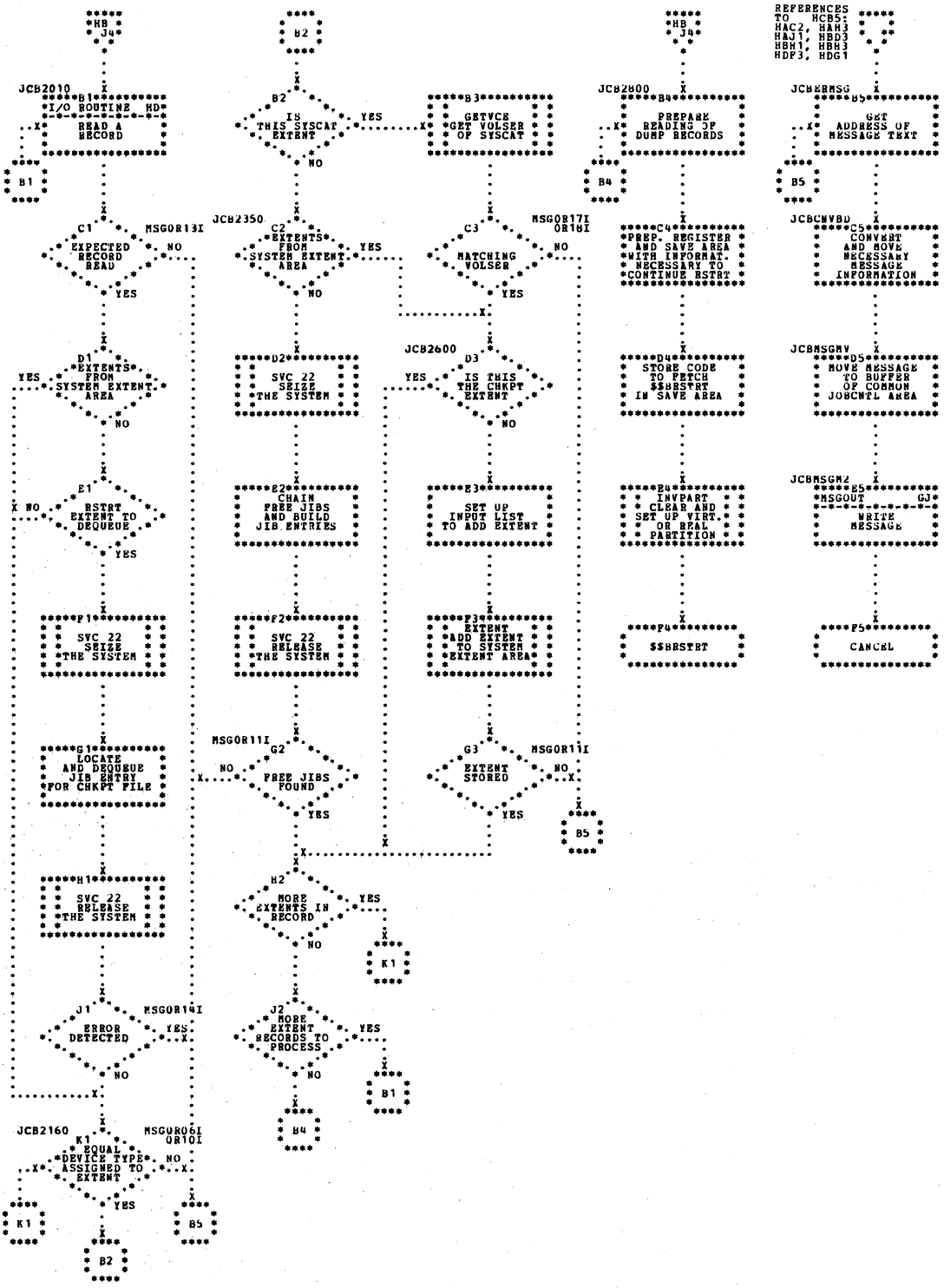


Chart HD. $JOBCTLB - Restart Subroutines
Refer to Chart 09.

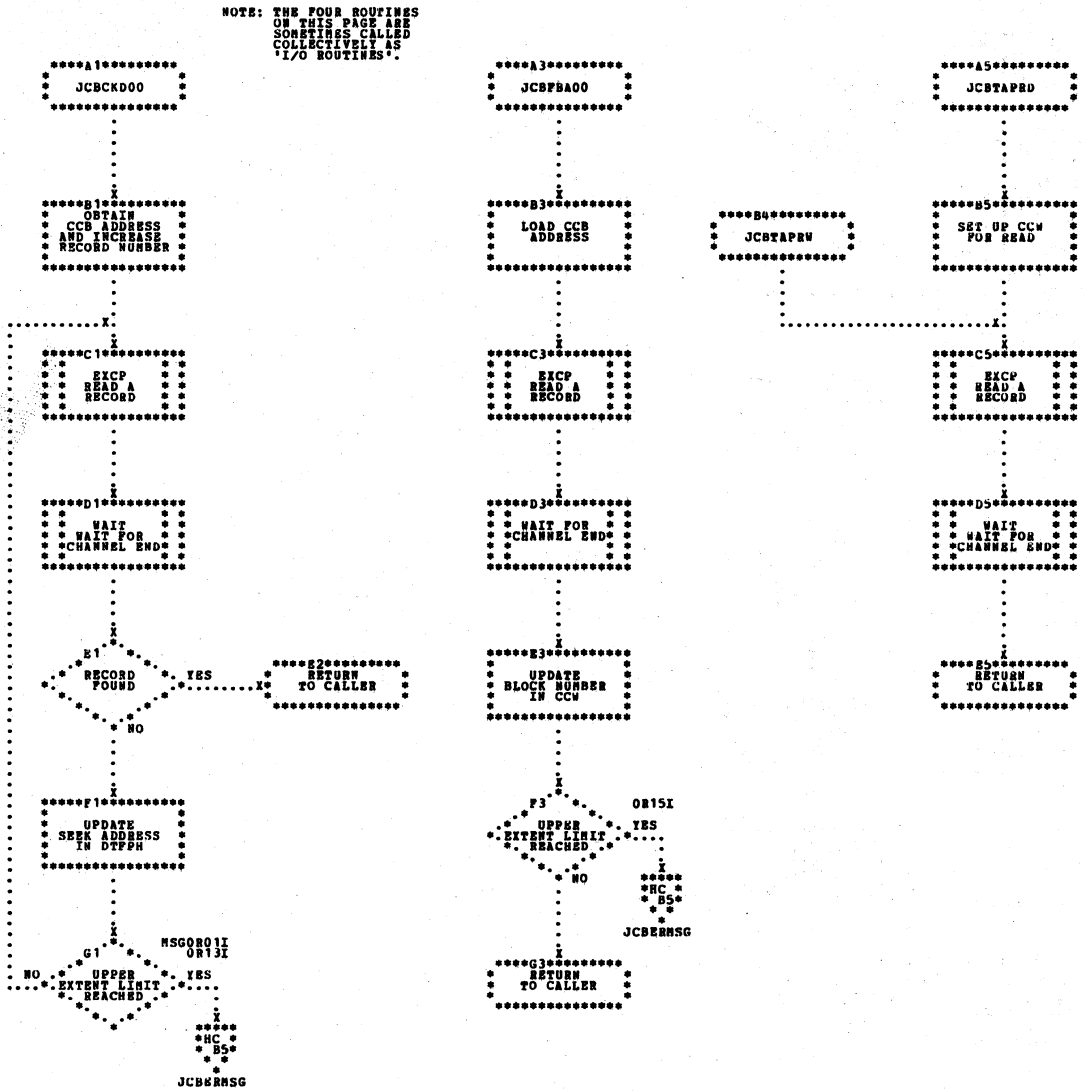


Chart HM. $JOBCTLC - Open Hard Copy File
Refer to Chart 09.1.

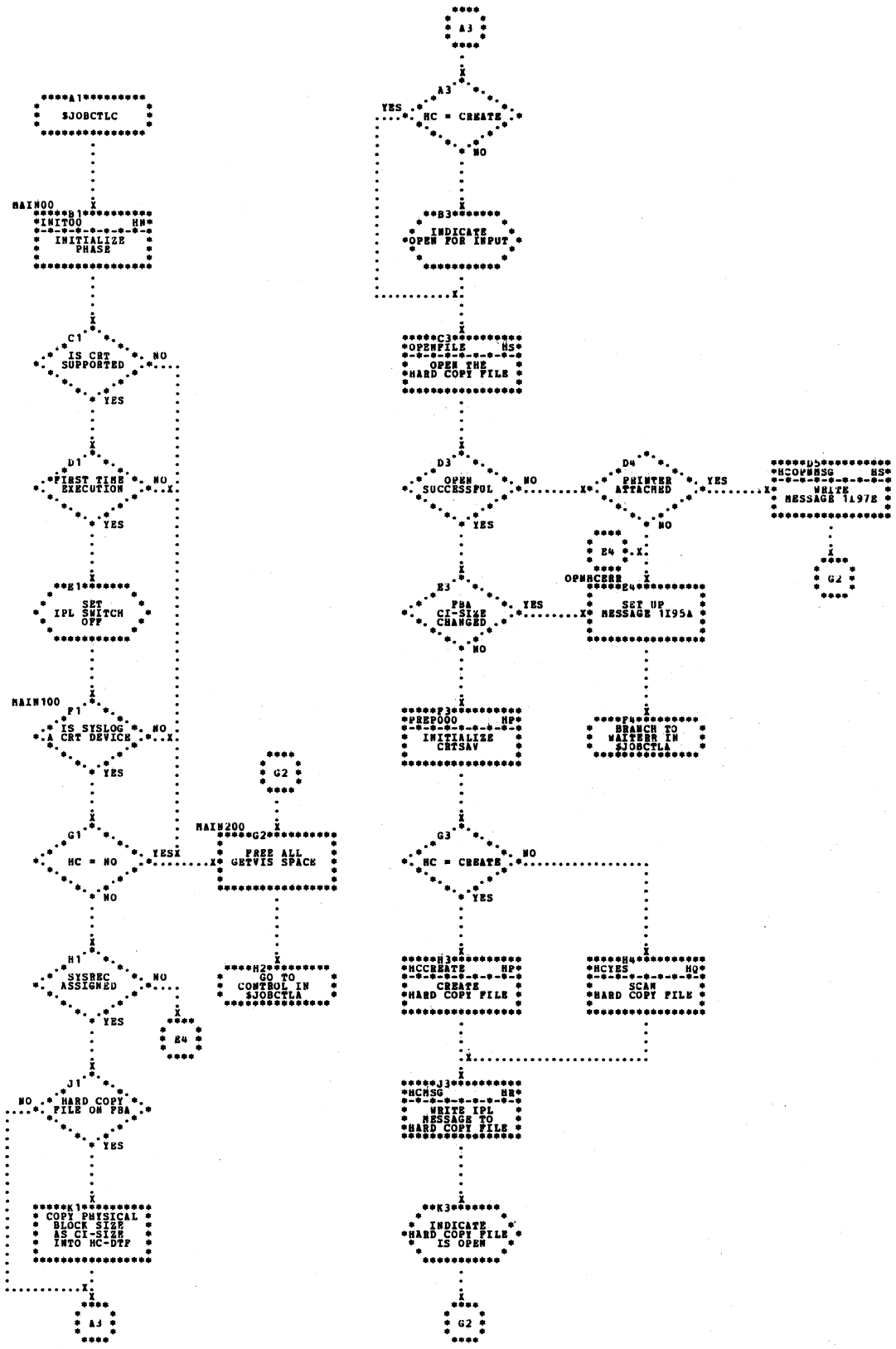


Chart HN. $JOBCTLC - ROD Command Processing and Initialization Routine
Refer to Chart 09.1.

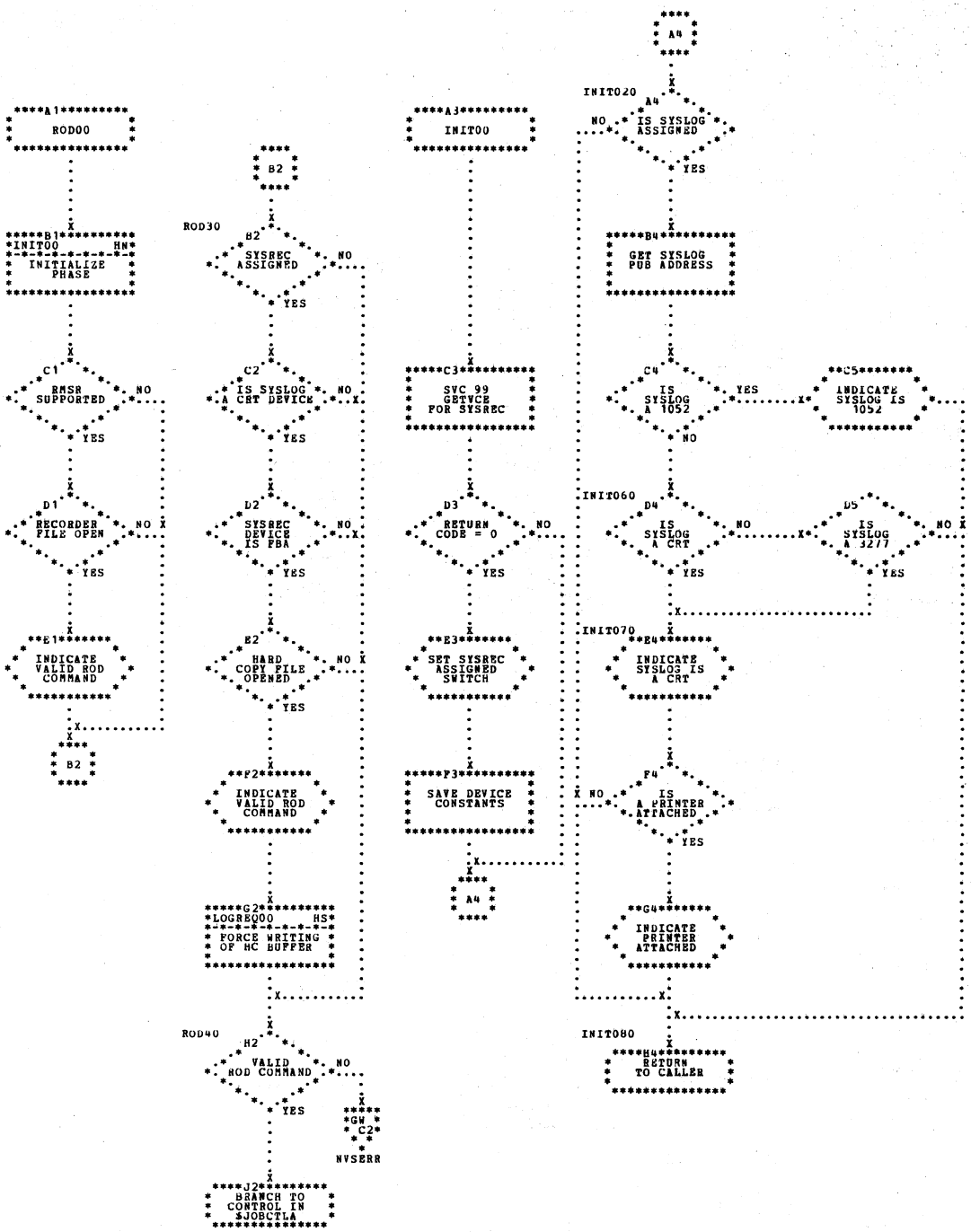


Chart HP. $JOBCTLC - Open Hard Copy File
Refer to Chart 09.1.

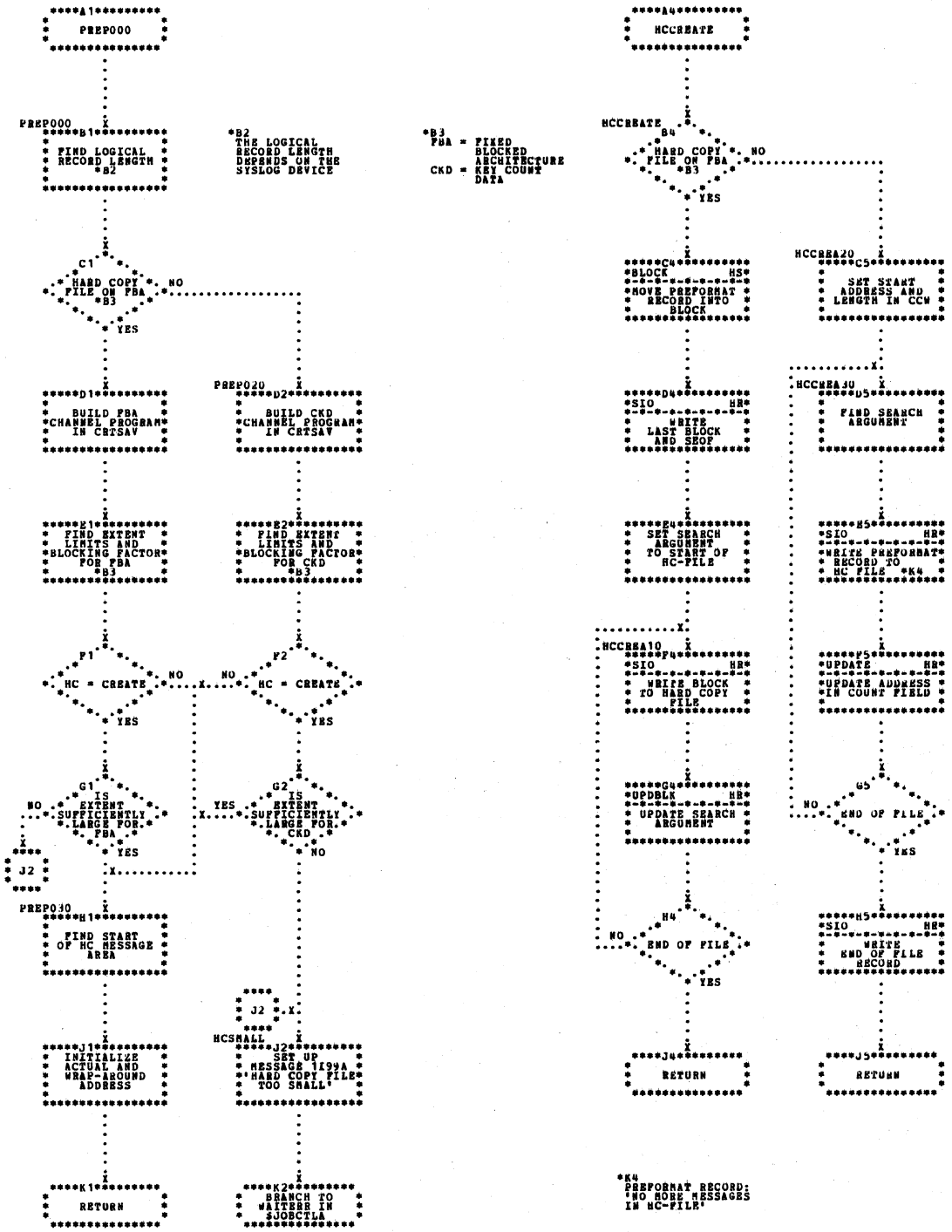


Chart HQ. $JOBCTLC - Open Hard Copy File
Refer to Chart 09.1.

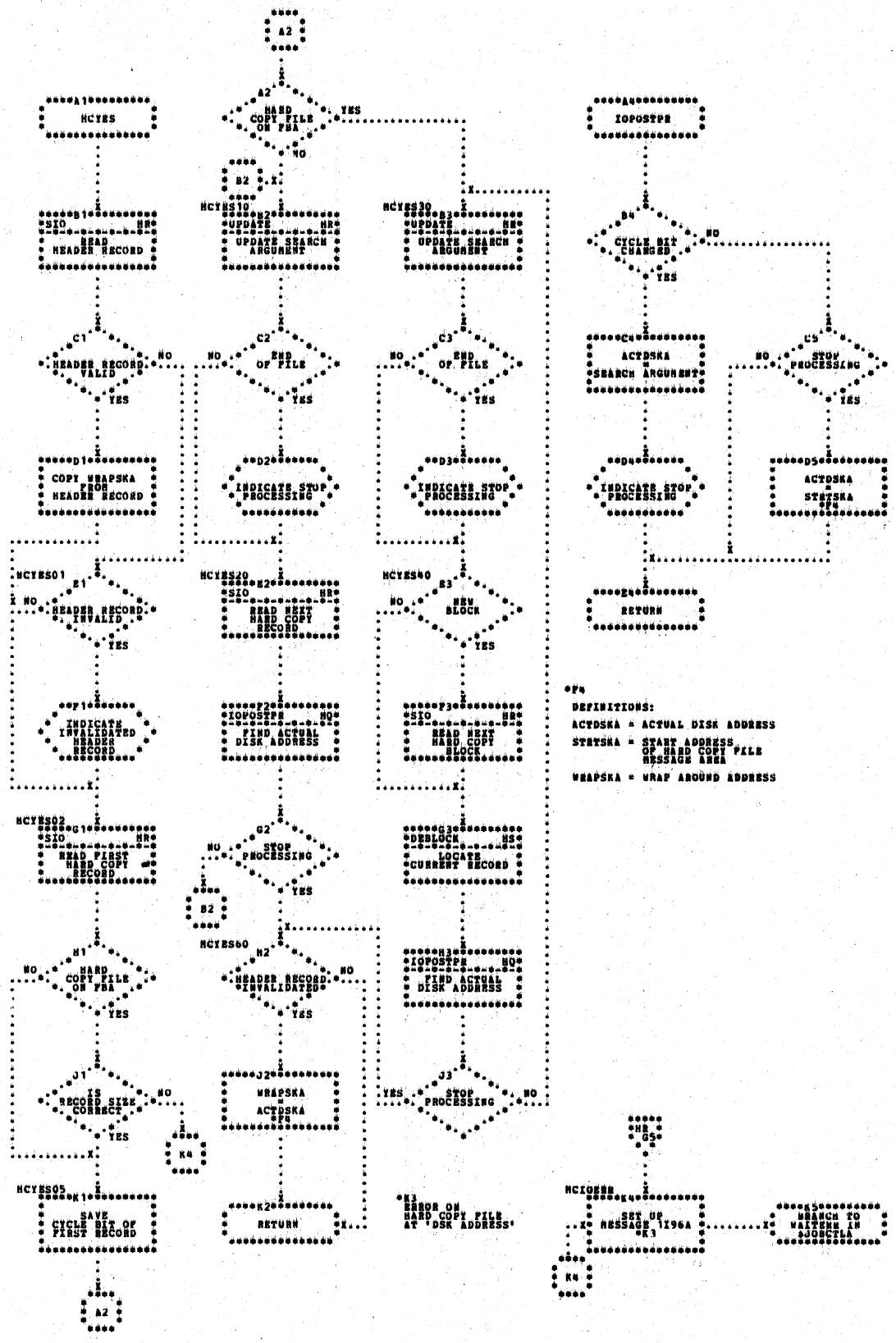


Chart HR. $JOBCTLC - Open Hard Copy File
Refer to Chart 09.1.

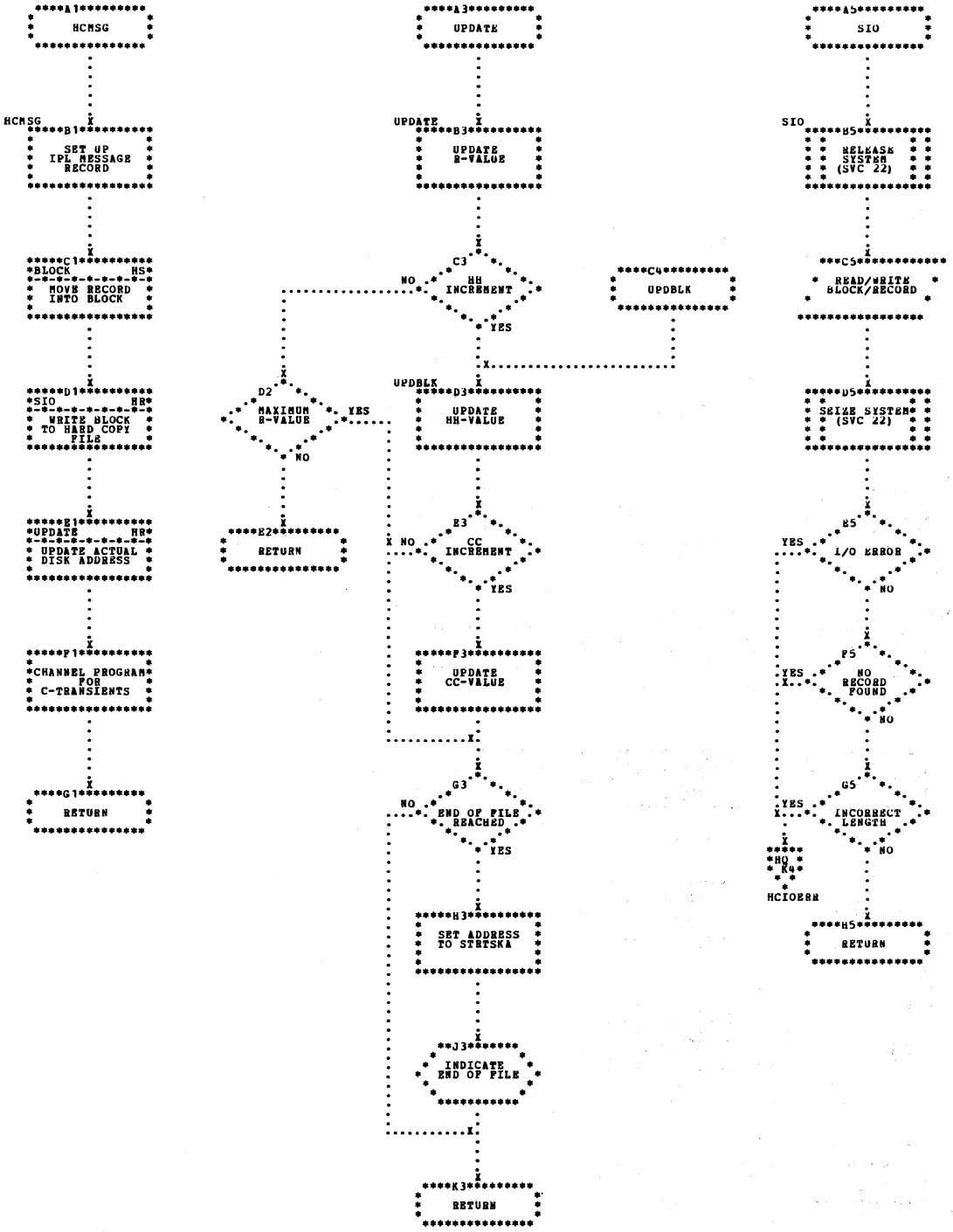


Chart HS. $JOBCTLC - Open Hard Copy File
Refer to Chart 09.1.

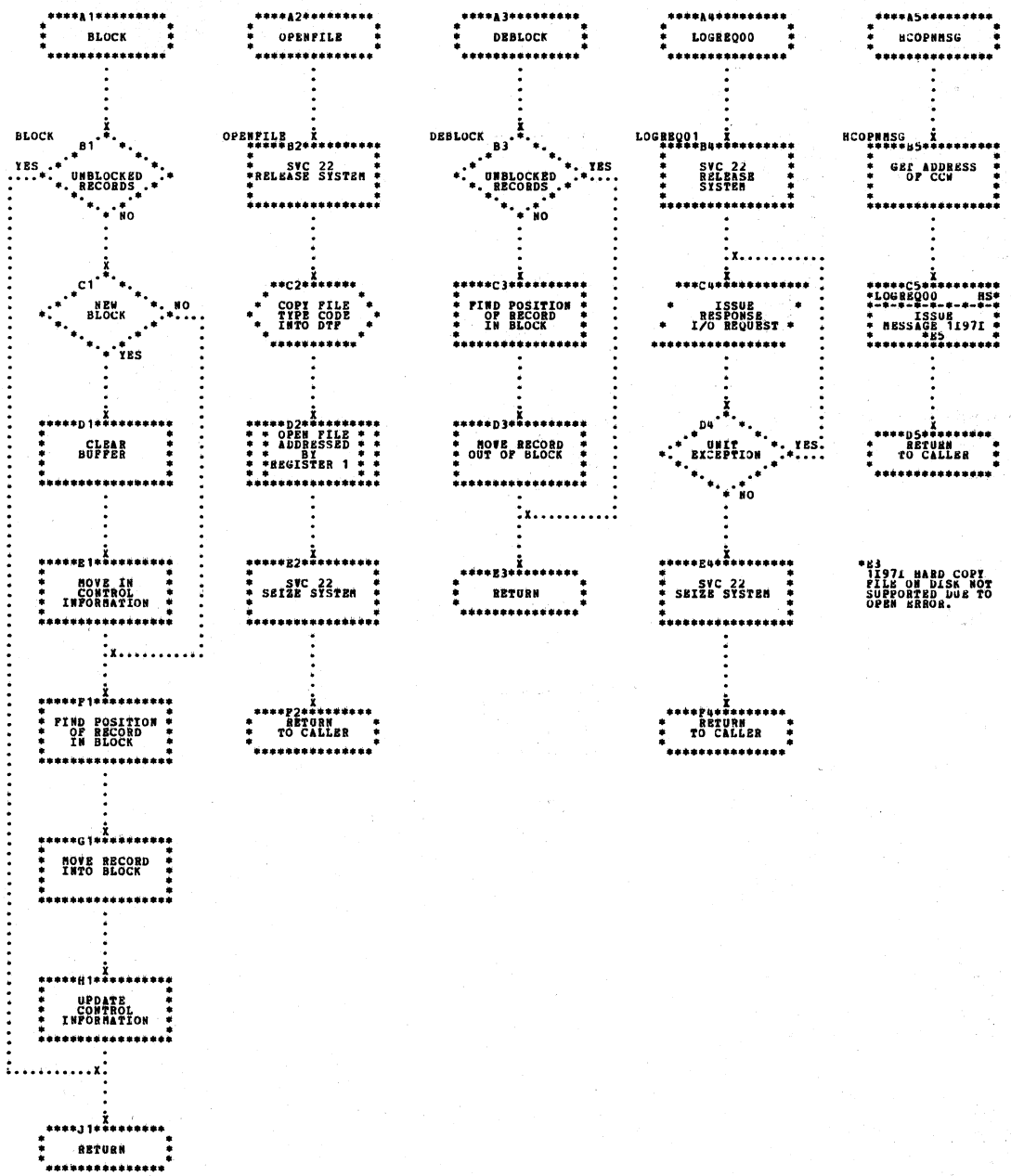


Chart IA. $JOBCTLD - Main CSECT
Refer to Charts 10 to 13.

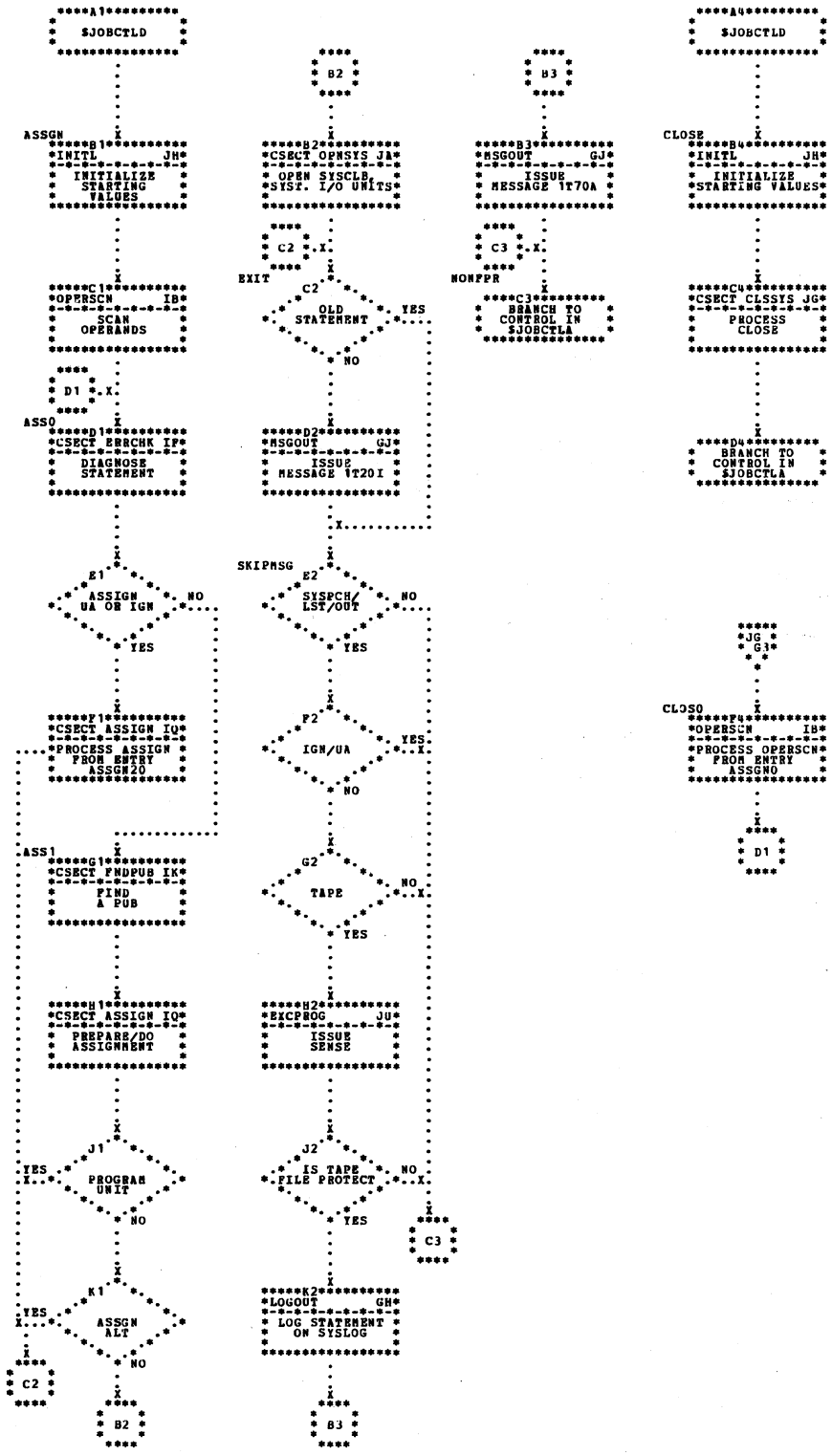


Chart IB. $JOBCTLD - Scan Operands of ASSGN Statement (Part 1 of 4)
Refer to Charts 10 to 13.

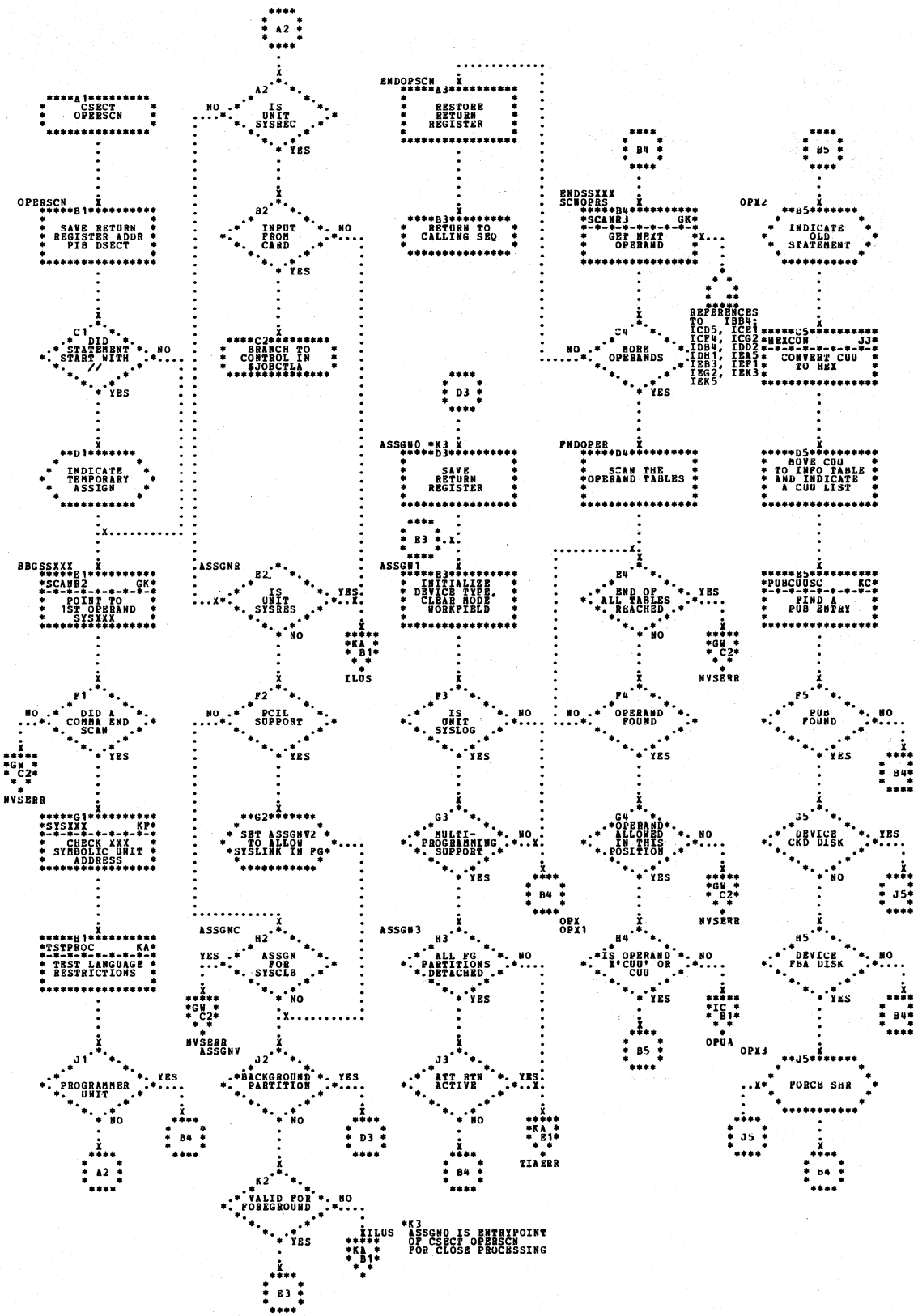


Chart IC. $JOBCTLD - Scan Operands of ASSGN Statement (Part 2 of 4)
Refer to Charts 10 to 13.

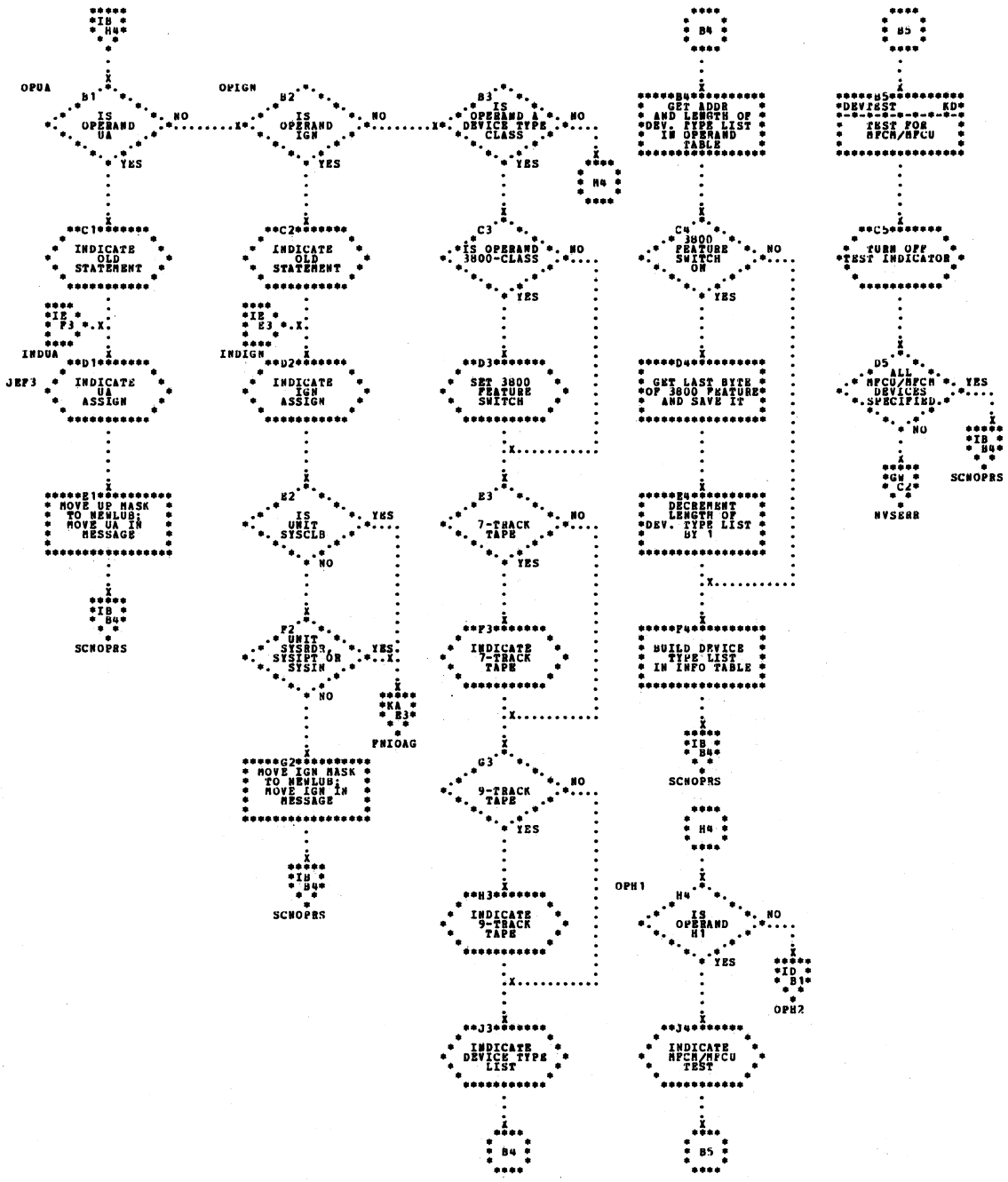


Chart ID. $JOBCTLD - Scan Operands of ASSGN Statement (Part 3 of 4)
Refer to Charts 10 to 13.

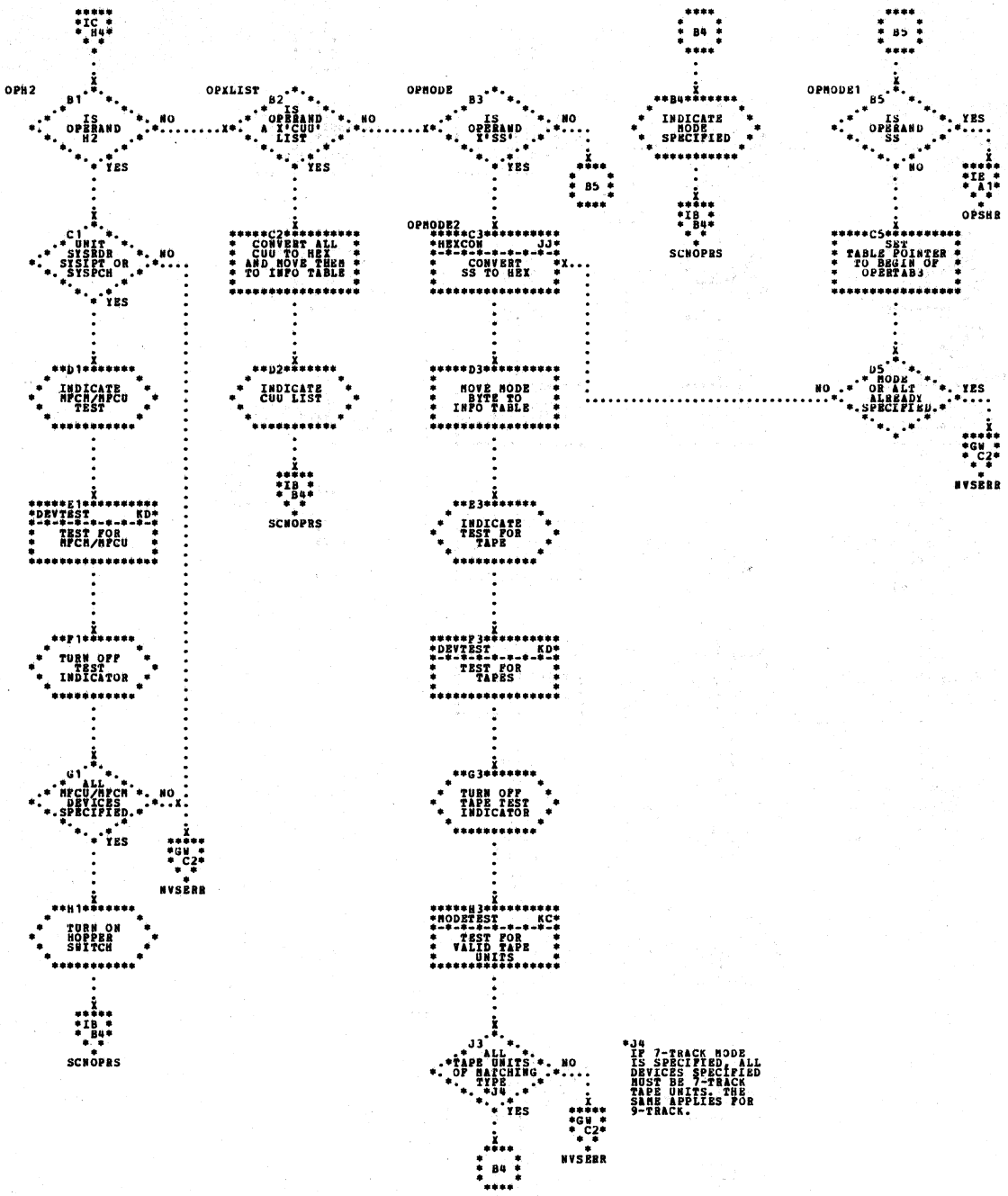


Chart IE. $JOBCTLD - Scan Operands of ASSGN Statement (Part 4 of 4)
Refer to Charts 10 to 13.

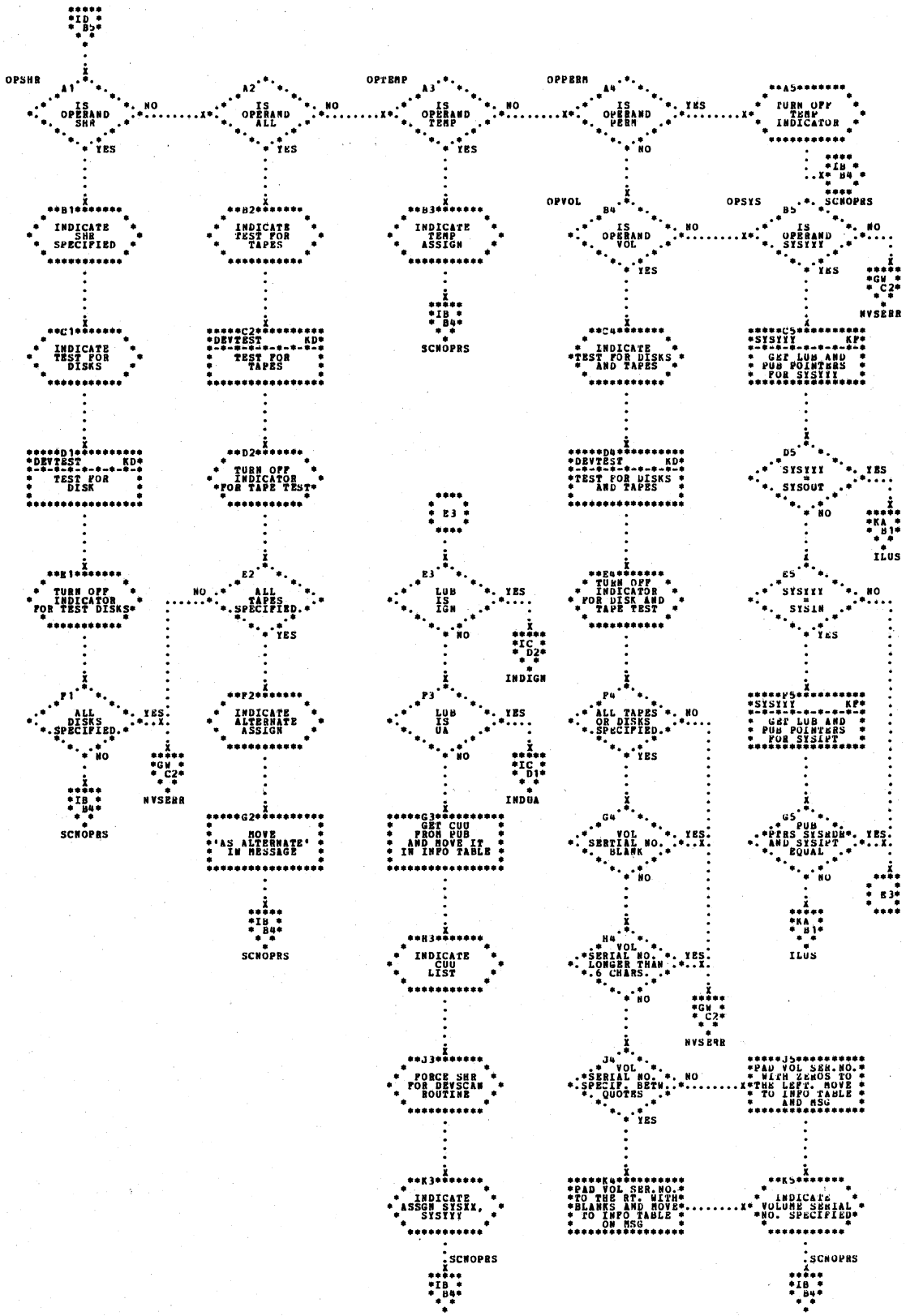


Chart IF. $JOBCTLD - Error Checking
Refer to Charts 10 to 13.

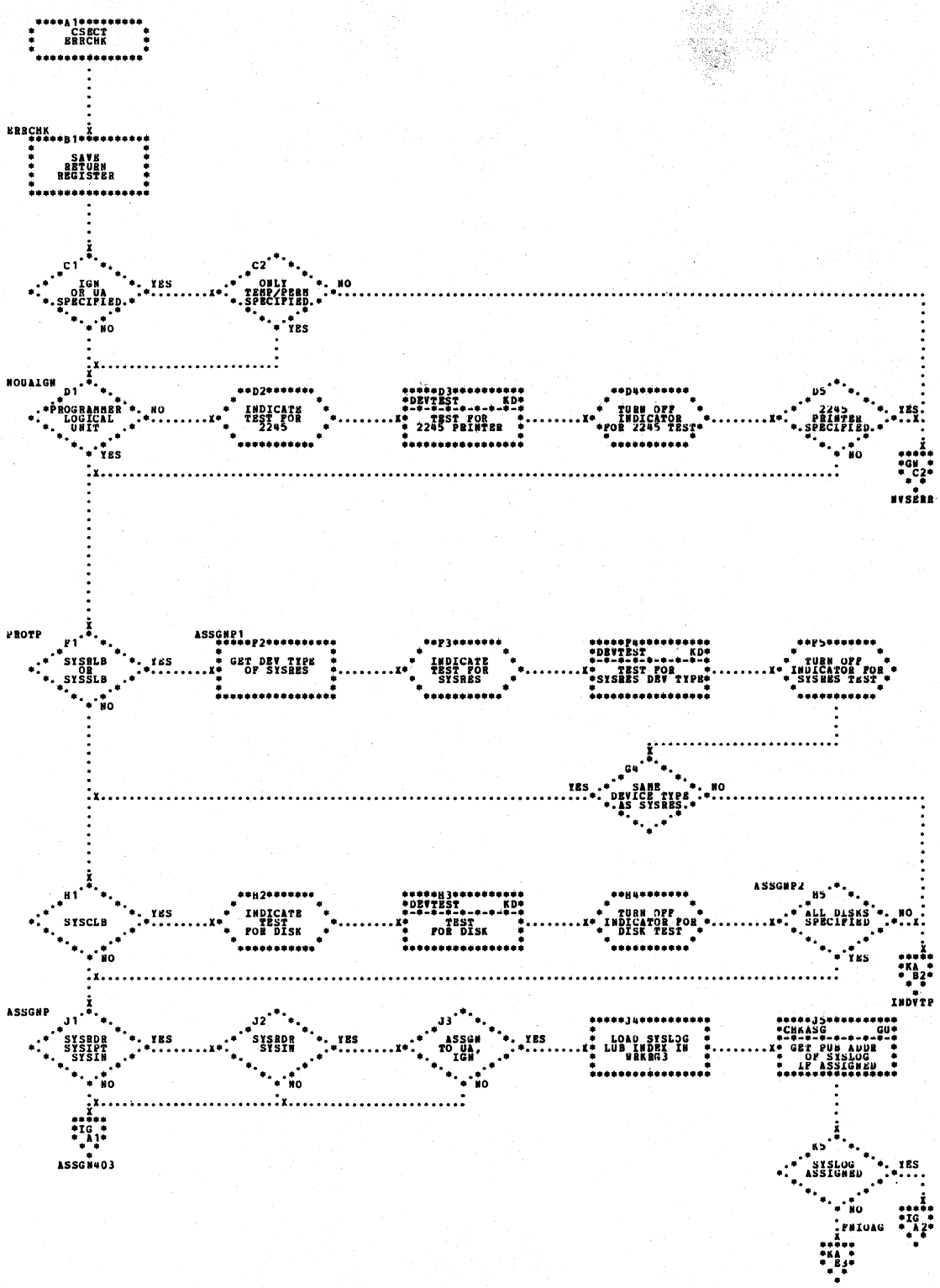


Chart IG. $JOBCTLD - Error Checking
Refer to Charts 10 to 13.

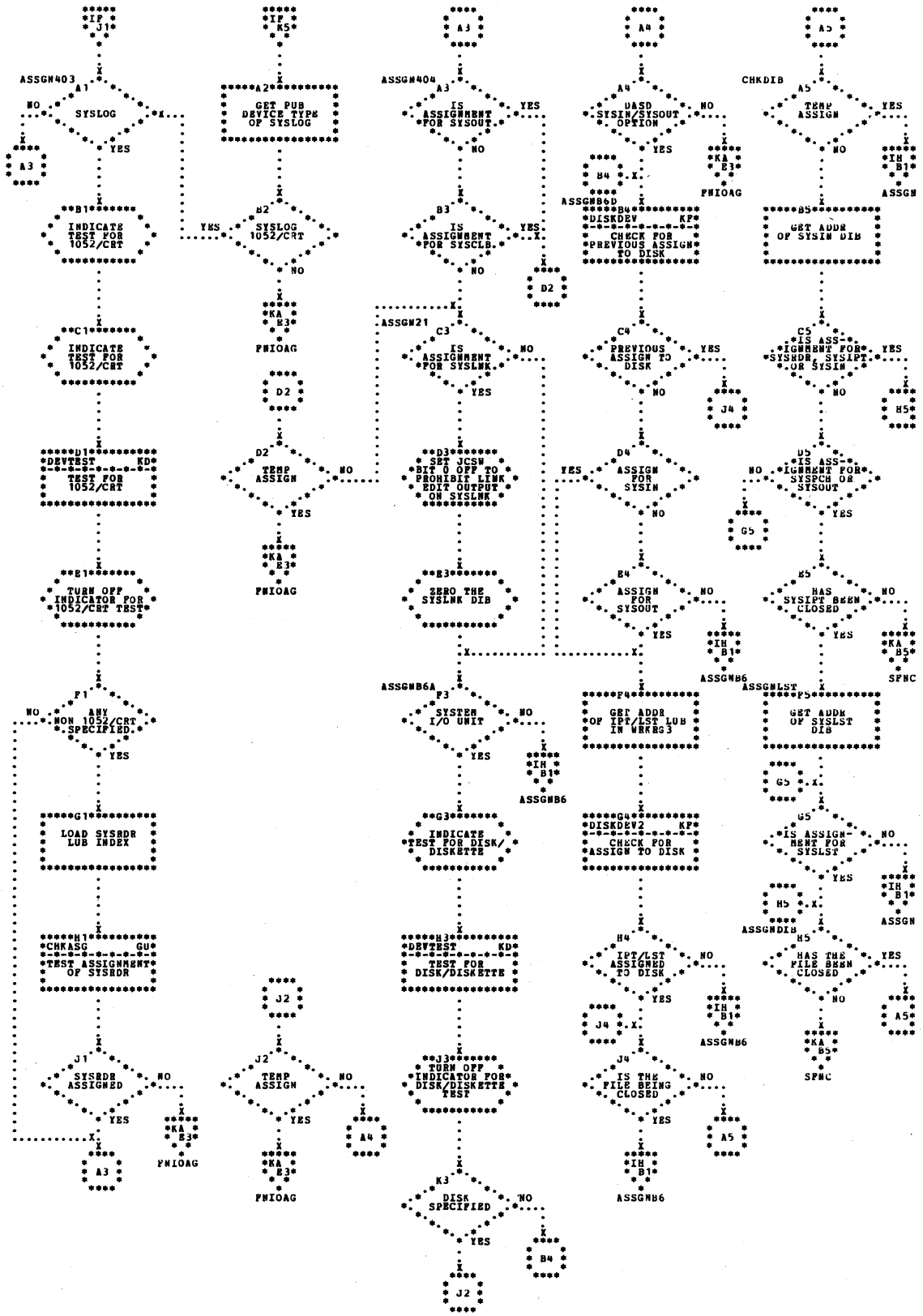


Chart IH. $JOBCTLD - Error Checking
Refer to Charts 10 to 13.

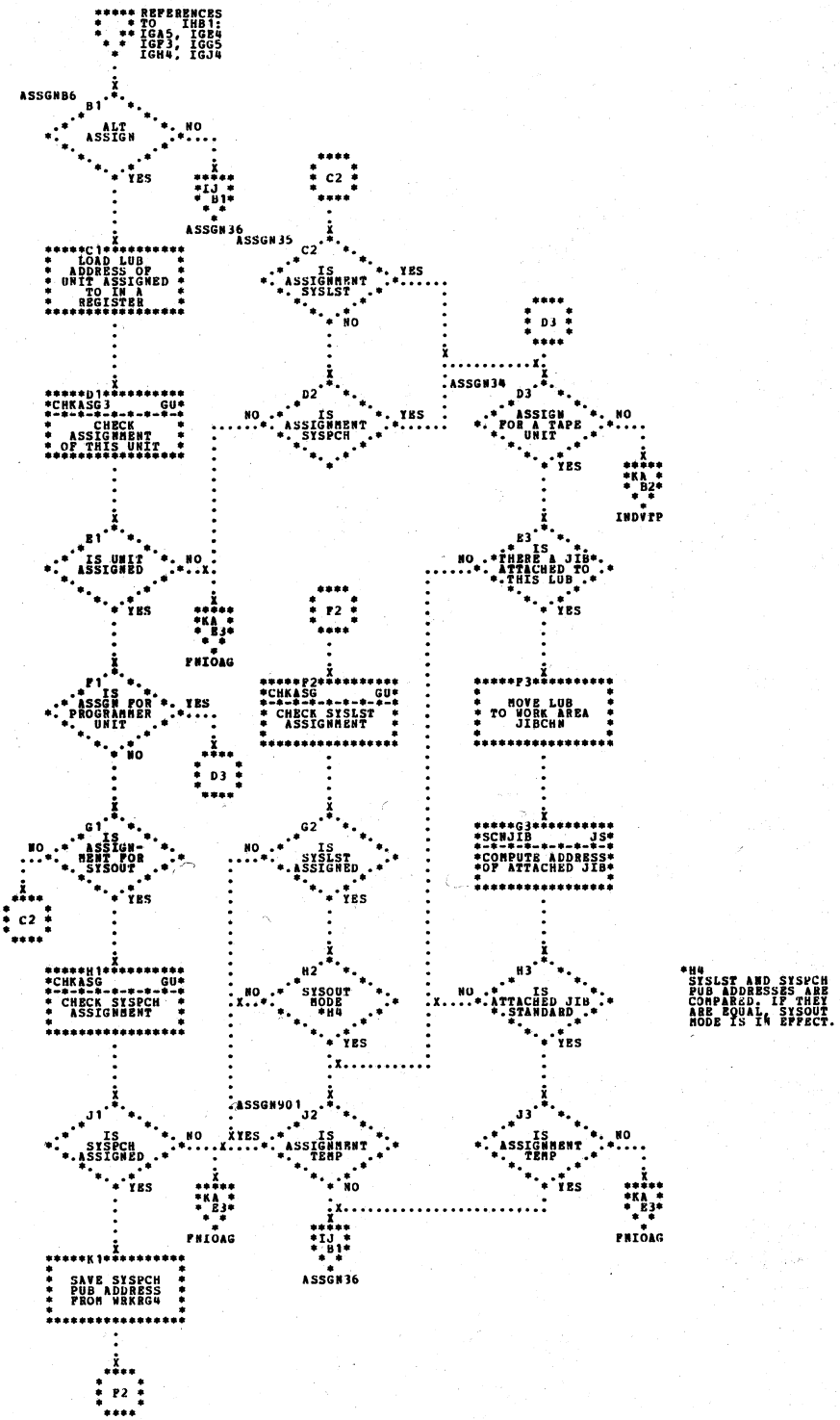


Chart IJ. $JOBCTLD - Error Checking
Refer to Charts 10 to 13.

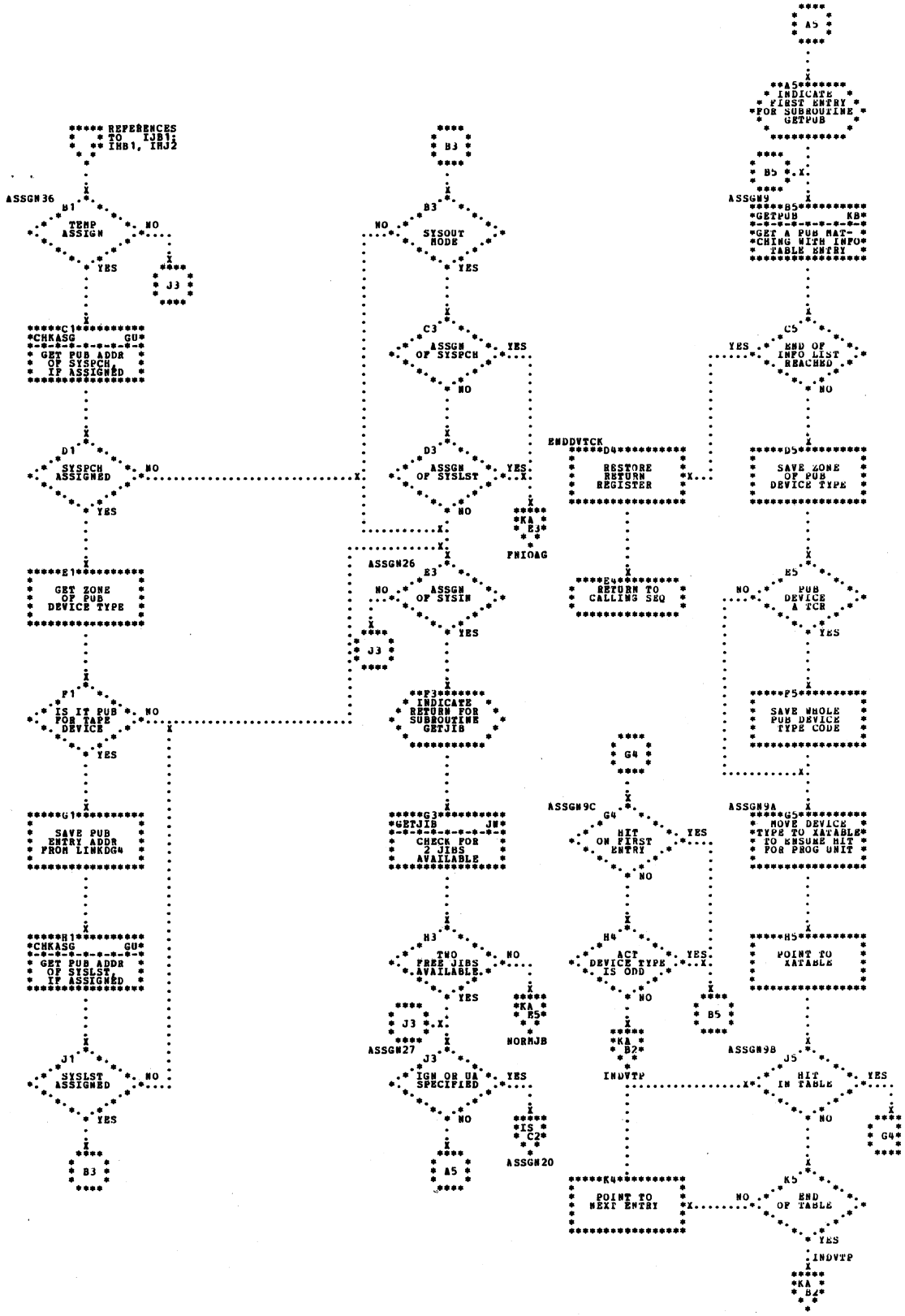


Chart IK. $JOBCTLD - Find a PUB to Be Assigned
Refer to Charts 10 to 13.

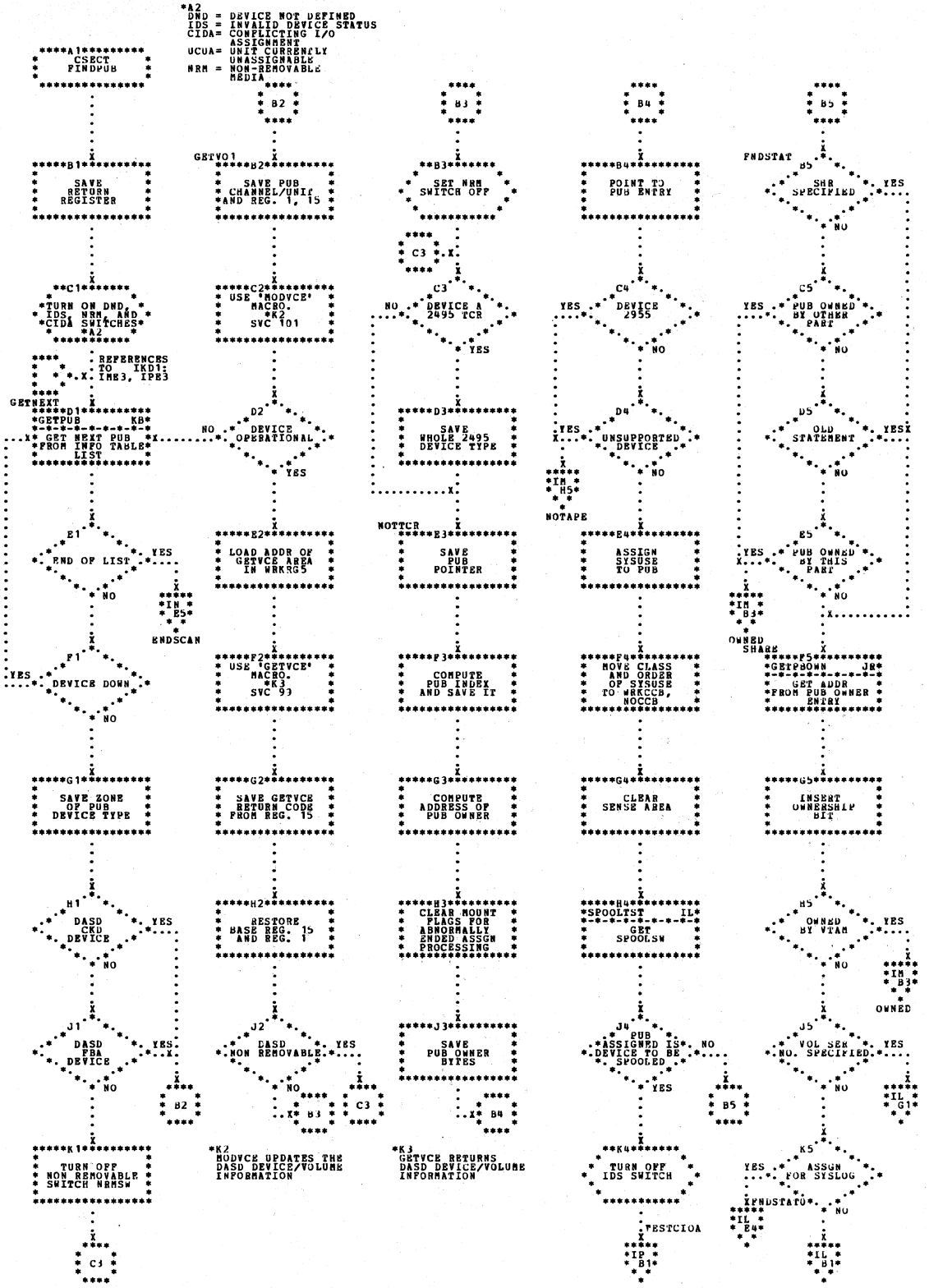


Chart IL. $JOBCTLD - Find a PUB to be Assigned
Refer to Charts 10 to 13.

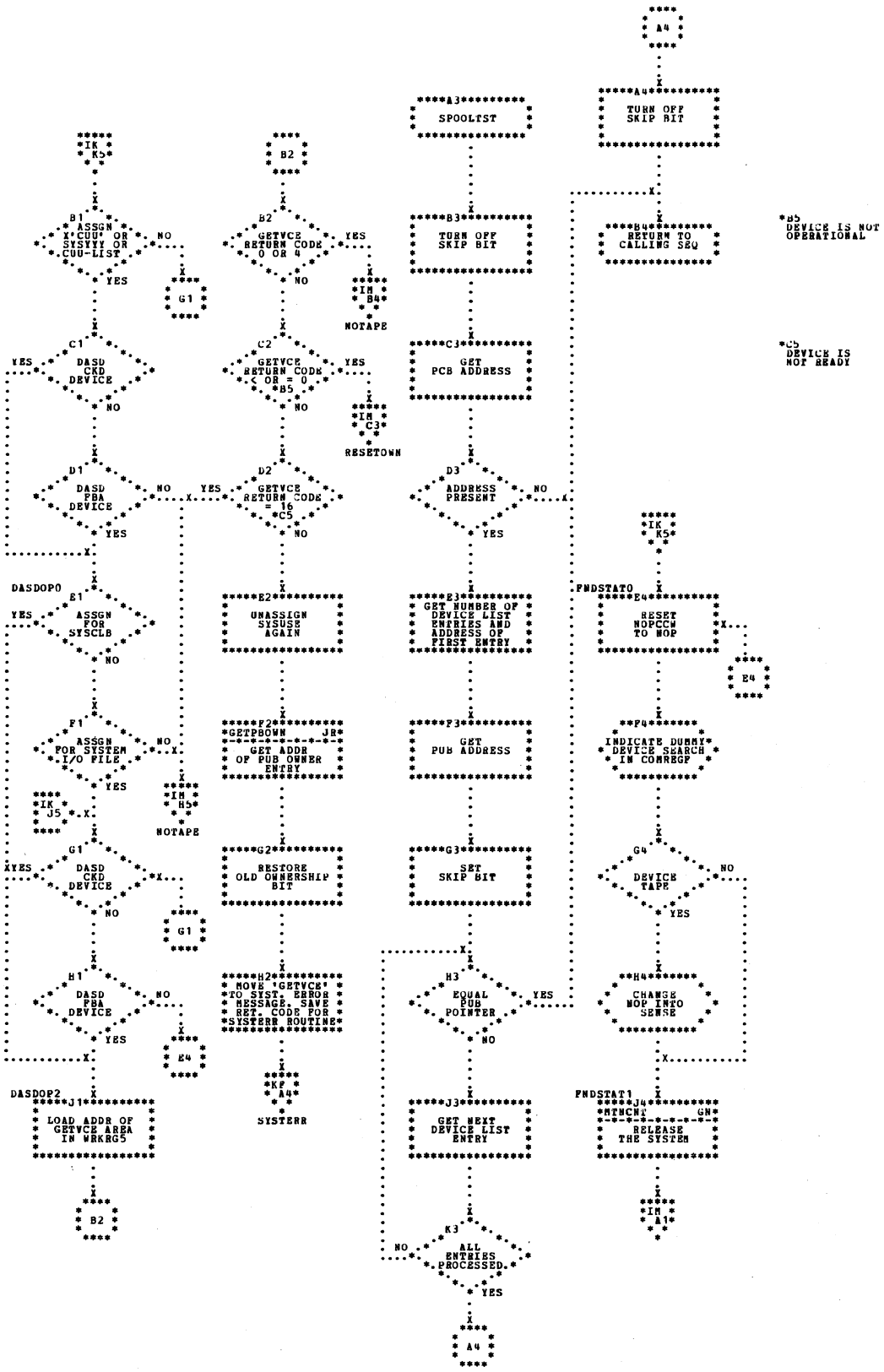


Chart IM. $JOBCTLD - Find a PUB to be Assigned
Refer to Charts 10 to 13.

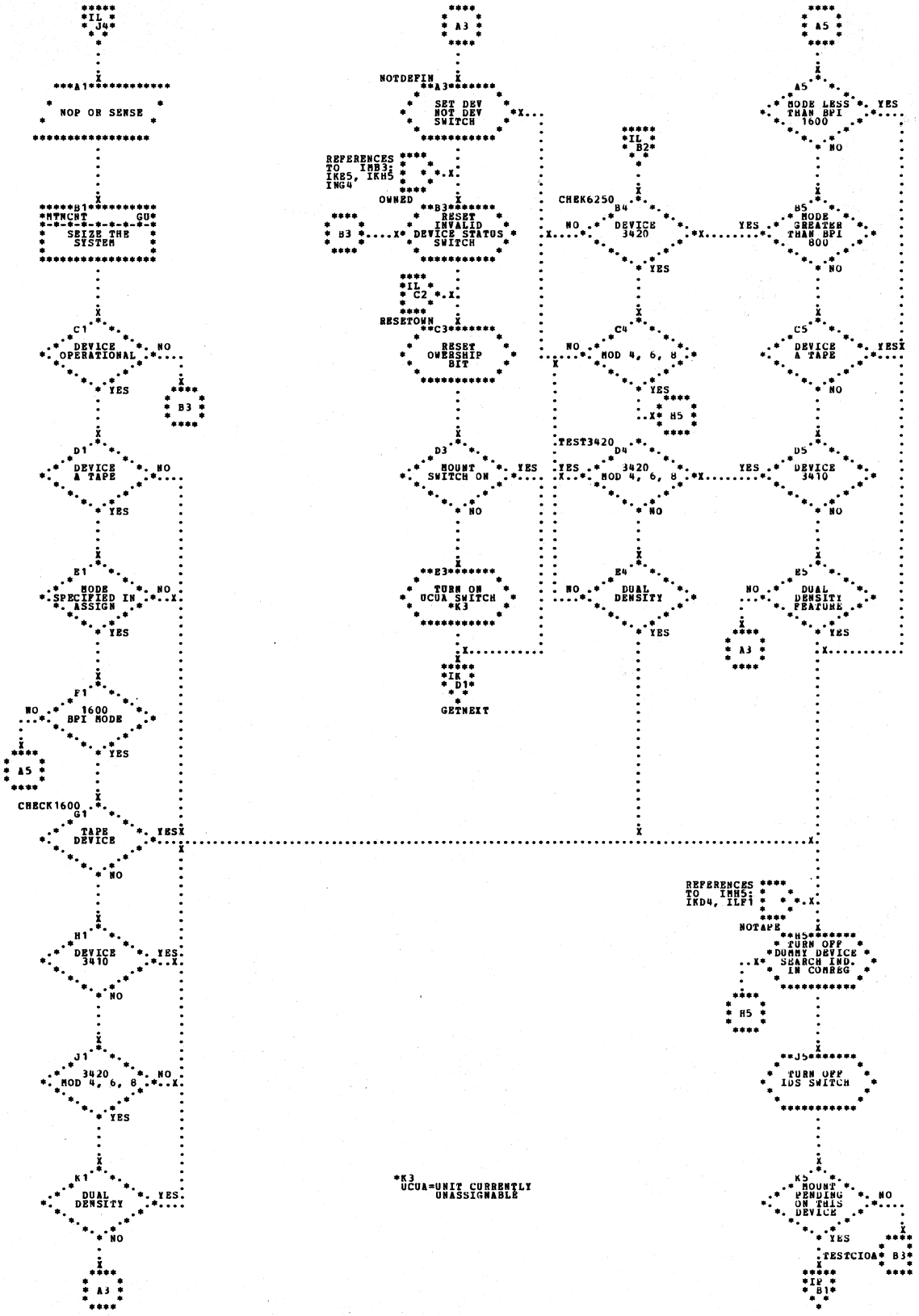


Chart IN. $JOBCTLD - Find a PUB to be Assigned
Refer to Charts 10 to 13.

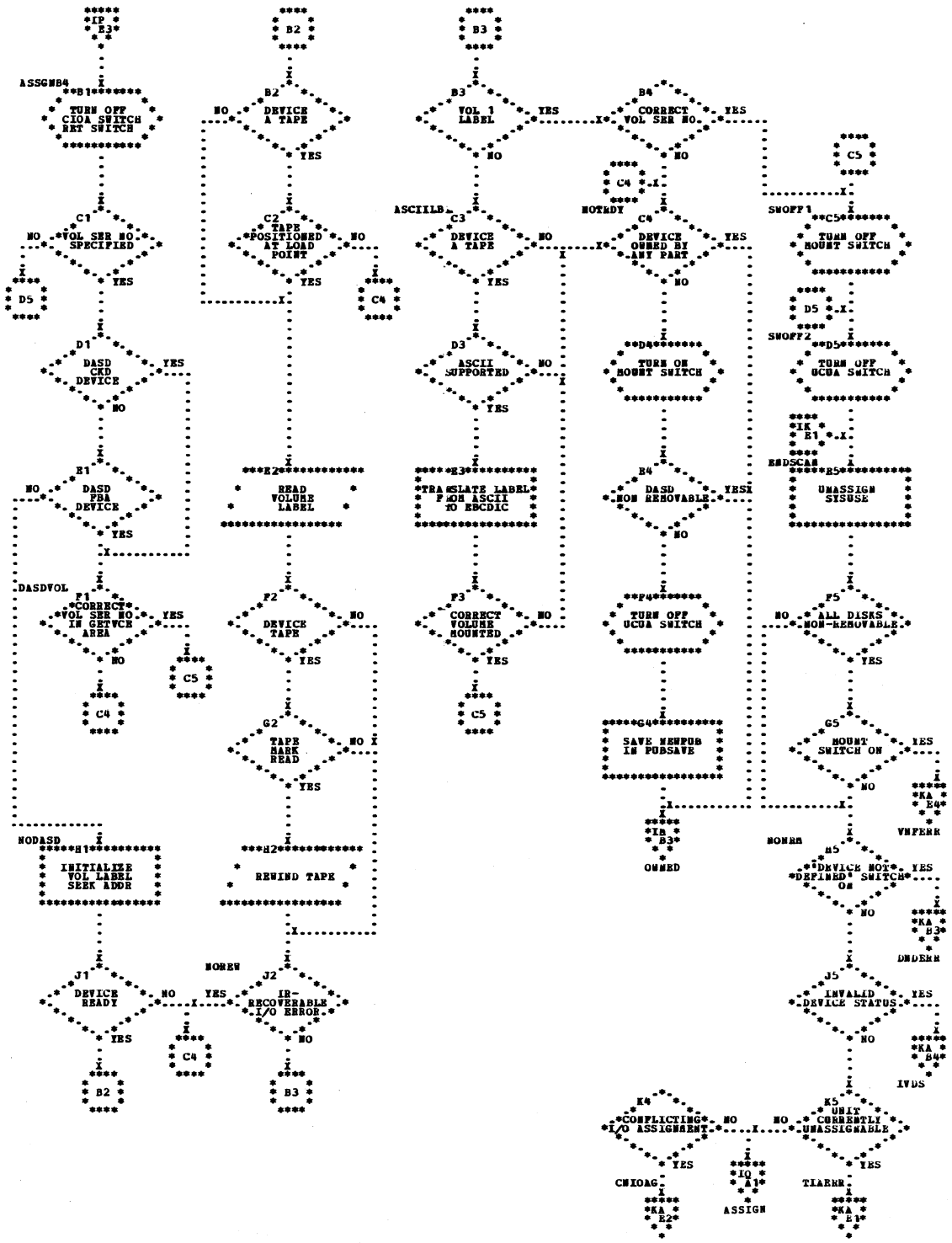


Chart IP. $JOBCTLD - Find a PUB to be Assigned
Refer to Charts 10 to 13.

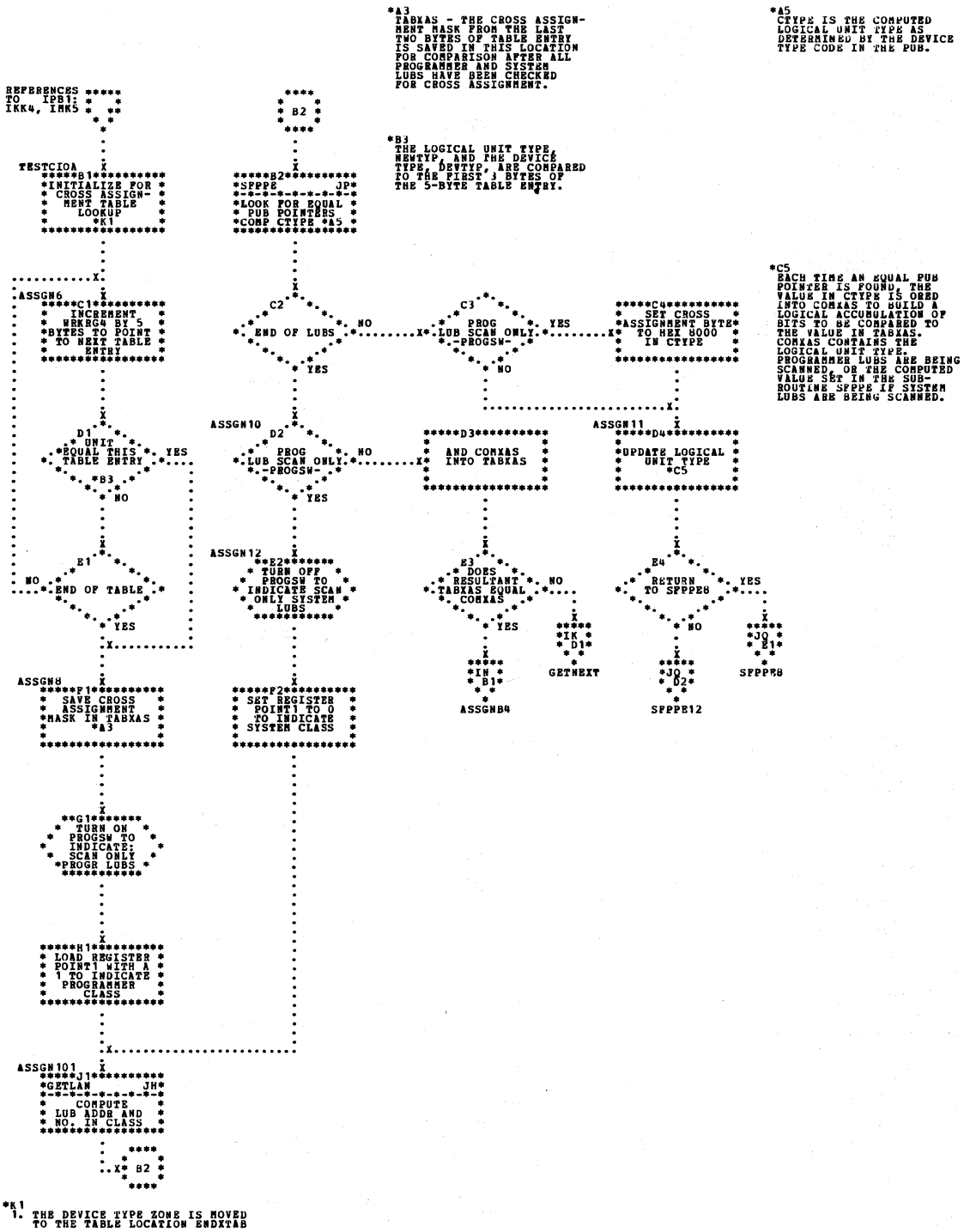


Chart IQ. $JOBCTLD - Prepare and Execute Assignment
Refer to Charts 10 to 13.

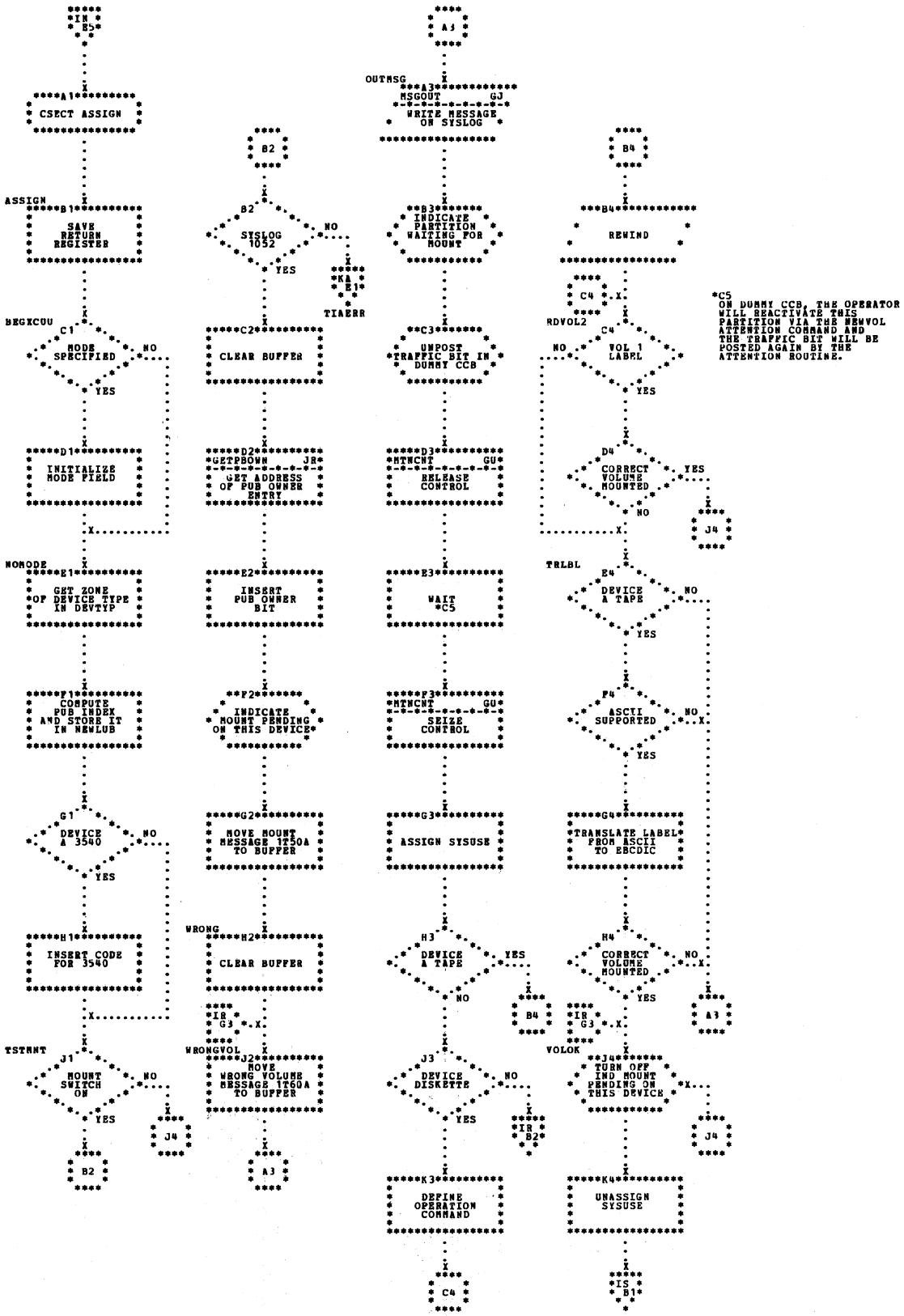


Chart IR. $JOBCTLD - Prepare and Execute Assignment
Refer to Charts 10 to 13.

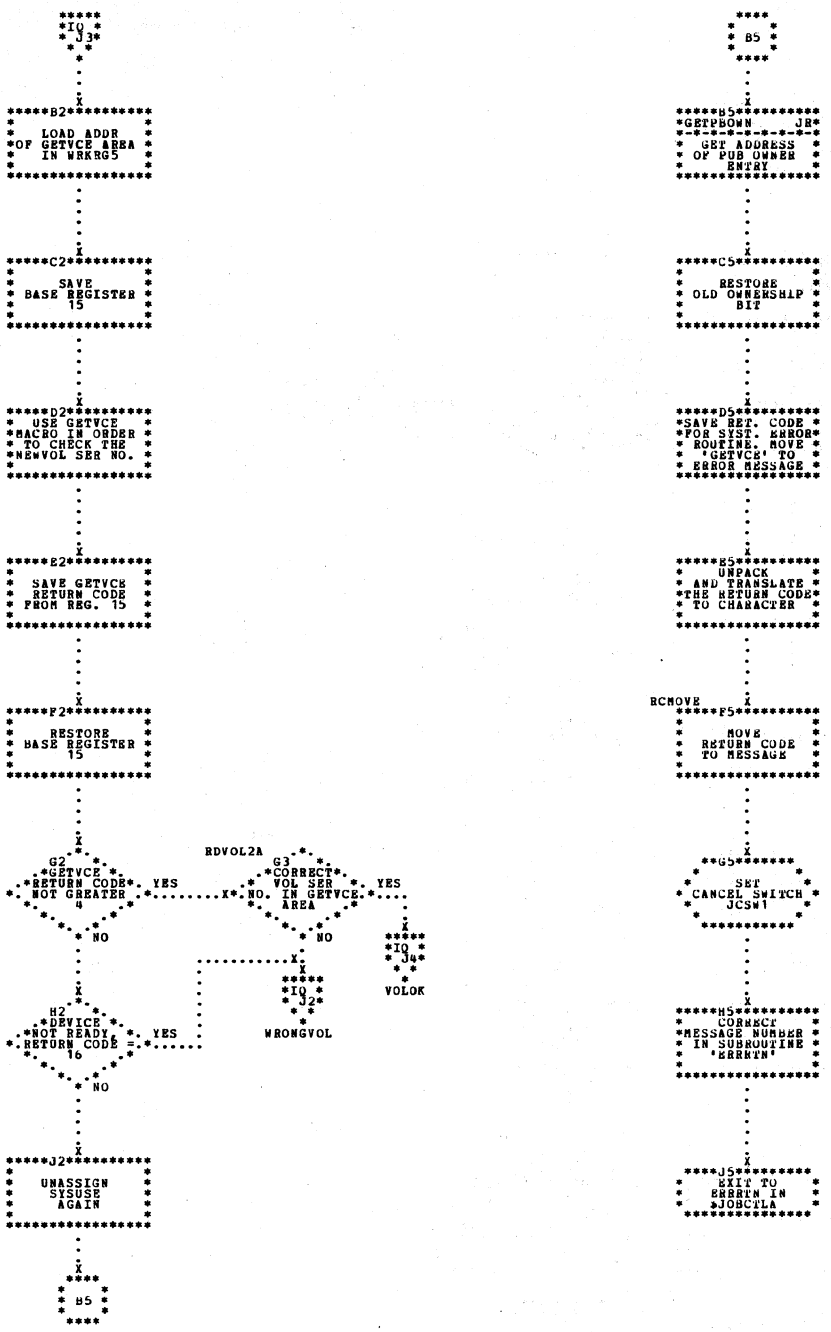


Chart IS. $JOBCTLD - Prepare and Execute Assignment
Refer to Charts 10 to 13.

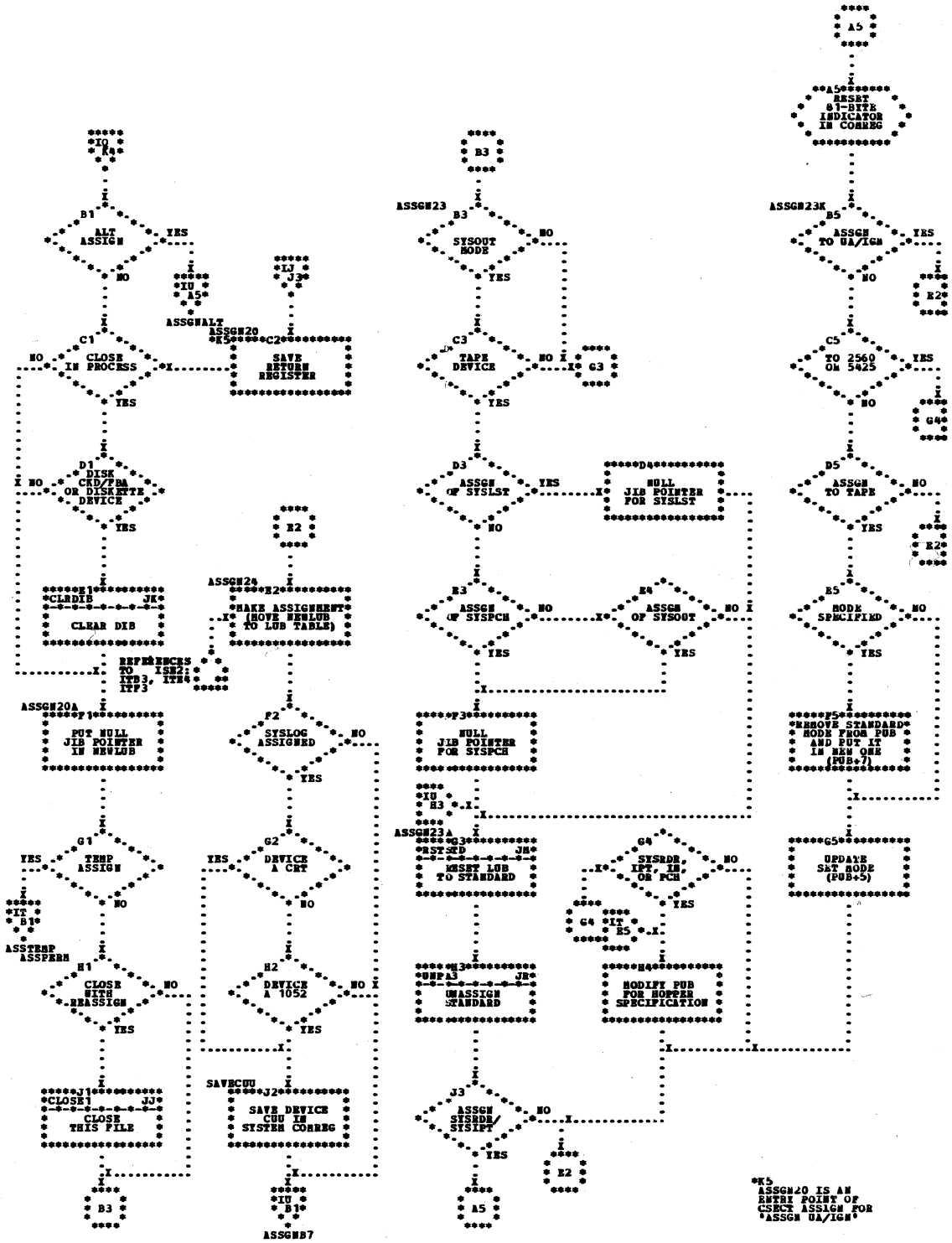


Chart IT. $JOBCTLD - Prepare and Execute Assignment
Refer to Charts 10 to 13.

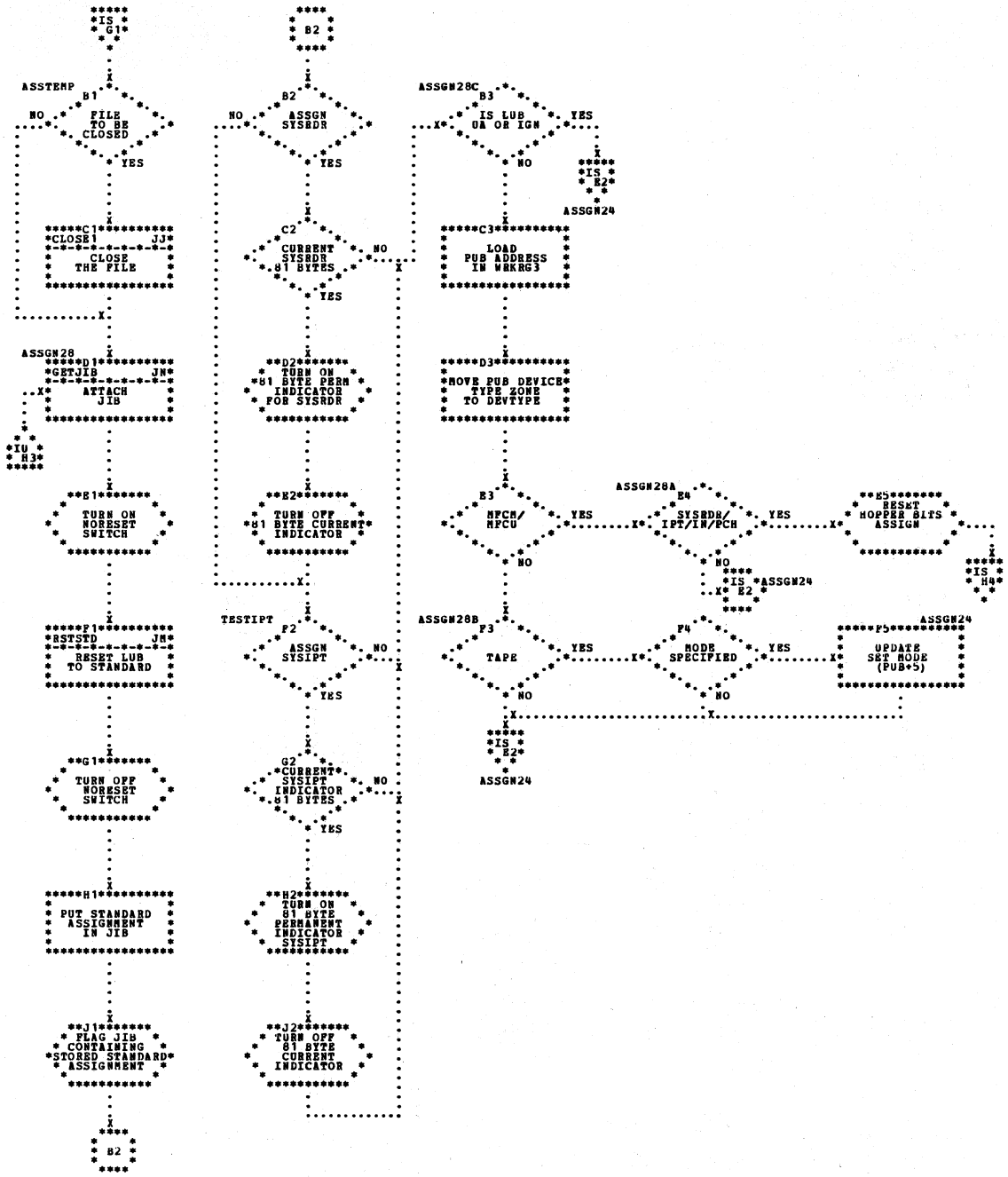


Chart IU. $JOBCTLD - Prepare and Execute Assignment
Refer to Charts 10 to 13.

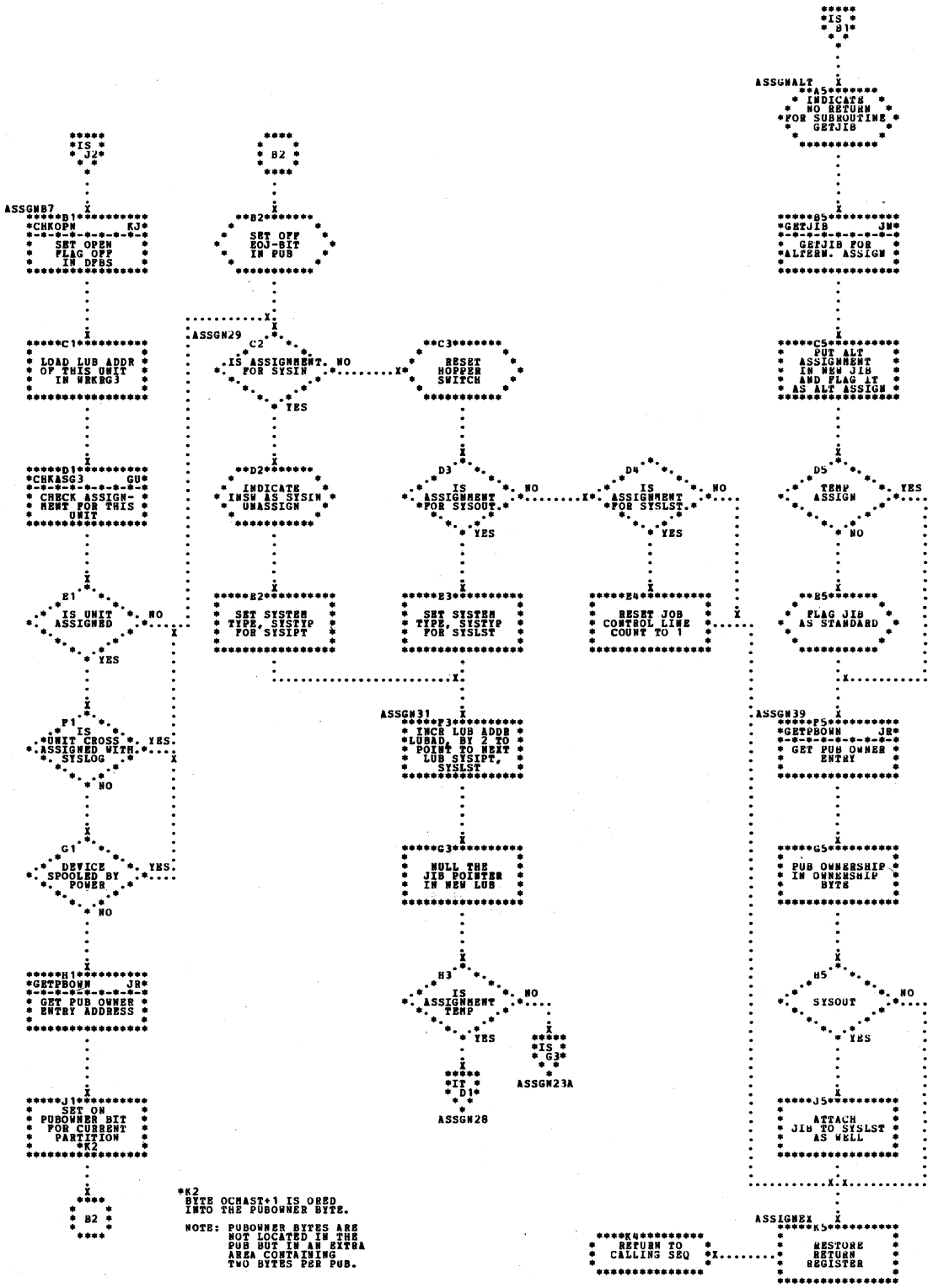


Chart JA. $JOBCTLD - Open System Files and Private CIL
Refer to Charts 10 to 13.

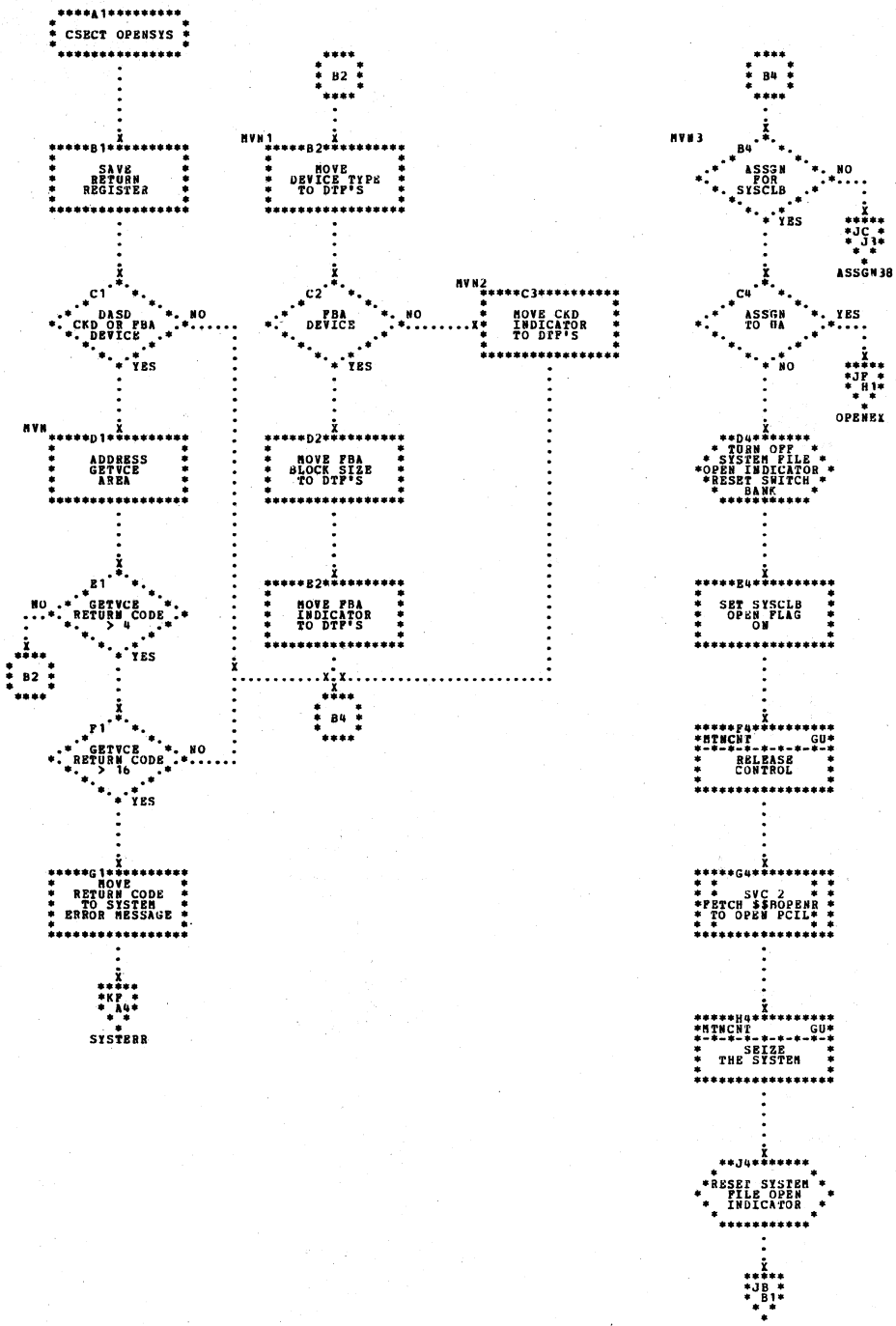


Chart JB. $JOBCTLD - Open System Files and Private CIL
Refer to Charts 10 to 13.

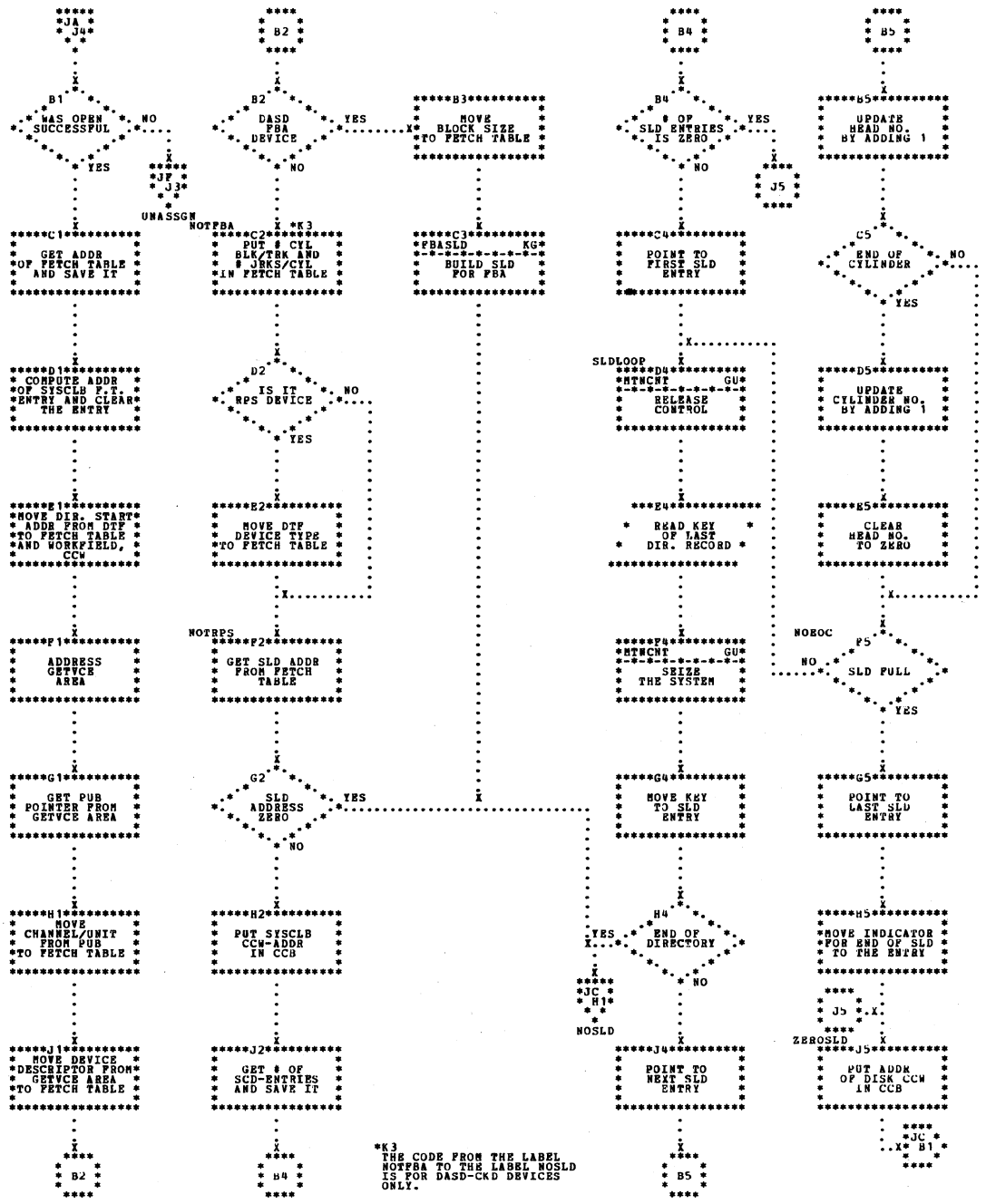


Chart JC. $JOBCTLD - Open System Files and Private CIL
Refer to Charts 10 to 13.

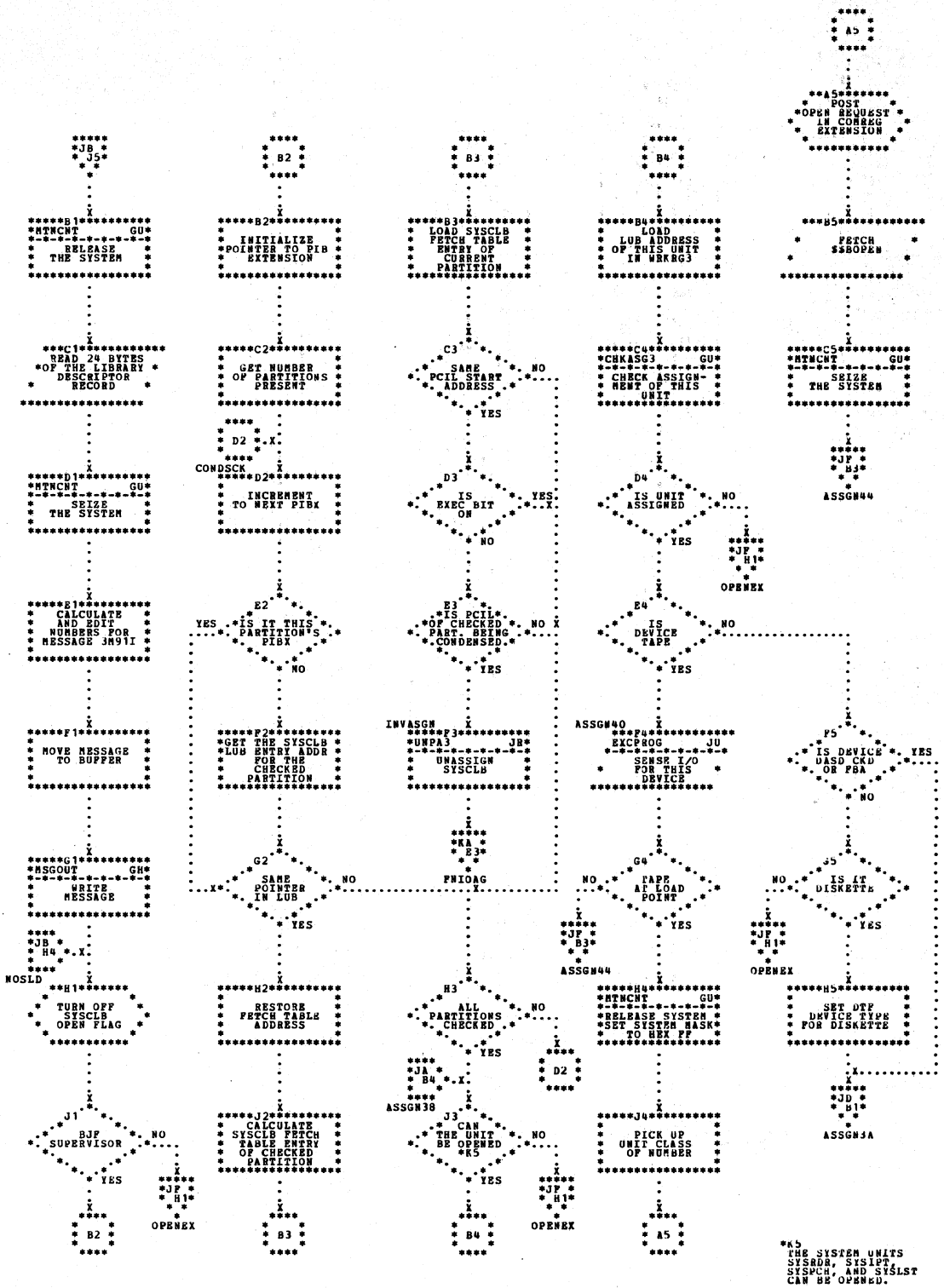


Chart JD. $JOBCTLD - Open System Files and Private CIL
Refer to Charts 10 to 13.

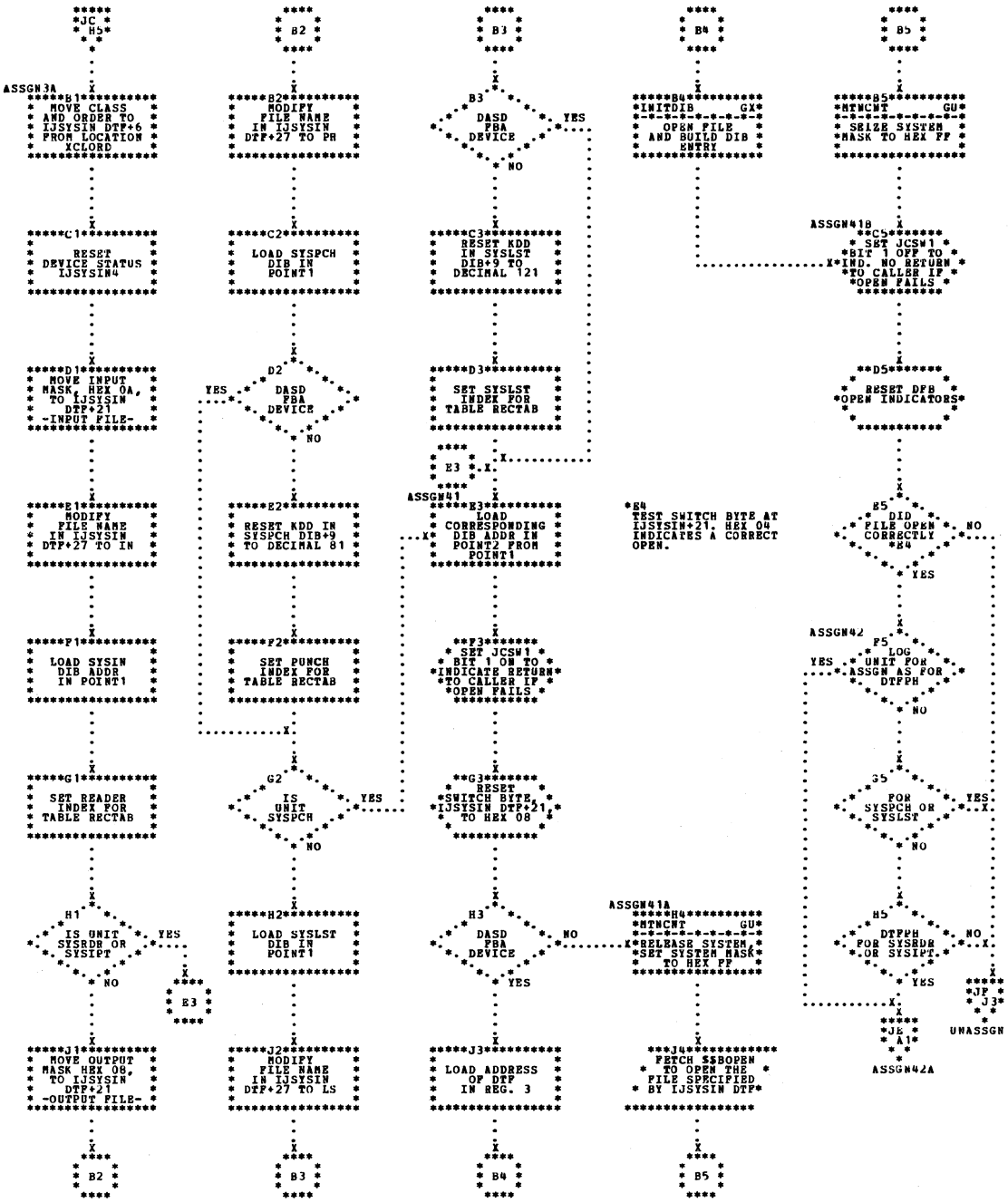


Chart JE. $JOBCTLD - Open System Files and Private CIL
Refer to Charts 10 to 13.

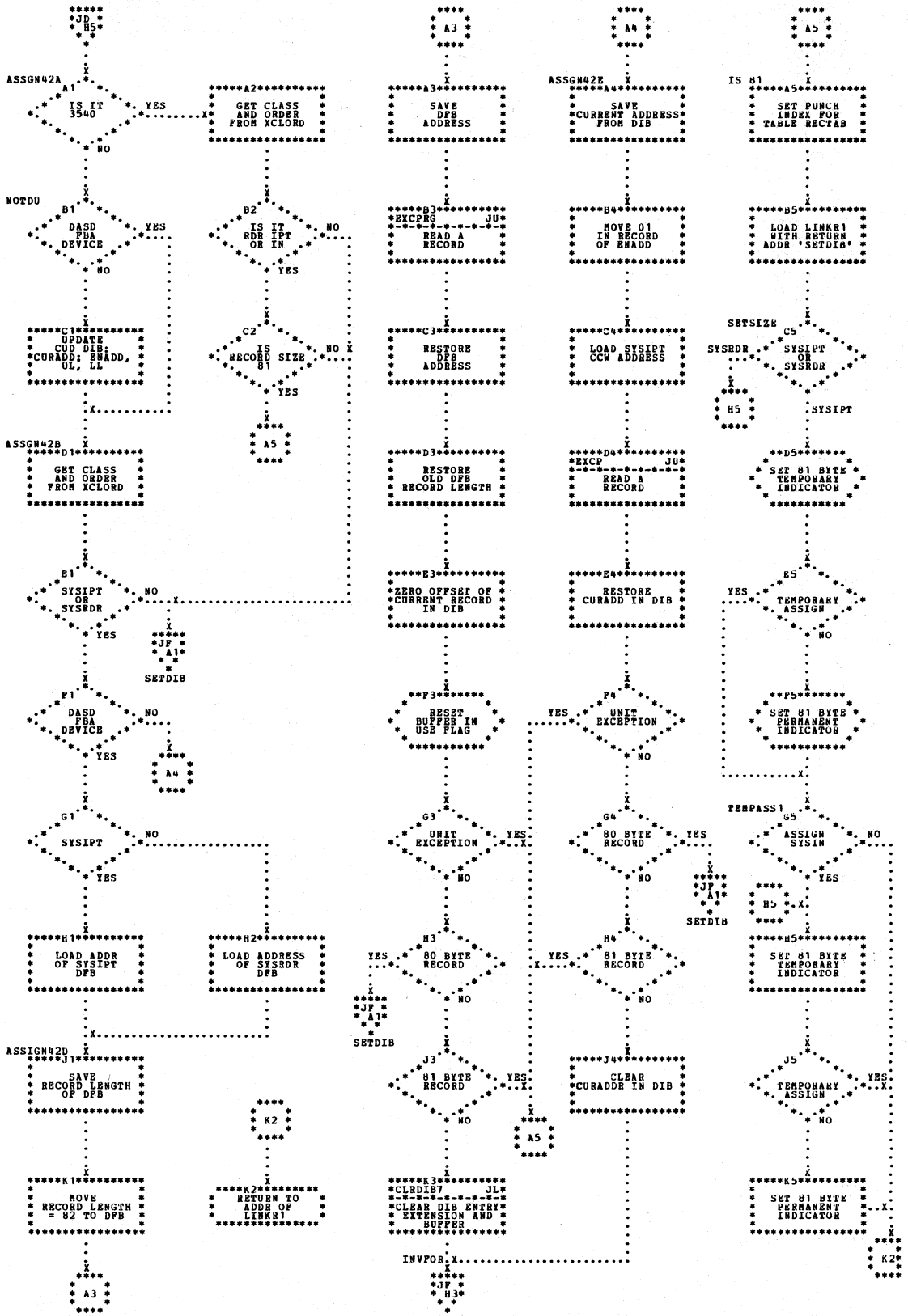


Chart JF. $JOBCTLD - Open System Files and Private CIL
Refer to Charts 10 to 13.

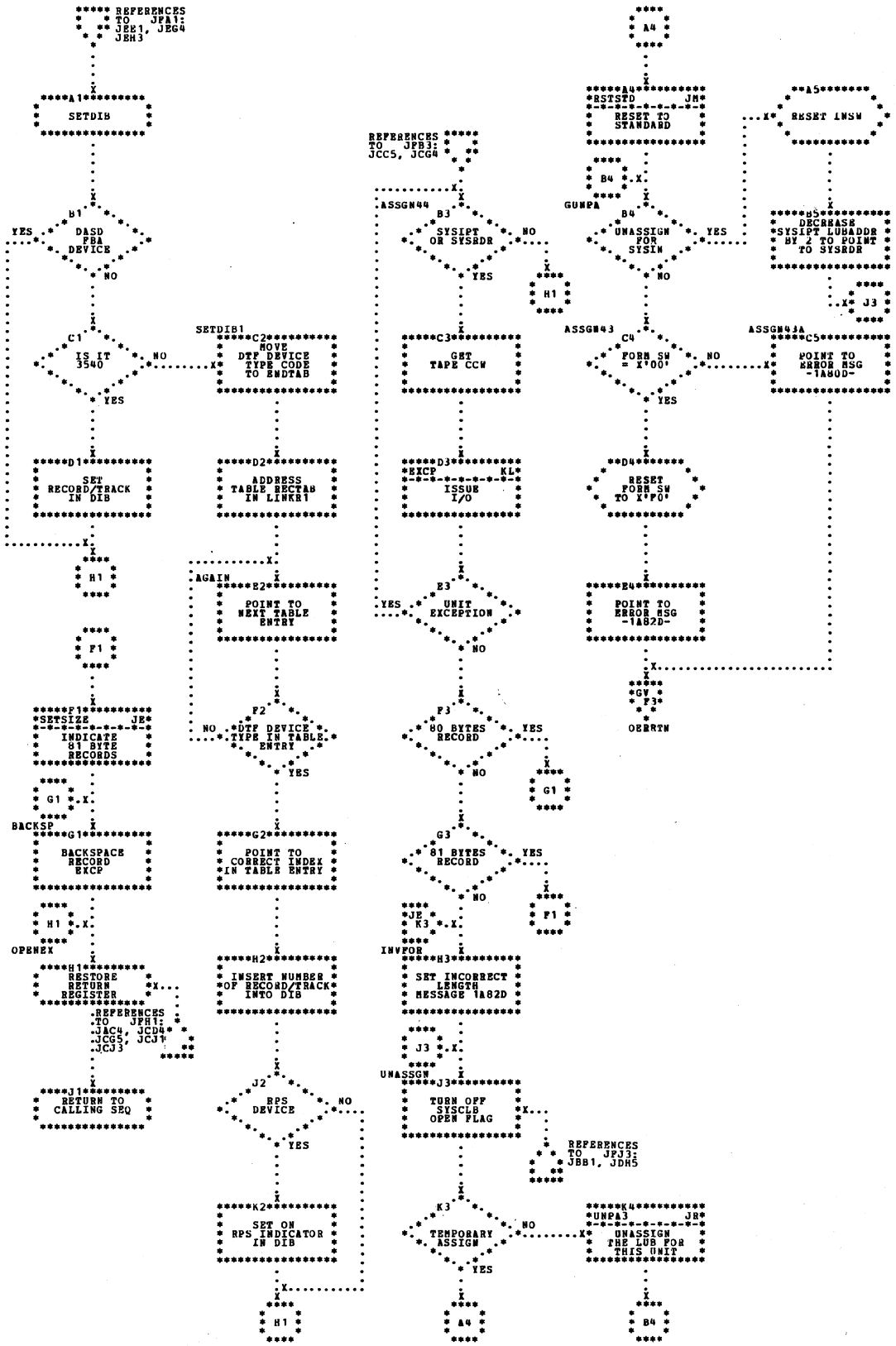


Chart JG. $JOBCTLD - Process CLOSE Command
Refer to Charts 10 to 13.

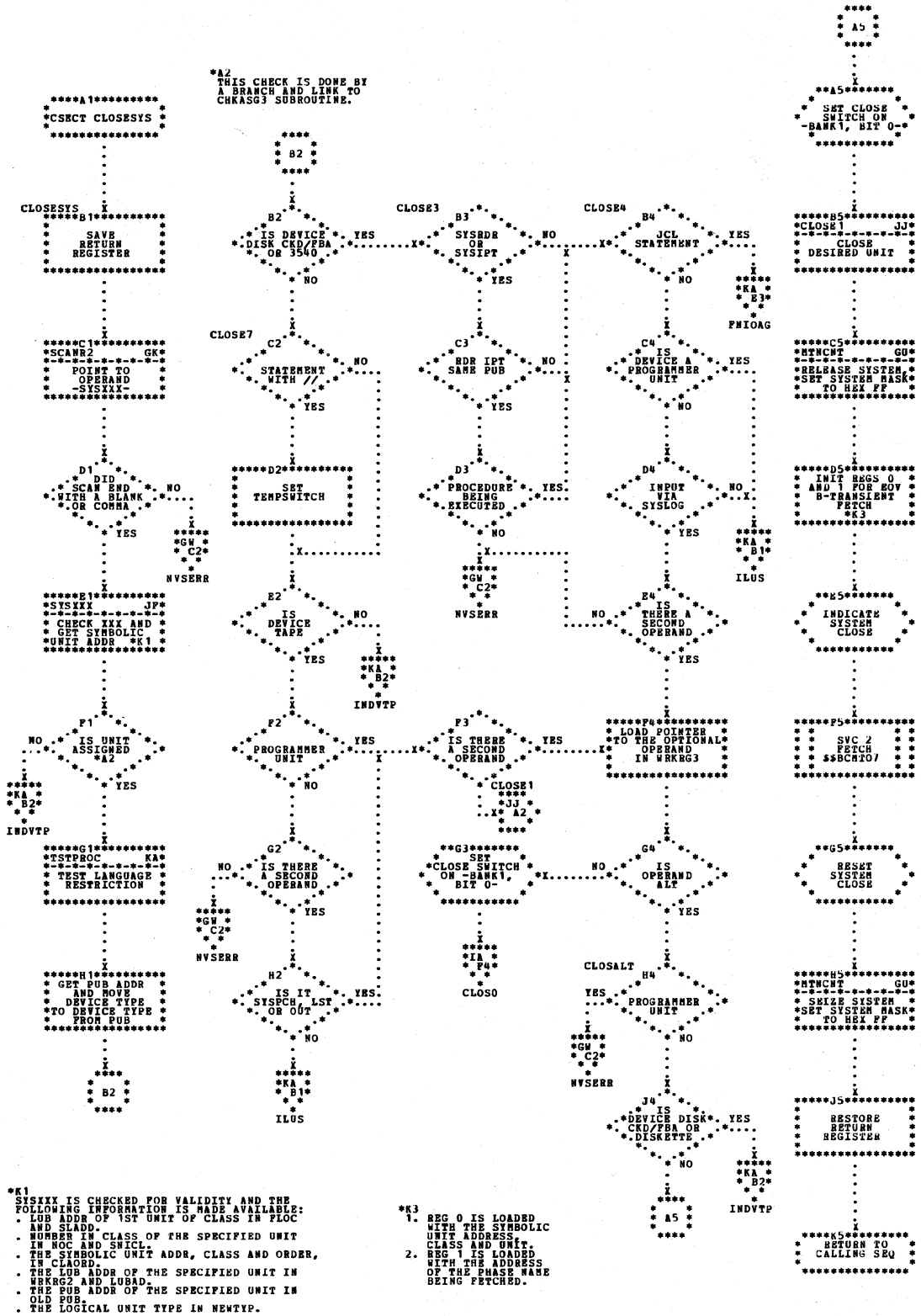
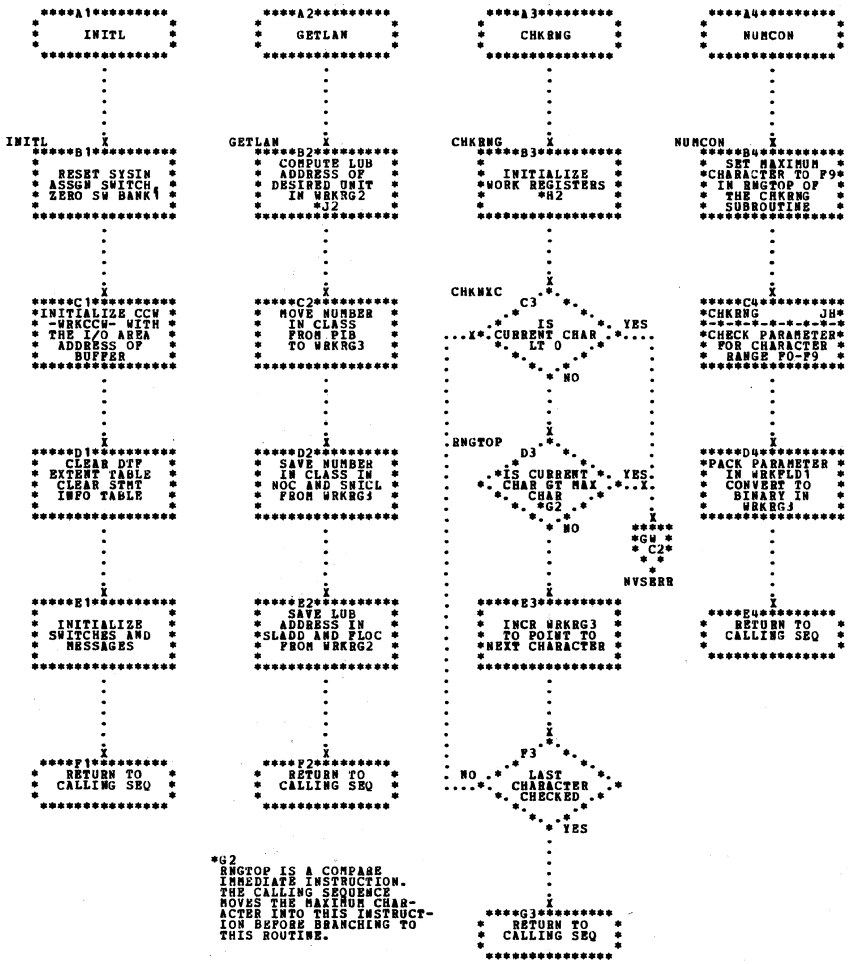


Chart JH. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.



*G2
RNGTOP IS A COMPARE IMMEDIATE INSTRUCTION. THE CALLING SEQUENCE MOVES THE MAXIMUM CHARACTER INTO THIS INSTRUCTION BEFORE BRANCHING TO THIS ROUTINE.

*H2
1. WRKRG3 IS SET TO THE PARAMETER START ADDRESS FROM POINT1 (NOTE 1).
2. WRKRG4 IS SET TO THE NUMBER OF CHARACTERS IN THE PARAMETER BY LOADING A VALUE 1 GREATER THAN THE CONTENTS OF POINT3.

*J2
. BG . F1 . F2 ETC..
NICLS.SS.PP.SS.PP.SS.PP. ..
FICLS.SS.PP.SS.PP.SS.PP.....

THE FORMAT FOR LOCATIONS FICLS AND NICLS WHERE--
SS = SYSTEM CLASS
PP = PROGRAMMER CLASS
(NOTES 2 AND 3)

NOTES:
1. REGISTER POINT1 CONTAINS DISPLACEMENT FROM ENTRY POINT (MAY BE NEGATIVE).
2. CALCULATE PROPER FICLS AND NICLS USING PIK (PIK = '10' FOR BG, '20' FOR F2, '30' FOR F1).
3. FICL X 2 + LUBADD = LUB ADDRESS OF FIRST UNIT OF DESIRED CLASS IN DESIRED PARTITION.

Chart JJ. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

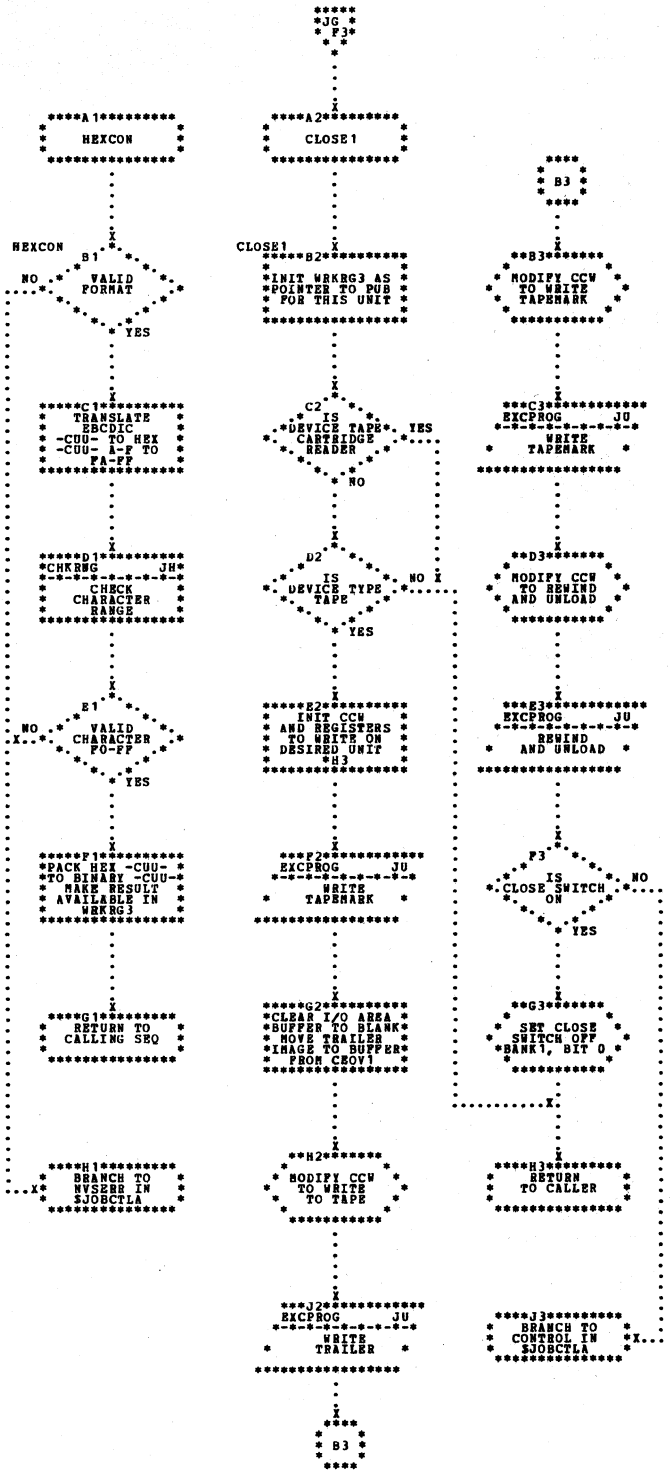


Chart JK. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

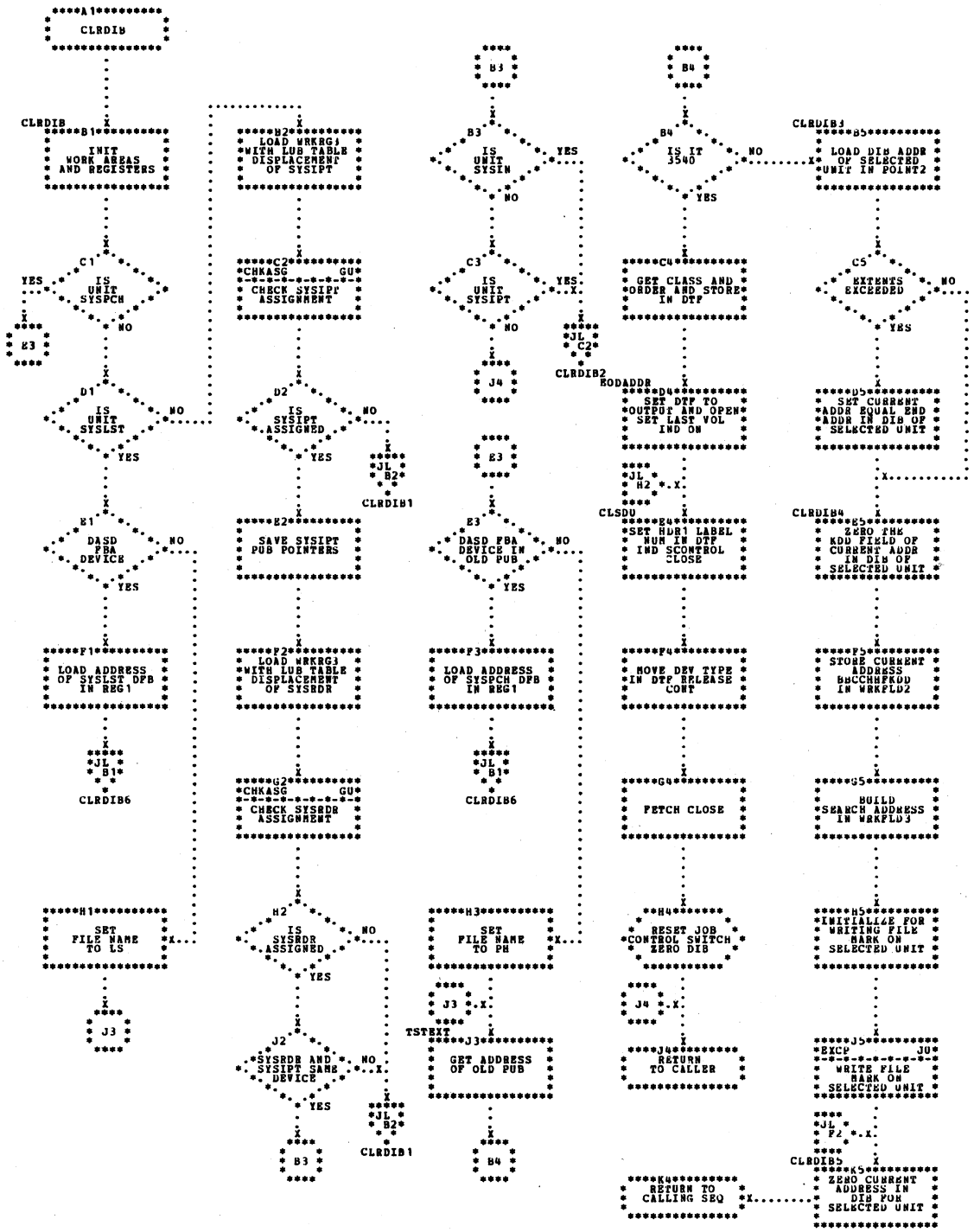


Chart JL. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

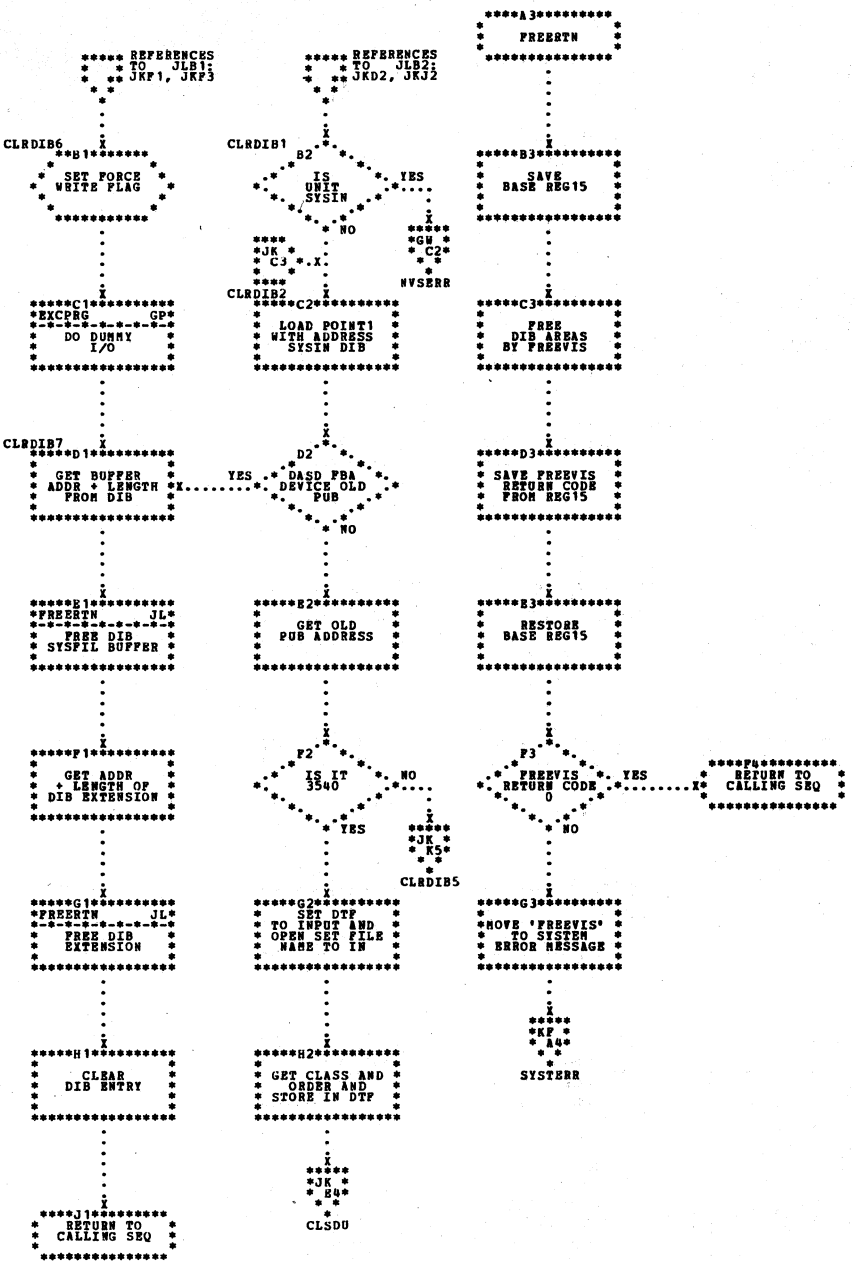


Chart JM. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

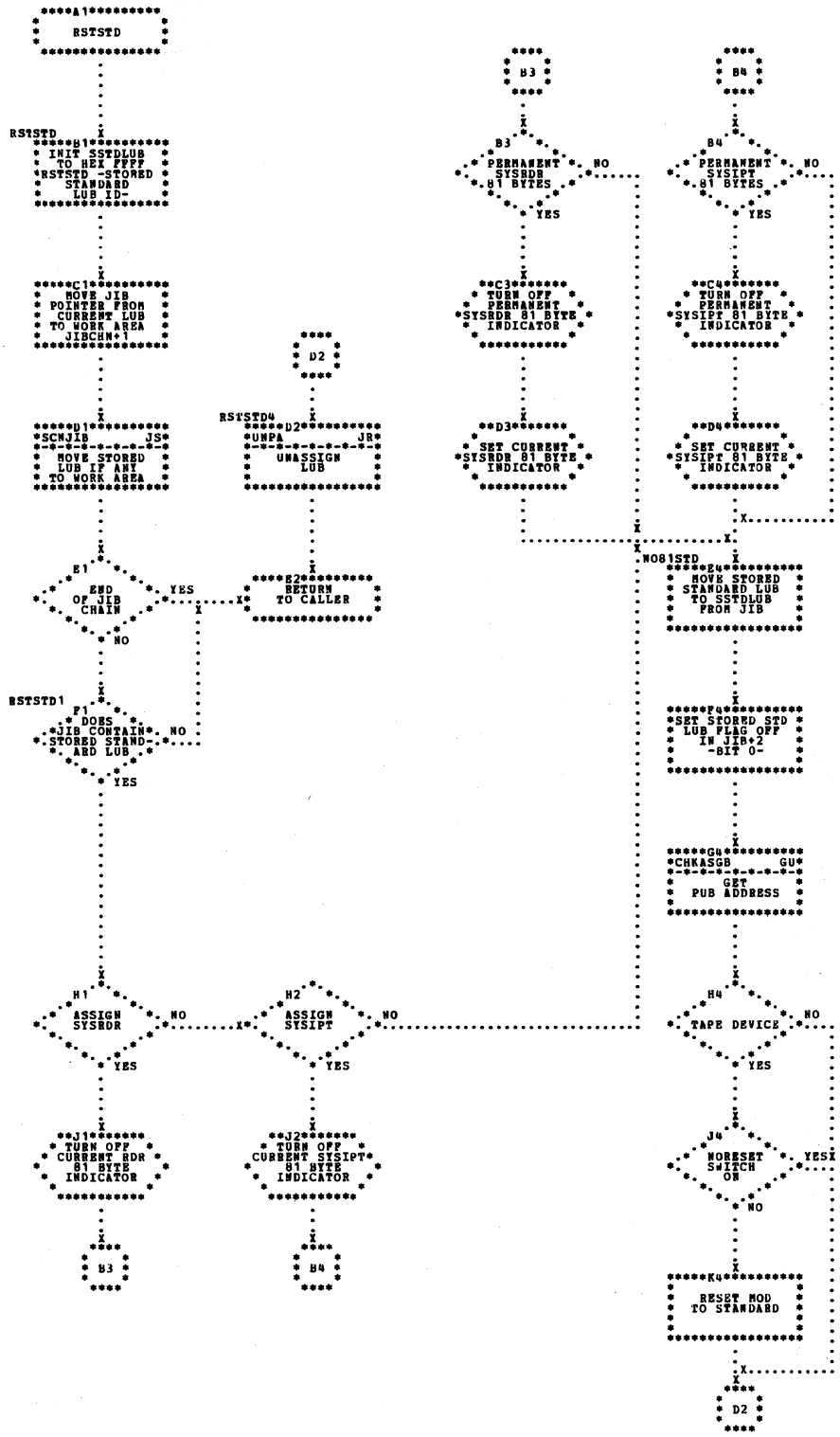


Chart JN. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

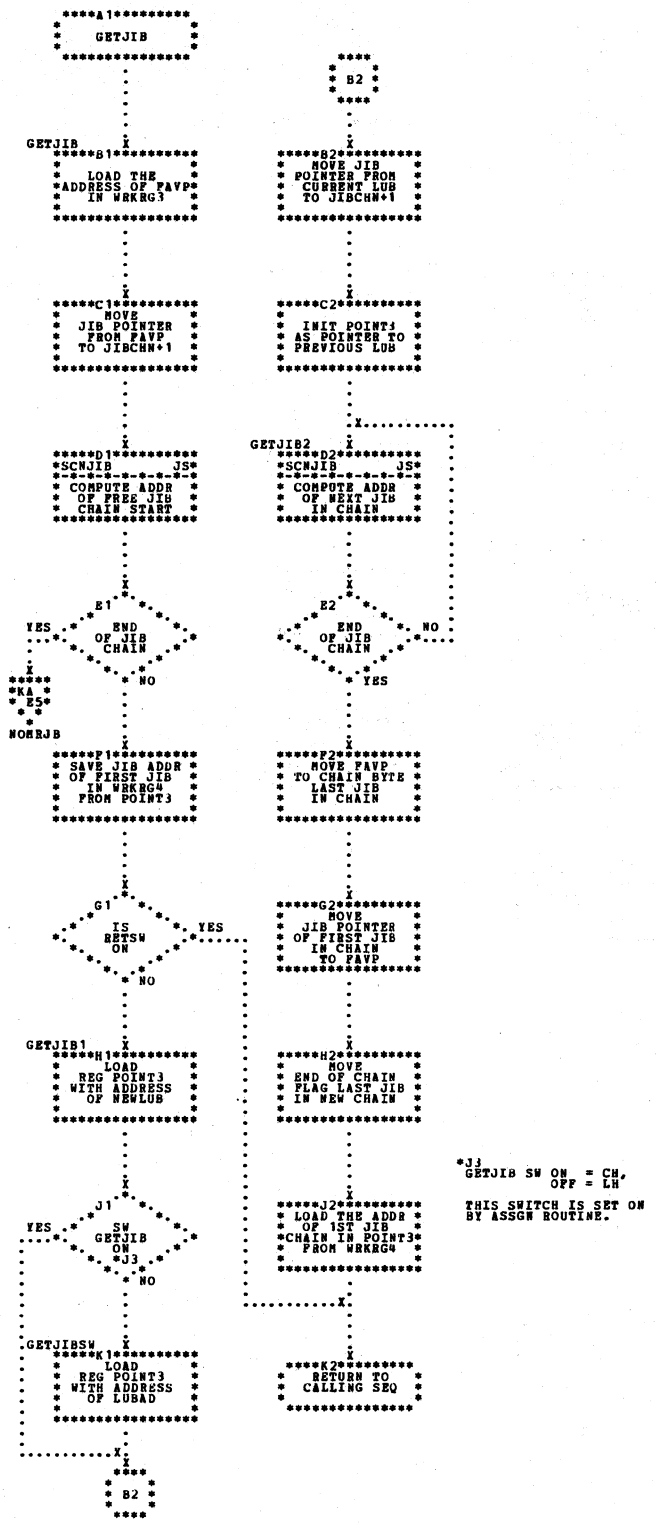


Chart JP. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

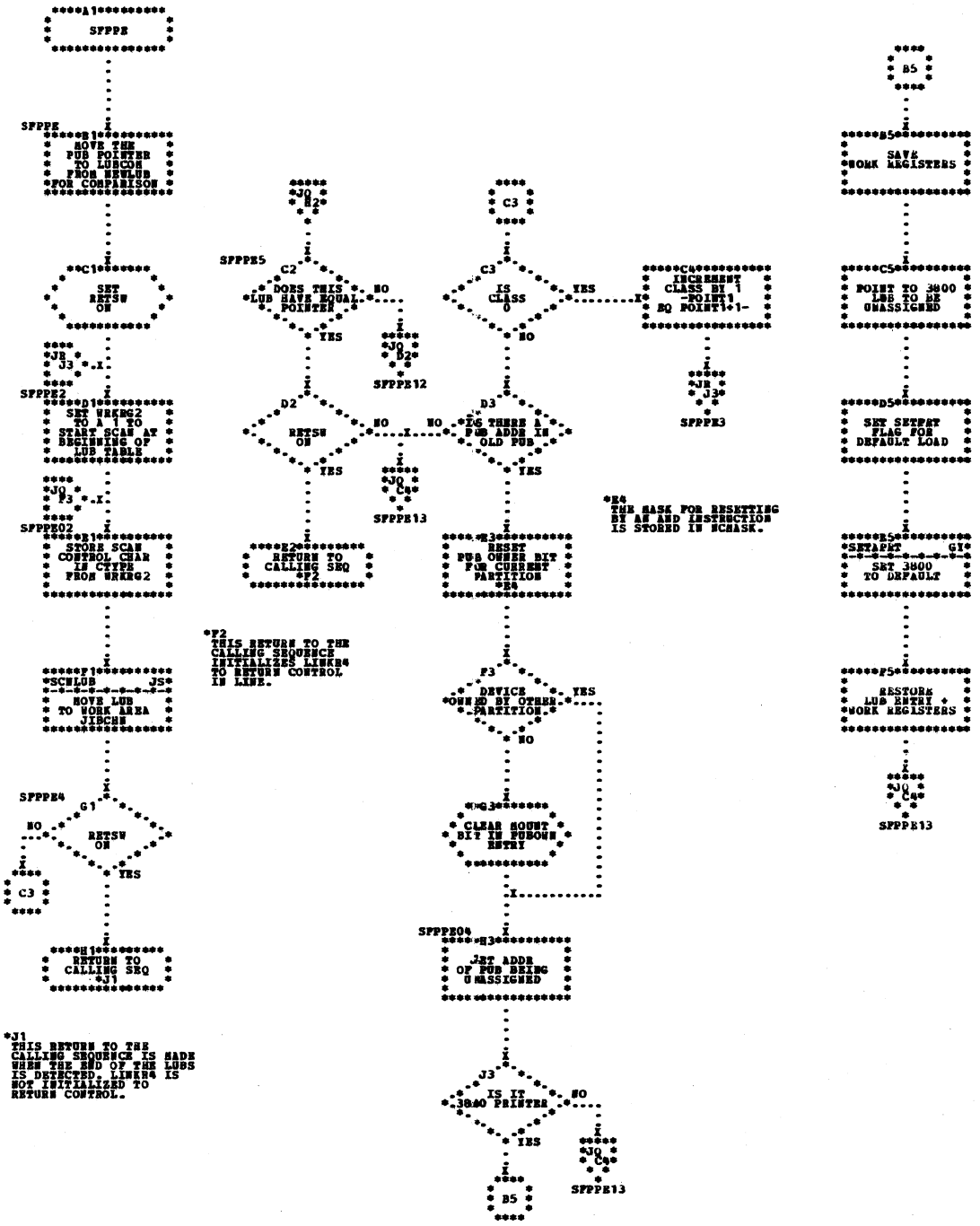


Chart JQ. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

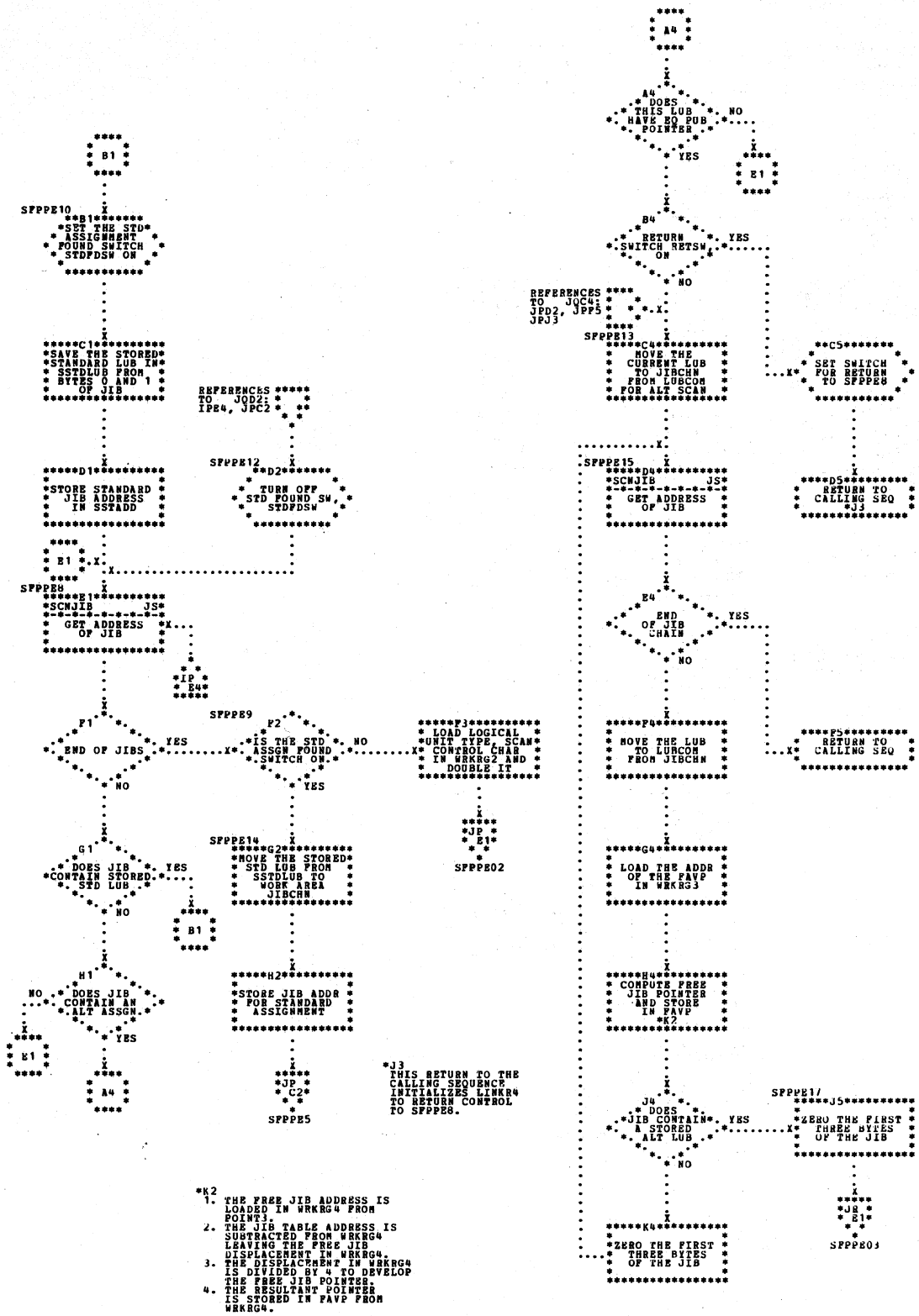


Chart JR. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

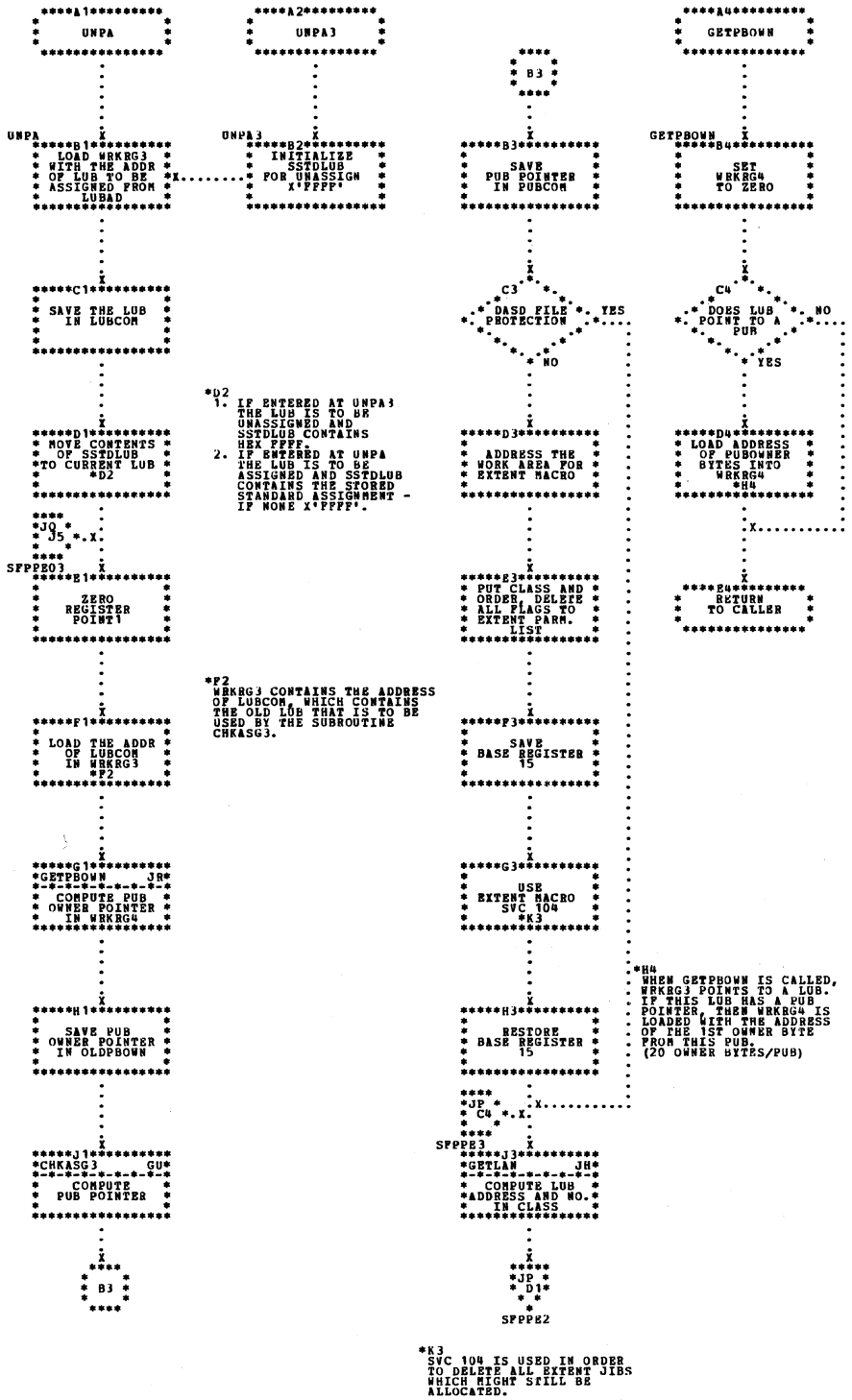


Chart JS. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

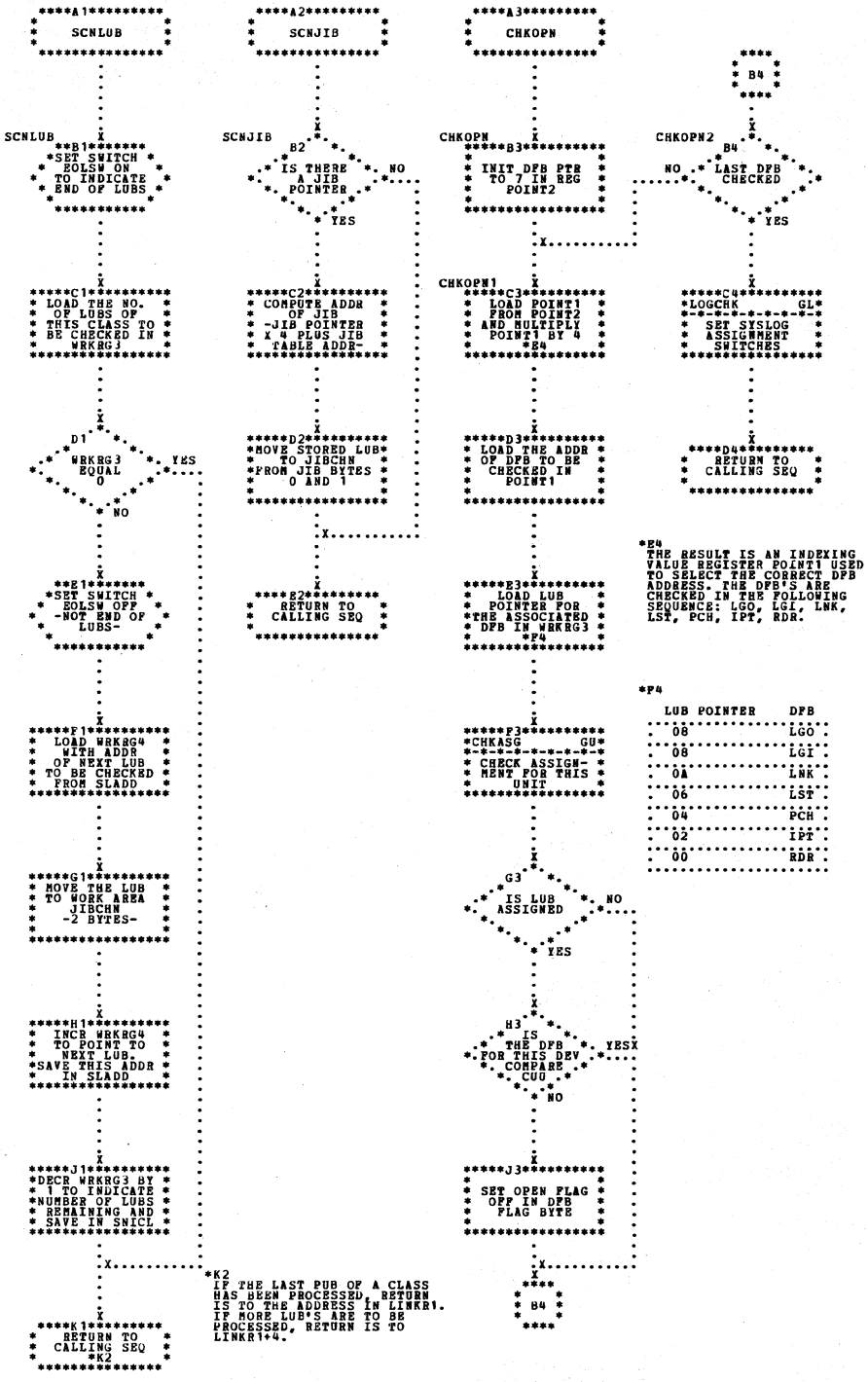


Chart JT. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

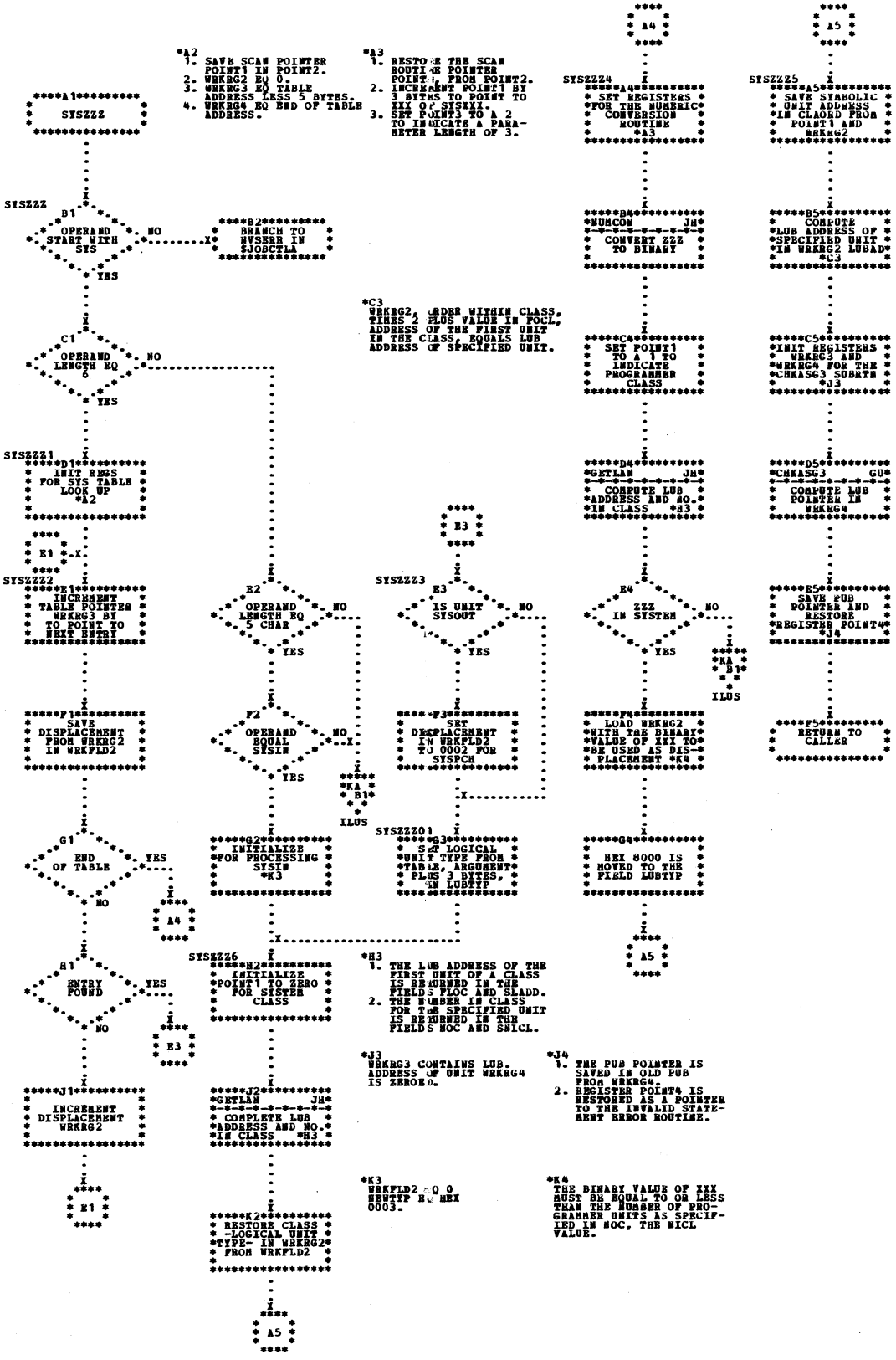


Chart JU. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

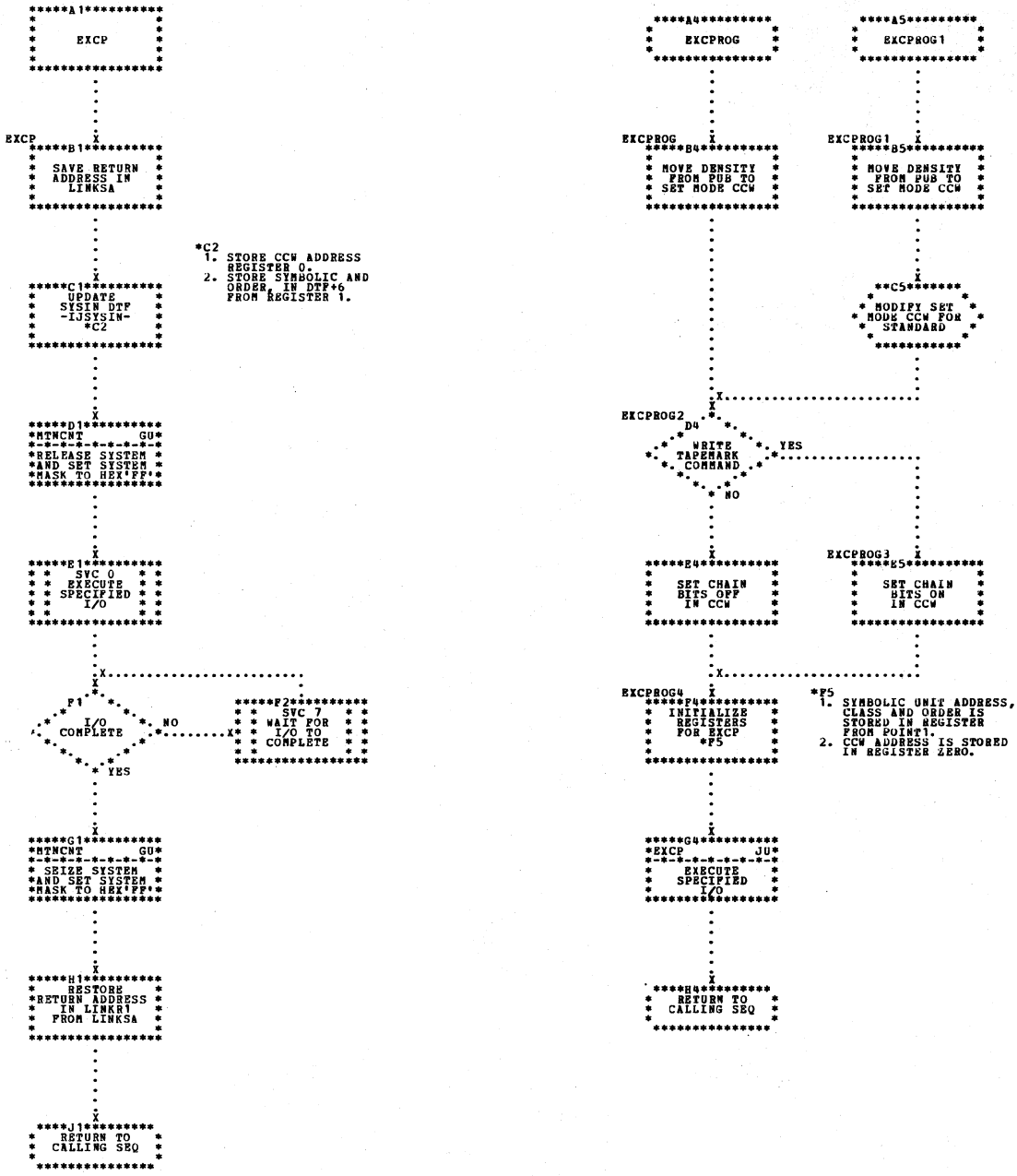


Chart KA. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

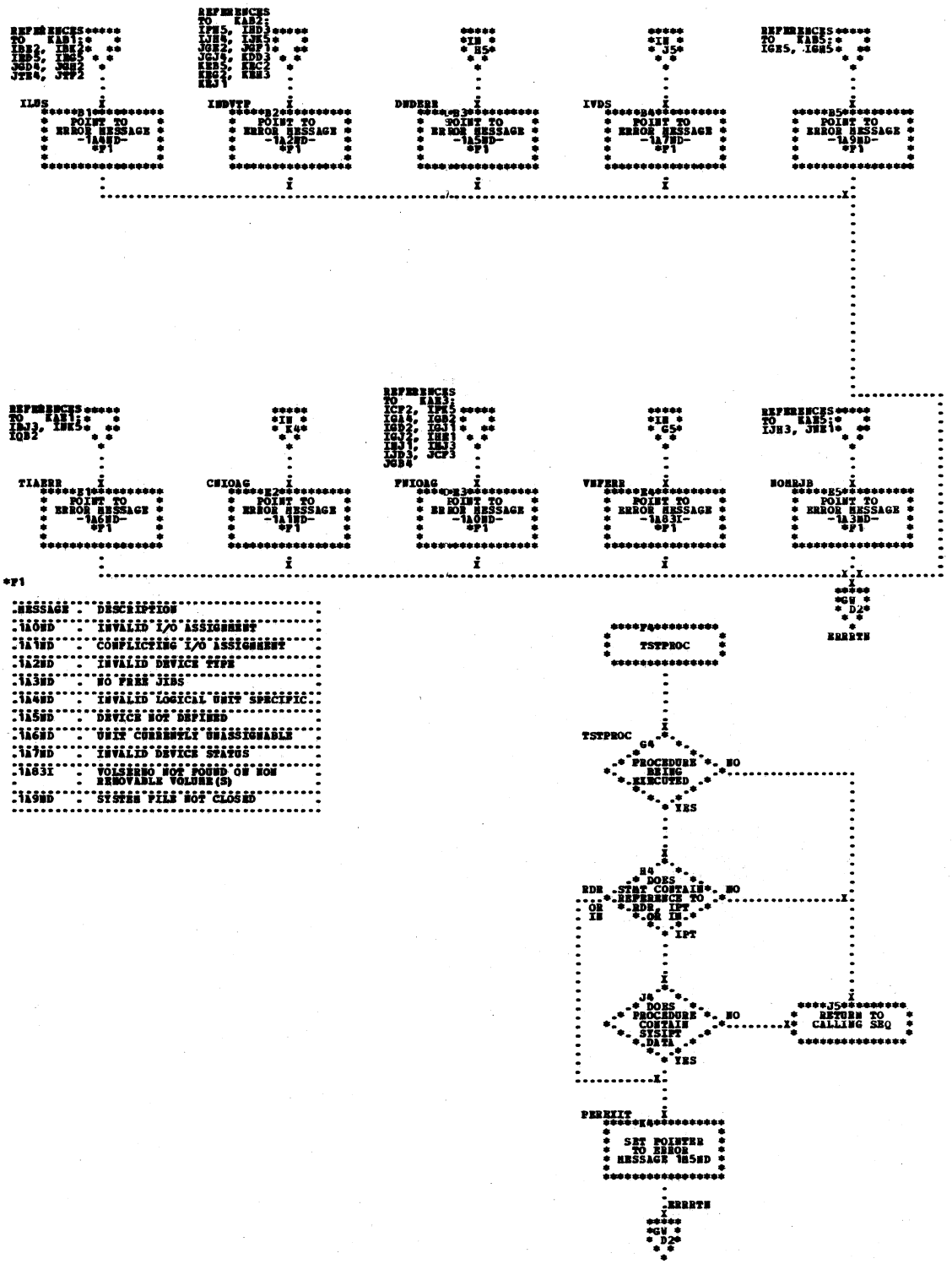


Chart KB. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

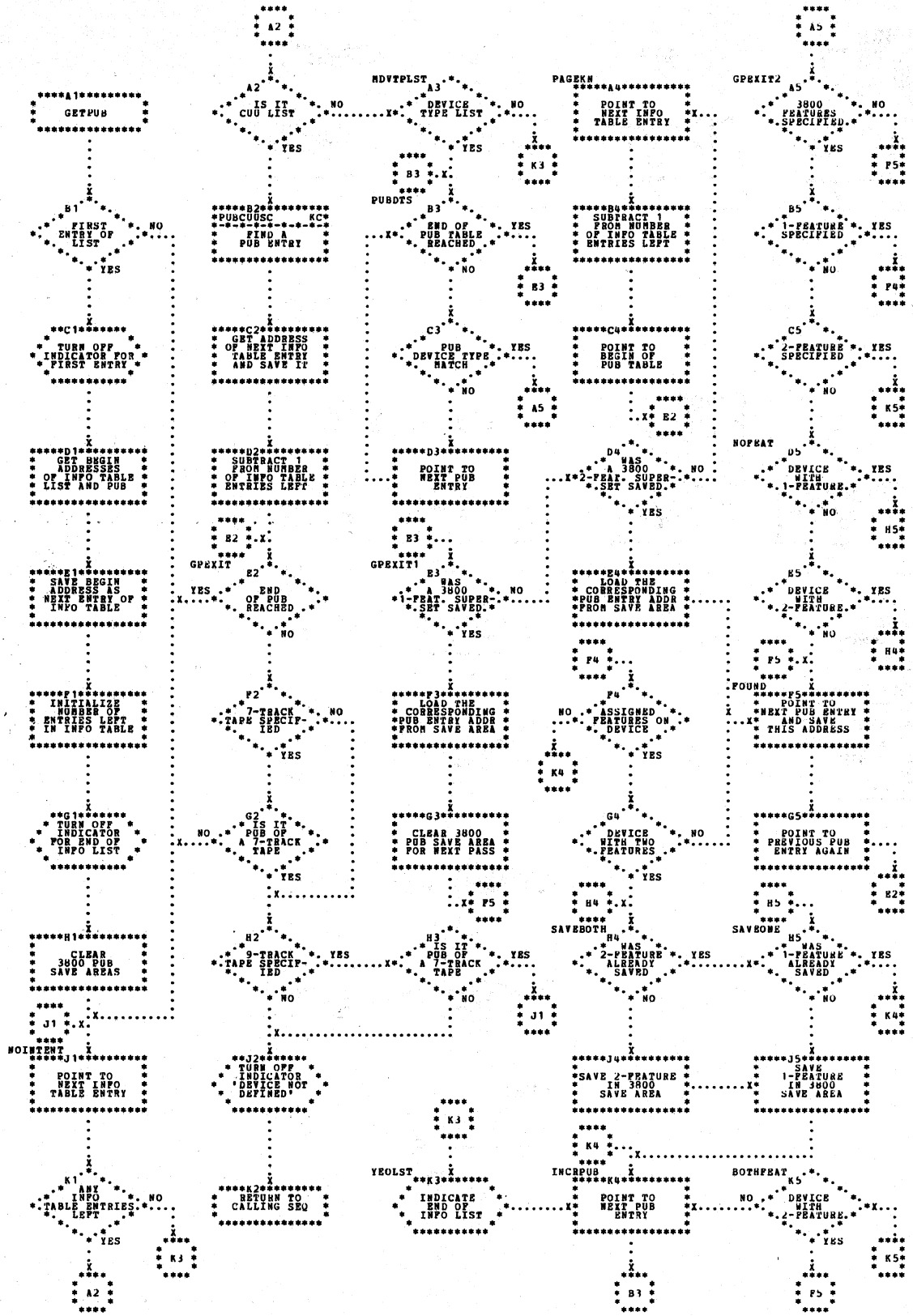


Chart KC. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

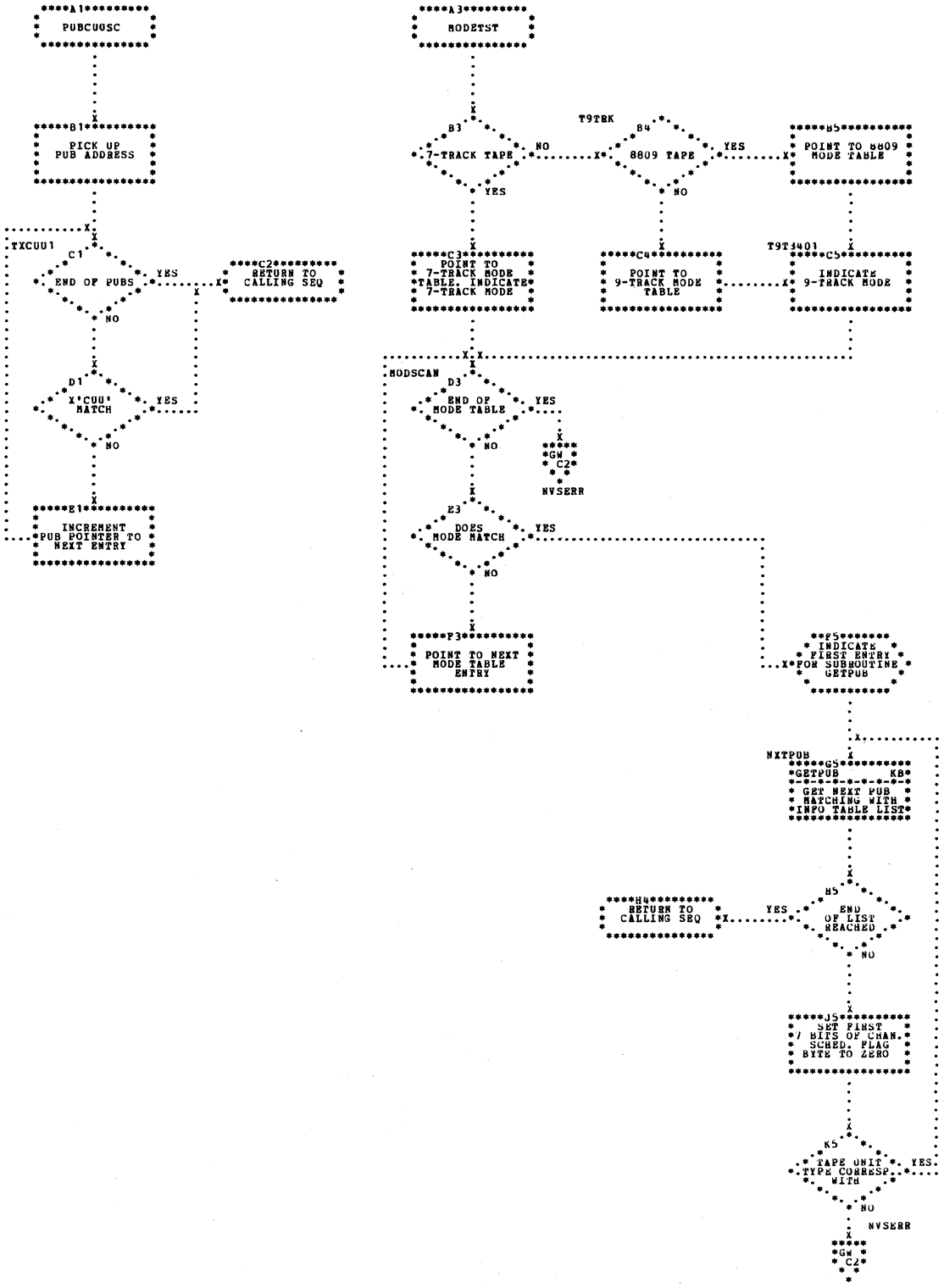


Chart KD. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

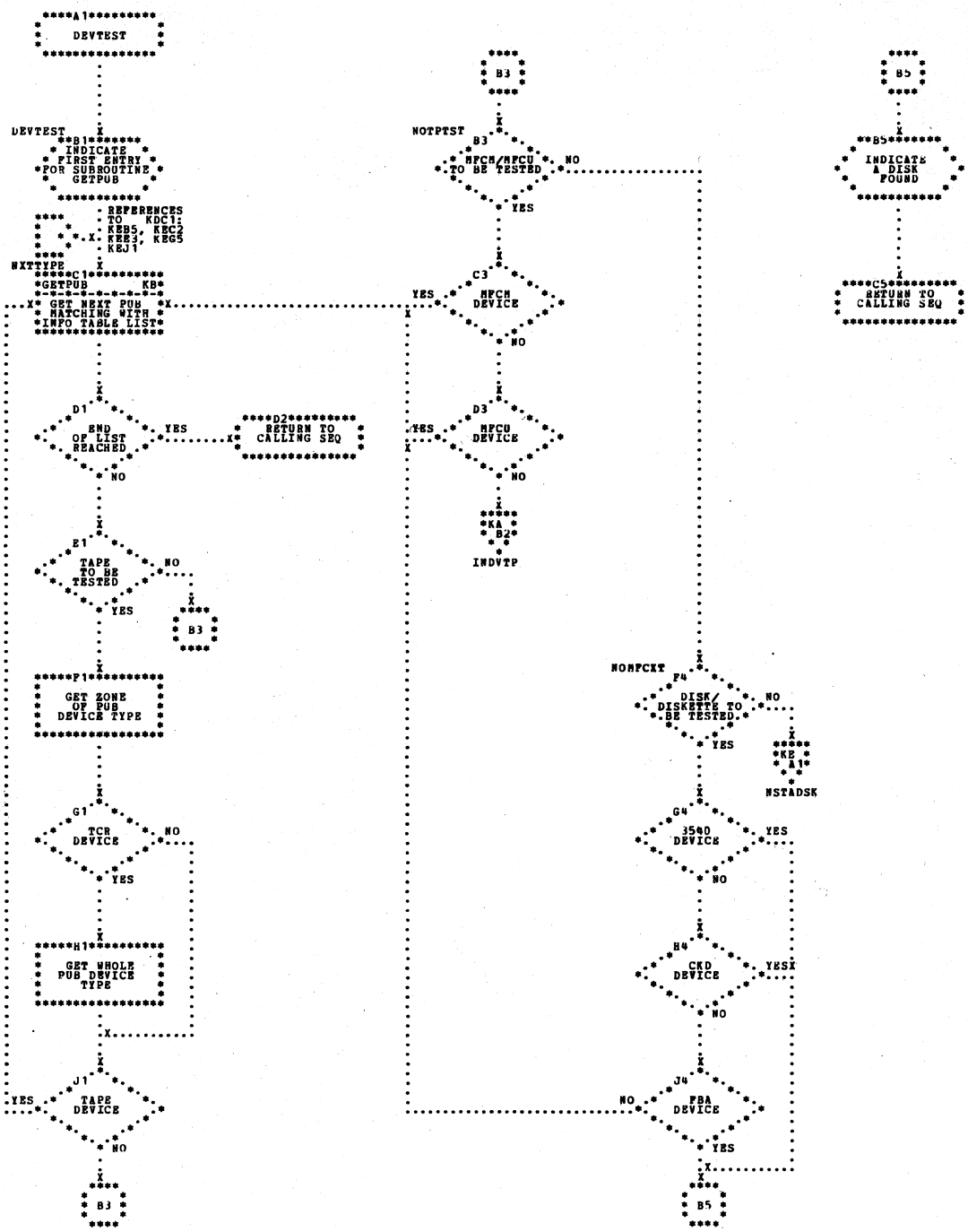


Chart KE. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

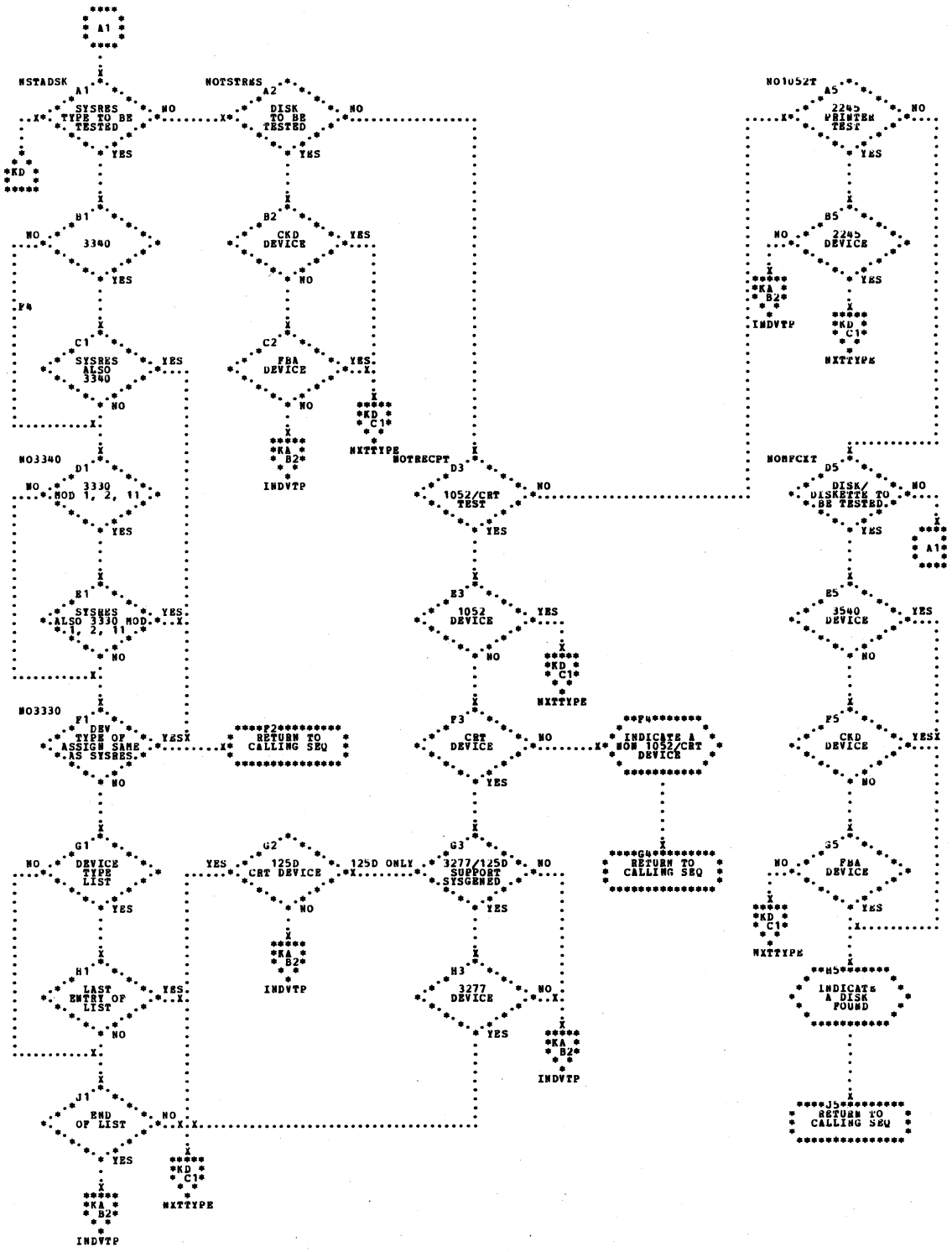


Chart KF. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

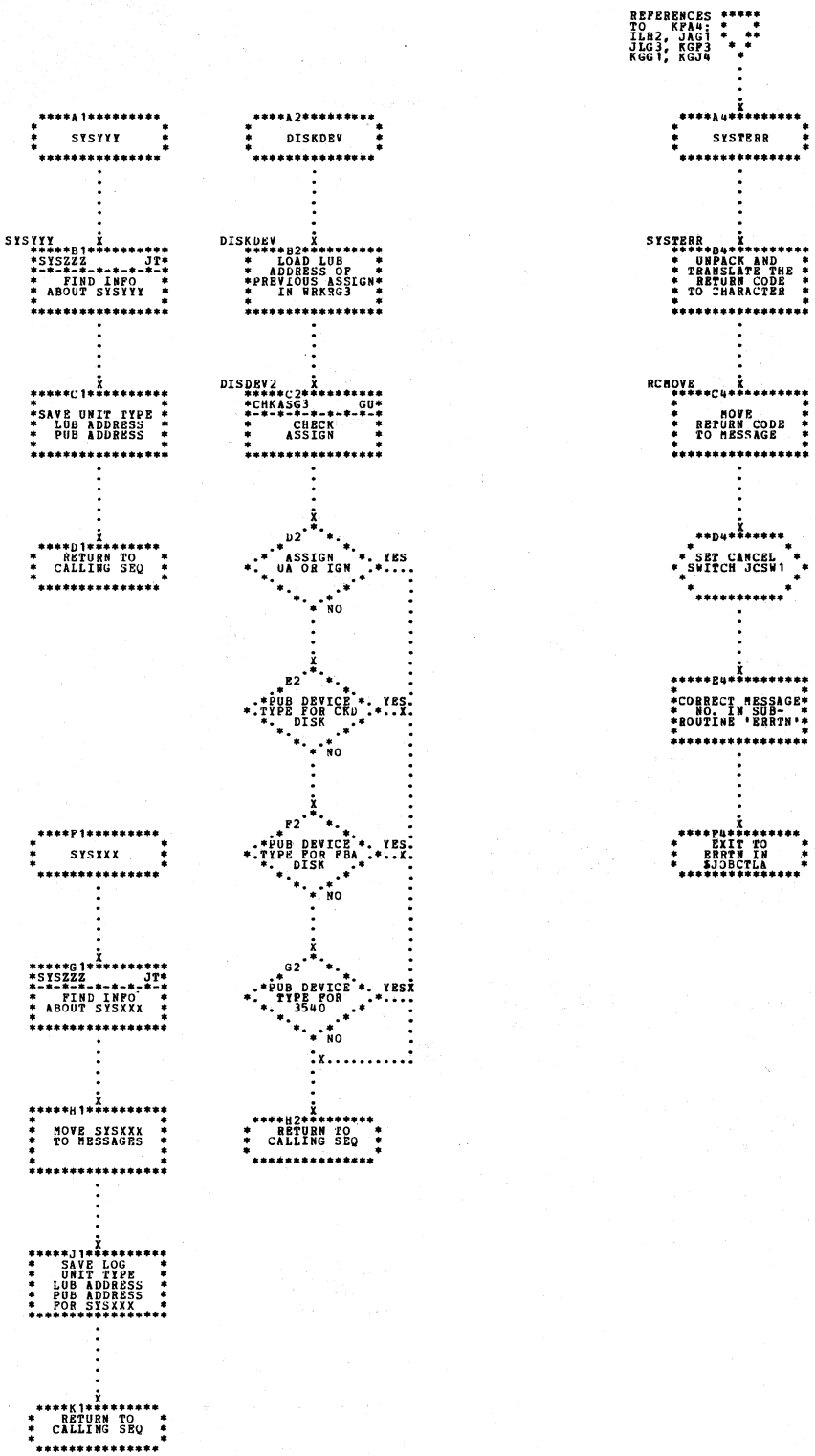


Chart KG. $JOBCTLD - Subroutines
Refer to Charts 10 to 13.

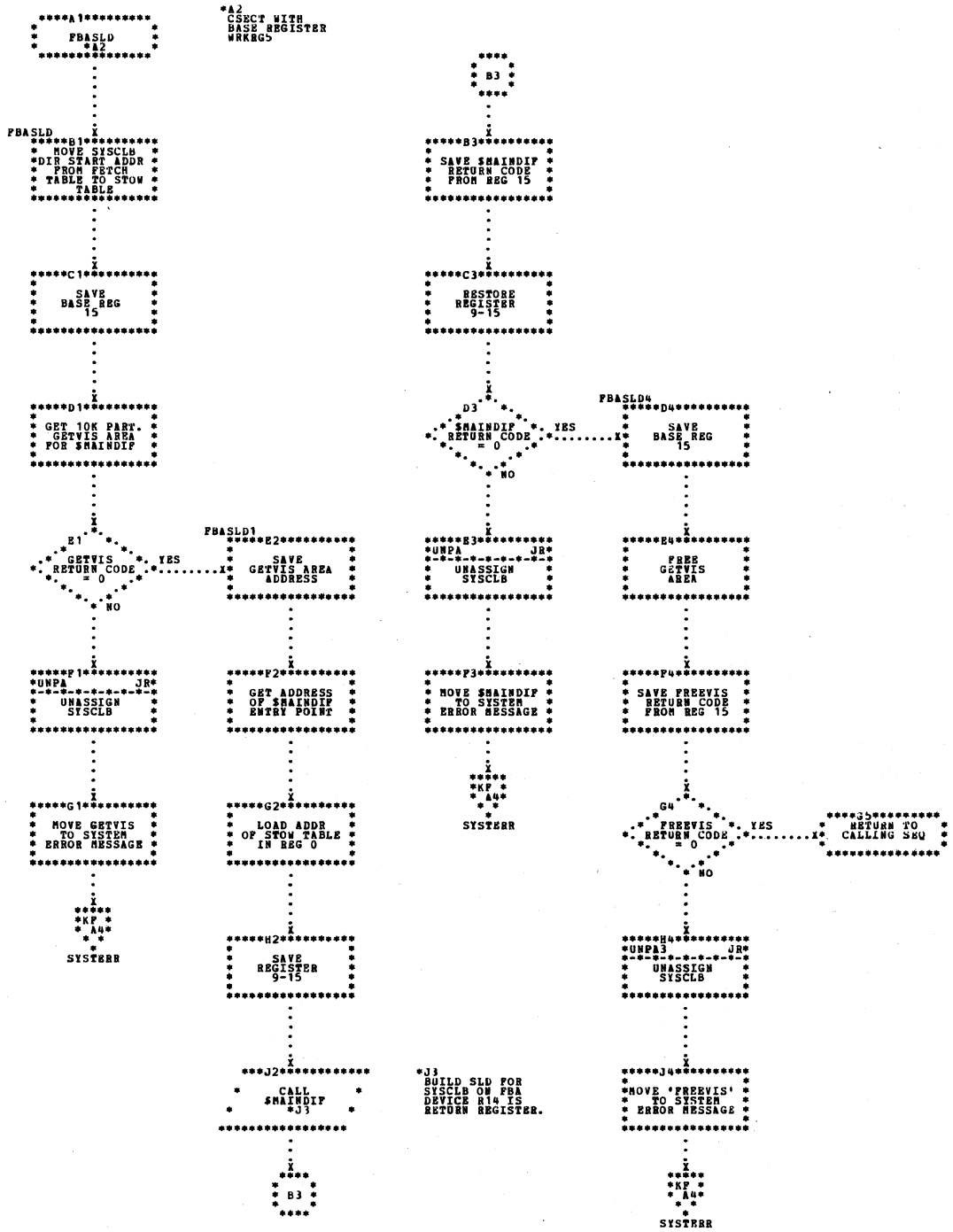


Chart LA. $JOBCTLE - EXEC Statement Processor (Part 1 of 6)
Refer to Chart 14.

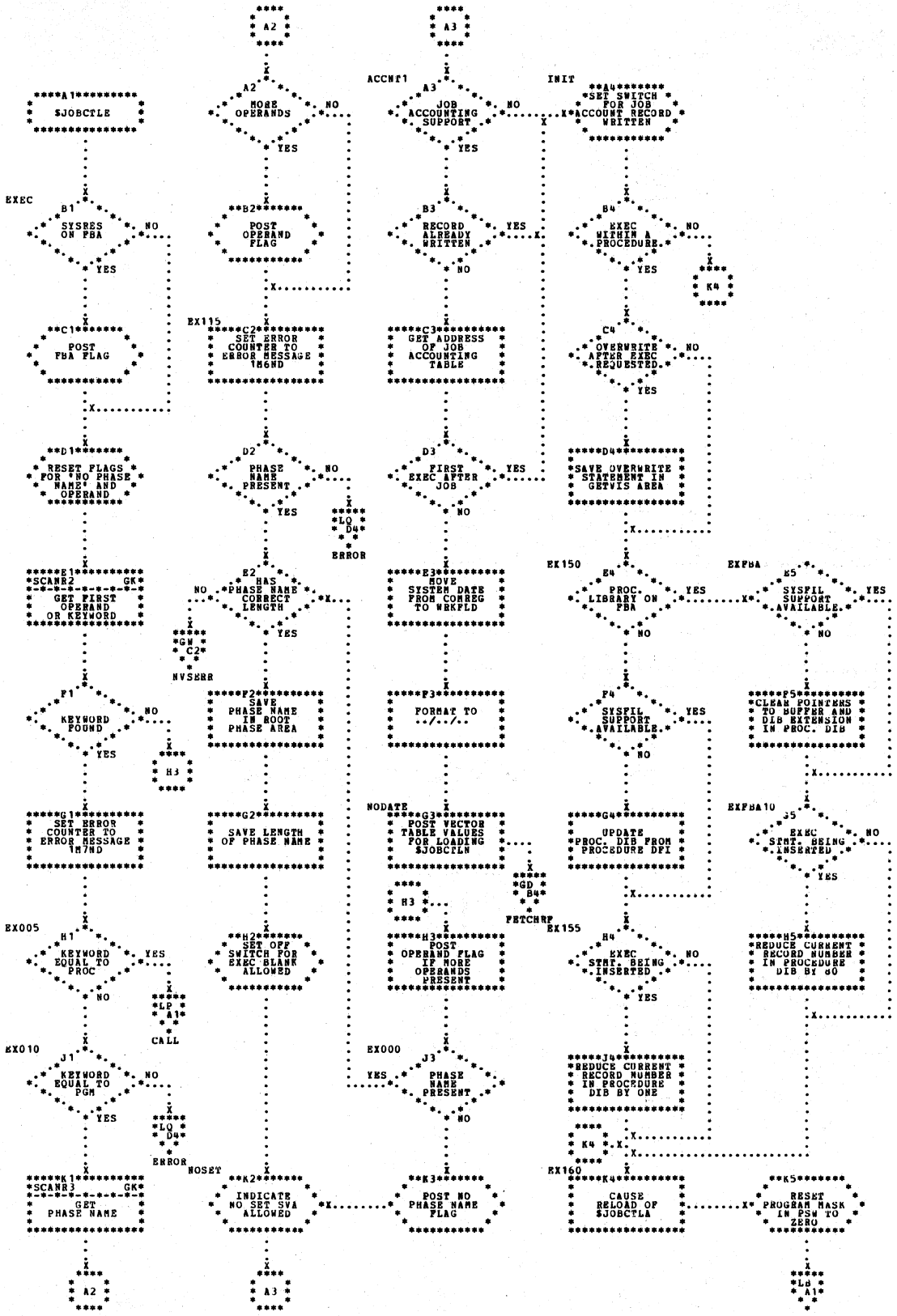


Chart LB. $JOBCTLE - EXEC Statement Processor (Part 2 of 6)
Refer to Chart 14.

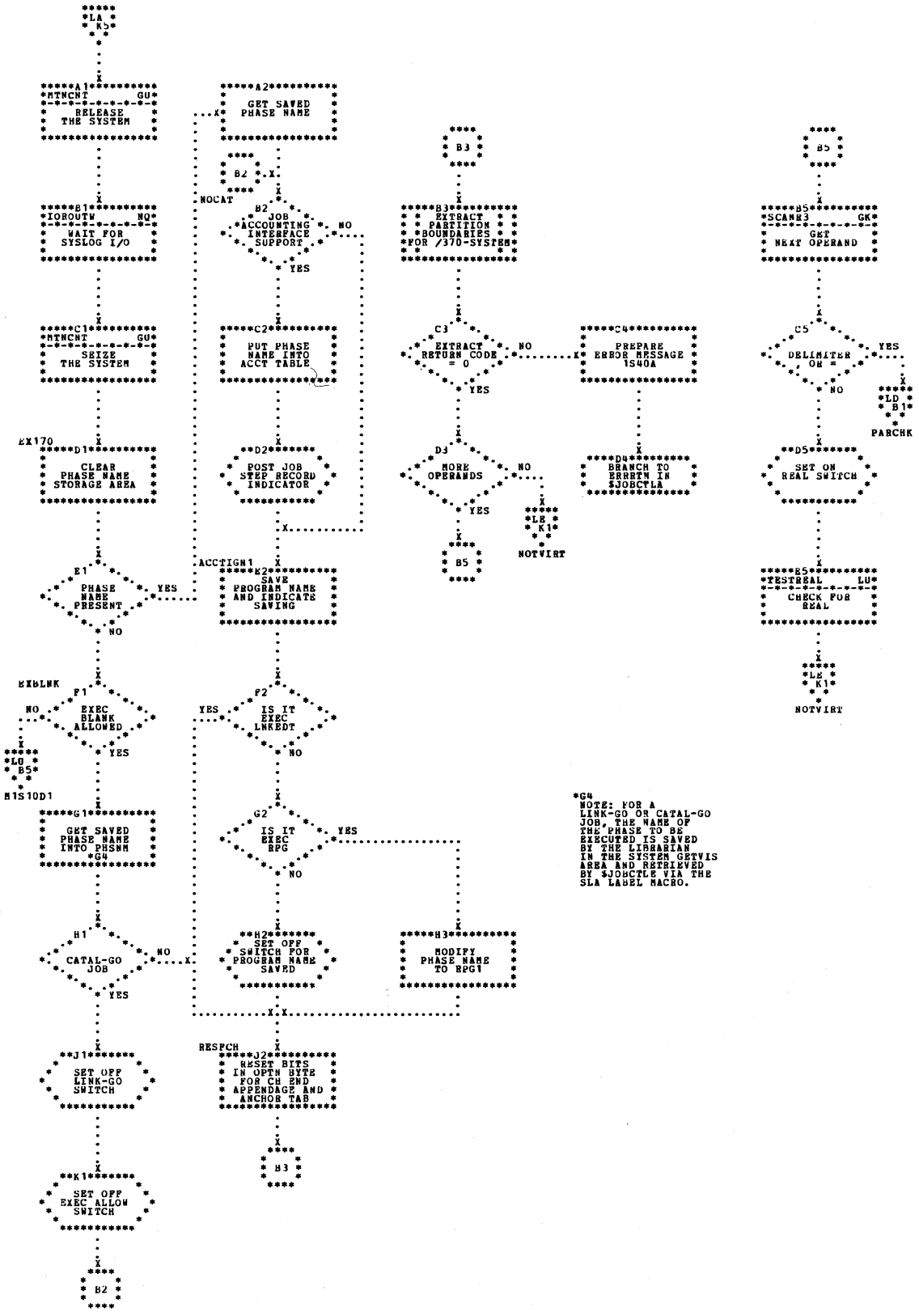


Chart LD. $JOBCTLE - EXEC Statement Processor (Part 3 of 6)
Refer to Chart 14.

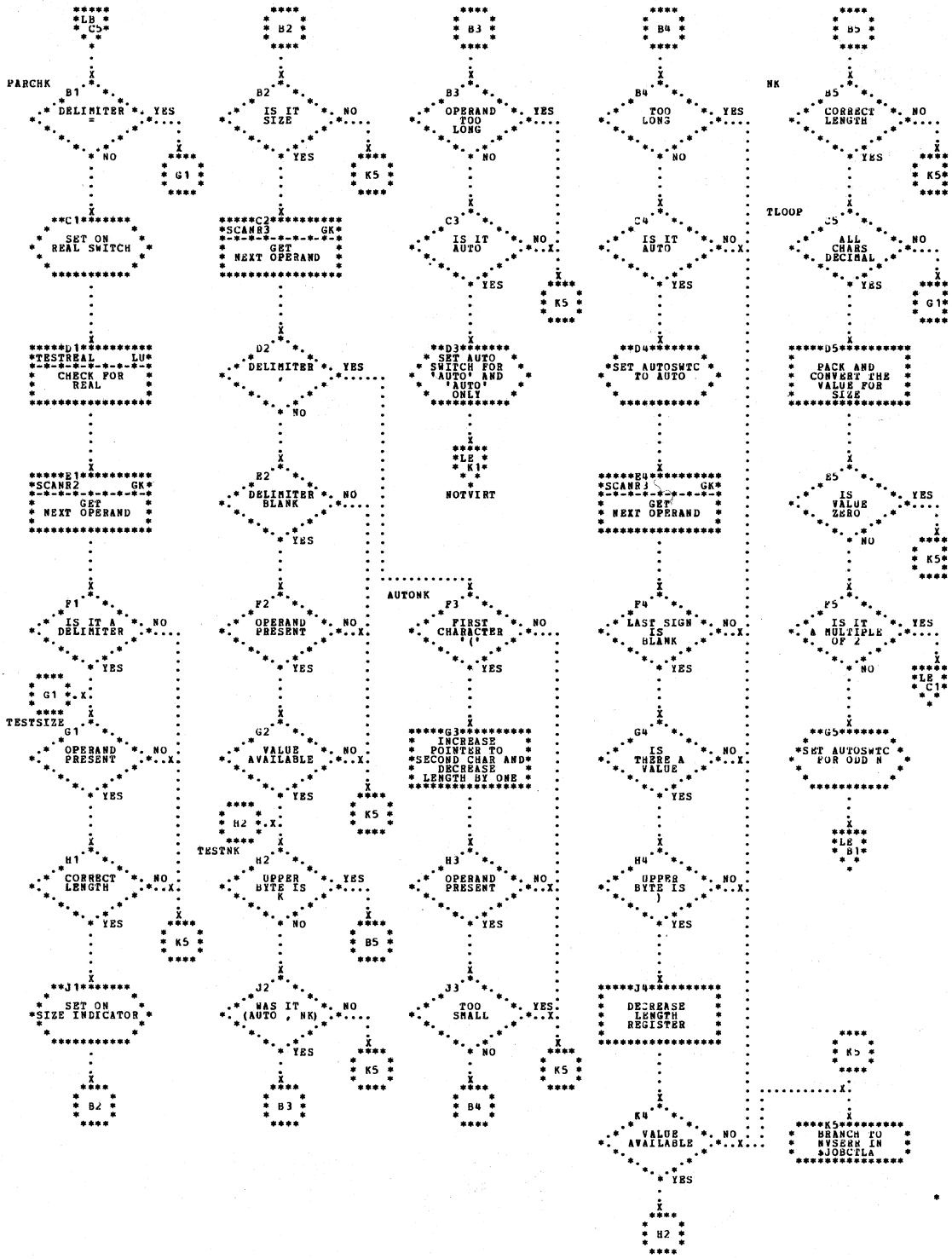


Chart LE. $JOBCTLE - EXEC Statement Processor (Part 4 of 6)
Refer to Chart 14.

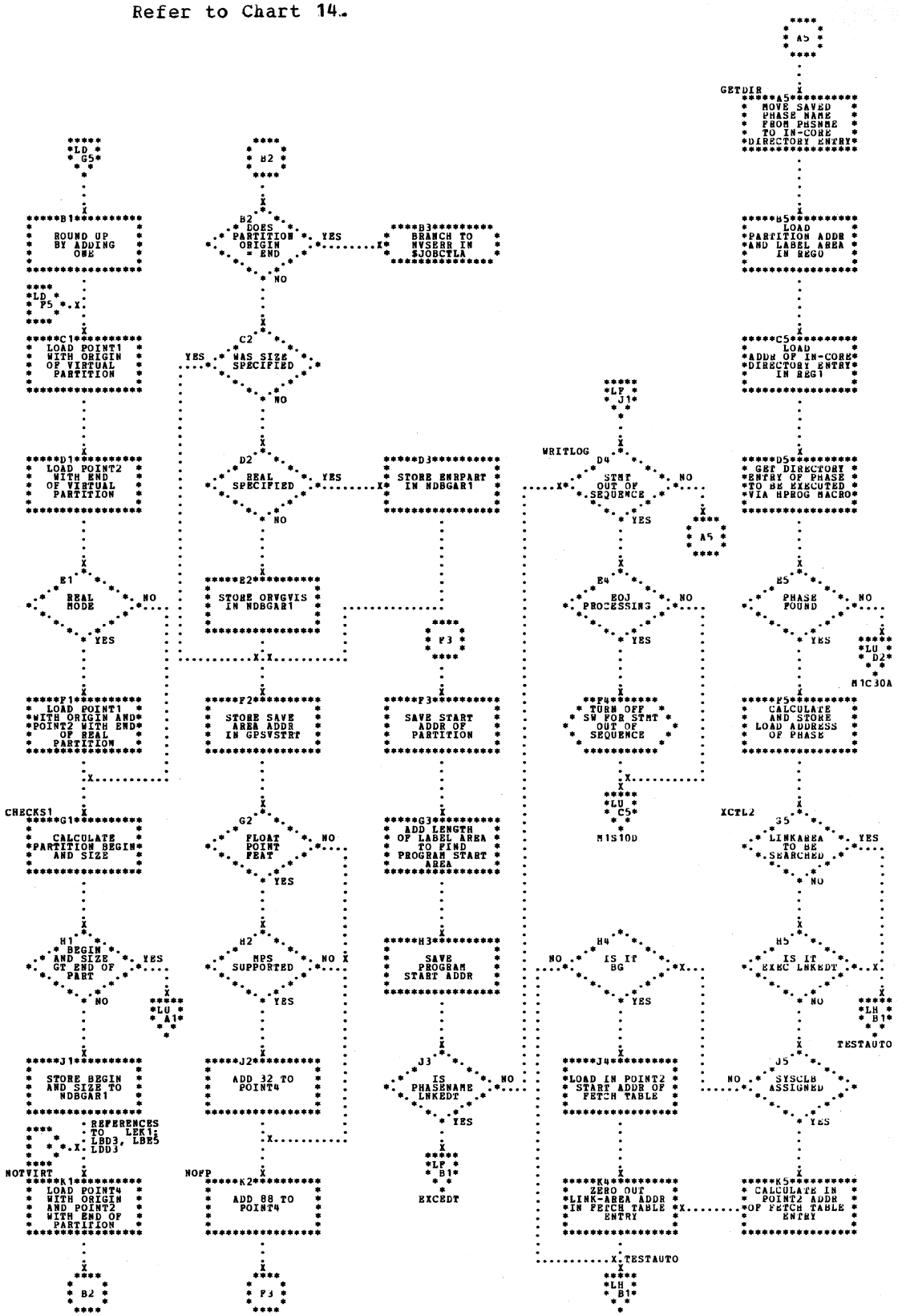


Chart LF. $JOBCTLE - EXEC Statement Processor (Part 5 of 6)
Refer to Chart 14.

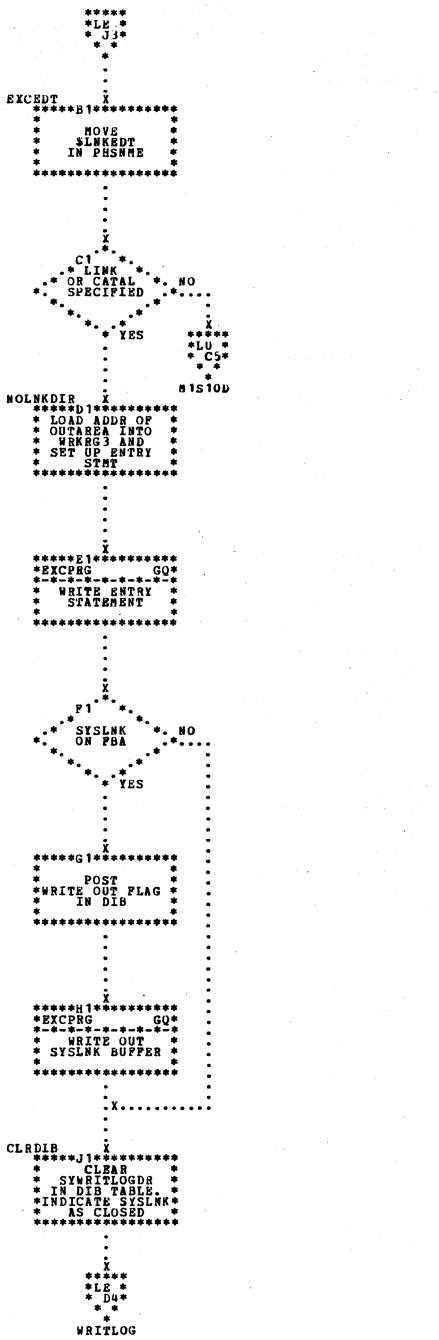


Chart LH. $JOBCTLE - EXEC Statement Processor (Part 6 of 6)
Refer to Chart 14.

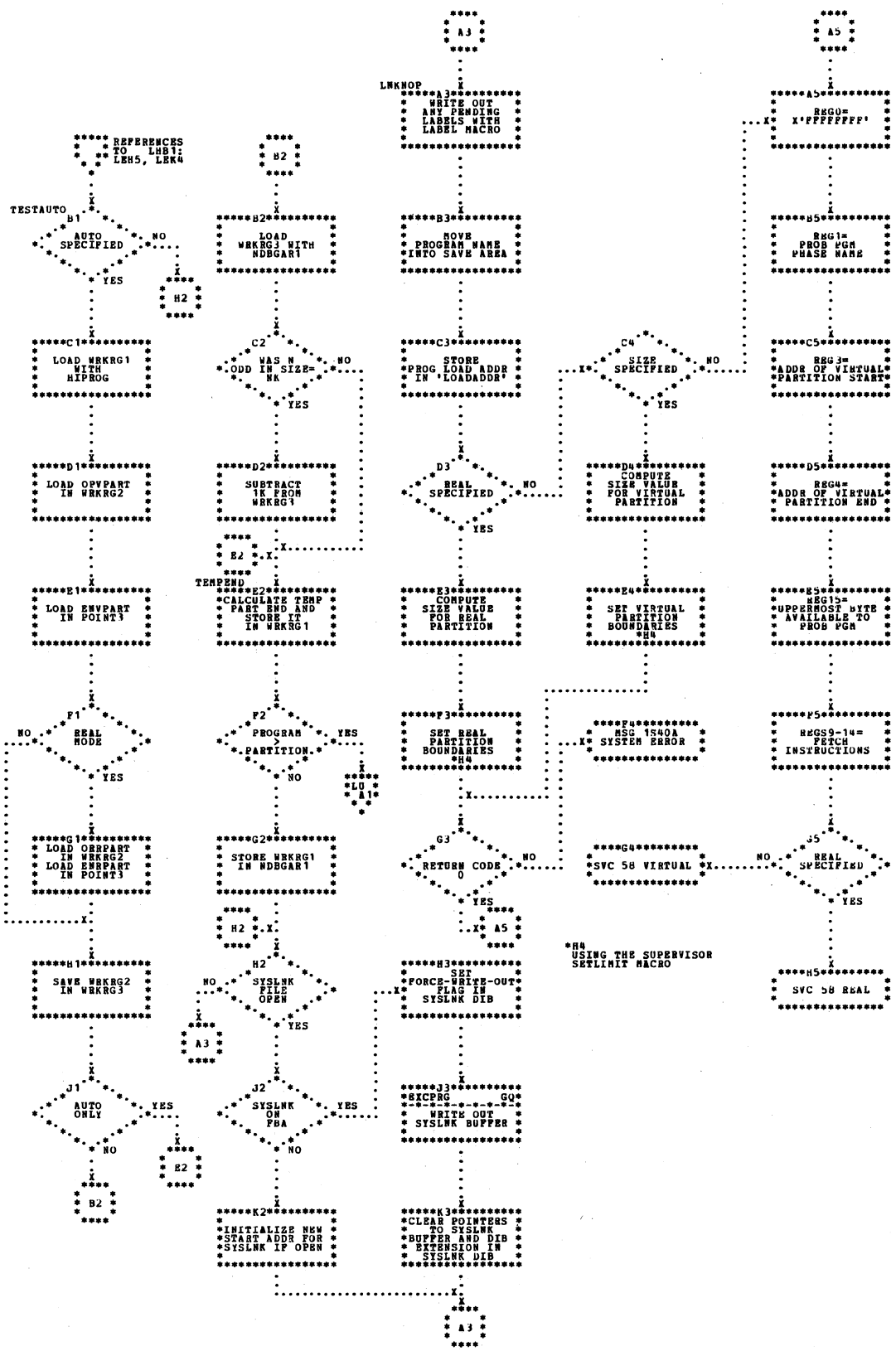


Chart LP. $JOBCTLE - EXEC PROC= Statement Processor
Refer to Chart 14.

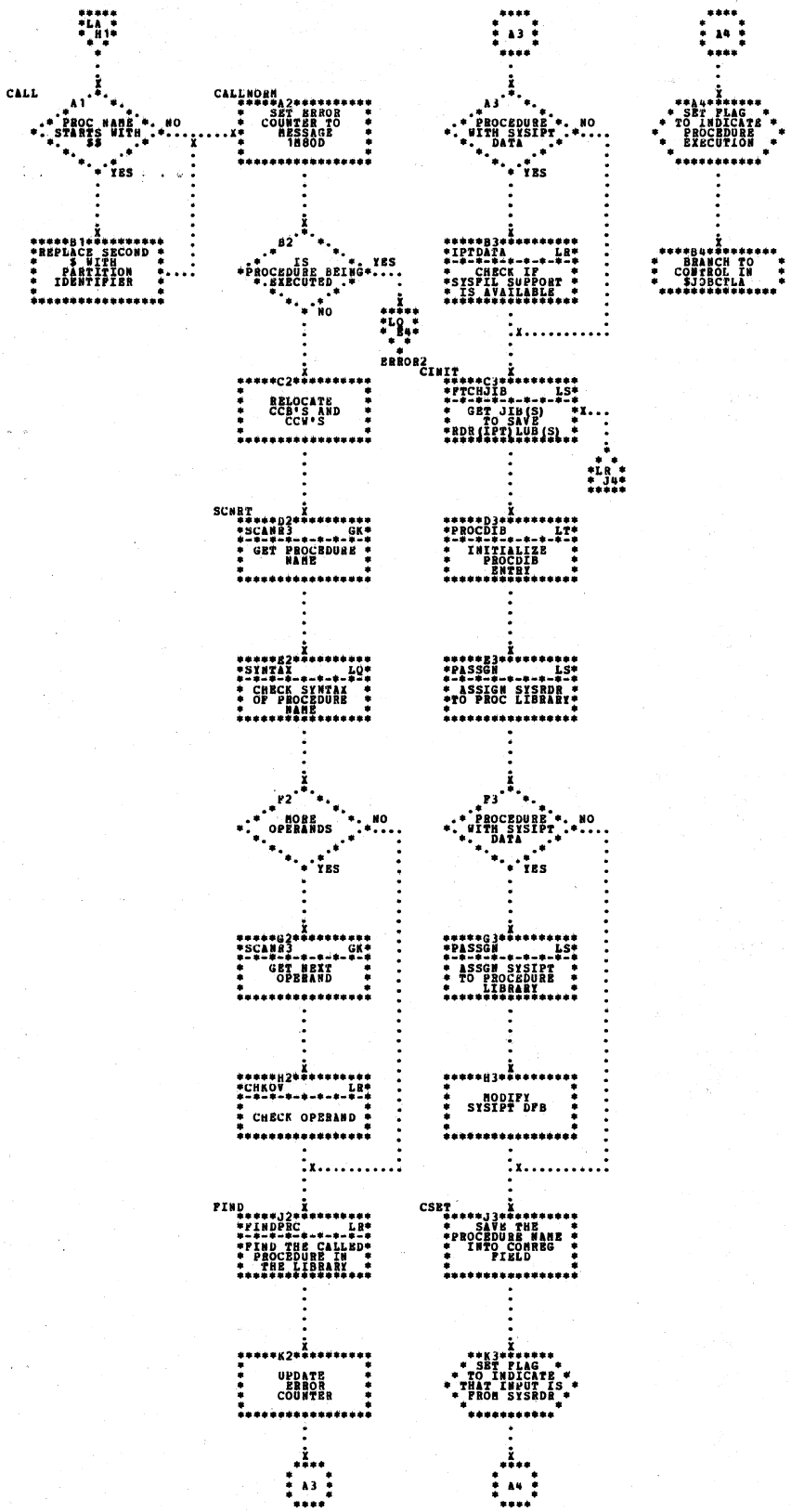


Chart LQ. $JOBCTLE - Syntax Checking Subroutine
Refer to Chart 14.

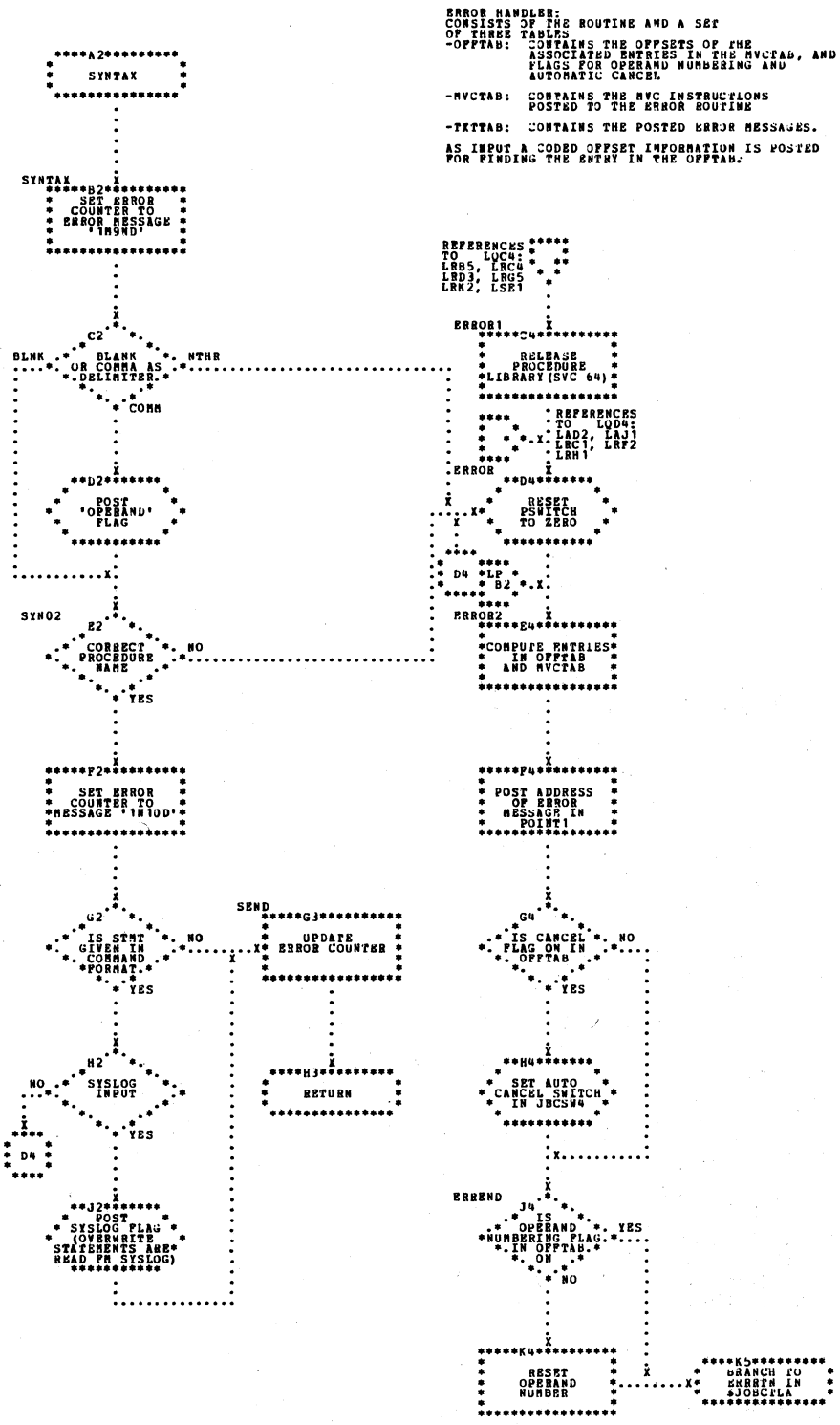


Chart LR. $JOBCTLE - Subroutines
Refer to Chart 14.

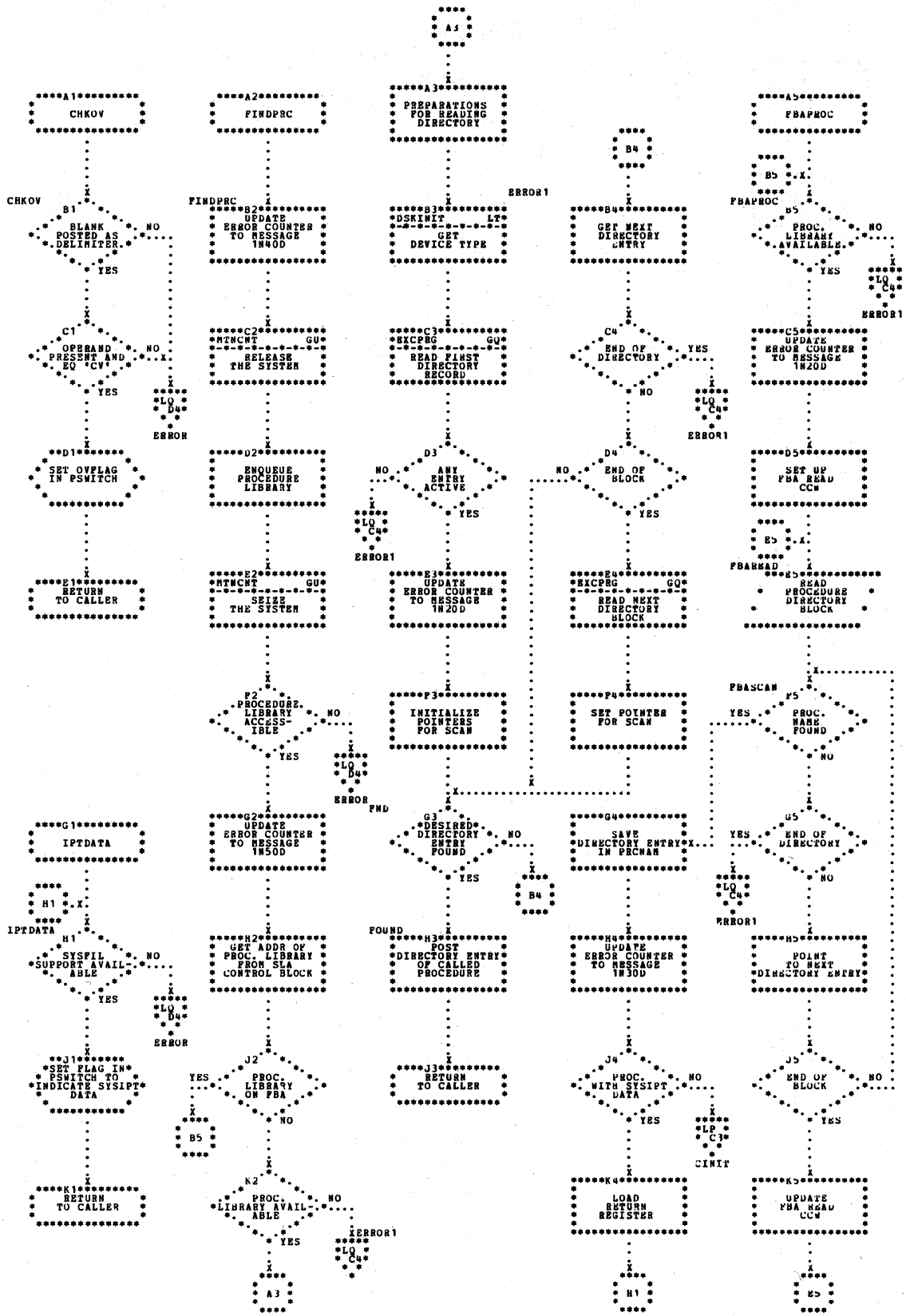


Chart LS. $JOBCTLE - Subroutines
Refer to Chart 14.

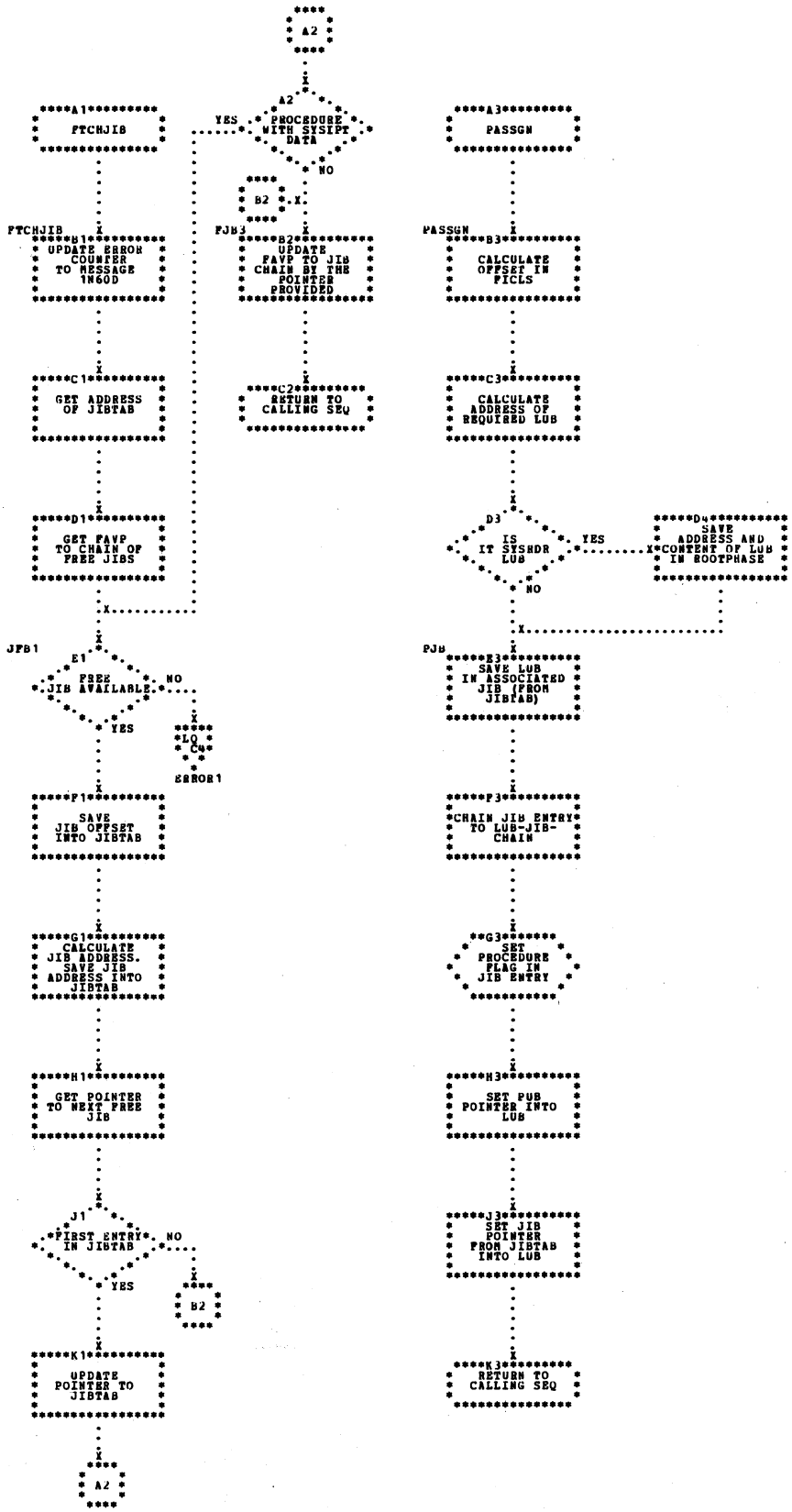


Chart LT. $JOBCTLE - Subroutines
Refer to Chart 14.

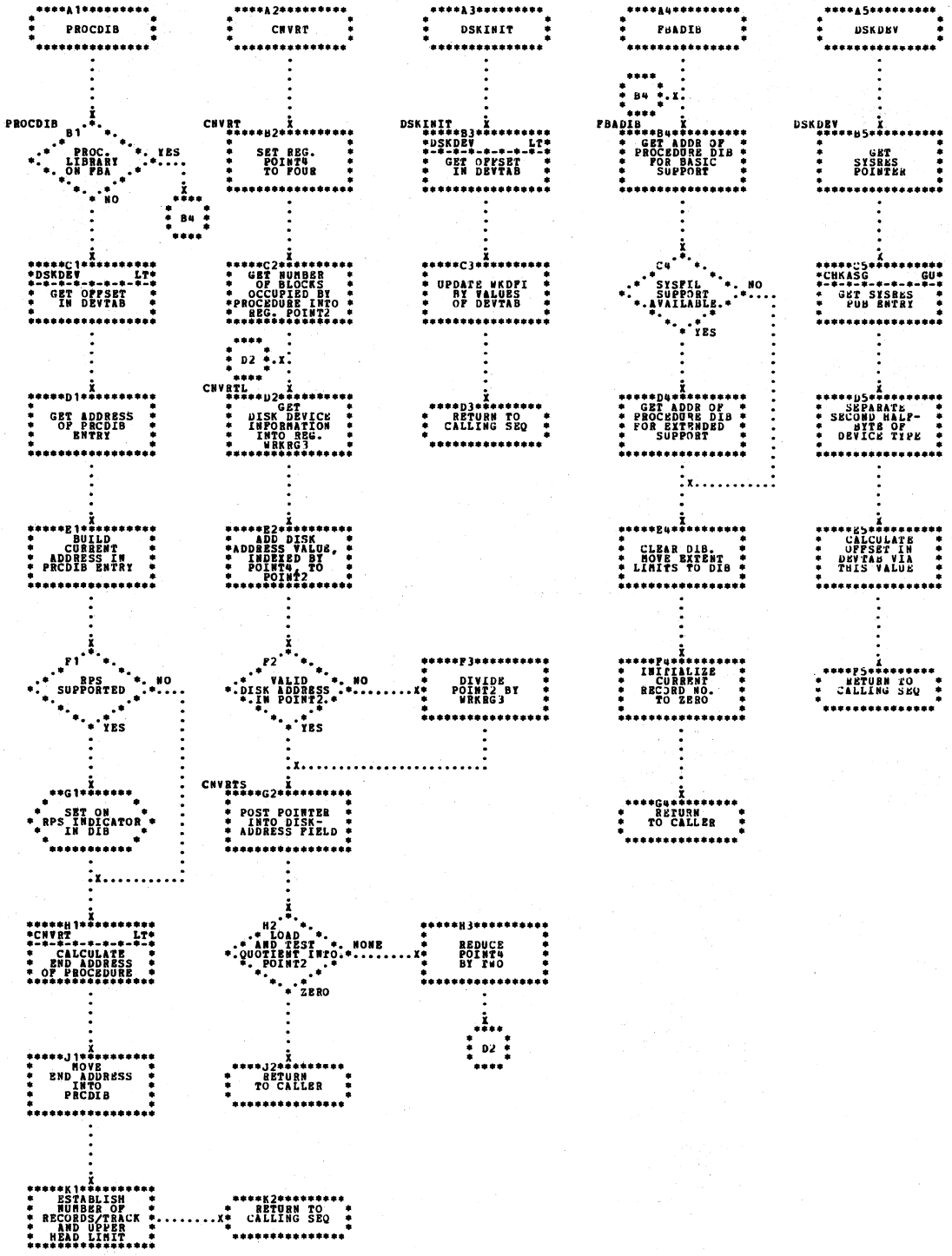


Chart LU. $JOBCTLE - Error Subroutines
Refer to Chart 14.

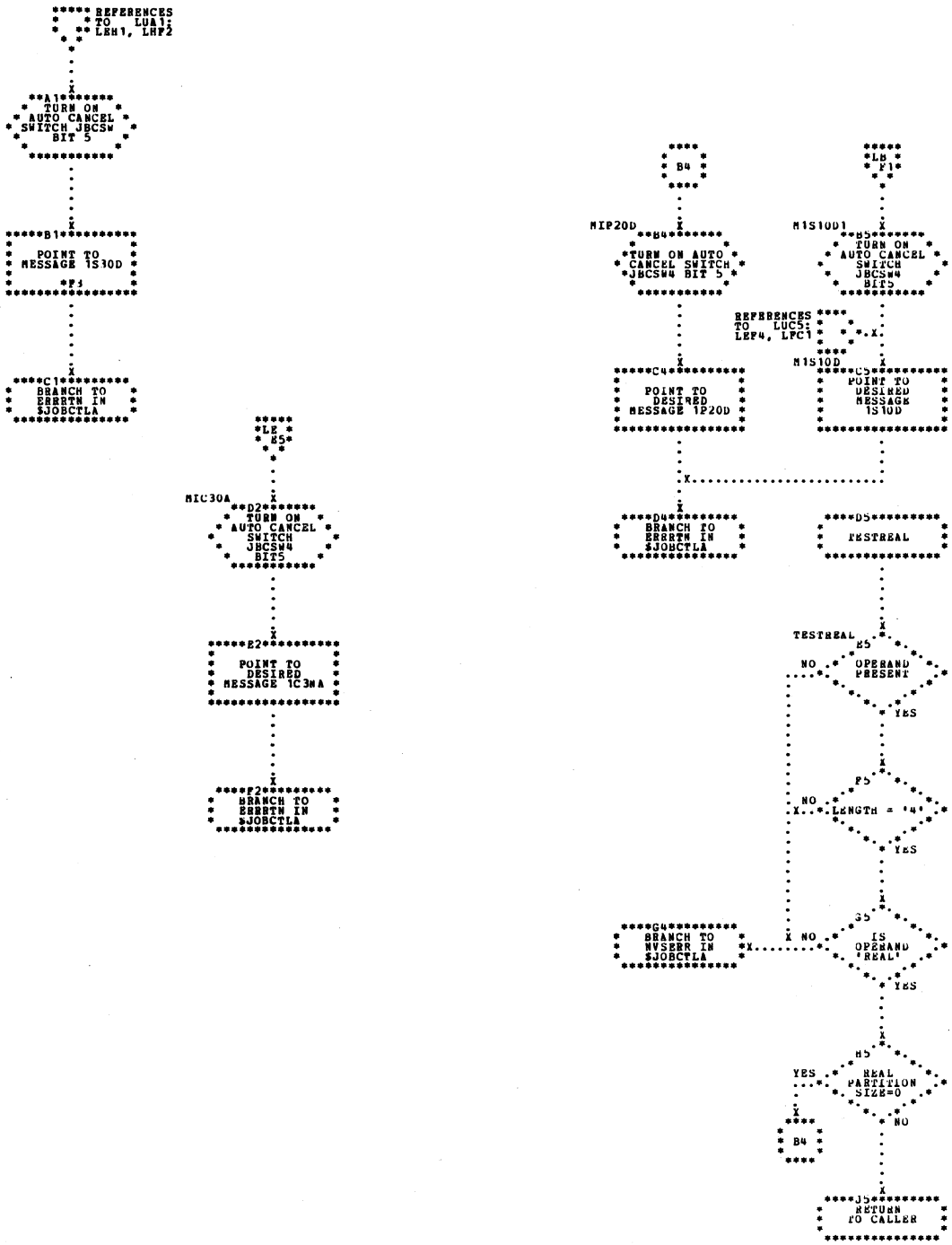


Chart MA. $JOBCTLF - LISTIO Statement Processor (Part 1 of 5)
Refer to Chart 15.

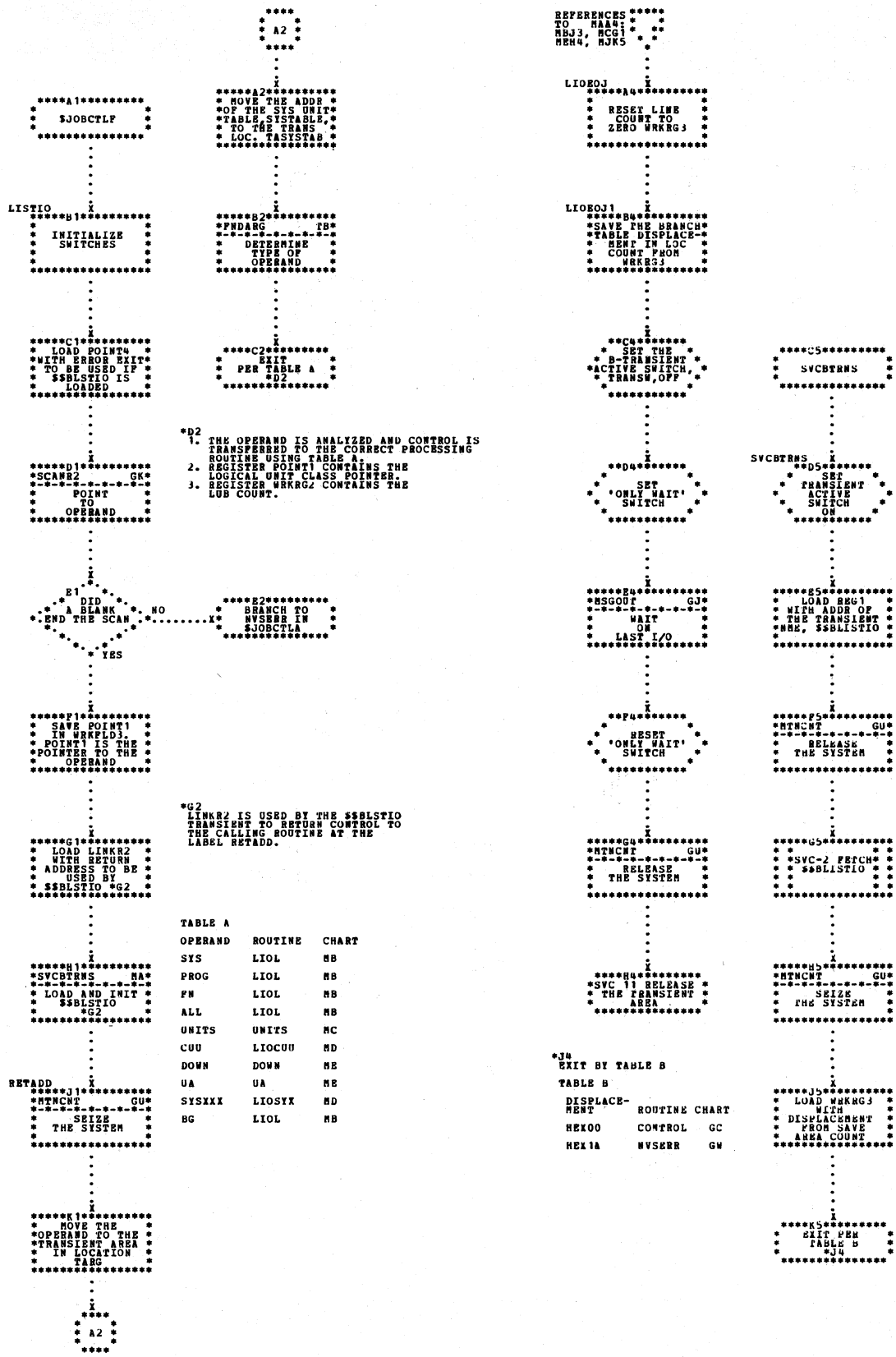


Chart MB. $JOBCTLF - LISTIO Statement Processor (Part 2 of 5)
Refer to Chart 15.

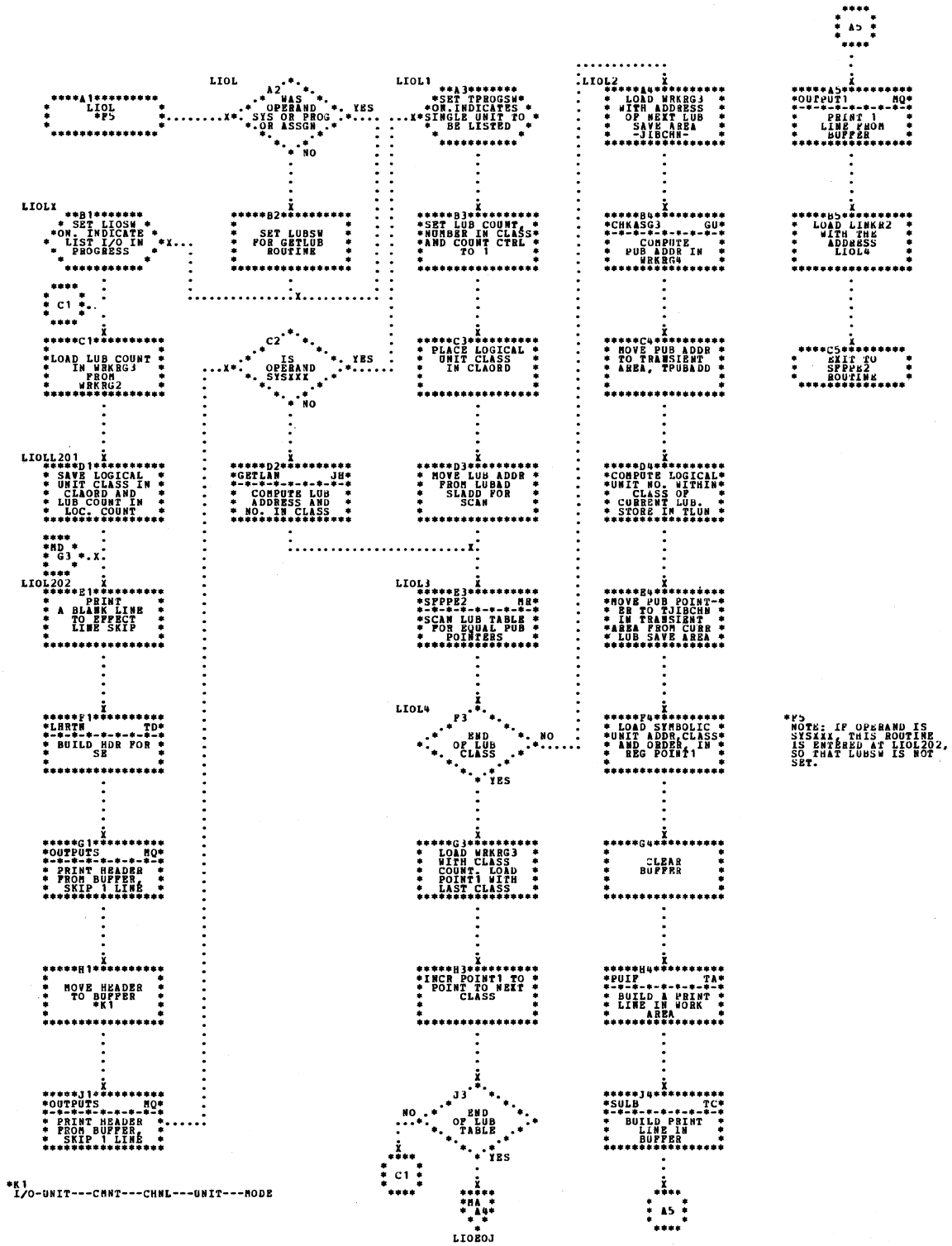


Chart MC. &JOBCTLF - LISTIO Statement Processor (Part 3 of 5)
Refer to Chart 15.

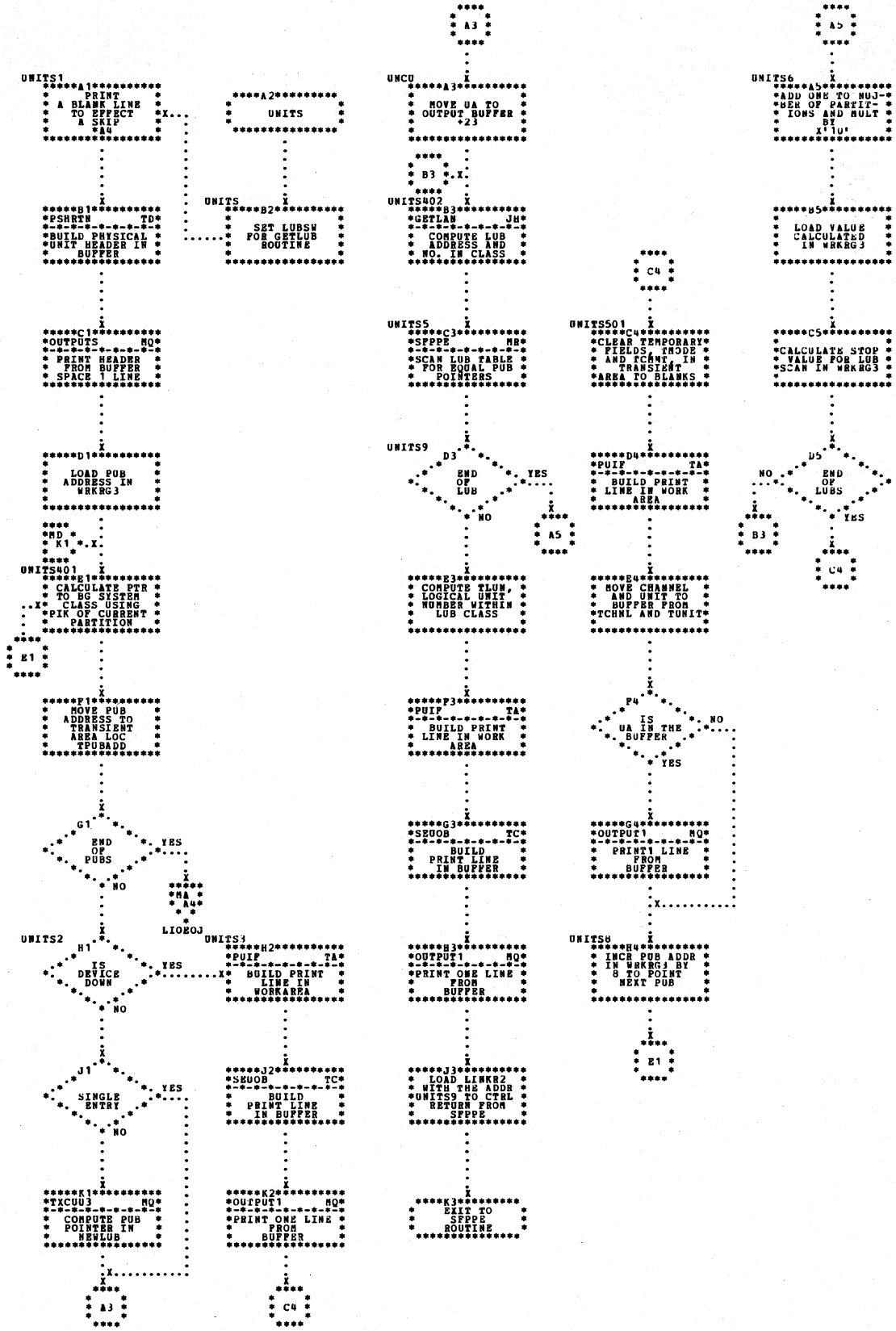


Chart MD. $JOBCTLF - LISTIO Statement Processor (Part 4 of 5)
Refer to Chart 15.

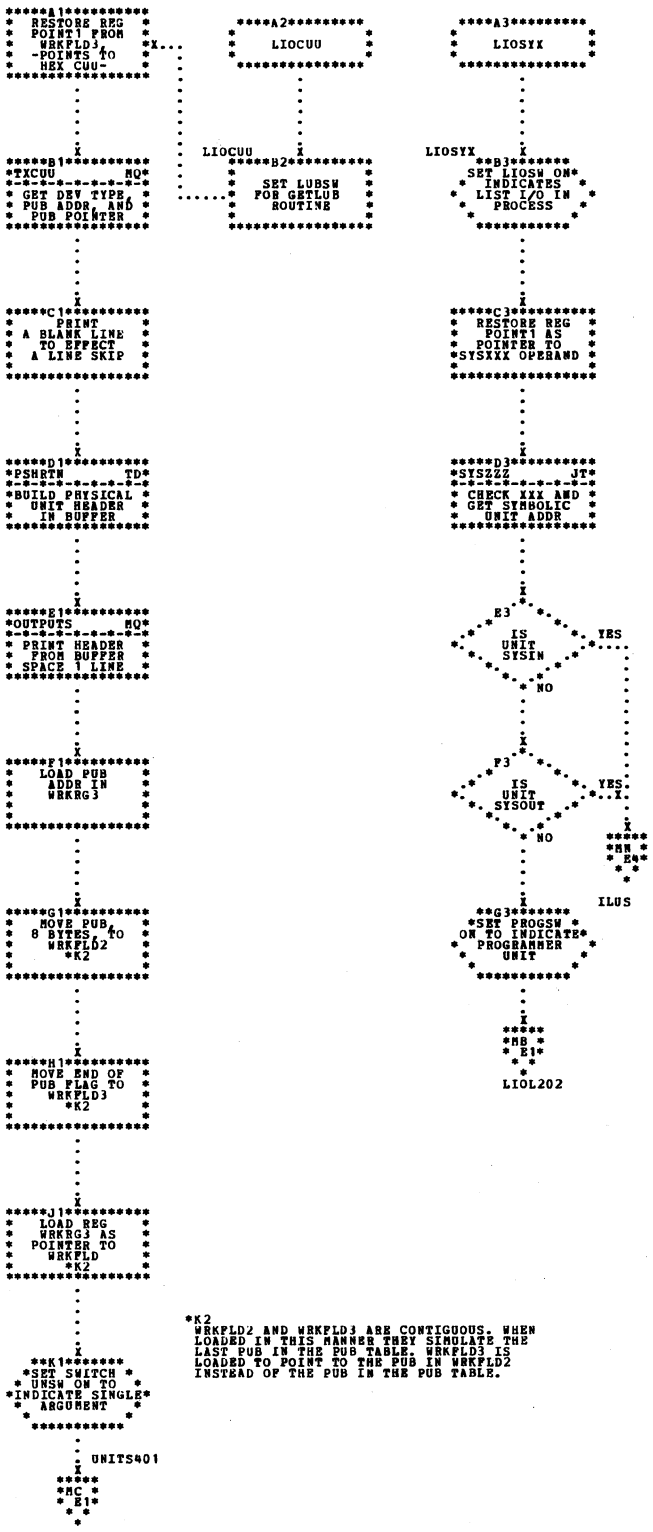


Chart ME. **$JOBCTLF** - LISTIO Statement Processor (Part 5 of 5)
Refer to Chart 15.

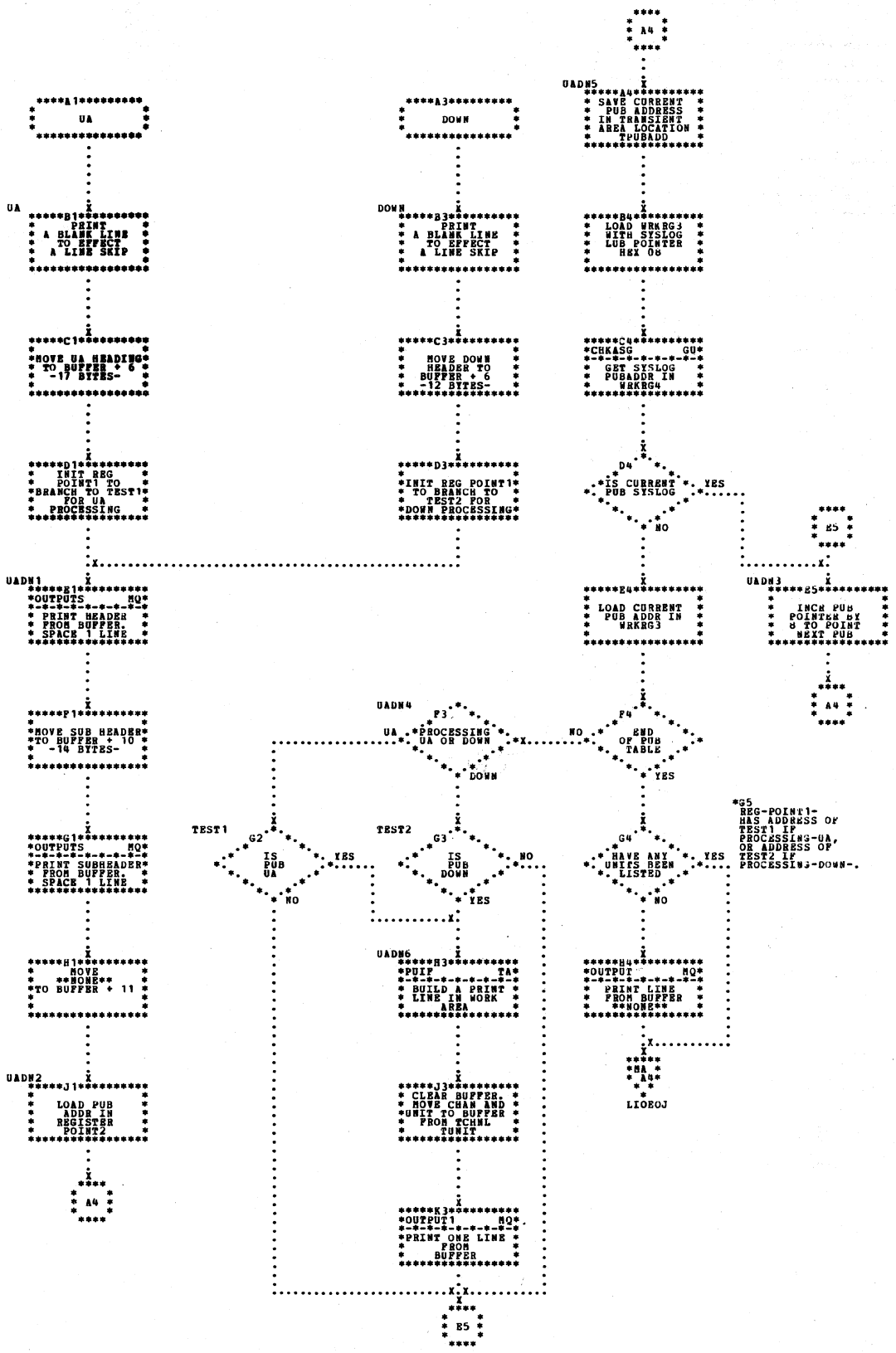


Chart MF. $JOBCTLF - RESET Statement Processor (Part 1 of 2)
Refer to Chart 15.

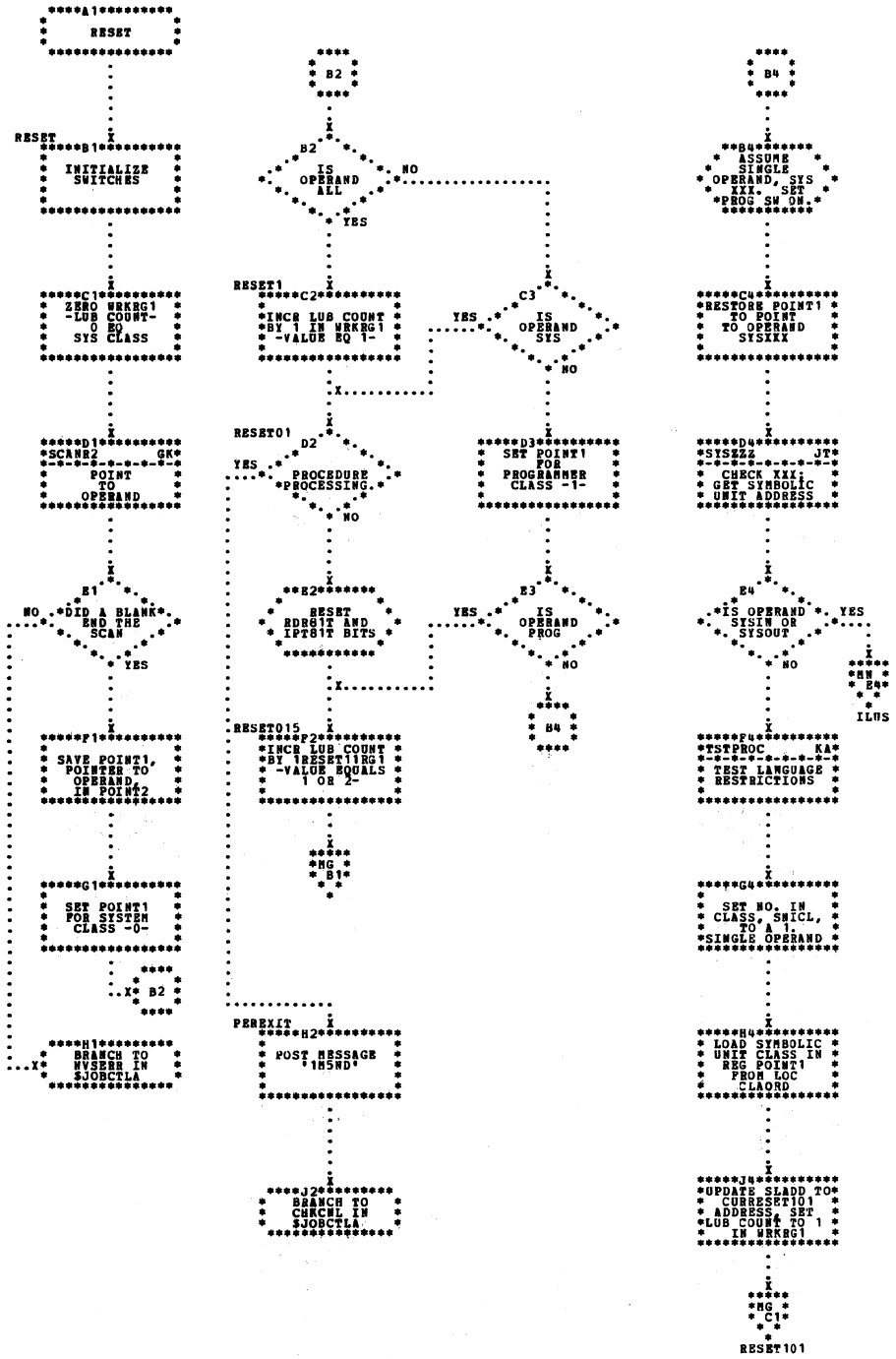


Chart MG. $JOBCTLF - RESET Statement Processor (Part 2 of 2)
Refer to Chart 15.

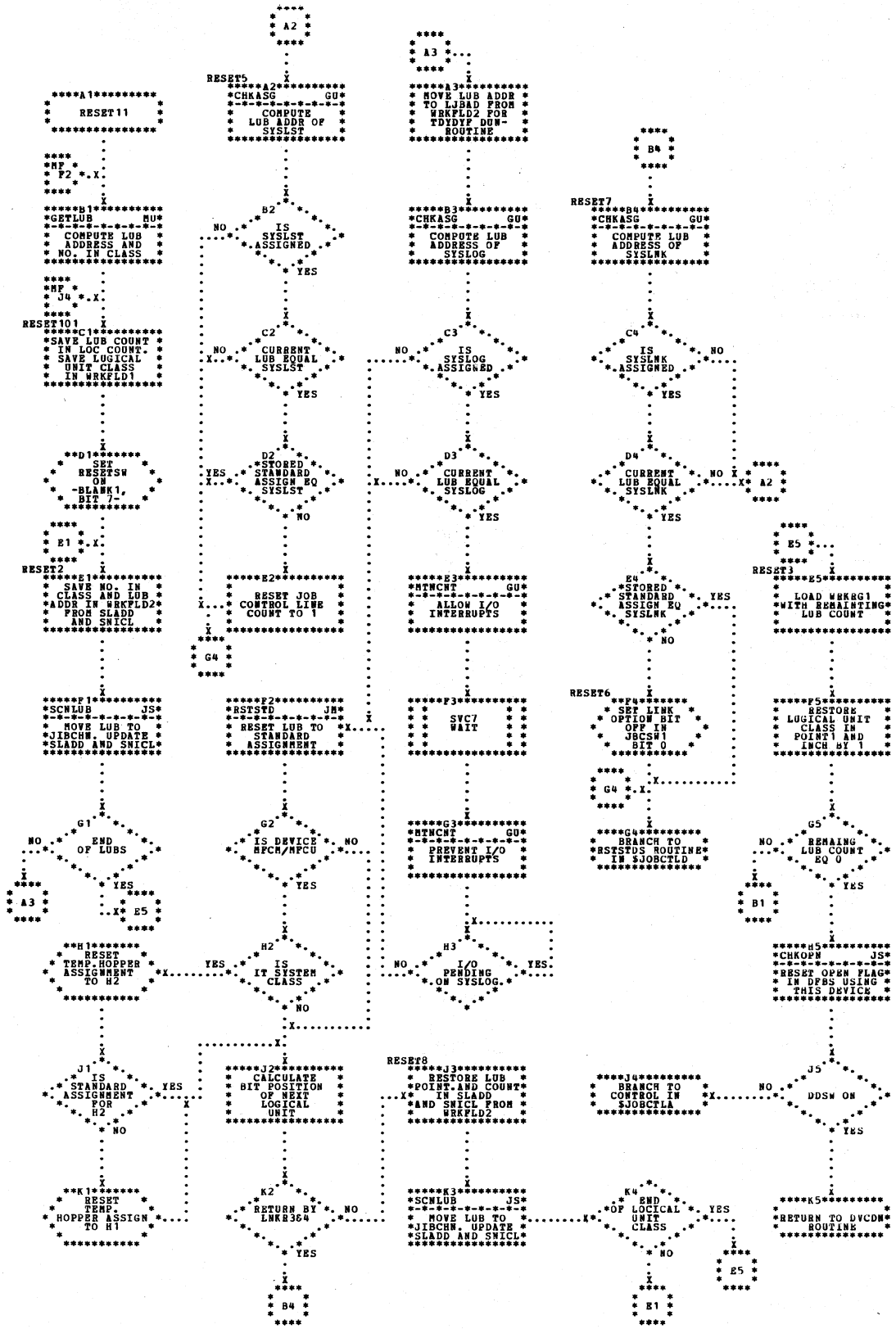


Chart MH. $JOBCTLF - DVCDN Statement Processor (Part 1 of 2)
Refer to Chart 15.

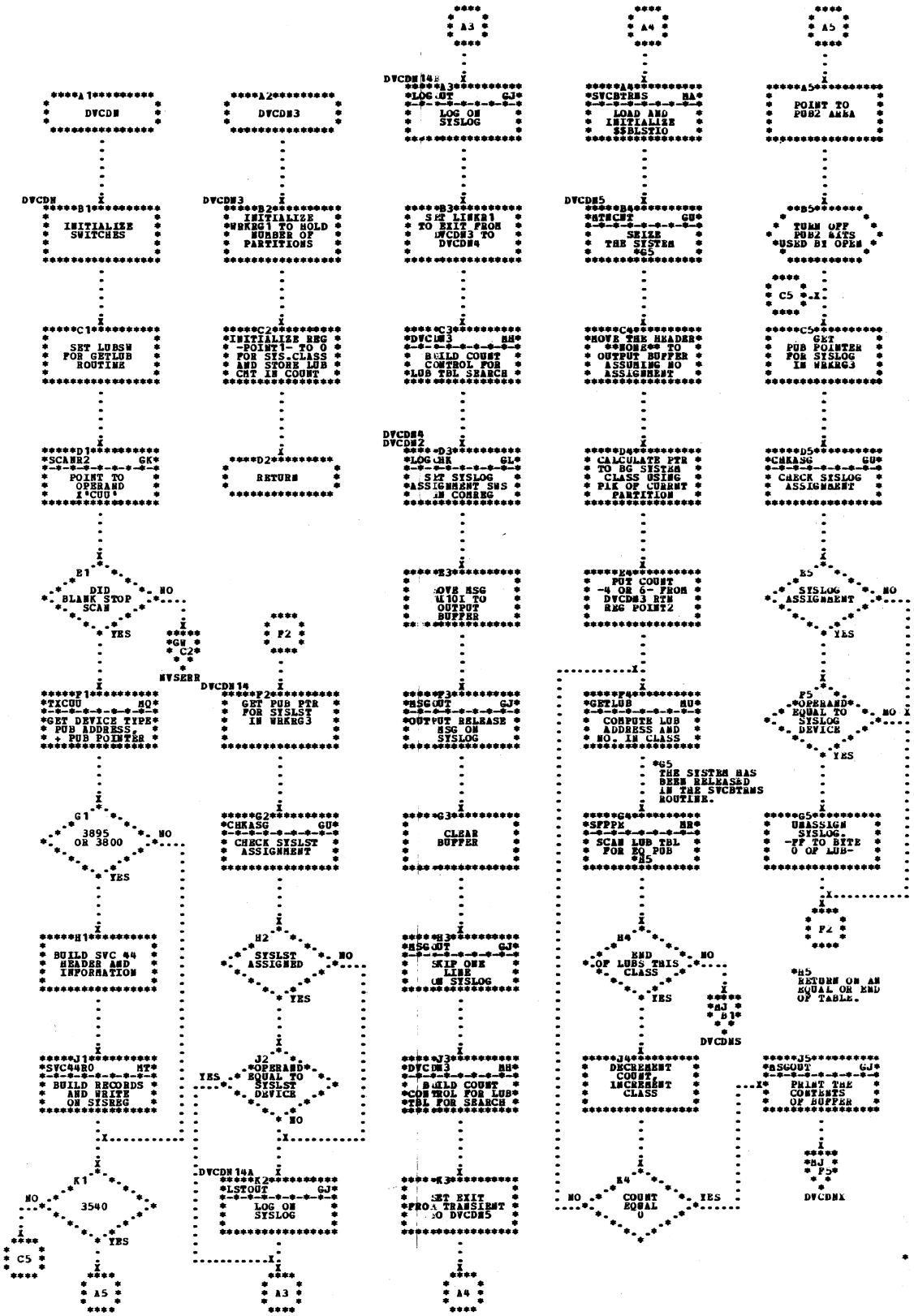


Chart MJ. $JOBCTLF - DVCDN Statement Processor (Part 2 of 2)
Refer to Chart 15.

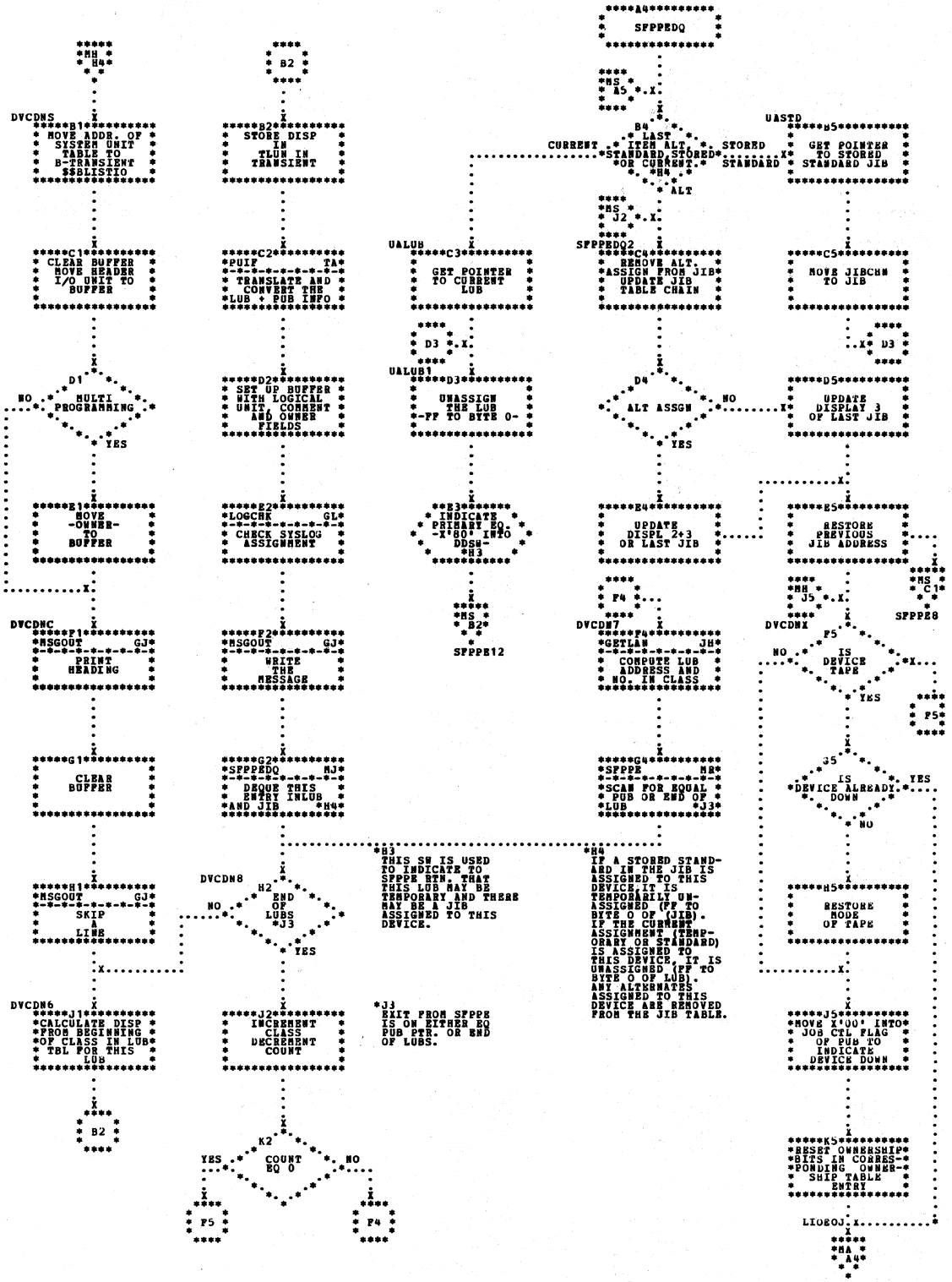


Chart MK. $JOBCTLF - MAP Command Processor
Refer to Chart 15.

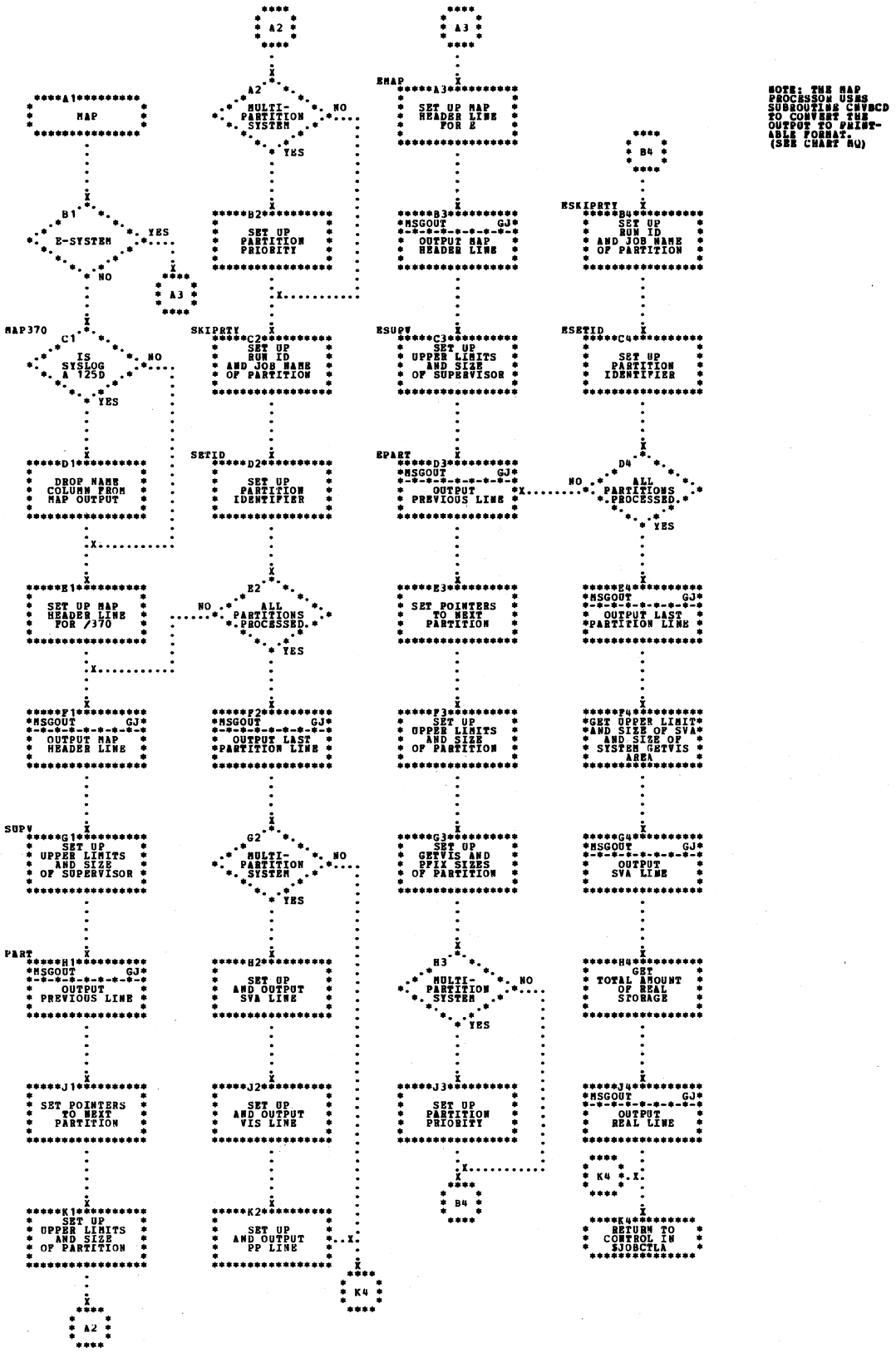


Chart ML. $JOBCTLF - DVCUP Statement Processor
Refer to Chart 15.

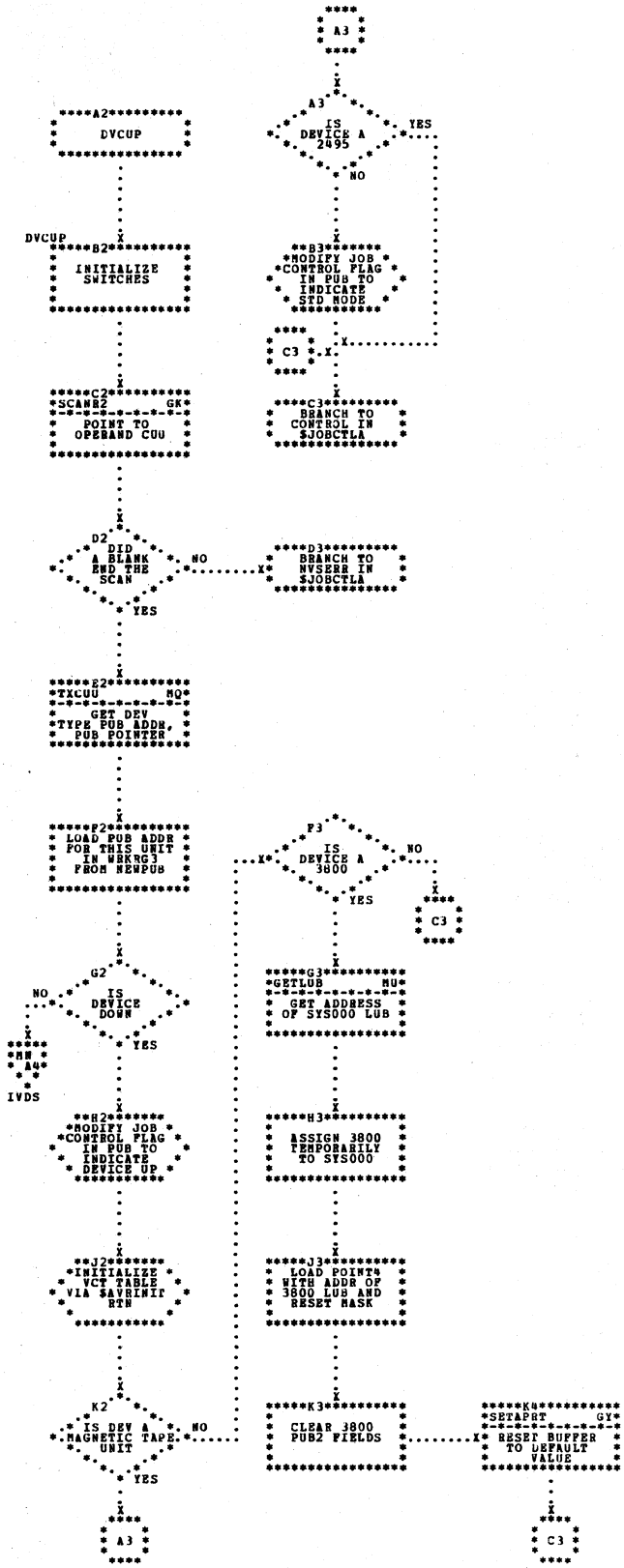


Chart MM. $JOBCTLF - UNBATCH Command Processor
Refer to Chart 15.

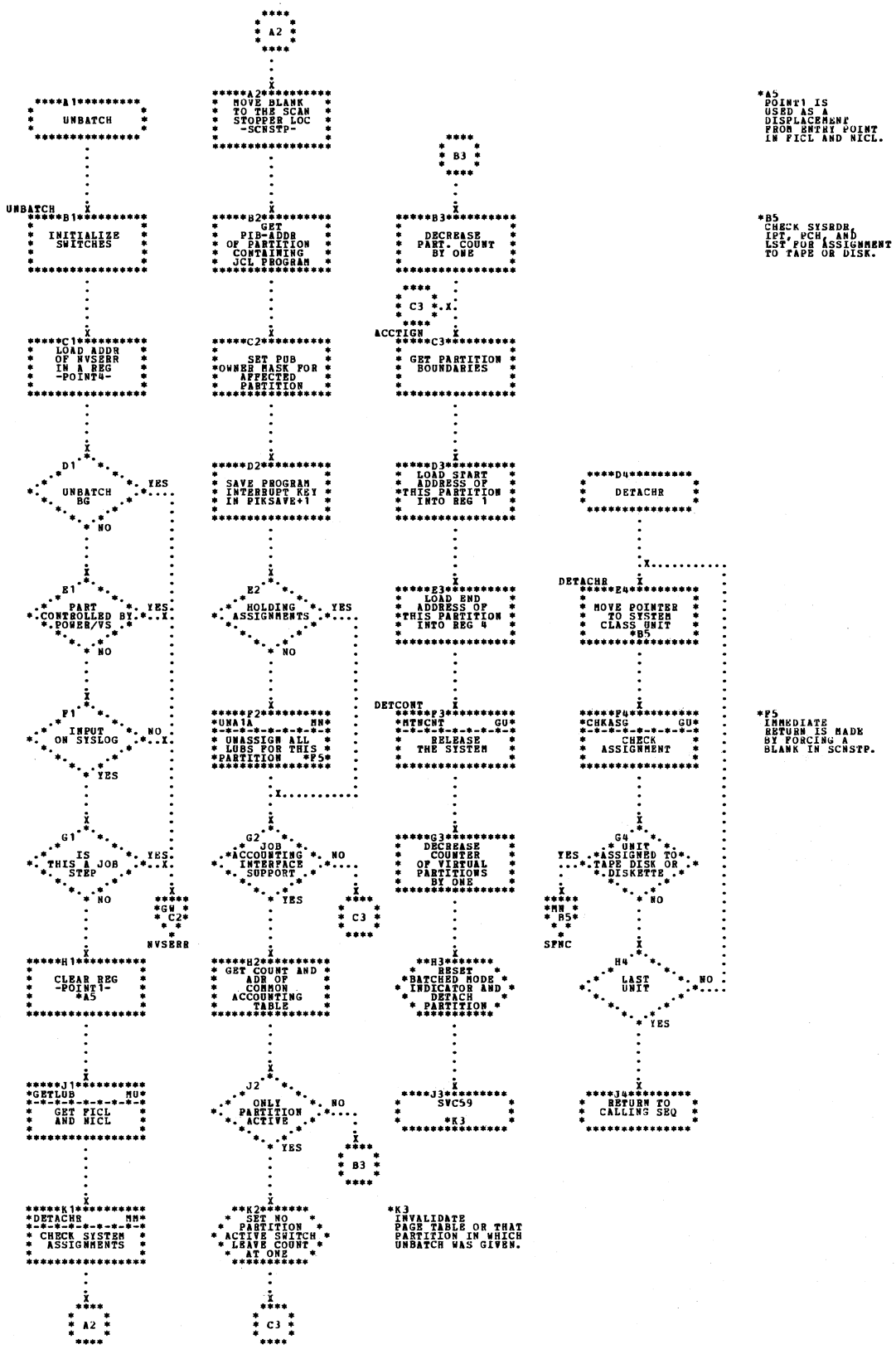


Chart MN. $JOBCTLF - UNA Statement Processor (Part 1 of 2)
Refer to Chart 15.

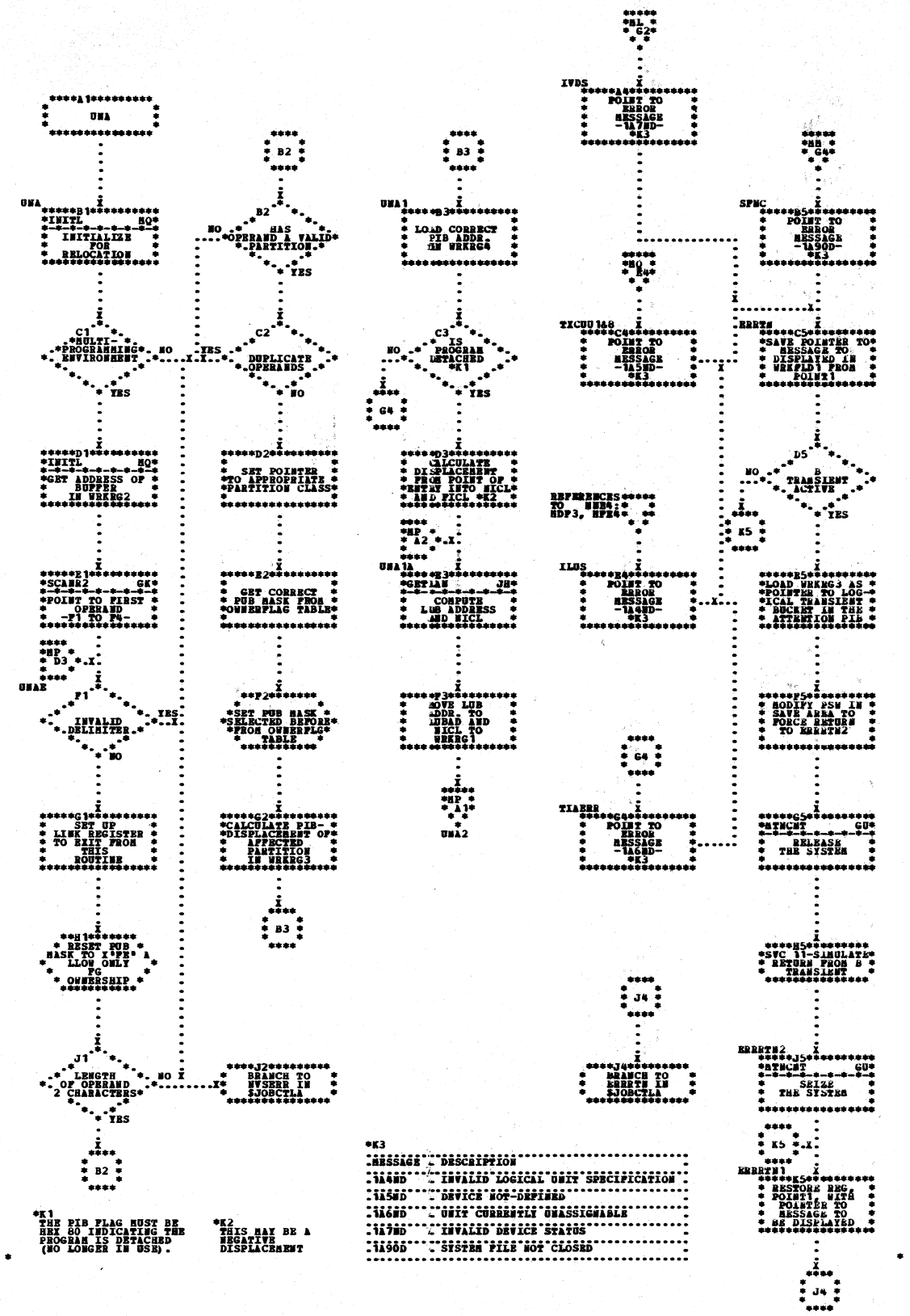


Chart MP. $JOBCTLF - UNA Statement Processor (Part 2 of 2)
Refer to Chart 15.

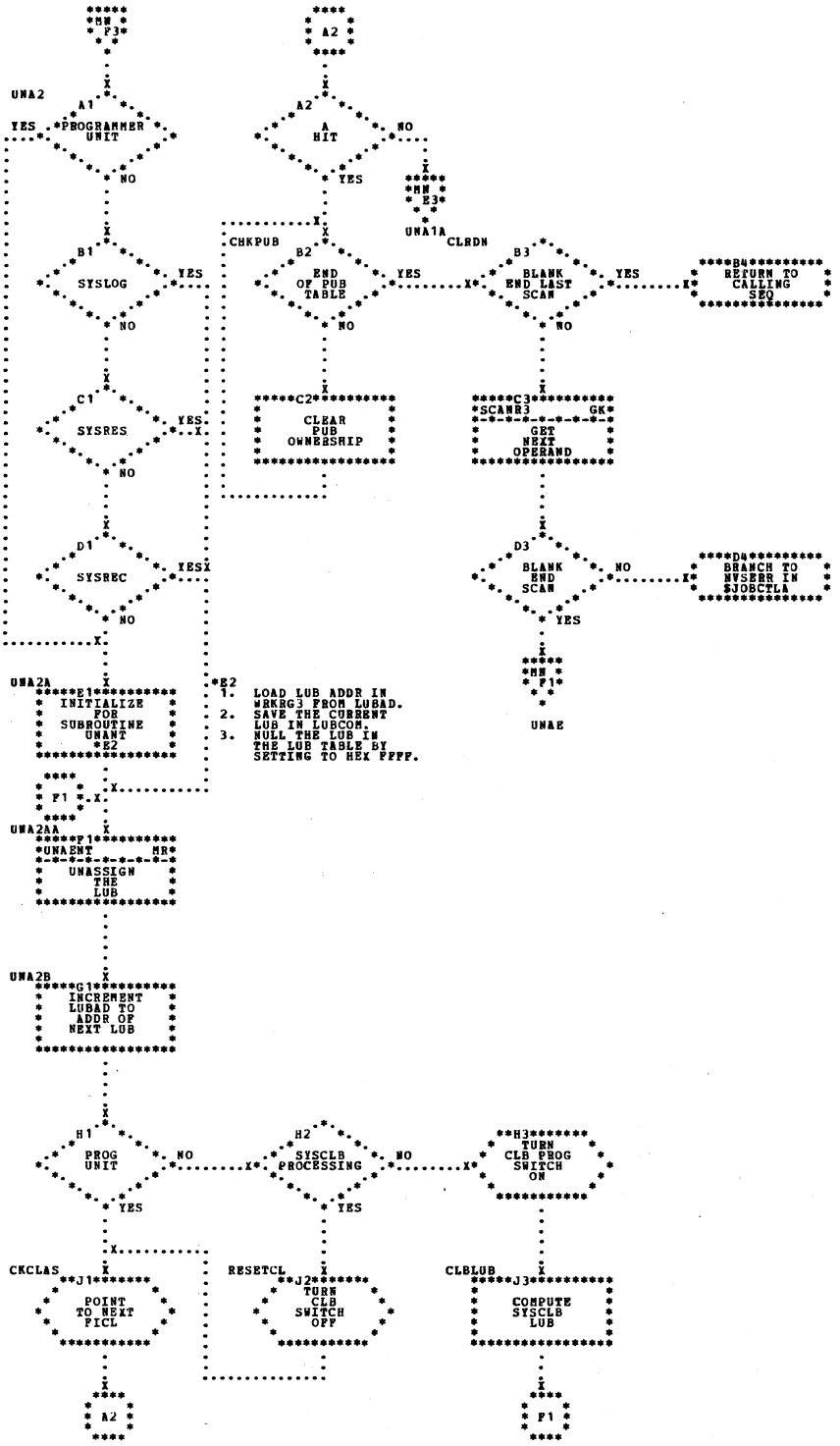


Chart MQ. $JOBCTLF - Subroutines
Refer to Chart 15.

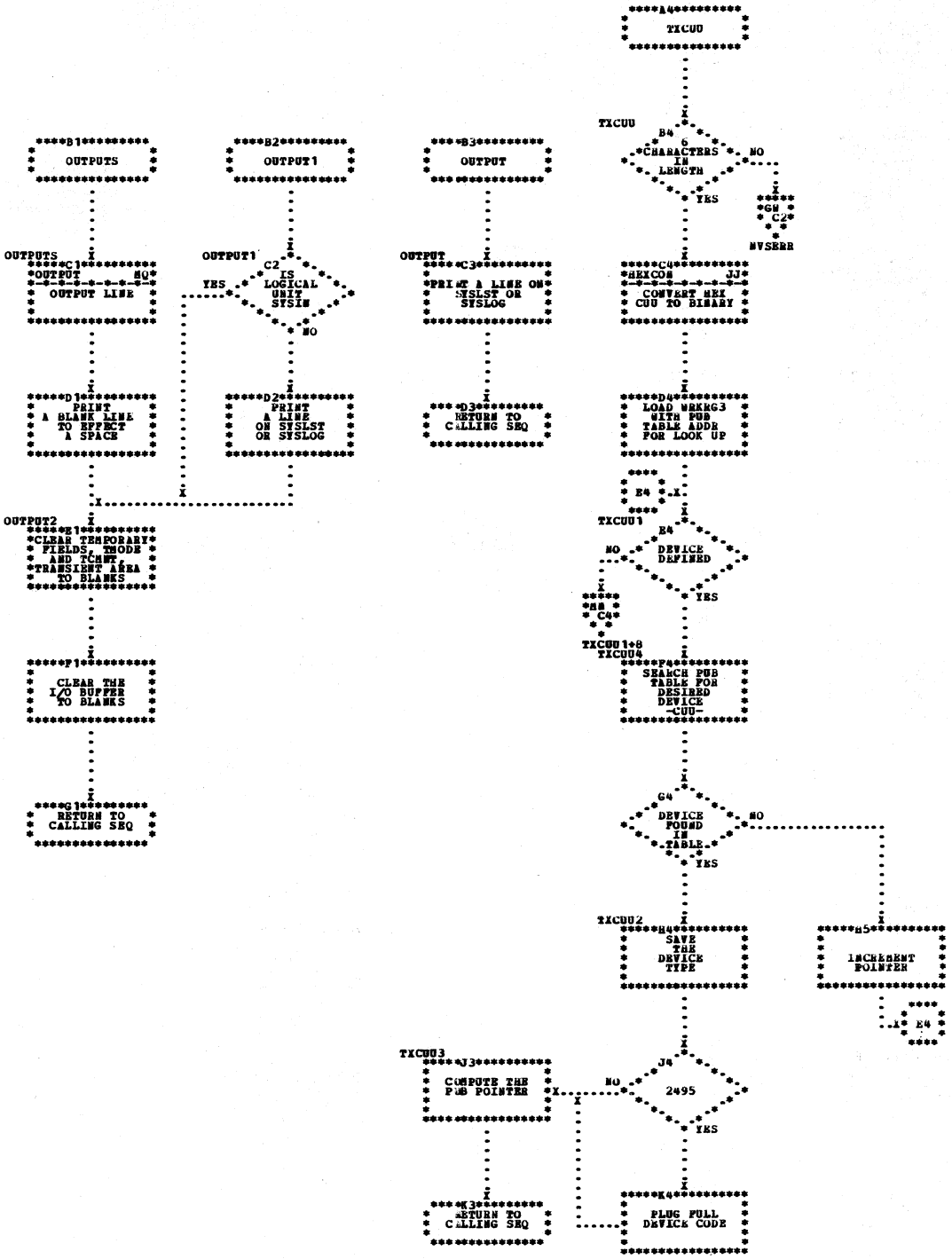


Chart MR. $JOBCTLF - Subroutines
Refer to Chart 15.

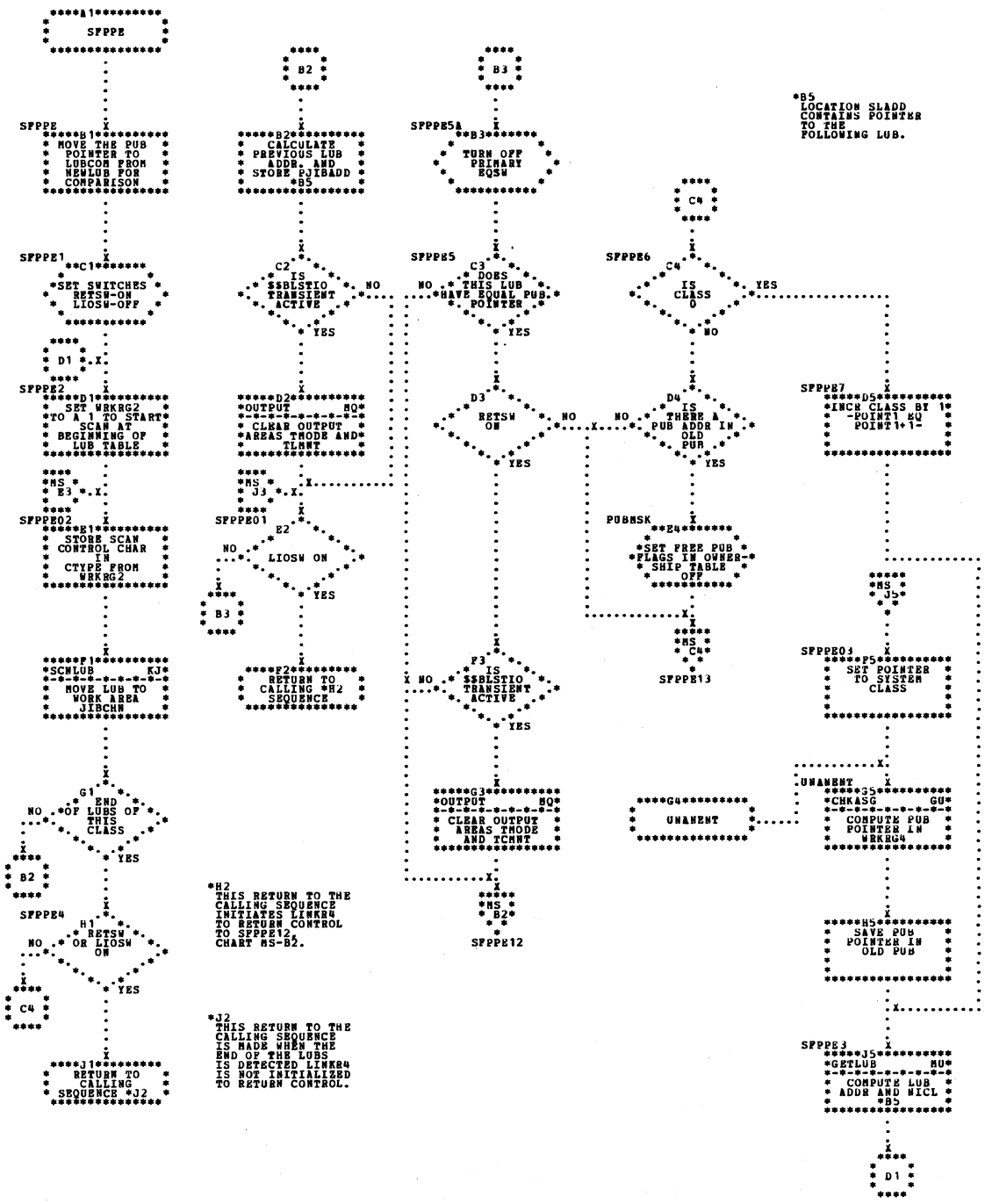


Chart MS. $JOBCTLF - Subroutines
Refer to Chart 15.

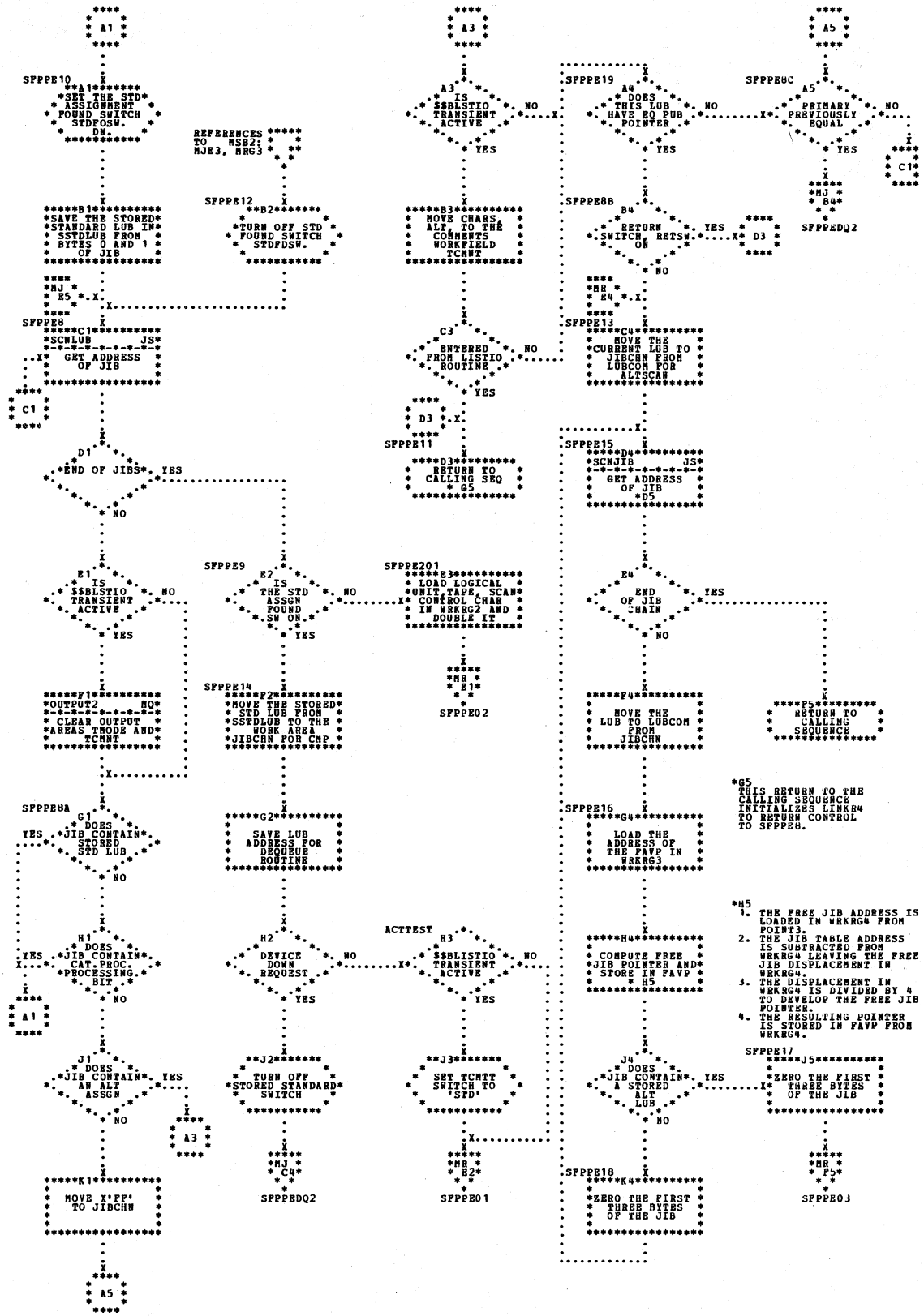


Chart MT. $JOBCTLF - Subroutines
Refer to Chart 15.

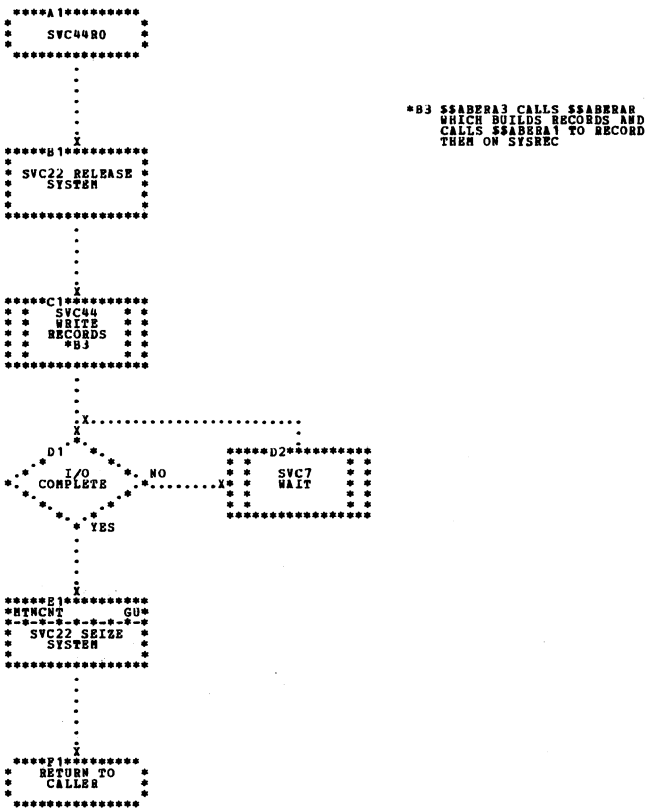


Chart MU. $JOBCTLF - Subroutines
Refer to Chart 15.

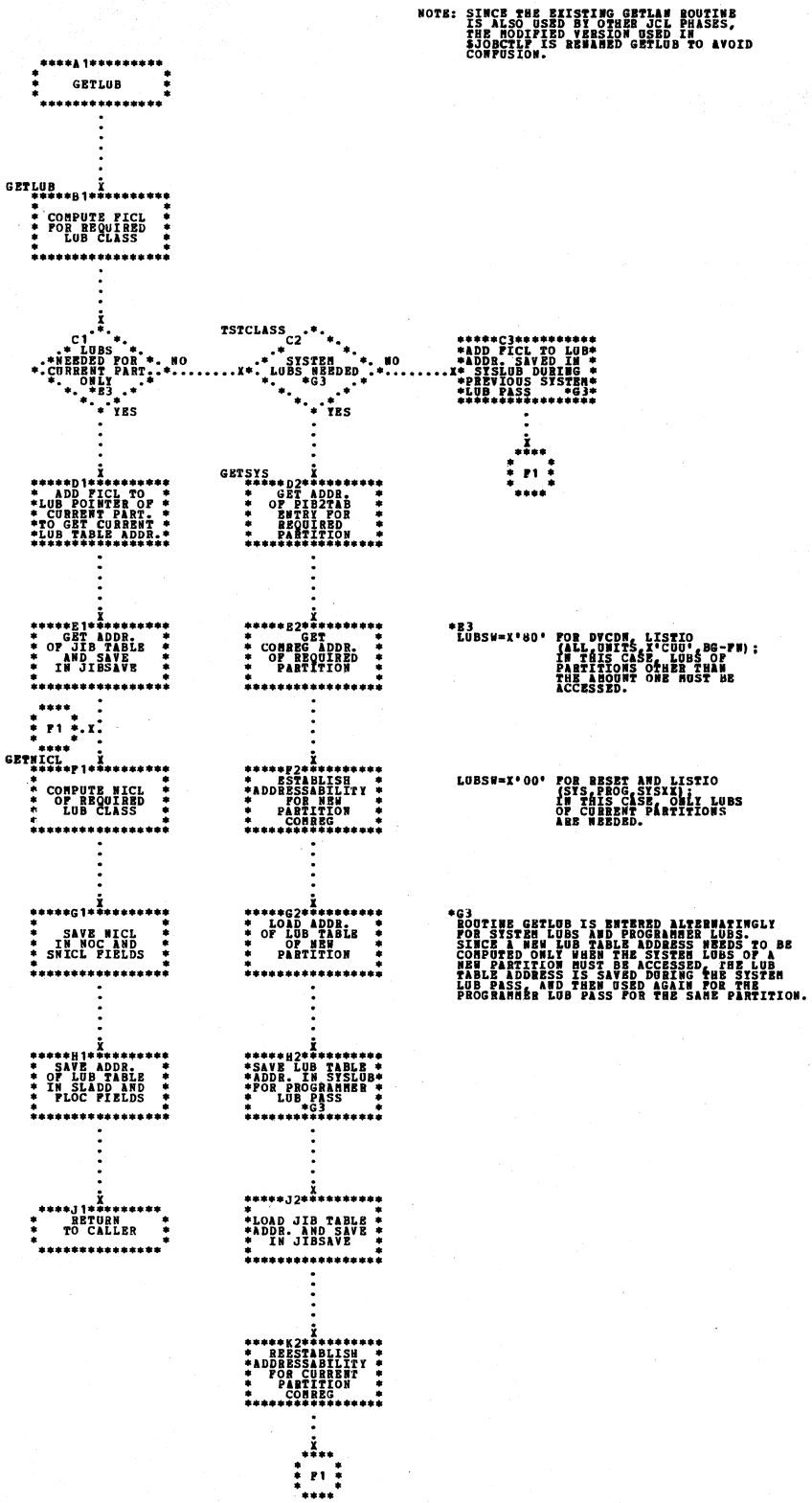


Chart NA. $JOBCTLG - JOB Statement Processor (Part 1 of 3)
Refer to Chart 17.

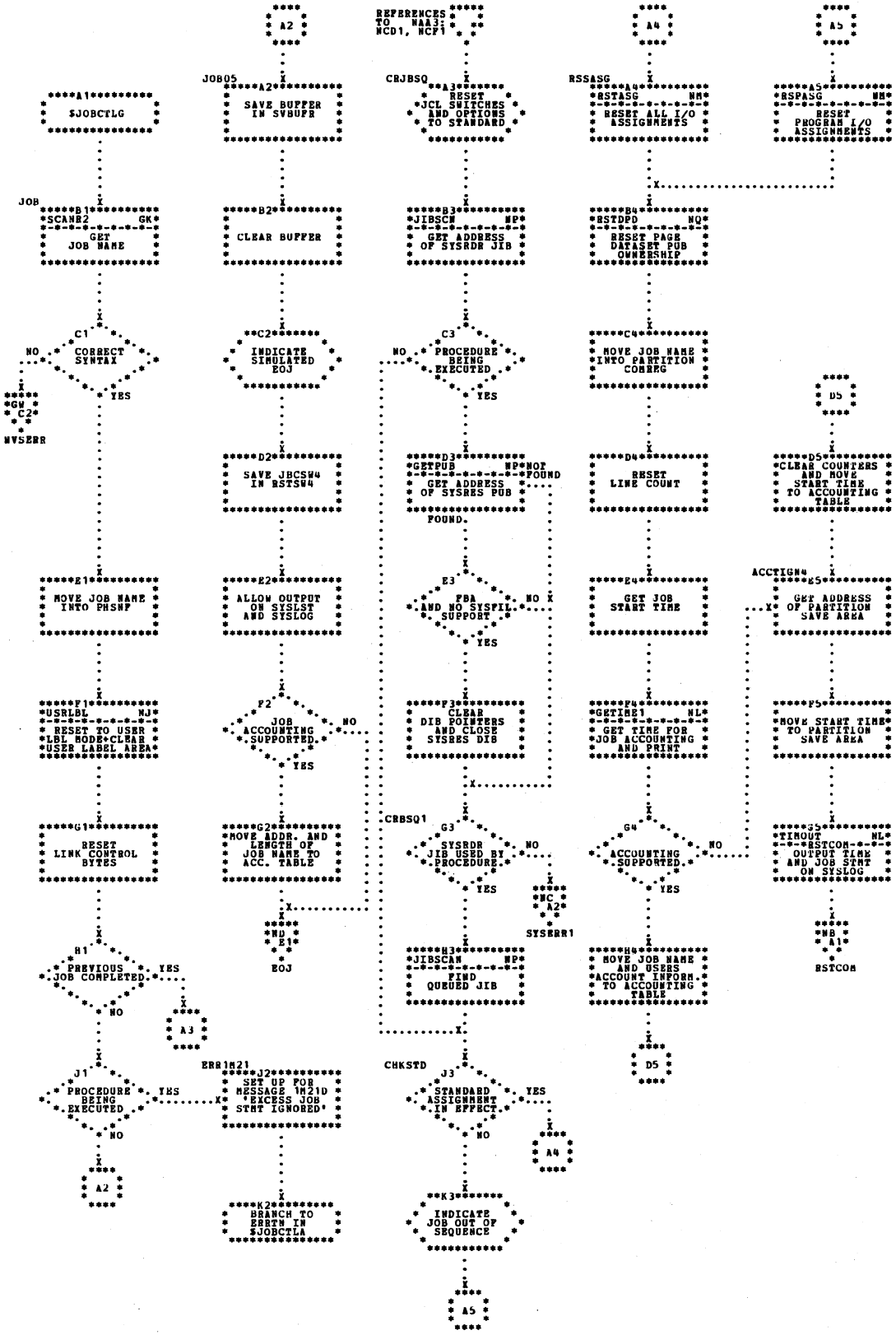


Chart NB. $JOBCTLG - JOB Statement Processor (Part 2 of 3)
Refer to Chart 17.

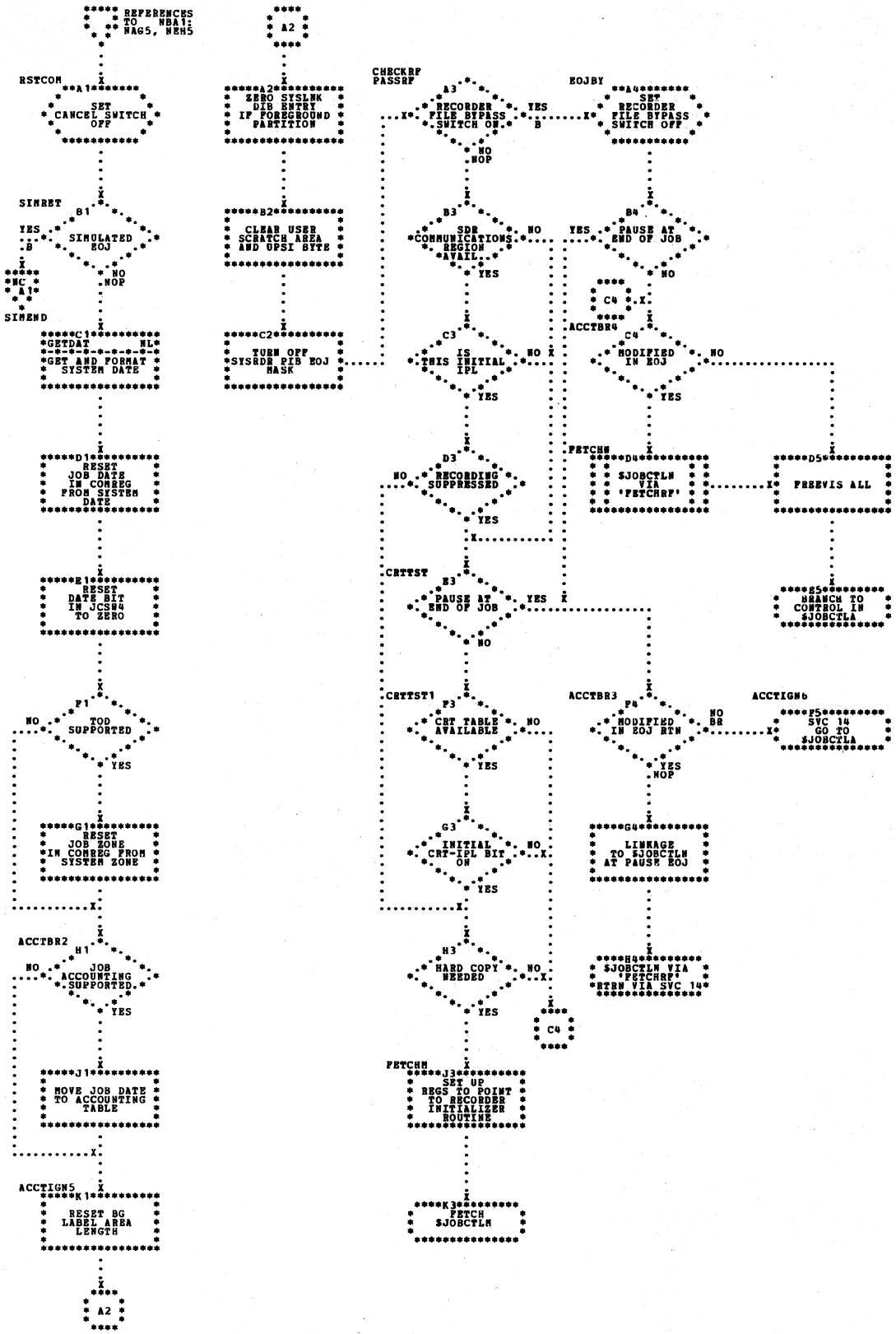


Chart NC. $JOBCTLG - JOB Statement Processor (Part 3 of 3)
Refer to Chart 17.

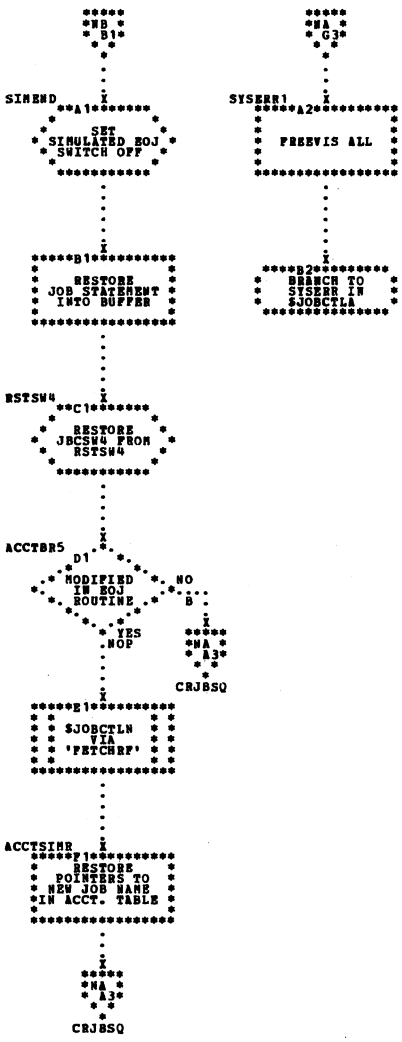


Chart ND. $JOBCTLG - /& Statement Processor (Part 1 of 2)
Refer to Chart 17.

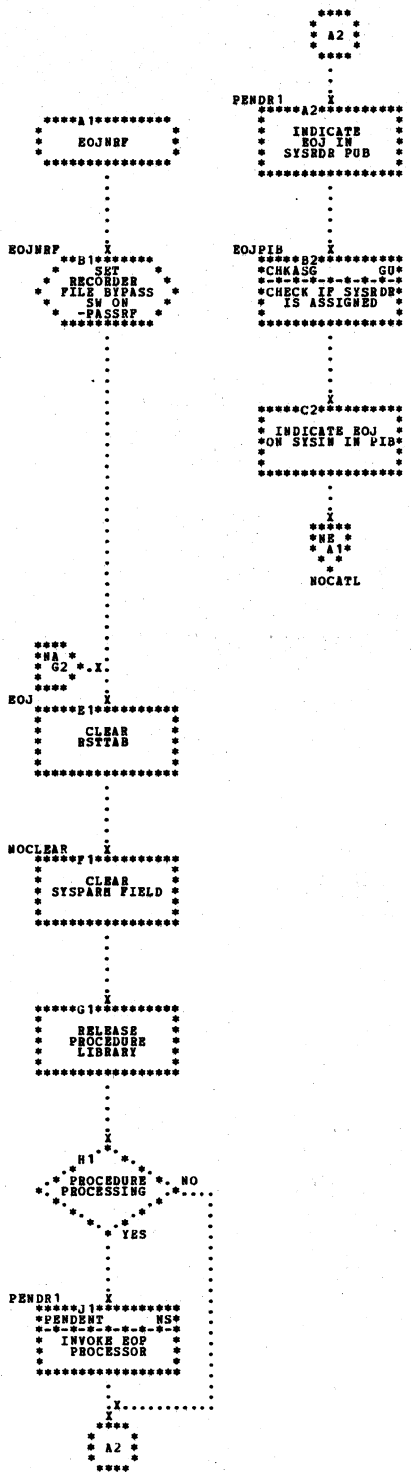


Chart NE. $JOBCTLG - /& Statement Processor (Part 2 of 2)
Refer to Chart 17.

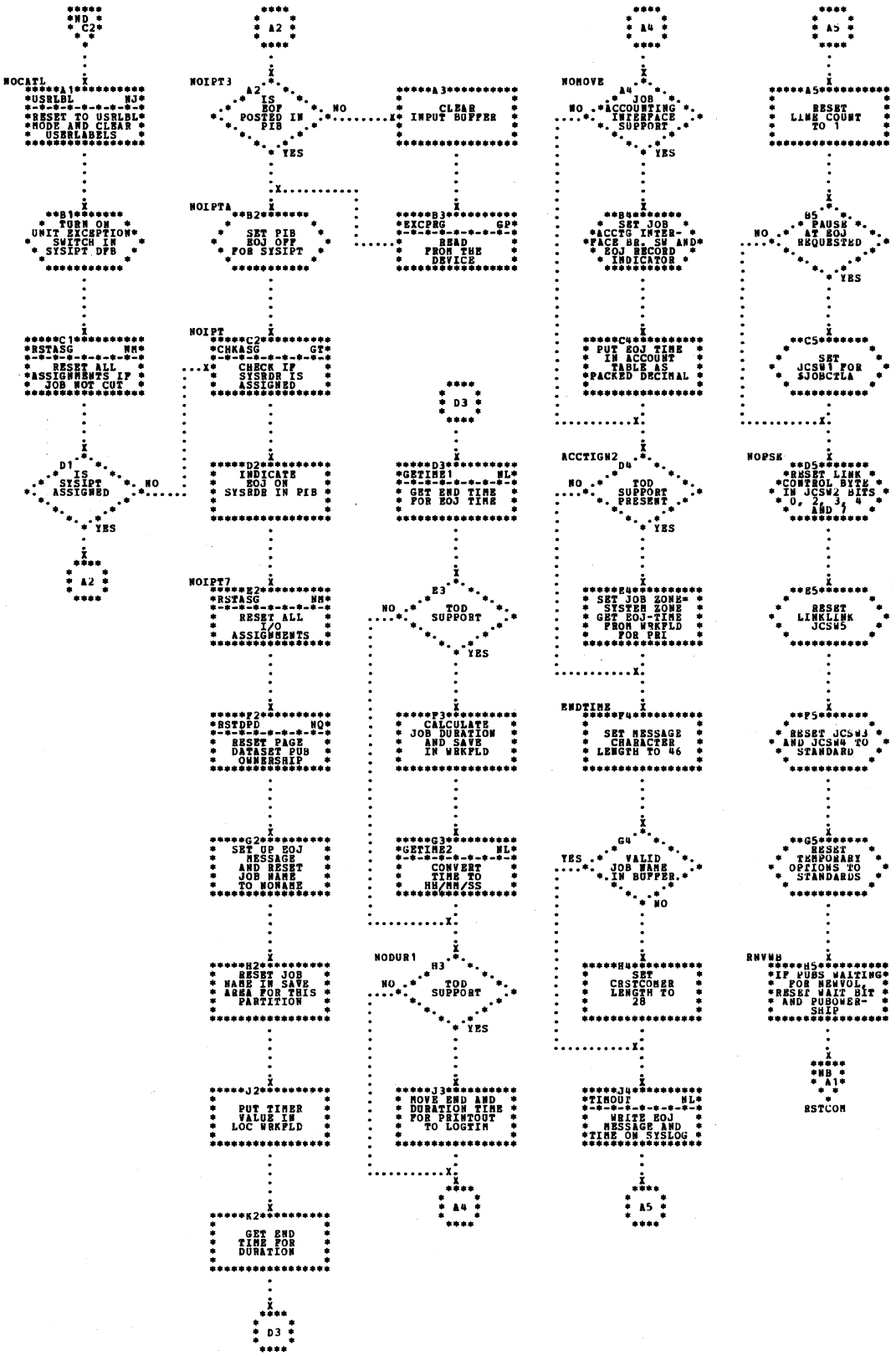


Chart NF. $JOBCTLG - CANCEL Processor
Refer to Chart 17.

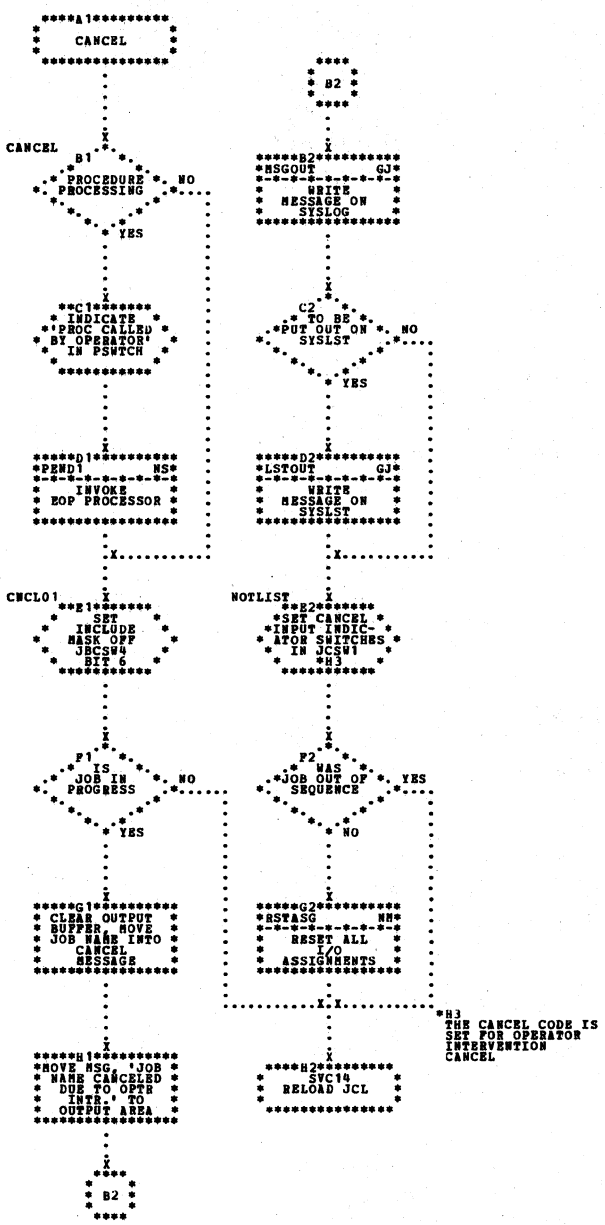
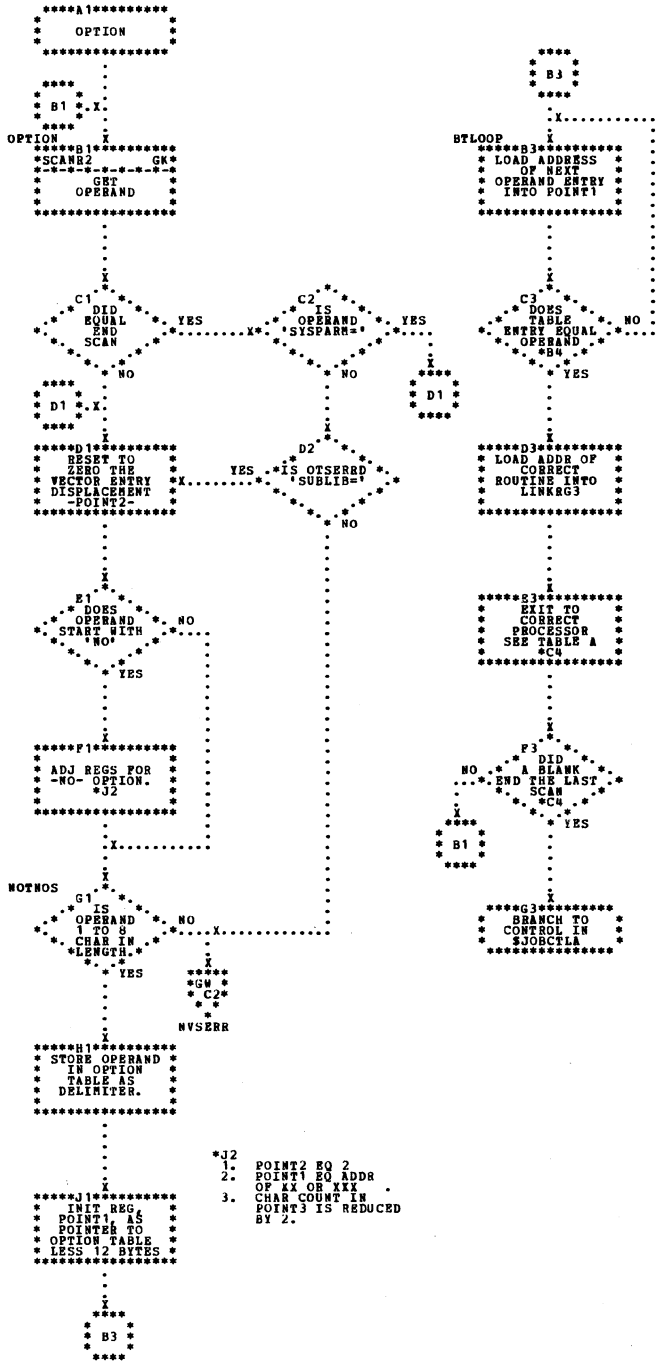


Chart NG. $JOBCTLG - OPTION Statement Processor (Part 1 of 5)
Refer to Chart 16.



*J2
1. POINT2 EQ 2
2. POINT1 EQ ADDR OF XX OR XXX .
3. CHAR COUNT IN POINT3 IS REDUCED BY 2.

*B4
BECAUSE THE OPERAND HAS BEEN STORED IN THE TABLE AS A DELIMITER, IT IS NOT POSSIBLE TO SEARCH THE ENTIRE TABLE WITHOUT FINDING AN EQUAL.

*C4
EACH PROCESSOR RETURNS CONTROL TO BLOCK F3 UNLESS AN INVALID OPERAND IS SPECIFIED. CONTROL IS TRANSFERRED TO NVOPTN AT BLOCK C2 IF AN INVALID OPERAND IS SPECIFIED.

TABLE A

OPERAND	CHART	FLOWCHART LABEL
DECK	NH	DECK
NODECK	NH	NODECK
LIST	NH	LIST
NOLIST	NH	NOLIST
LISTX	NH	LISTX
NLISTX	NH	NLISTX
SYM	NH	SYM
NOSYM	NH	NOSYM
XREF	NH	XREF
NOXREF	NH	NOXREF
ERRS	NH	ERRS
NOERRS	NH	NOERRS
48C	NH	C48
60C	NH	C60
LOG	NK	OPTLOG
NOLOG	NH	OPTNLOG
USRLABEL.	NJ	USRLBL
STDLABEL	NJ	STDLBL
DUMP	NK	DUMP
NODUMP	NK	NODUMP
LINK	NJ	LINK
NOLINK	NK	NOLINK
CATAL	NJ	CATAL
PARSTD	NJ	PARSTD
SYSPARM=	NK	SYSPRM
ALIGN	NH	ALIGN
NOALIGN	NH	NOALIGN
EDECK	NH	EDECK
NOEDECK	NH	NOEDECK
SUBLIB=	NK	SUBLIB
RLD	NK	RLD
NORLD	NH	NORLD
SXREF	NI	SXREF
SIZE	PS	SIZE
TERM	NI	TERM
NOTERM	NI	NOTERM
ONLINE	NI	OPTONL
NOONLINE	NI	OPTNOONL

Chart NH. $JOBCTLG - OPTION Statement Processor (Part 2 of 5)
Refer to Chart 16.

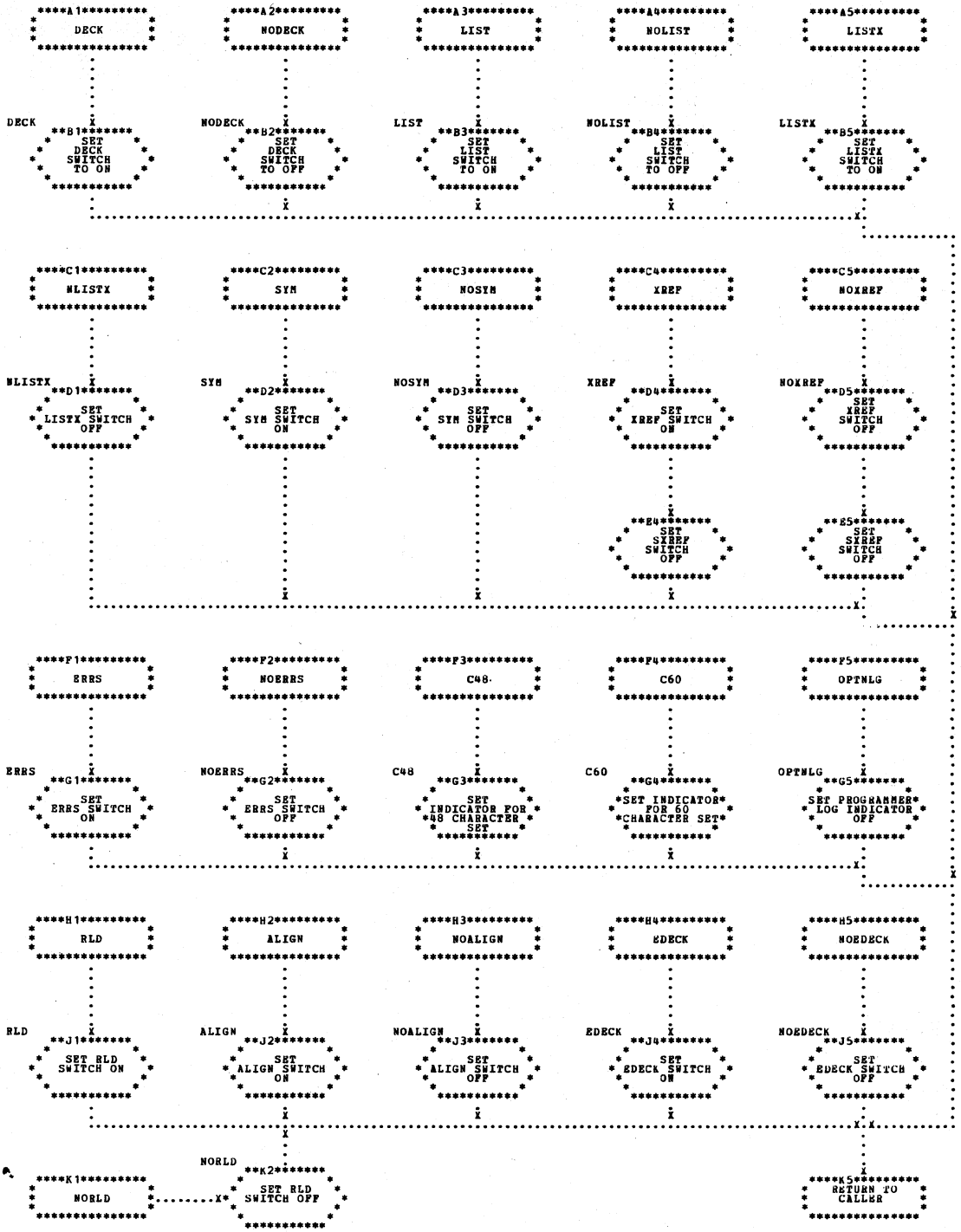
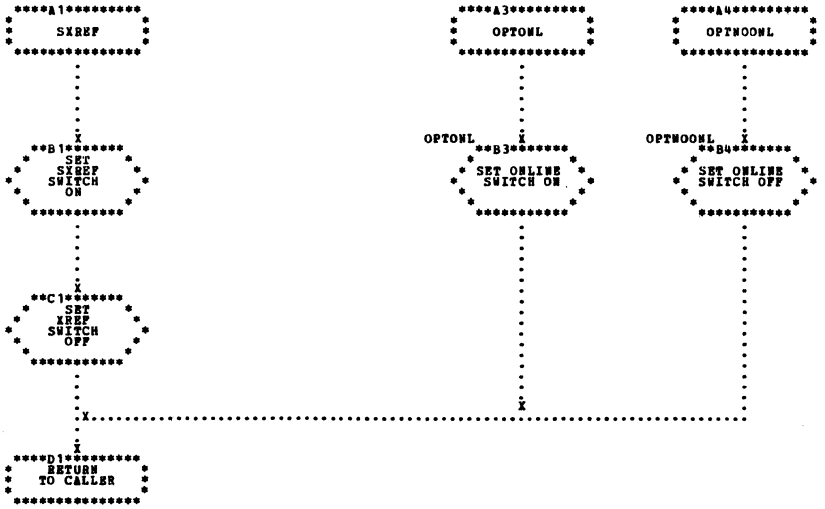


Chart NI. $JOBCTLG - OPTION Statement Processor (Part 3 of 5)
Refer to Chart 16.



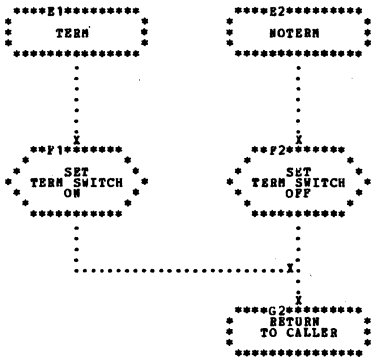
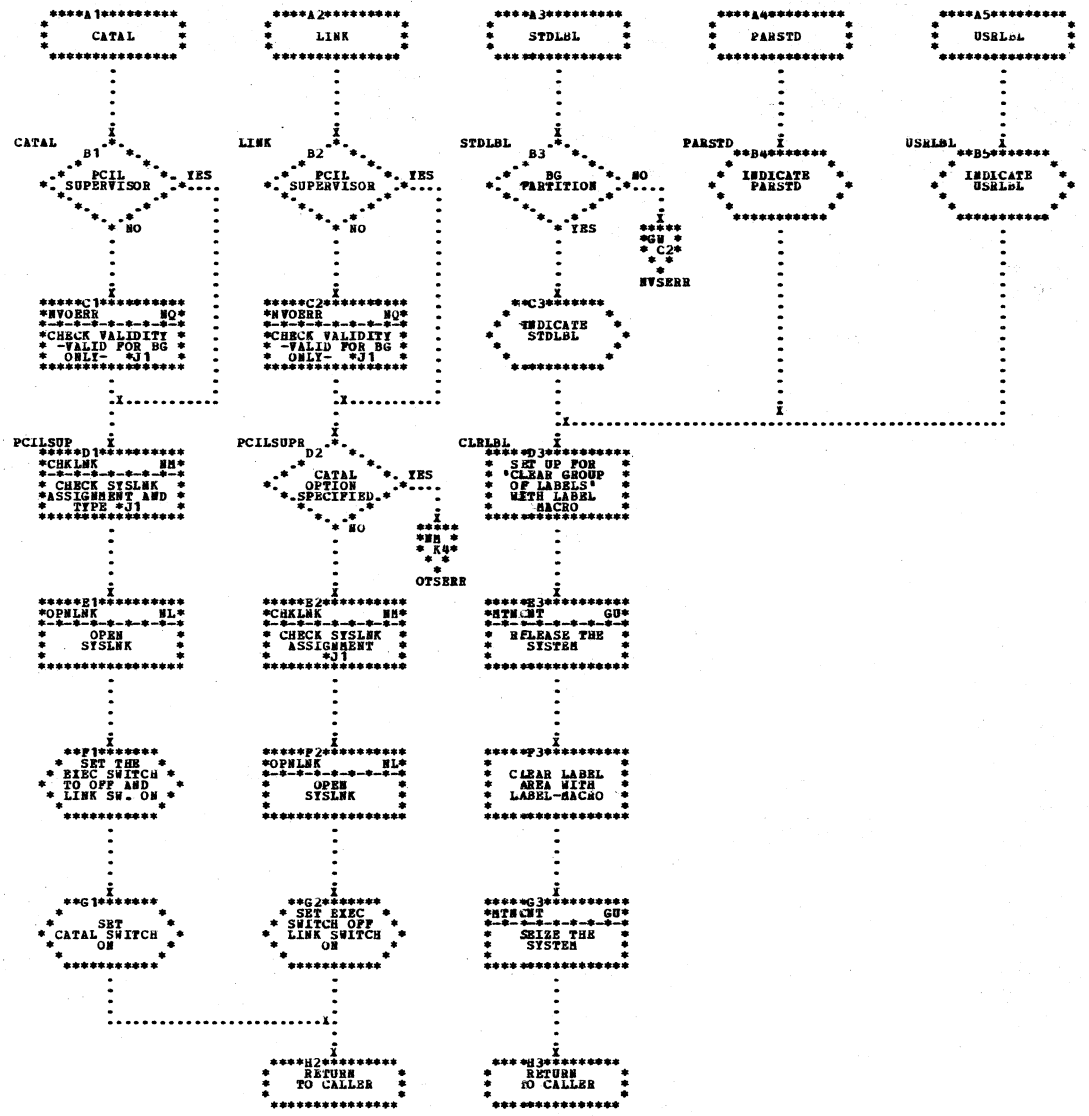


Chart NJ. $JOBCTLG - OPTION Statement Processor (Part 4 of 5)
Refer to Chart 16.



*J1
IF ANY ERROR IS FOUND, RETURN IS NOT TO THIS ROUTINE

Chart NK. $JOBCTLG - OPTION Statement Processor (Part 5 of 5)
Refer to Chart 16.

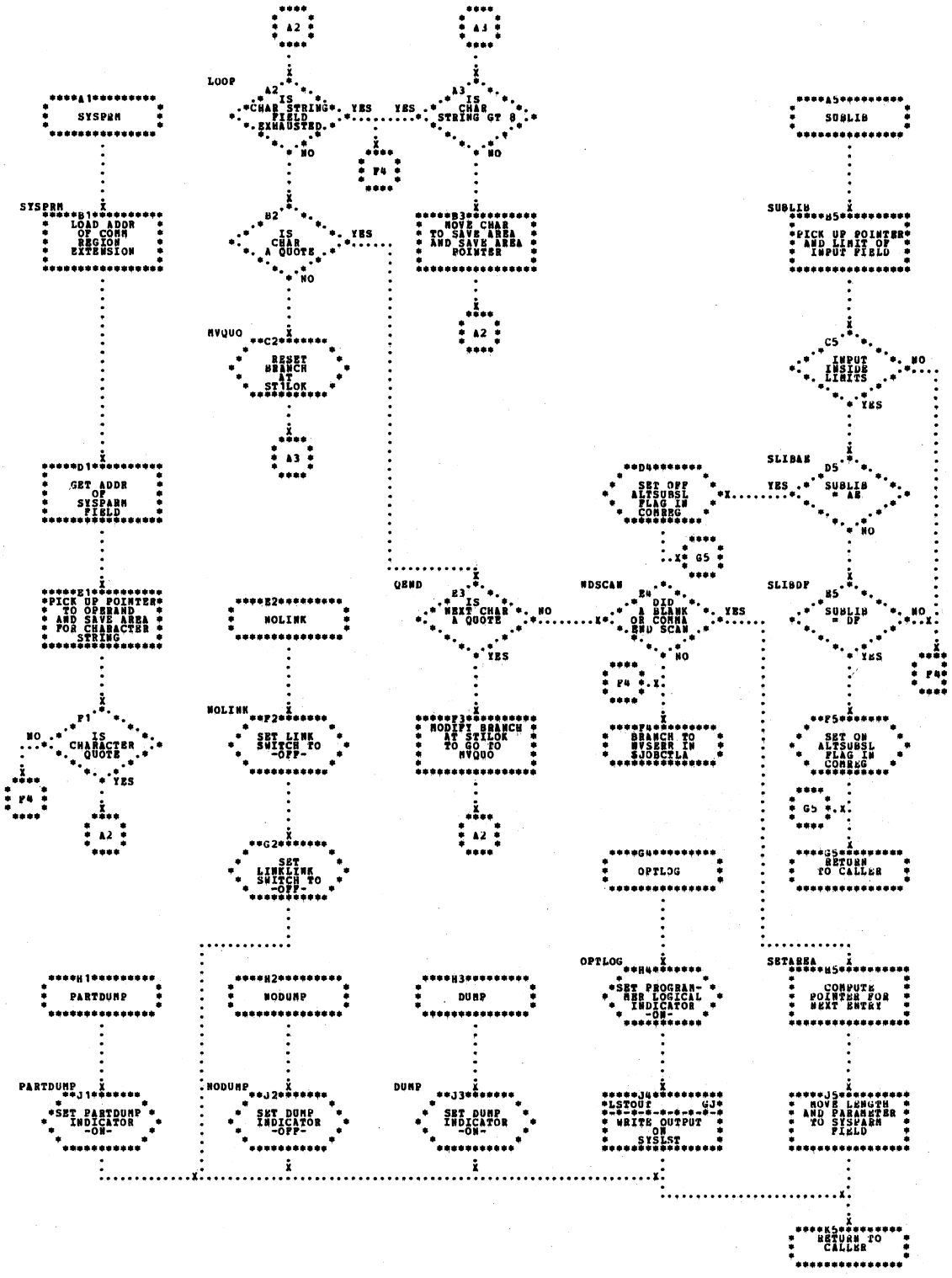


Chart NL. $JOBCTLG - Time Stamping Subroutines
Refer to Chart 17.

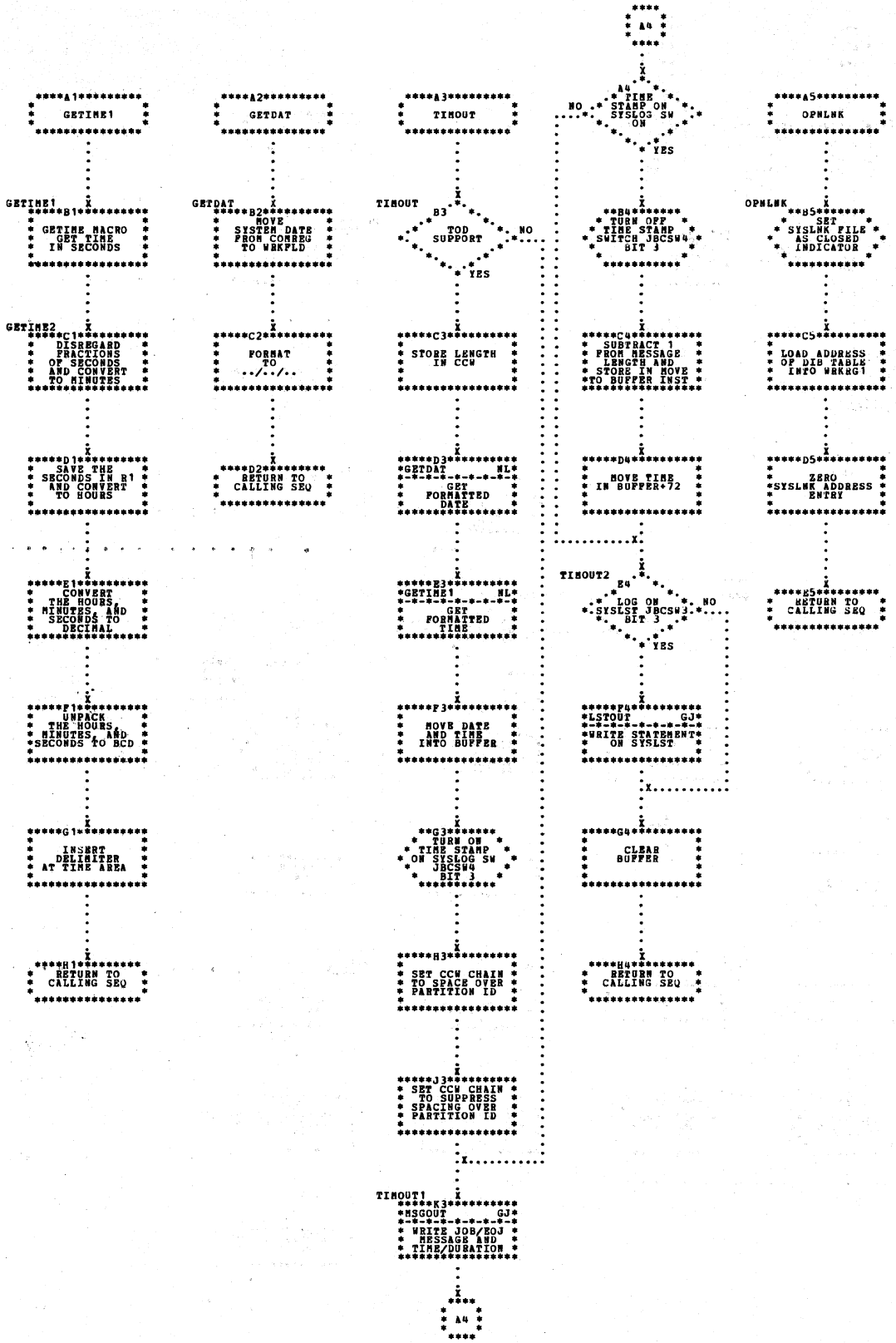


Chart NM. $JOBCTLG - Subroutines
Refer to Chart 17.

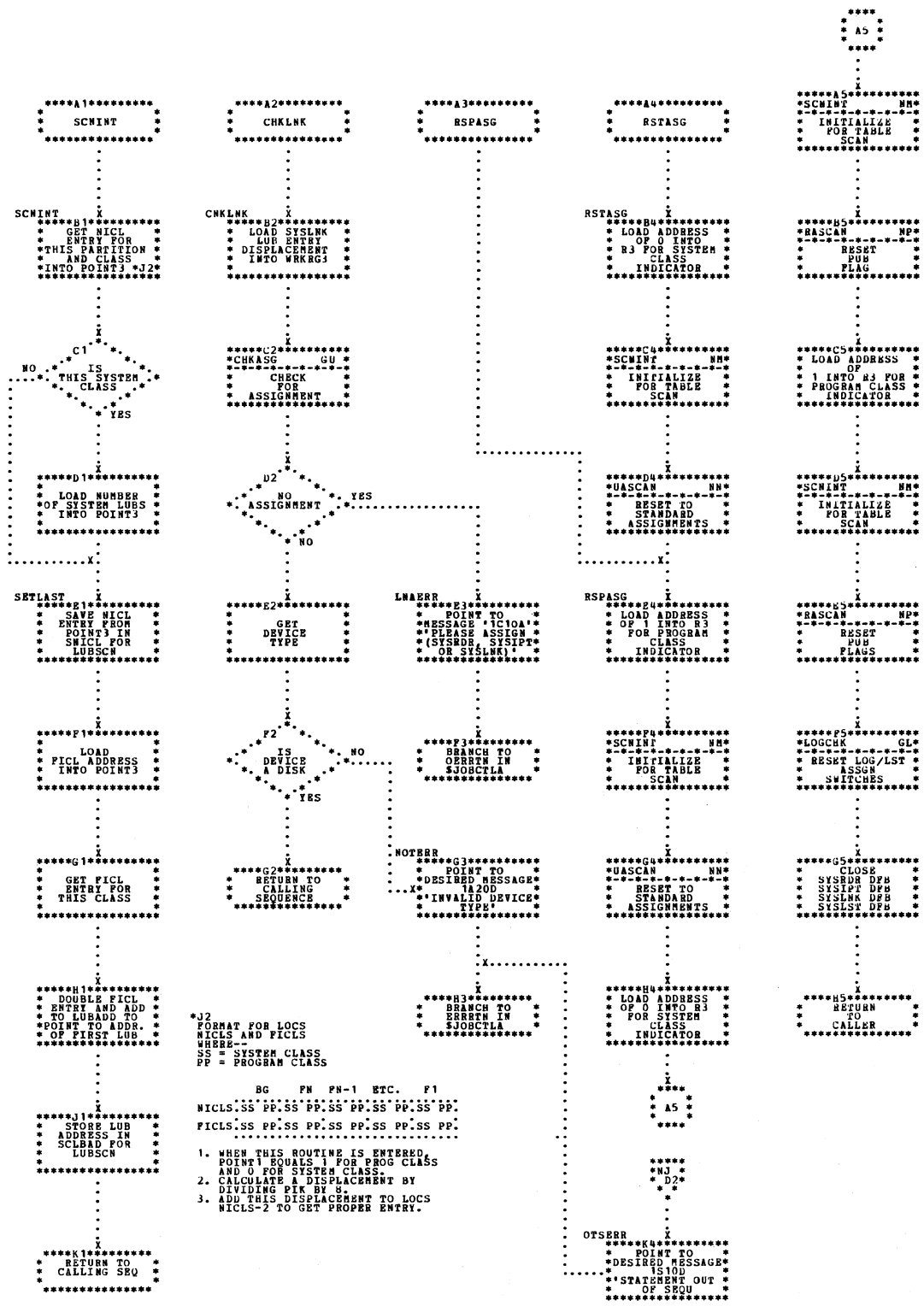


Chart NN. $JOBCTLG - Reset to Standard Assignment
Refer to Chart 17.

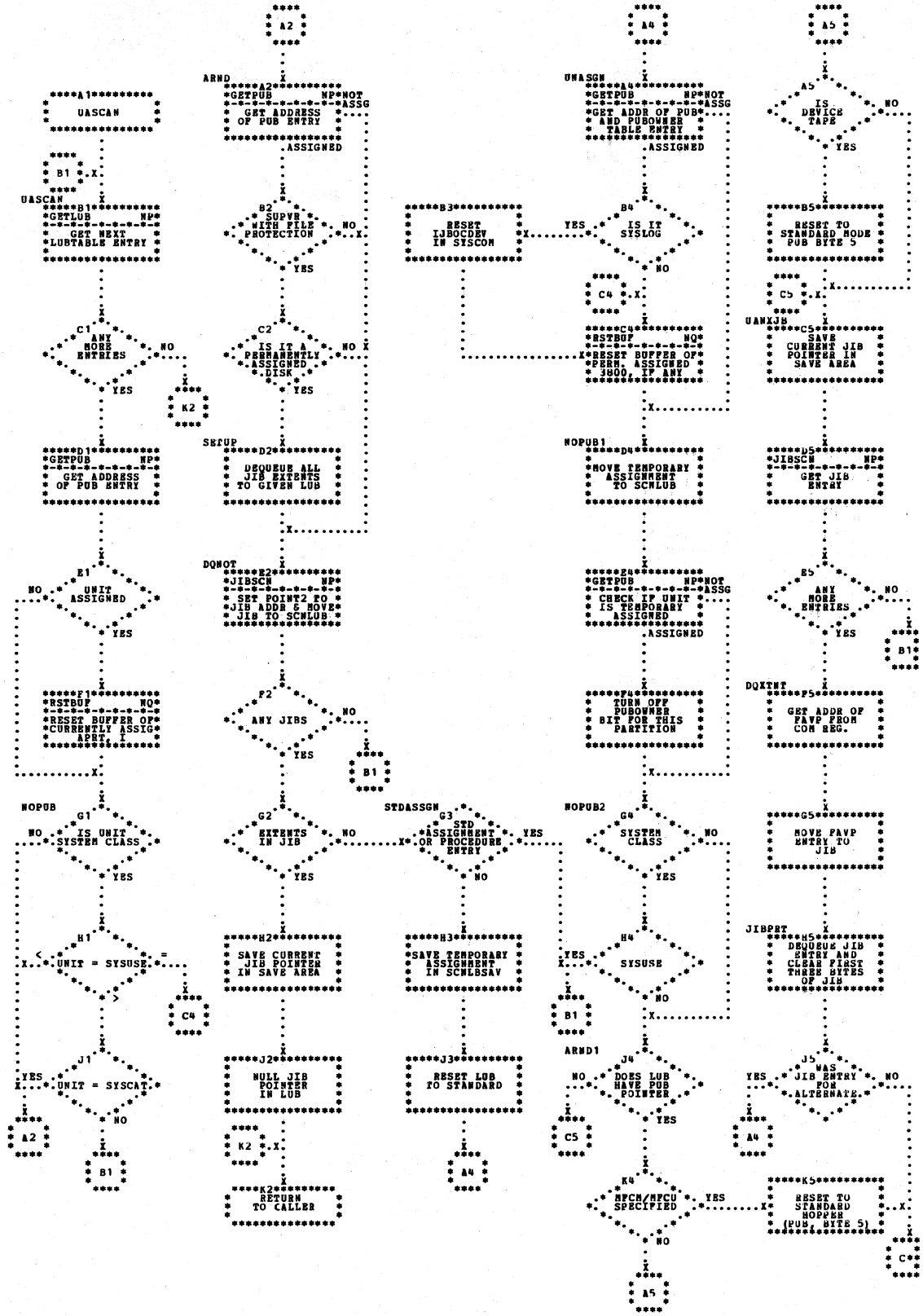


Chart NP. $JOBCTLG - Subroutines
Refer to Chart 17.

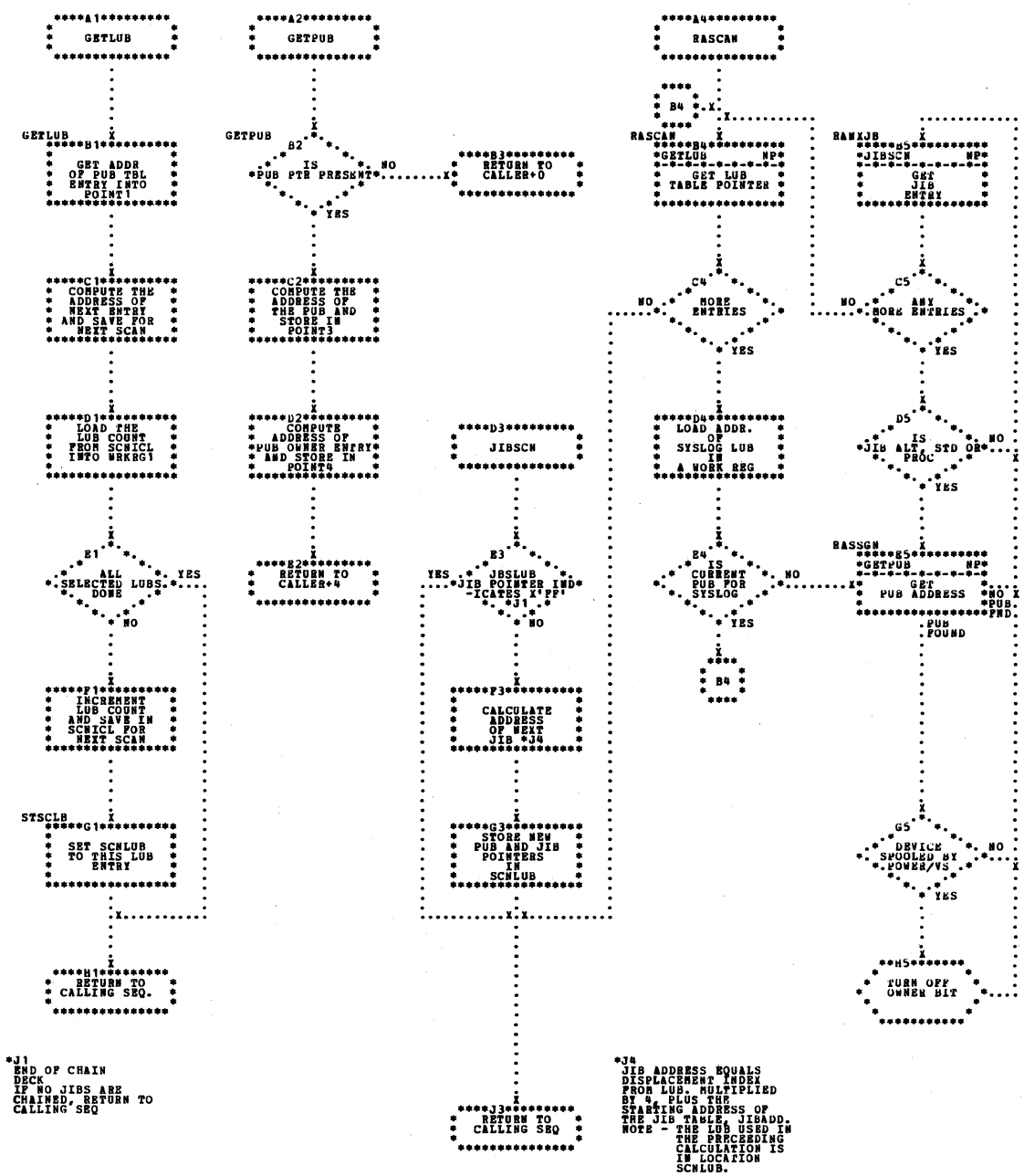


Chart NQ. $JOBCTLG - Subroutines
Refer to Chart 17.

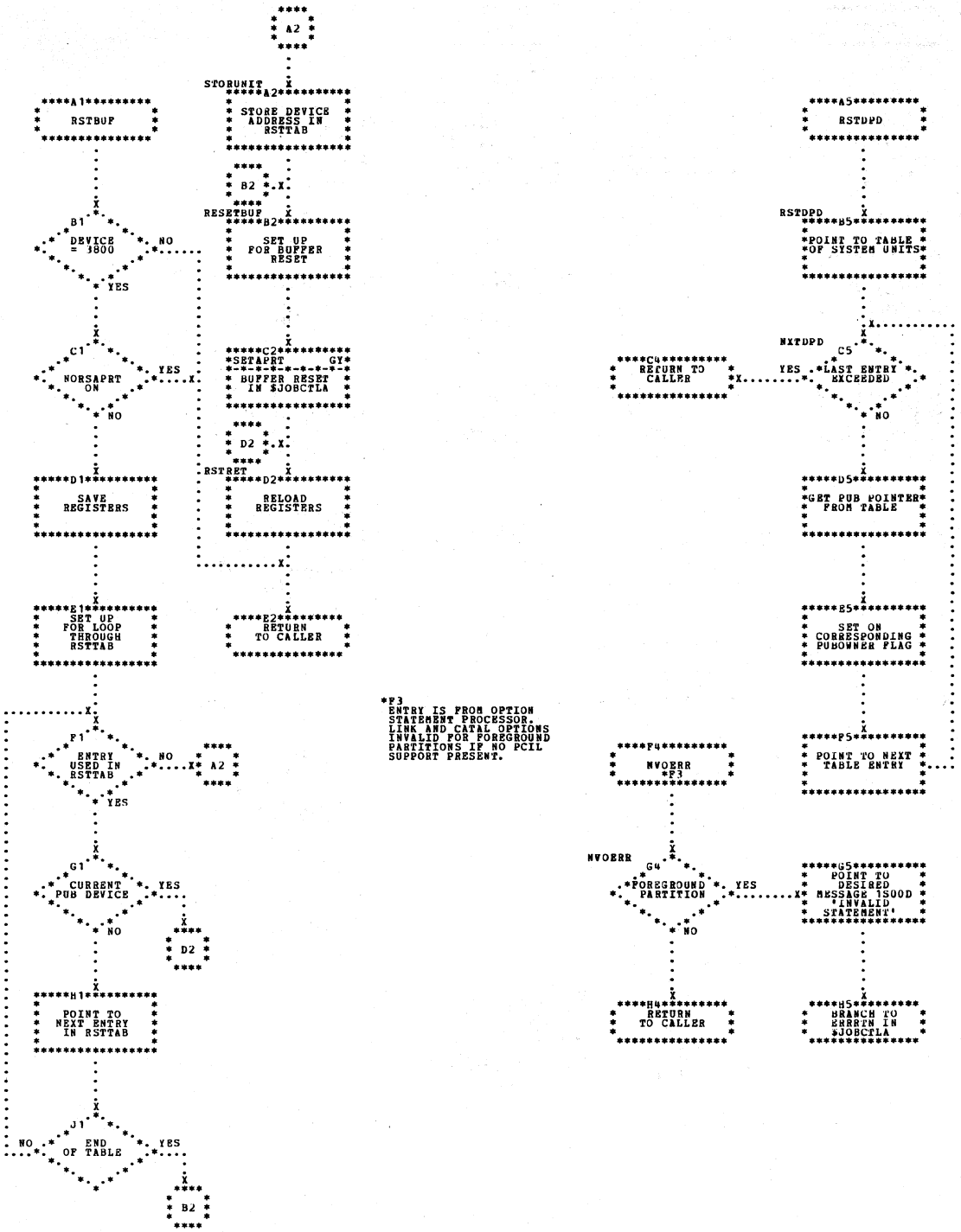


Chart NS. $JOBCTLG - /+ (End of Procedure) Statement Processor
Refer to Chart 17.

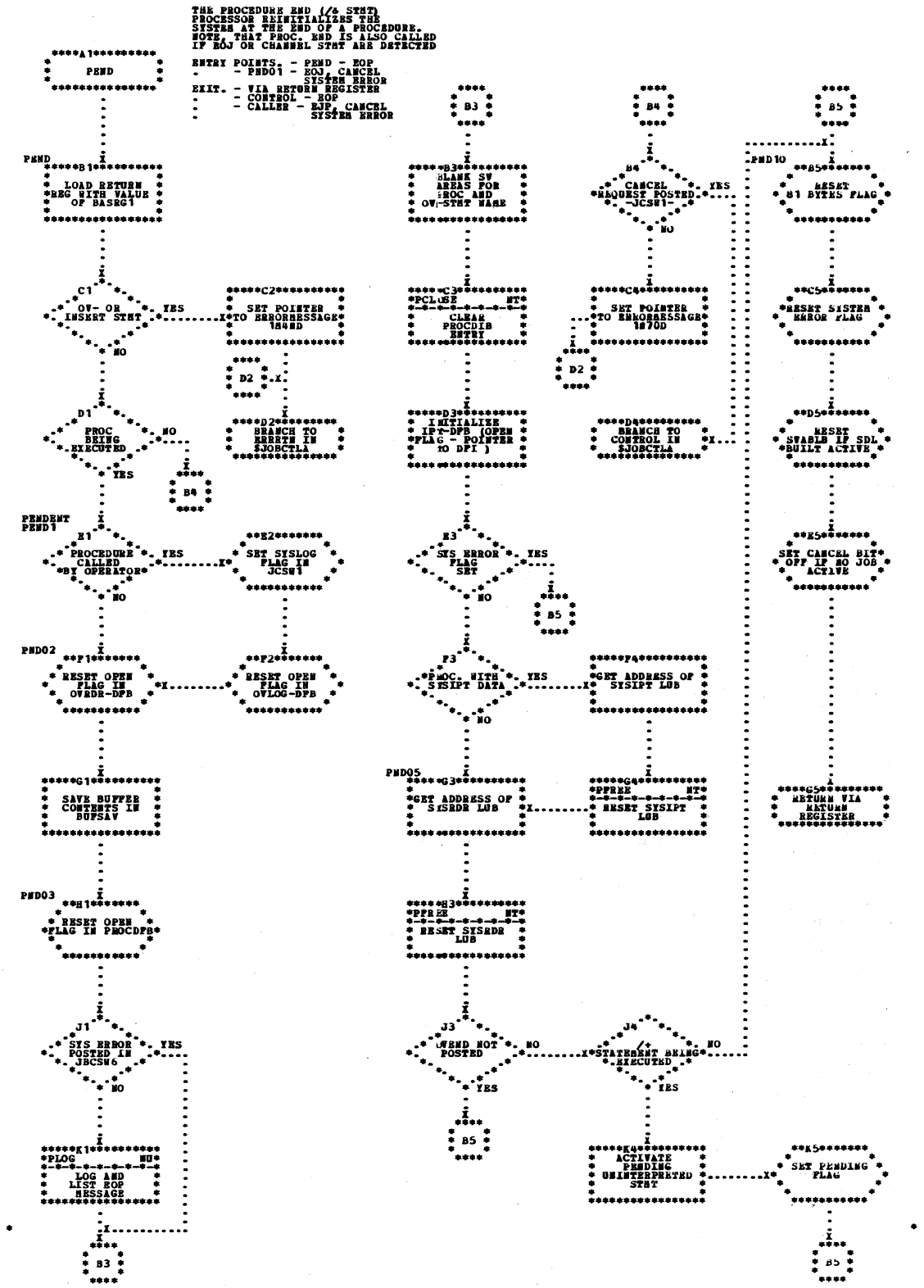


Chart NT. $JOBCTLG - EOP Subroutines (Part 1 of 2)
Refer to Chart 17.

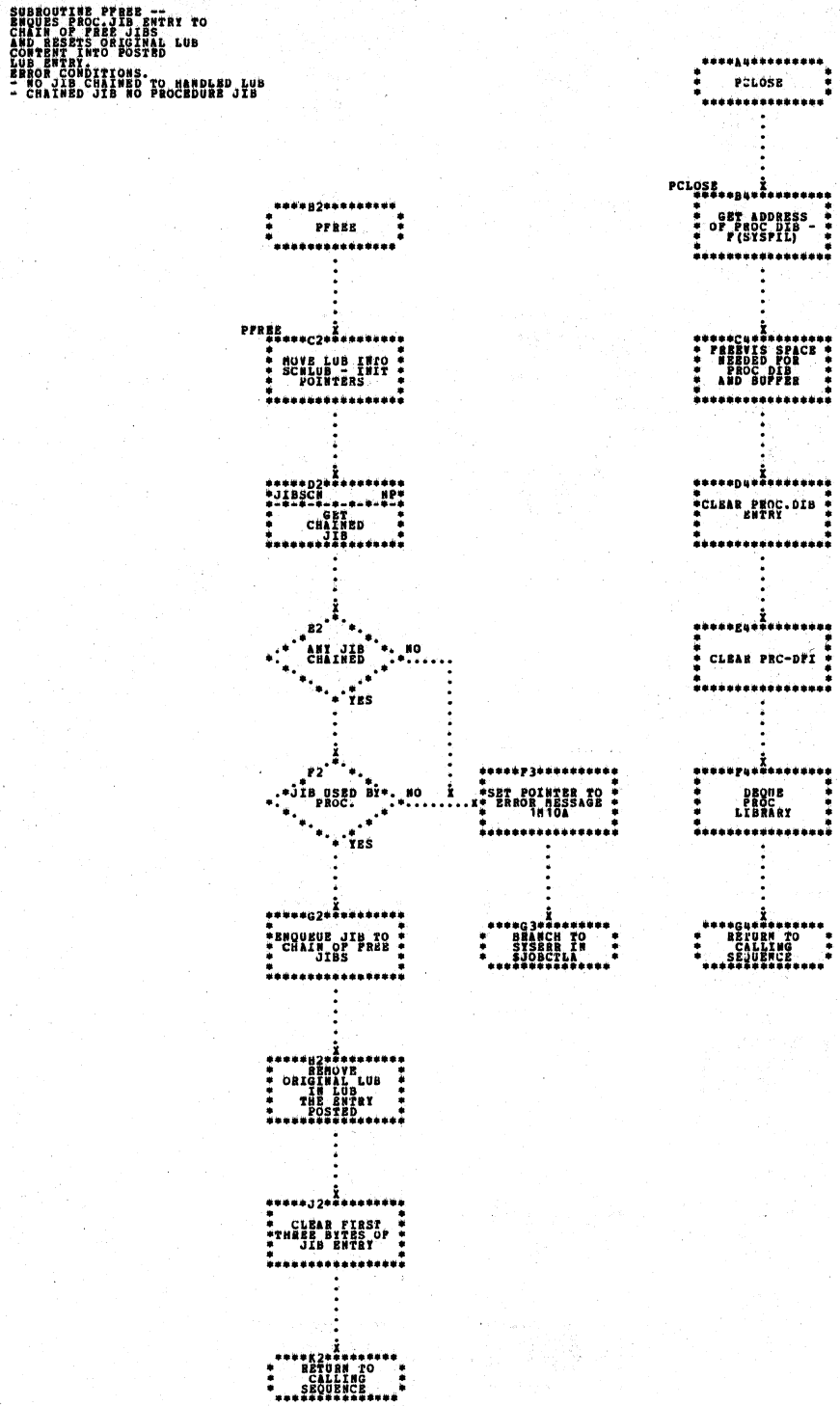


Chart NU. $JOBCTLG - EOP Subroutines (Part 2 of 2)
Refer to Chart 17.

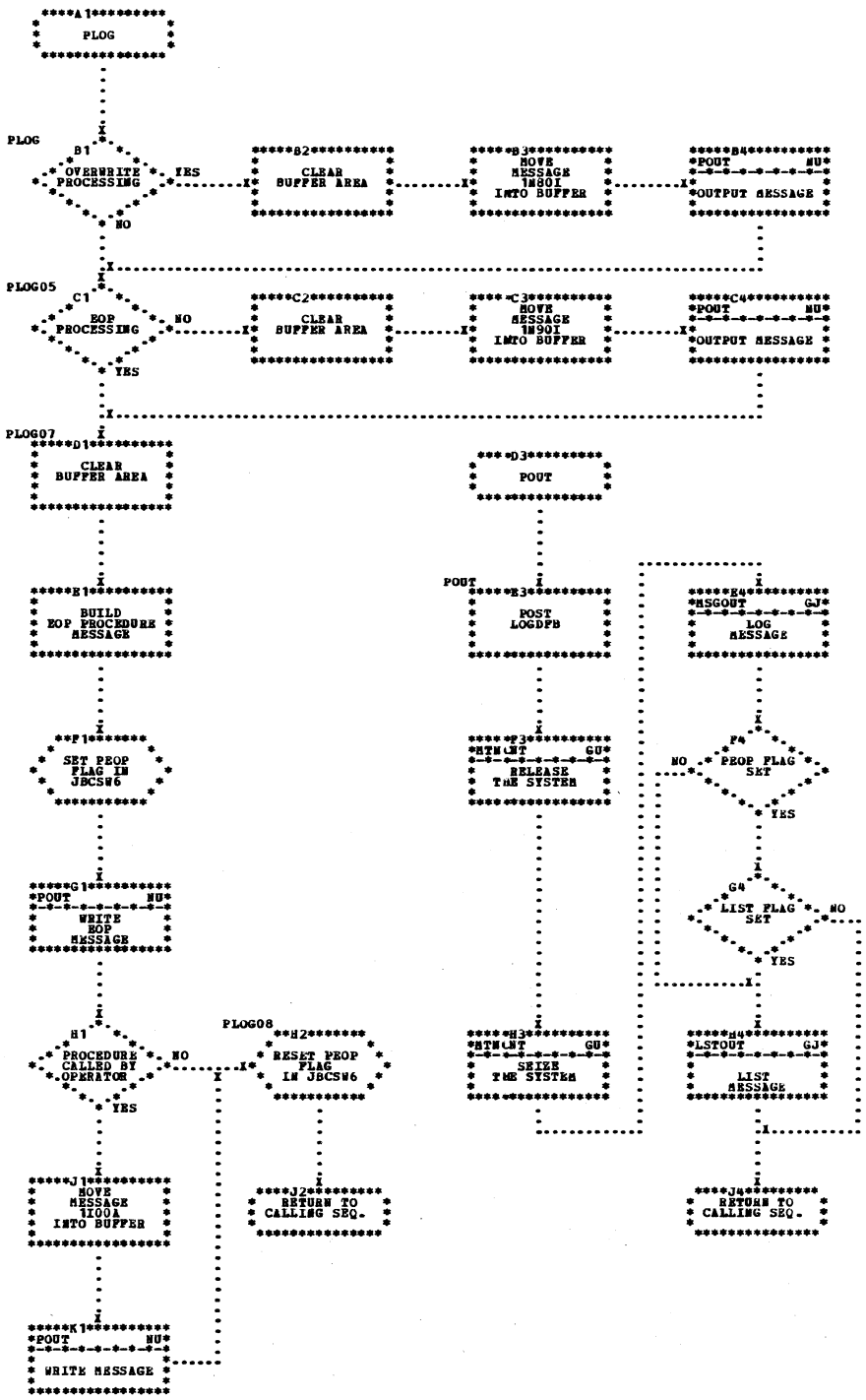


Chart PA. $JOBCTLJ - HOLD Command Processor
Refer to Charts 18 to 20.

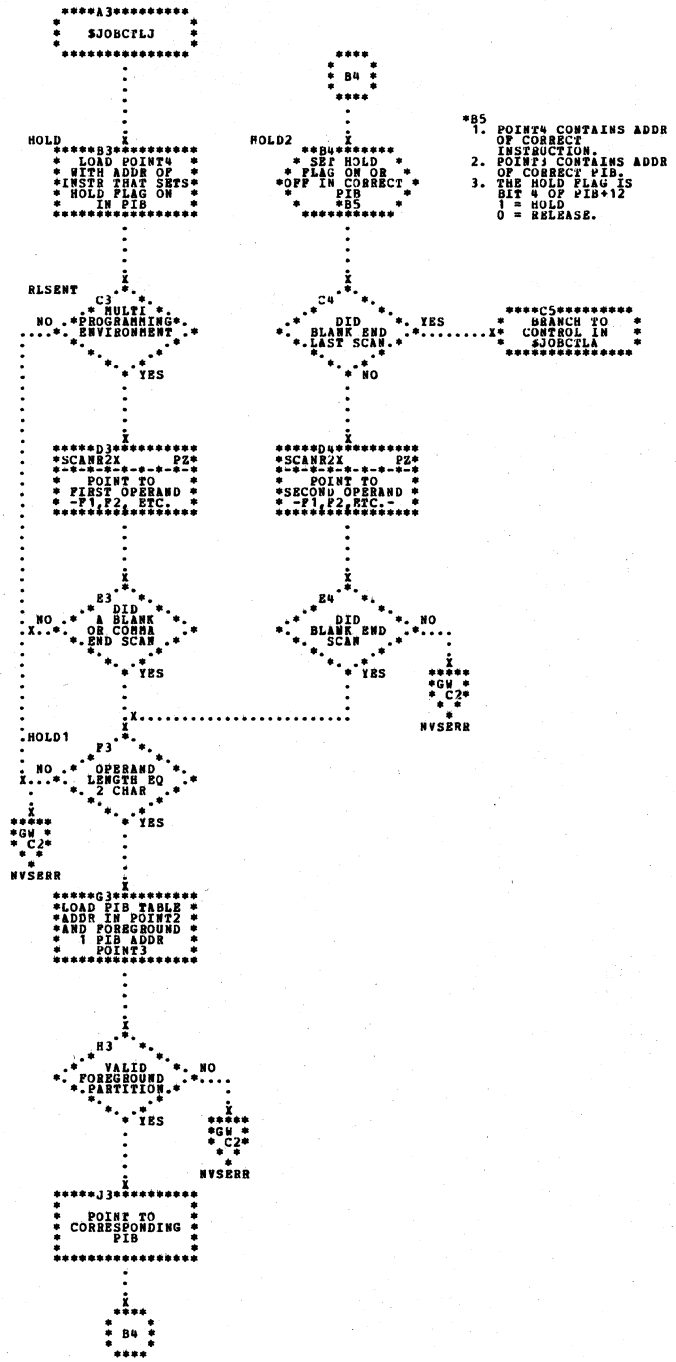


Chart PB. $JOBCTLJ - UCS Command Processor (Part 1 of 2)
Refer to Charts 18 to 20.

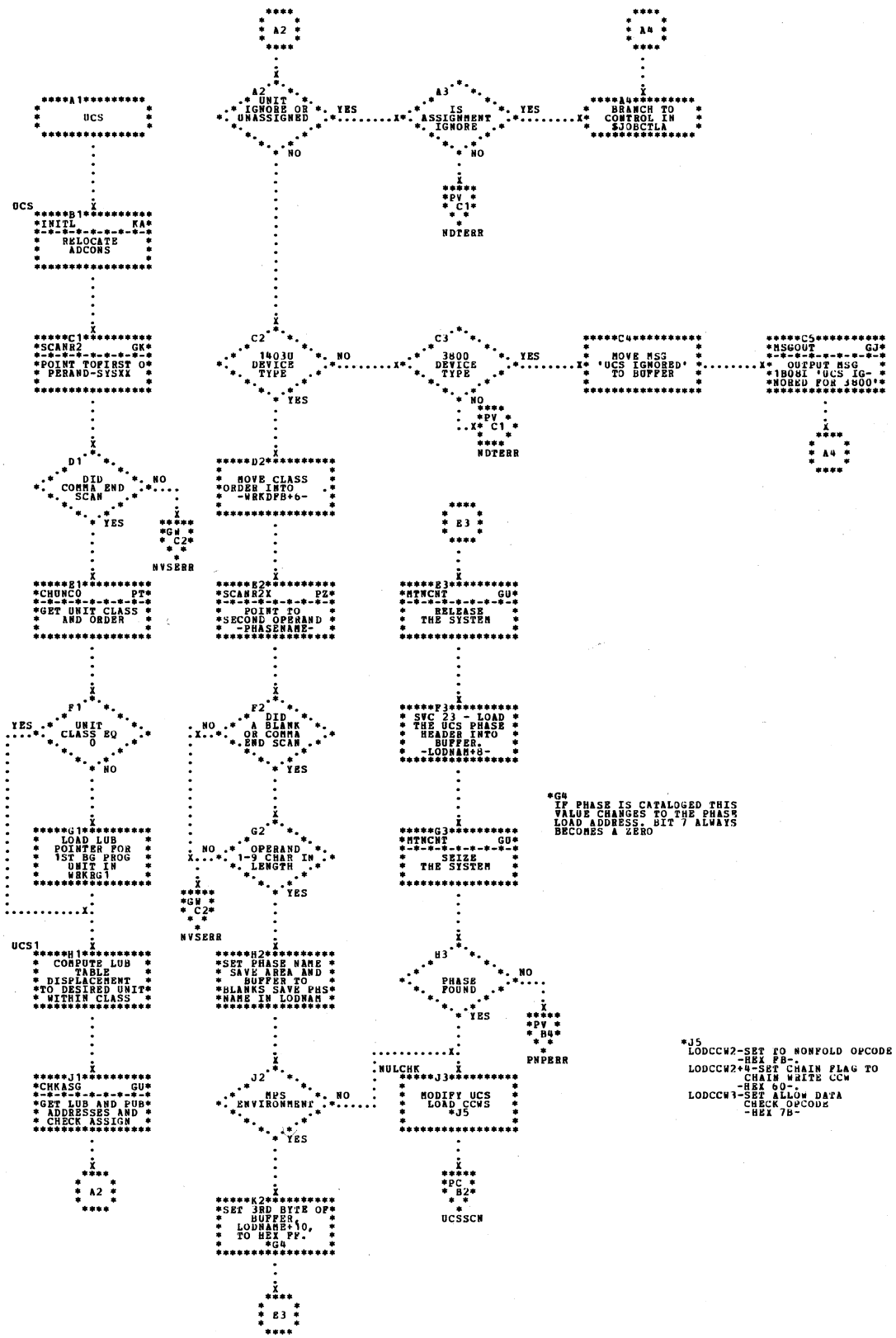


Chart PC. $JOBCTLJ - UCS Command Processor (Part 2 of 2)
Refer to Charts 18 to 20.

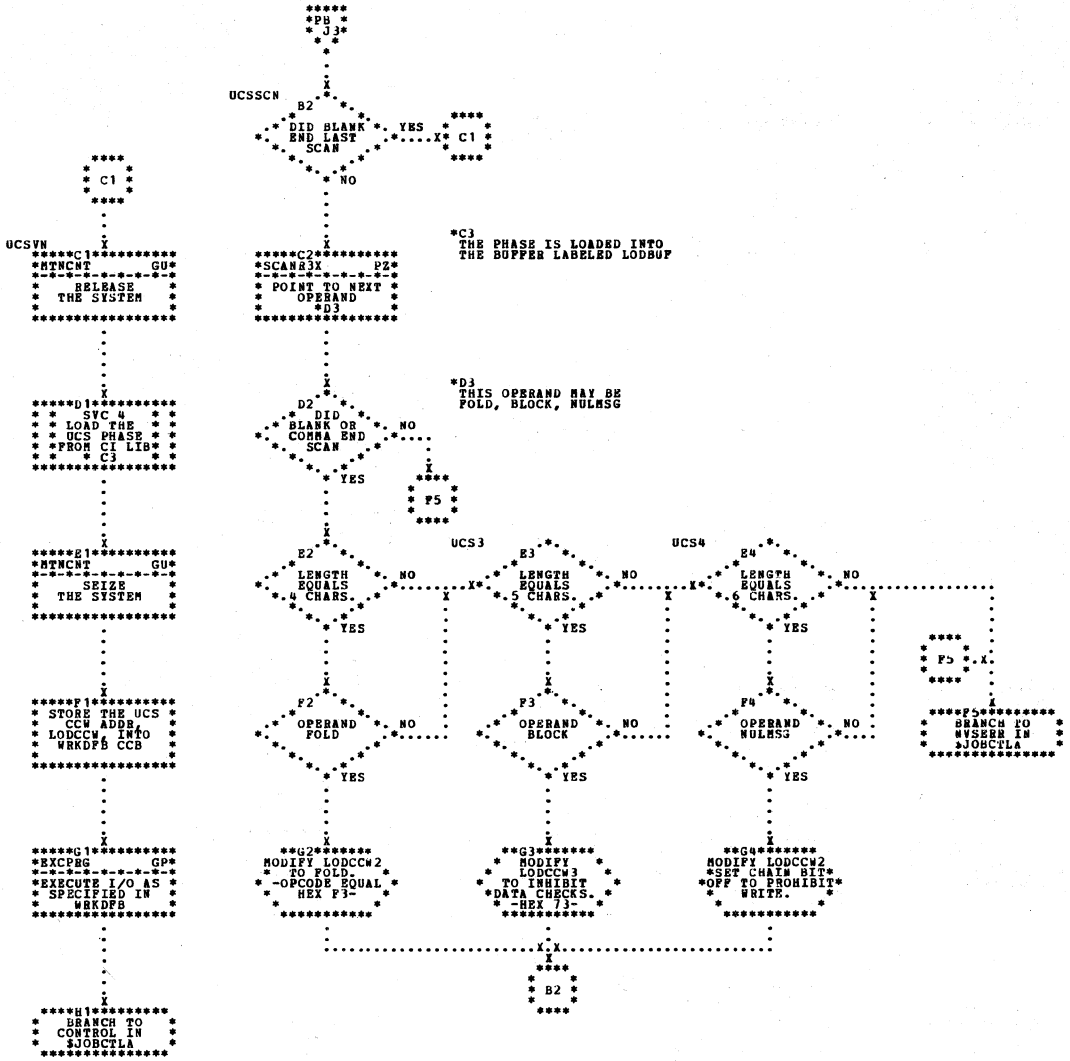


Chart PD. $JOBCTLJ - ACTION and INCLUDE Statement Processors
Refer to Charts 18 to 20.

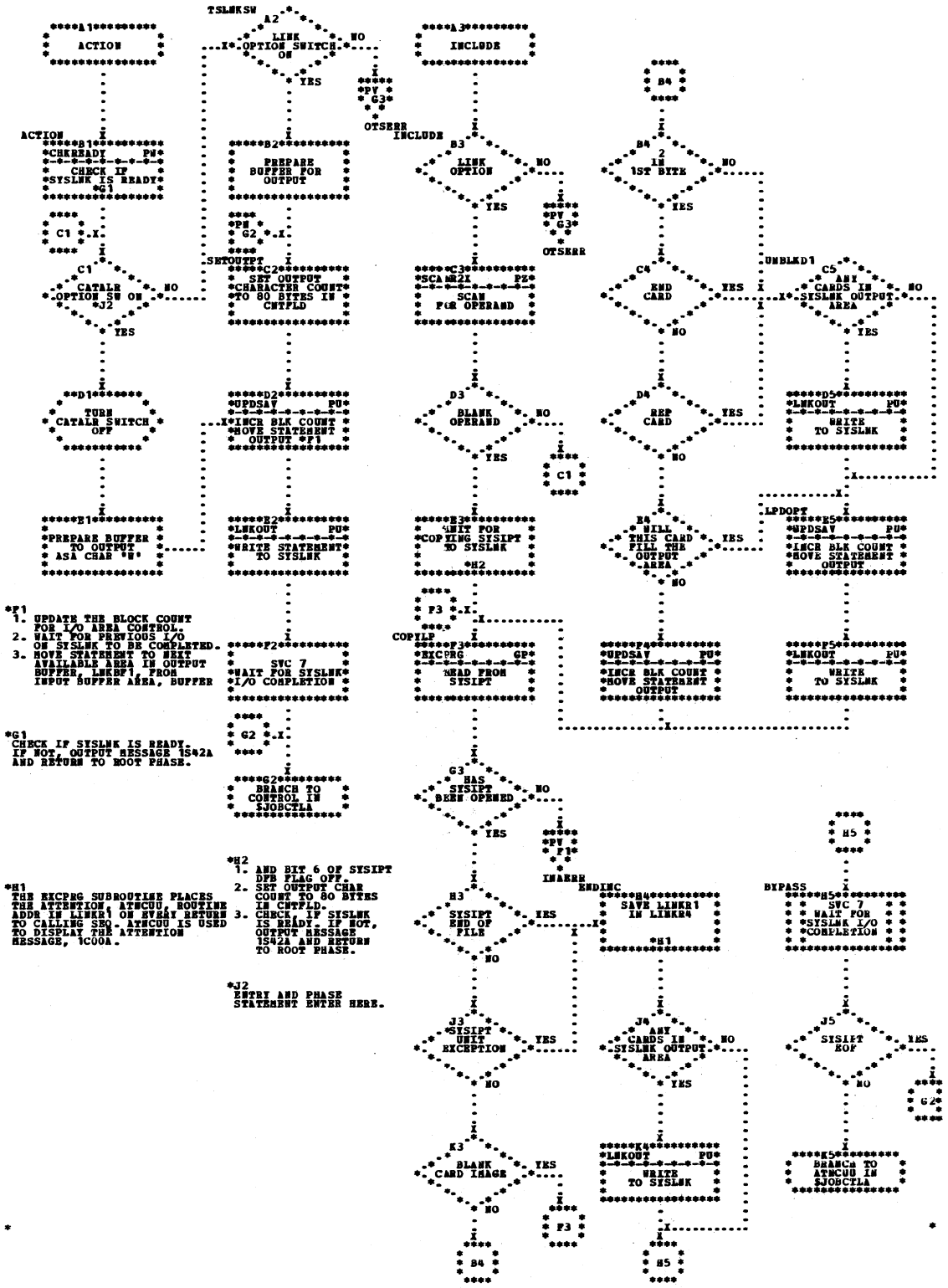


Chart PE. $JOBCTLJ - MTC Statement Processor (Part 1 of 2)
Refer to Charts 18 to 20.

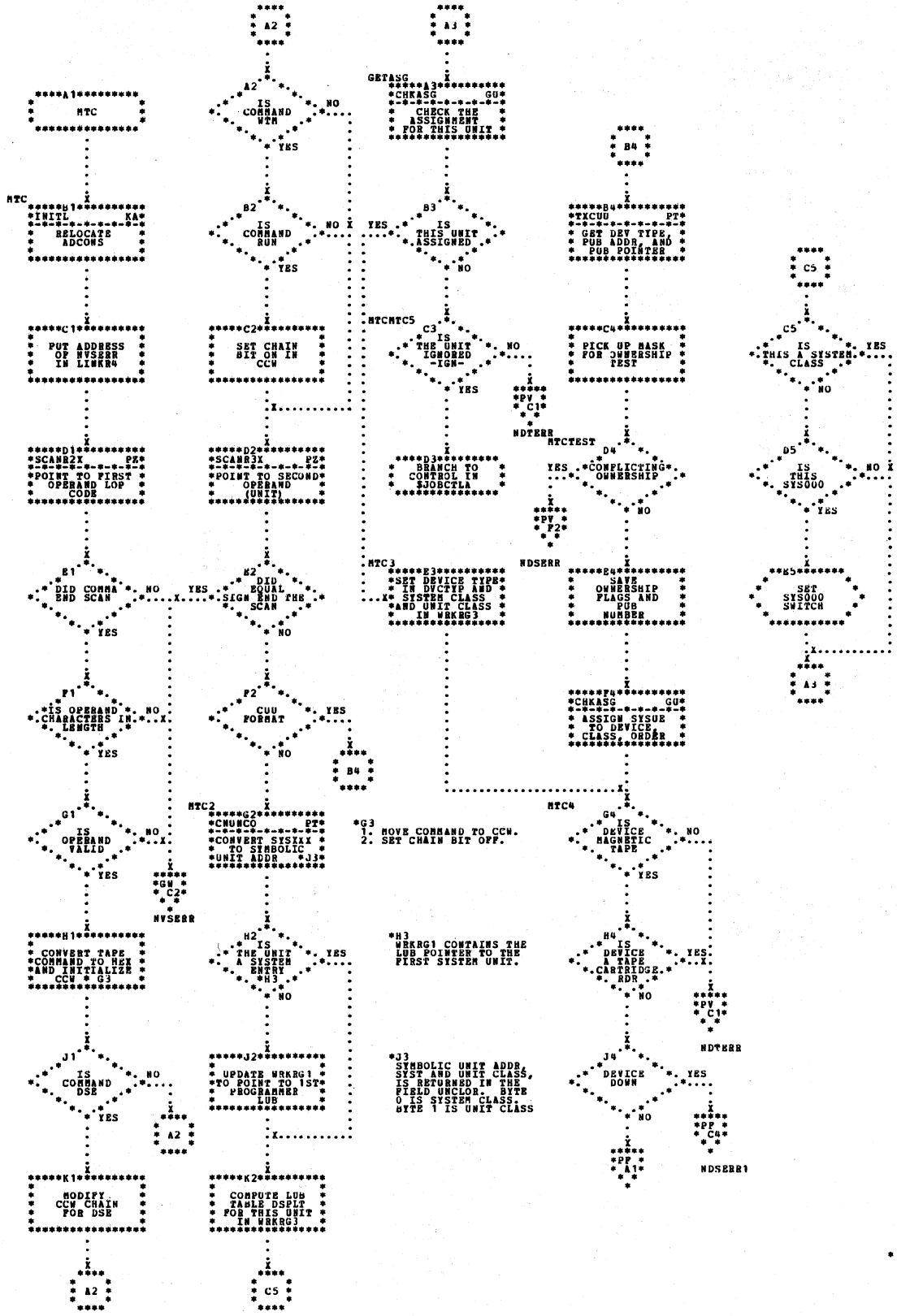


Chart PF. $JOBCTLJ - MTC Statement Processor (Part 2 of 2)
Refer to Charts 18 to 20.

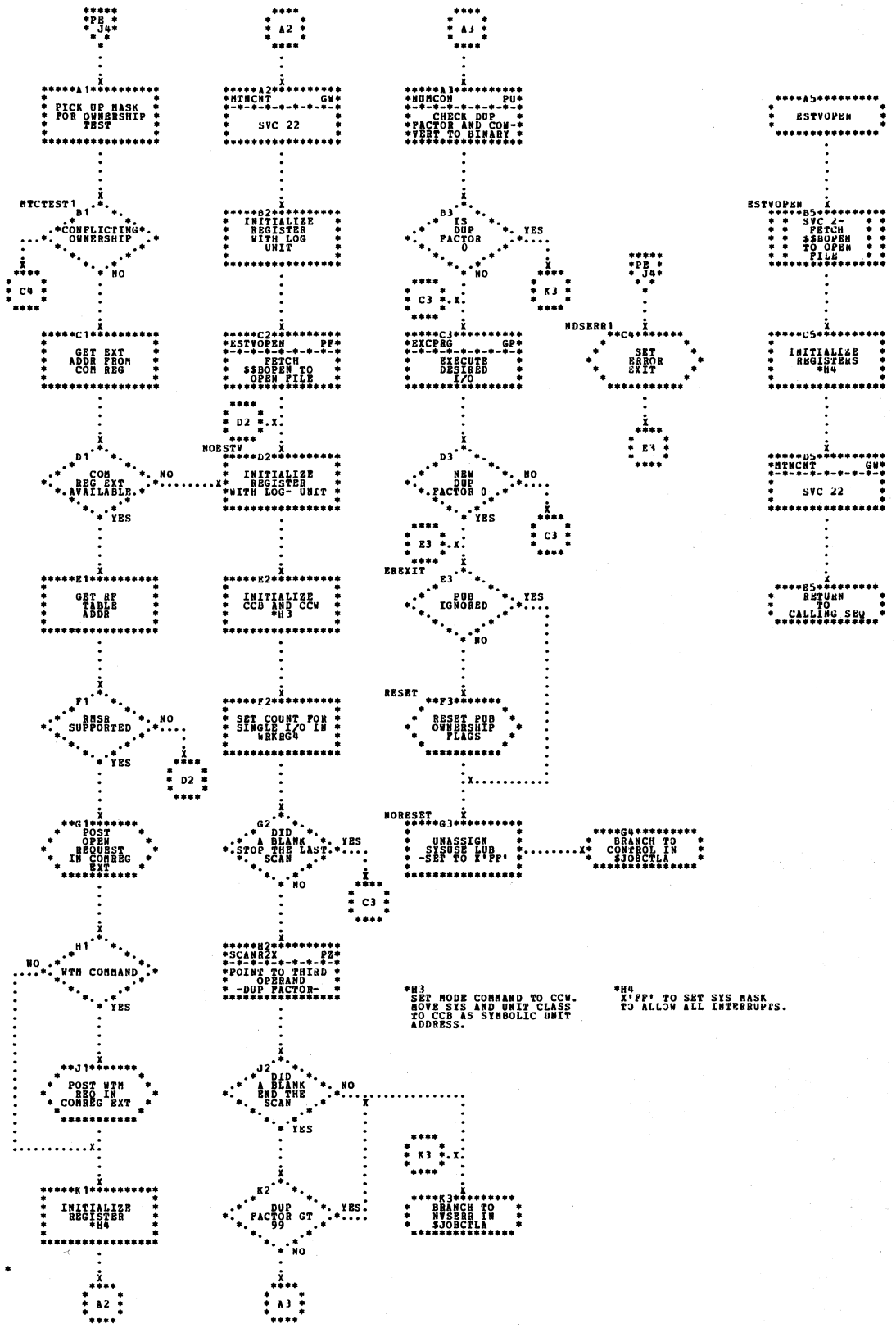


Chart PG. $JOBCTLJ - SET Statement Processor (Part 1 of 3)
Refer to Charts 18 to 20.

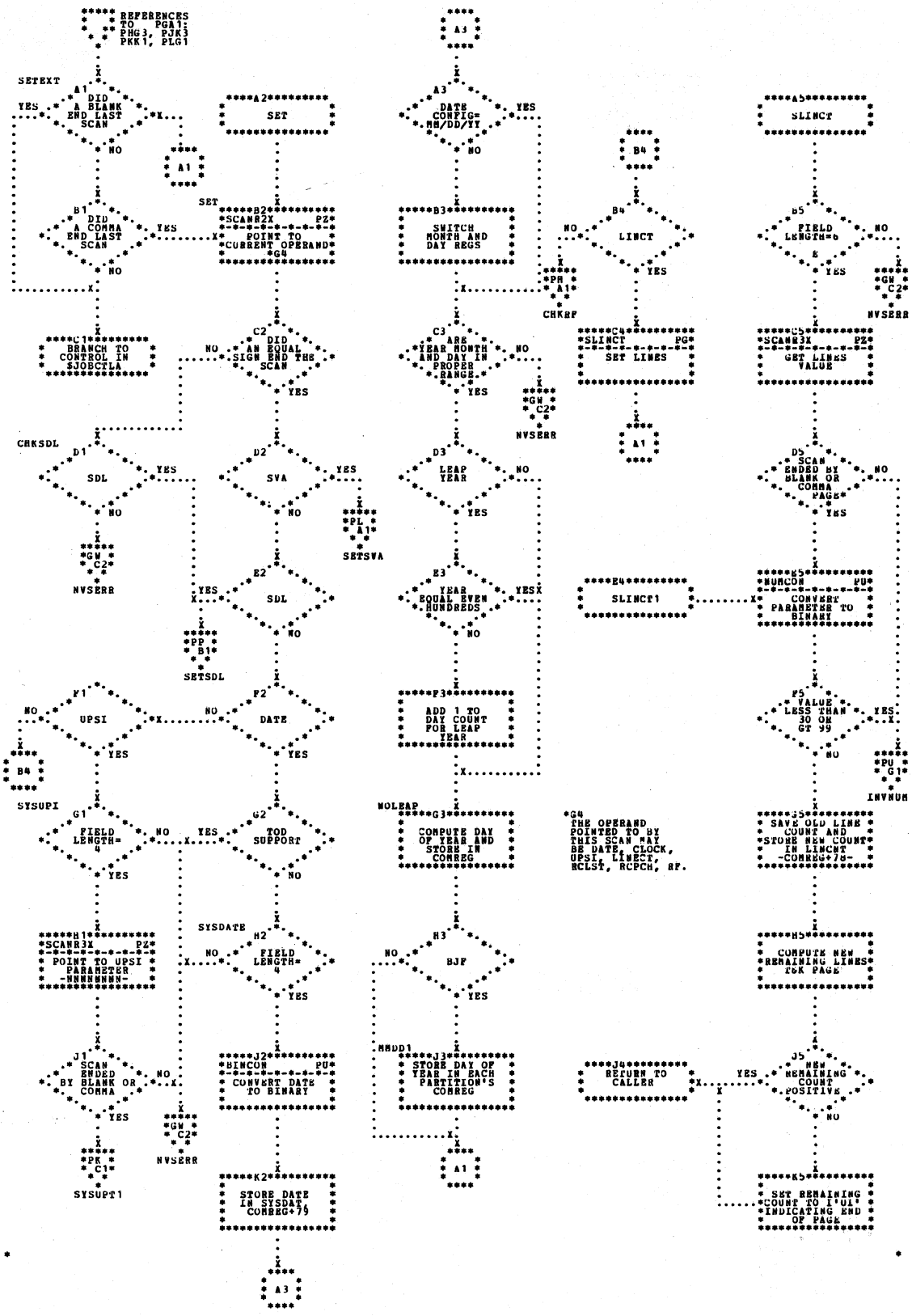


Chart PH. $JOBCTLJ - SET Statement Processor (Part 2 of 3)
Refer to Charts 18 to 20.

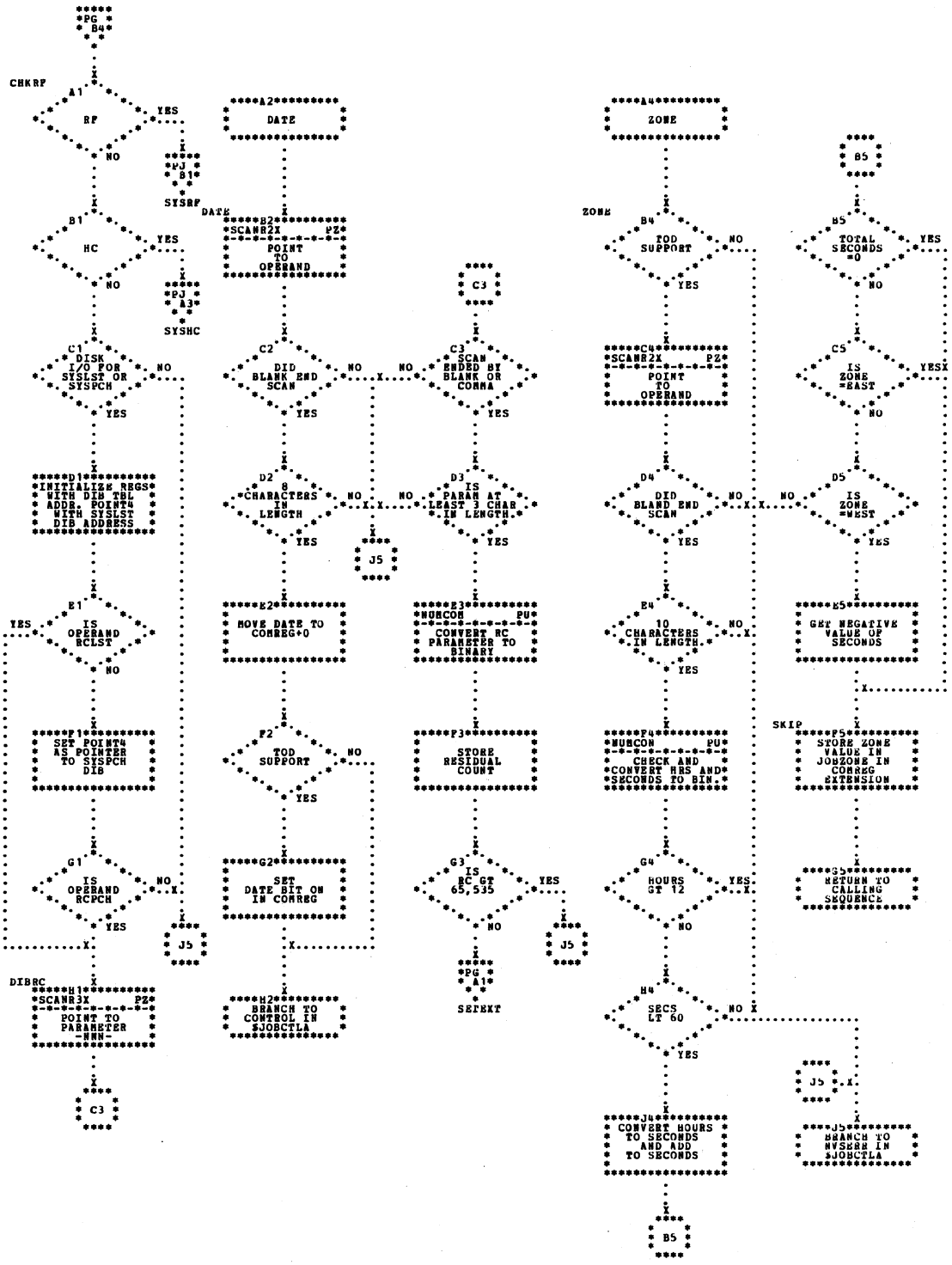


Chart PJ. $JOBCTLJ - SET Statement Processor (Part 3 of 3)
Refer to Charts 18 to 20.

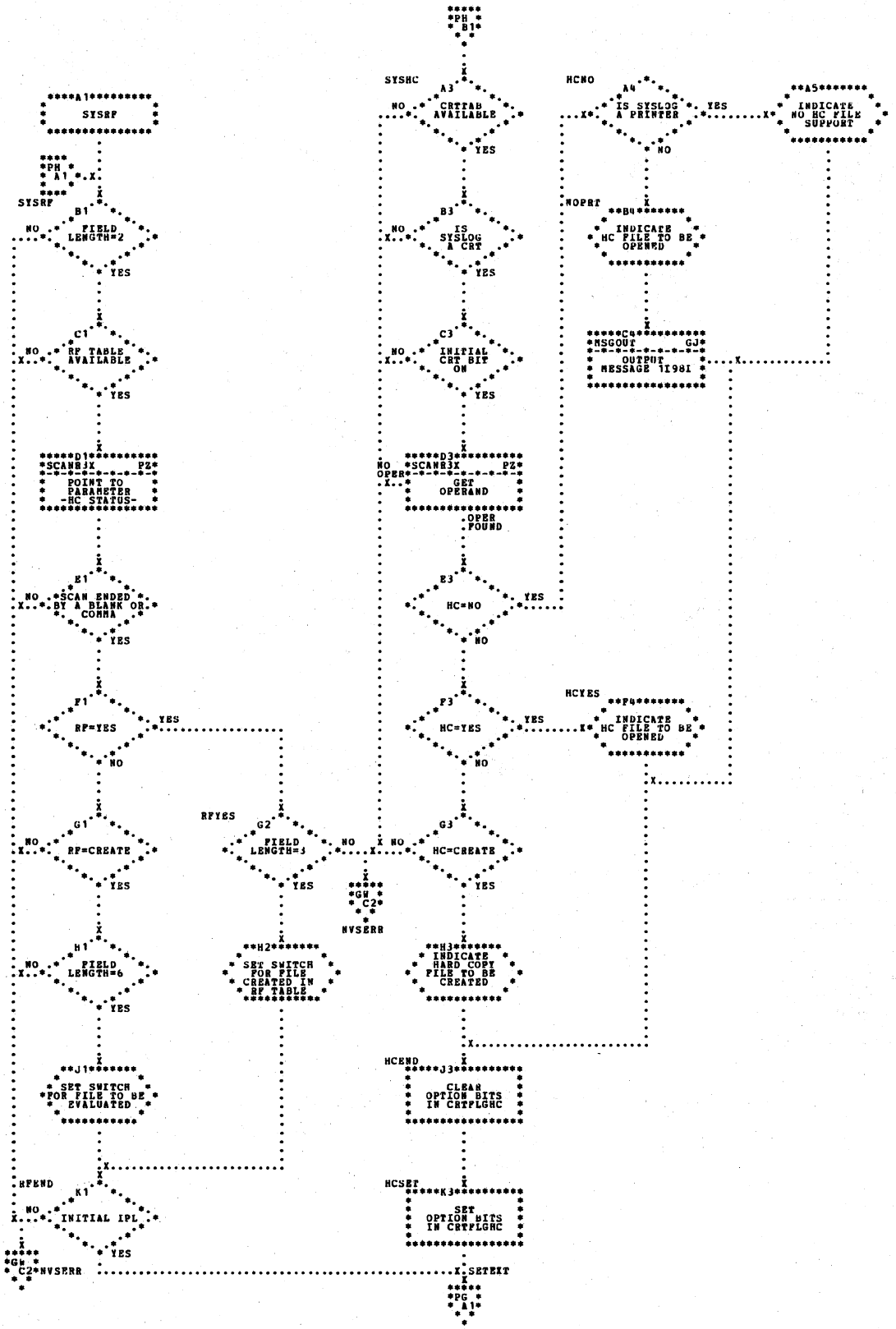


Chart PK. $JOBCTLJ - UPSI, PAUSE, LOG, and NOLOG Command
Refer to Charts 18 to 20.

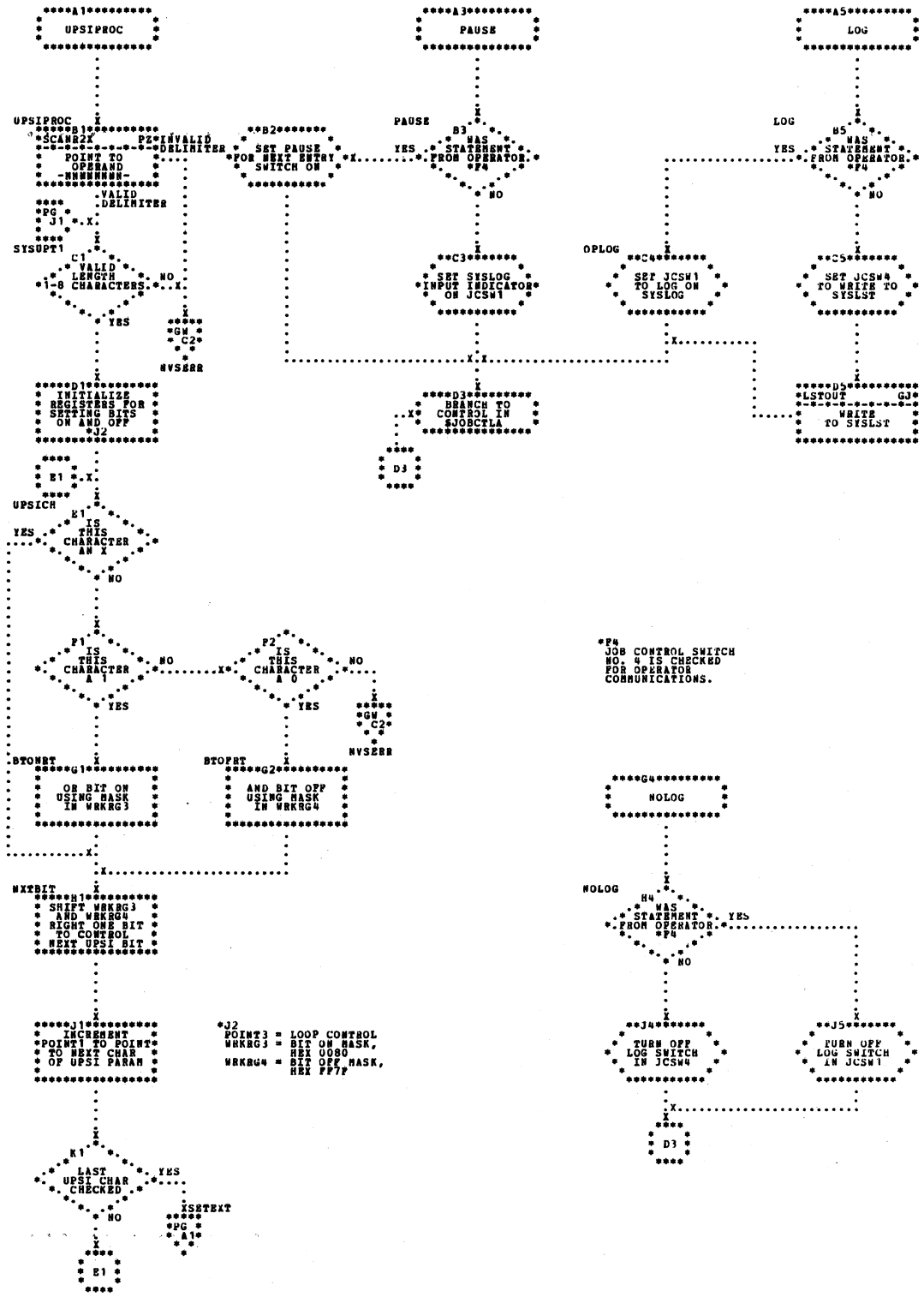


Chart PL. $JOBCTLJ - SET SVA Command Compatibility Processor
Refer to Charts 18 to 20.

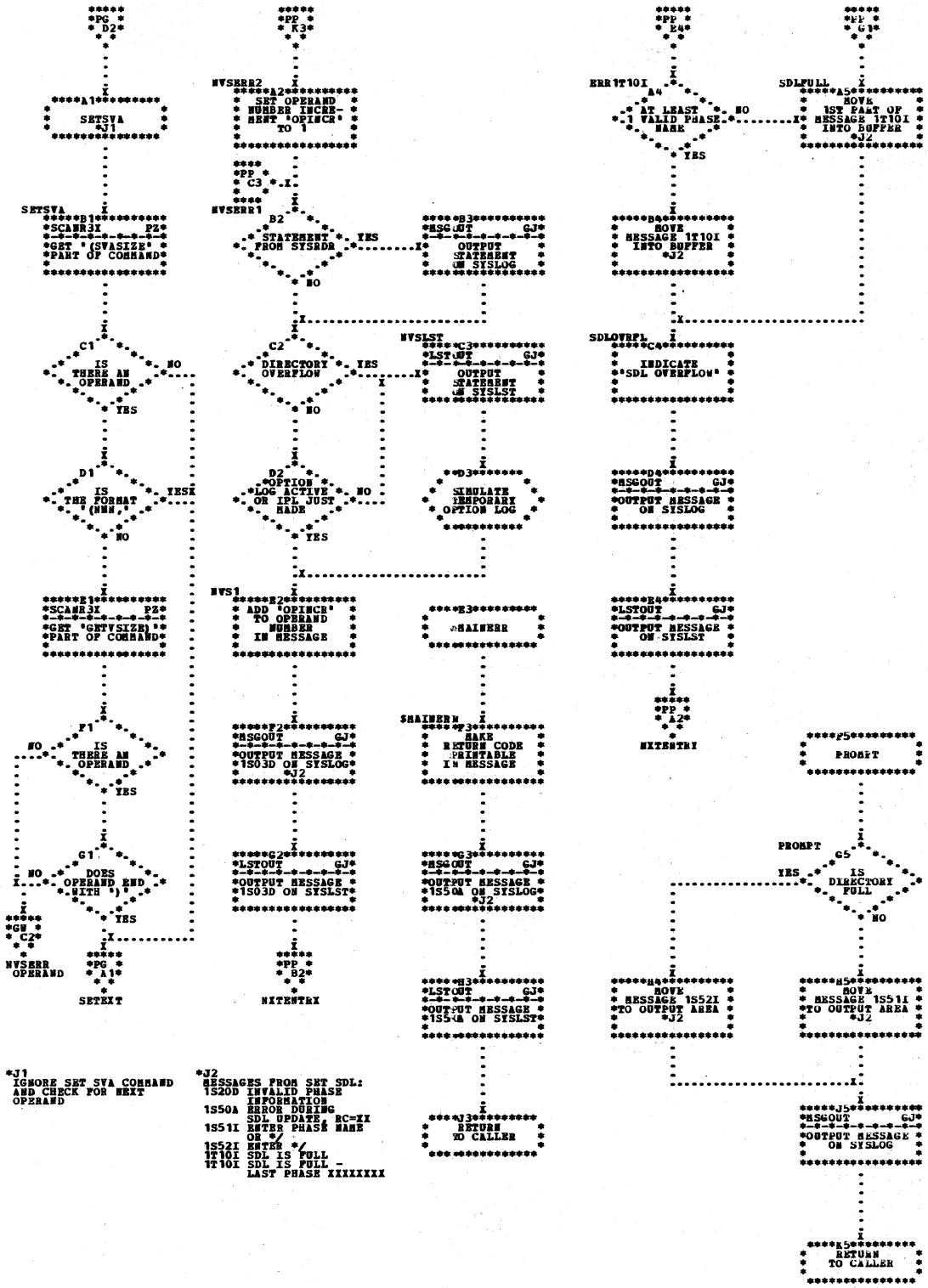


Chart PM. $JOBCTLJ - STOP Command Processor
Refer to Charts 18 to 20.

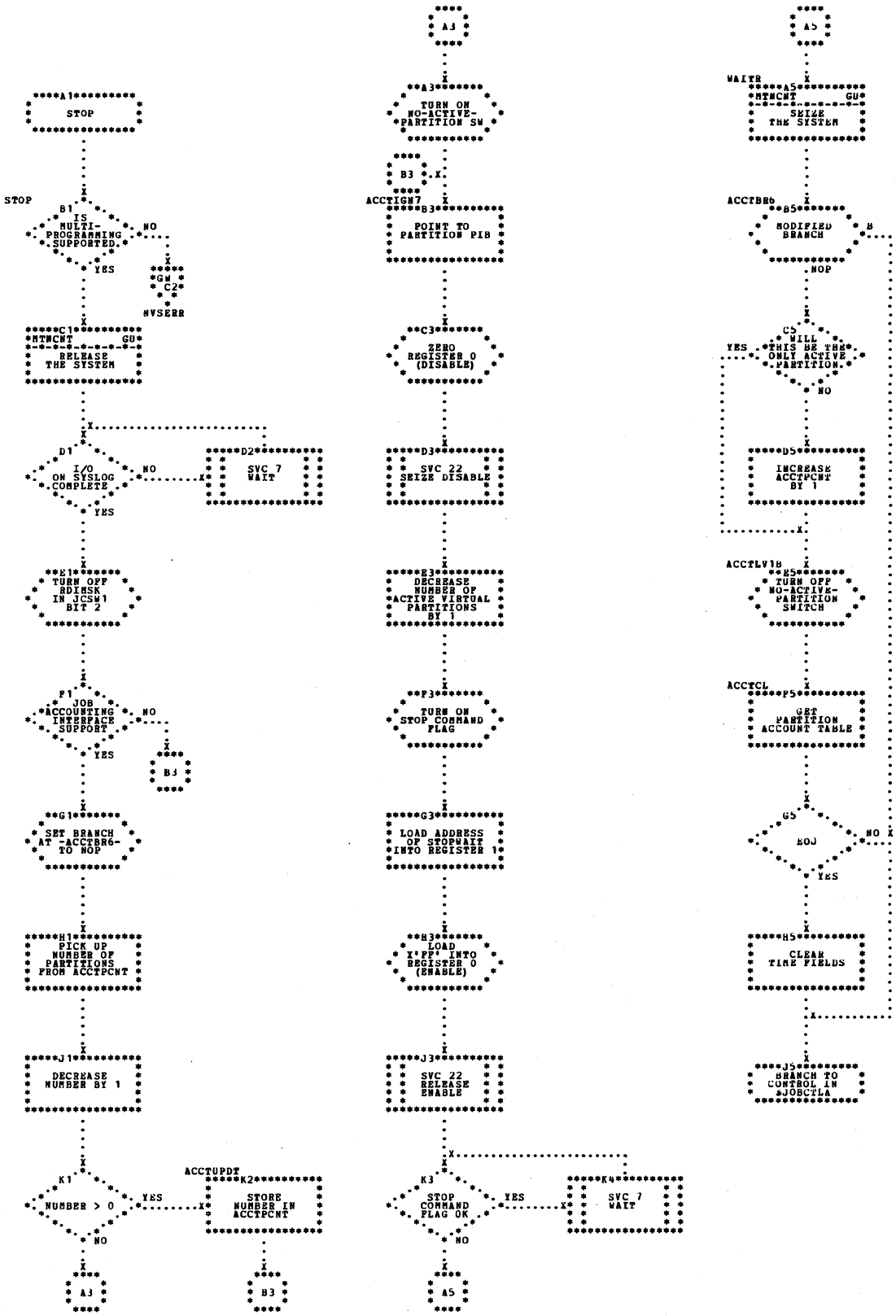


Chart PN. $JOBCTLJ - CATALR Statement Processor
Refer to Charts 18 to 20.

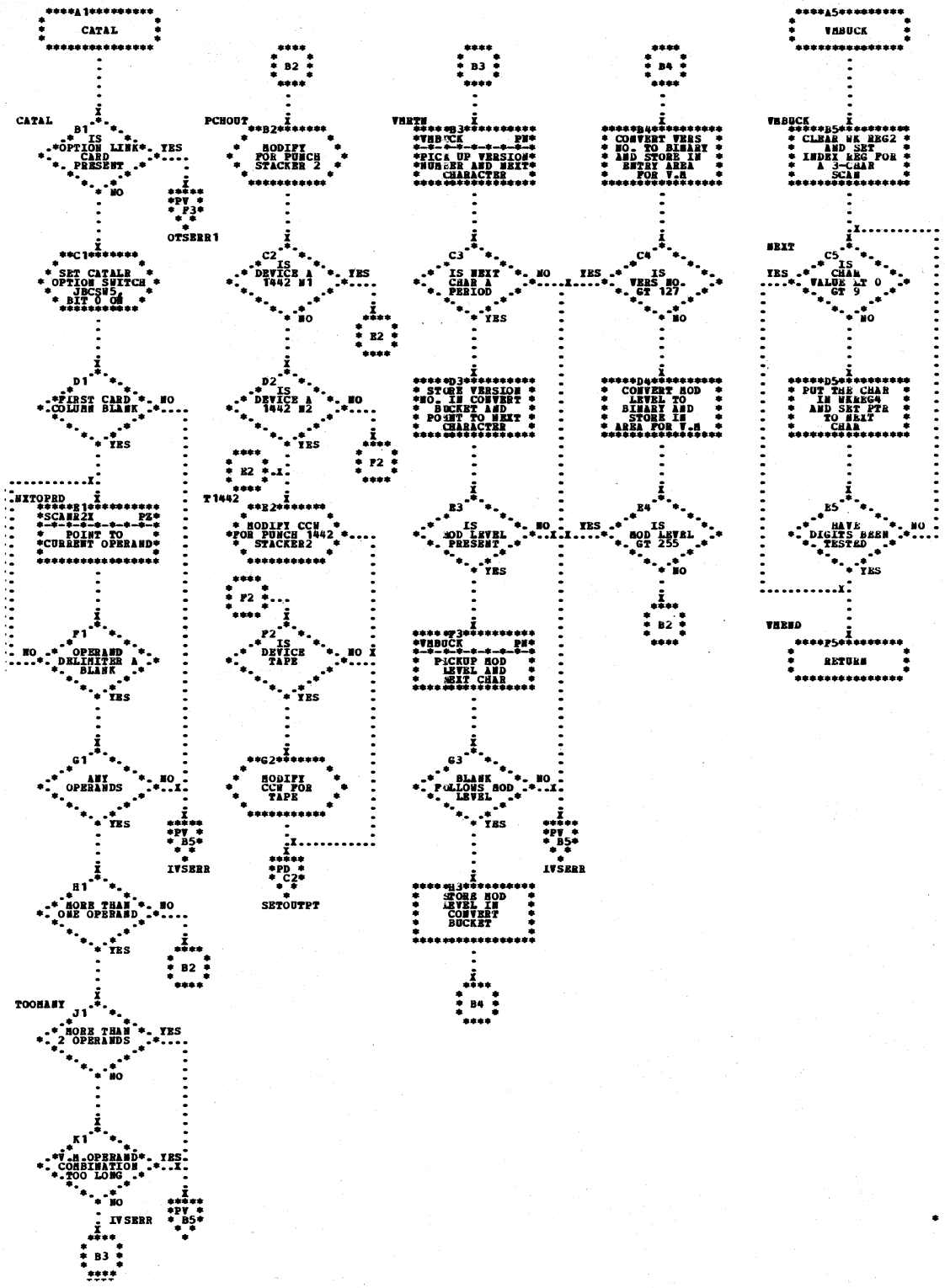


Chart PP. $JOBCTLJ - SETSDL Statement Processor
Refer to Charts 18 to 20.

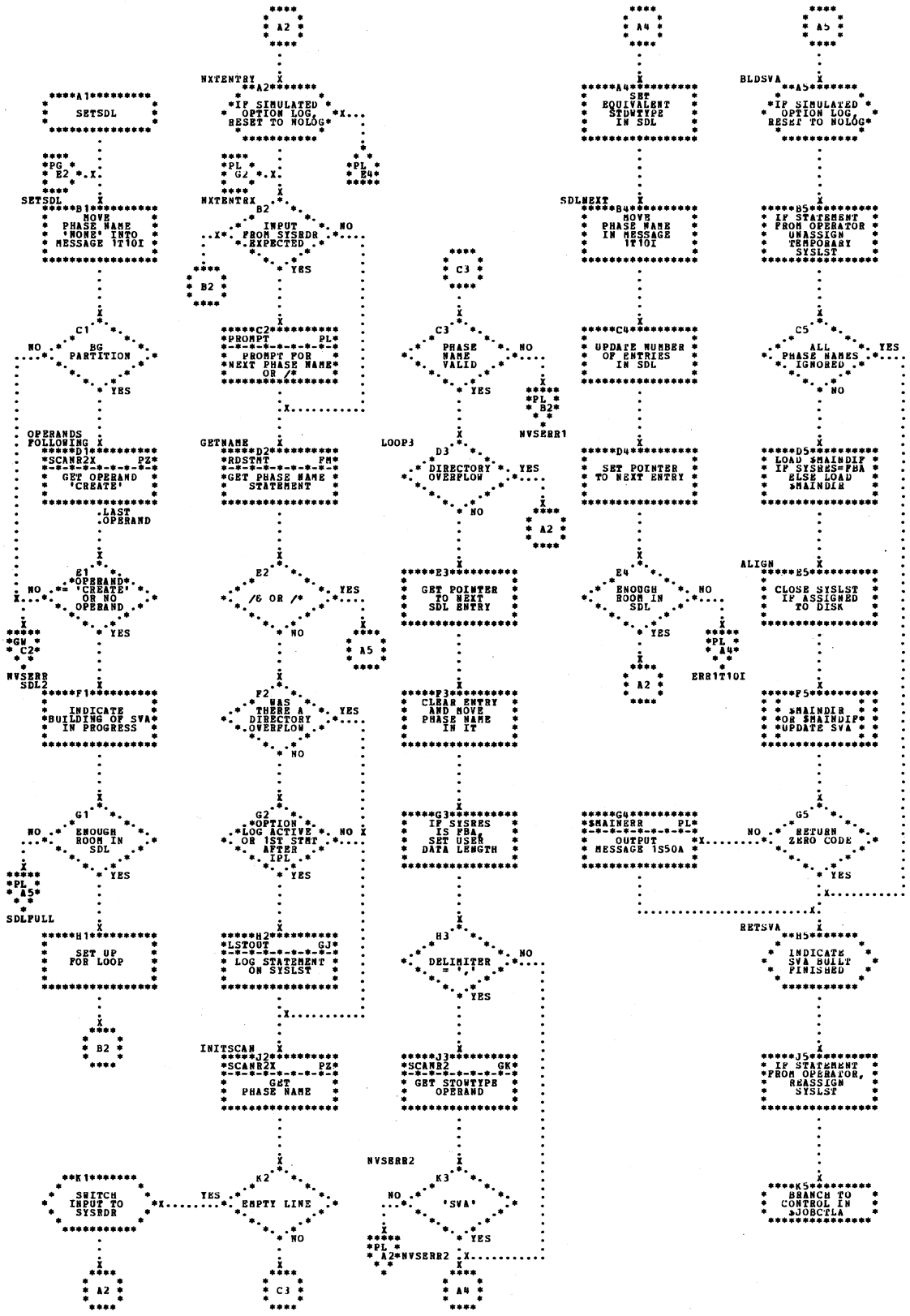


Chart PQ. $JOBCTLJ - ALLOC(R) Statement Processor (Part 1 of 2)
Refer to Charts 18 to 20.

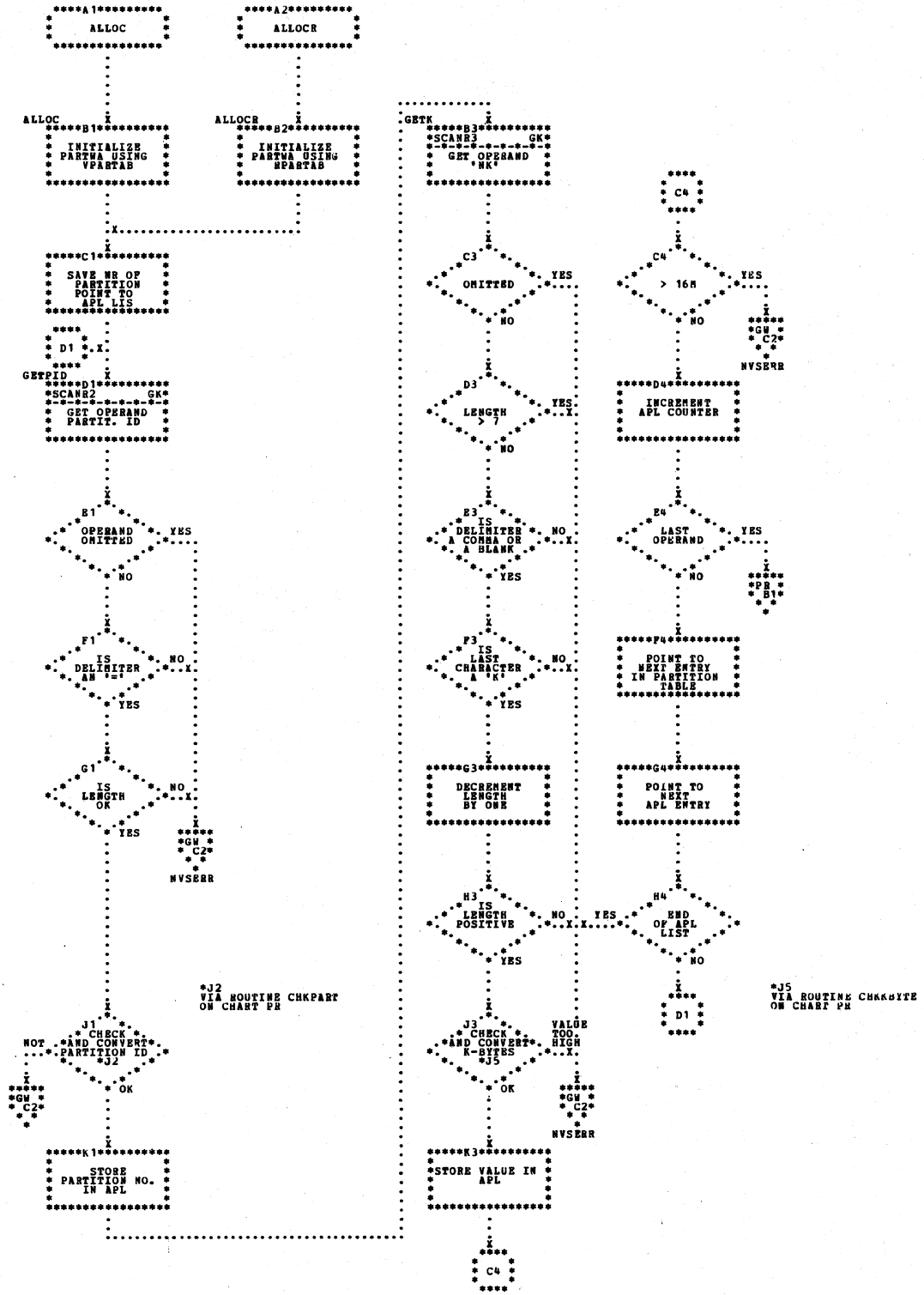


Chart PR. $JOBCTLJ - ALLOC(R) Statement Processor (Part 2 of 2)
Refer to Charts 18 to 20.

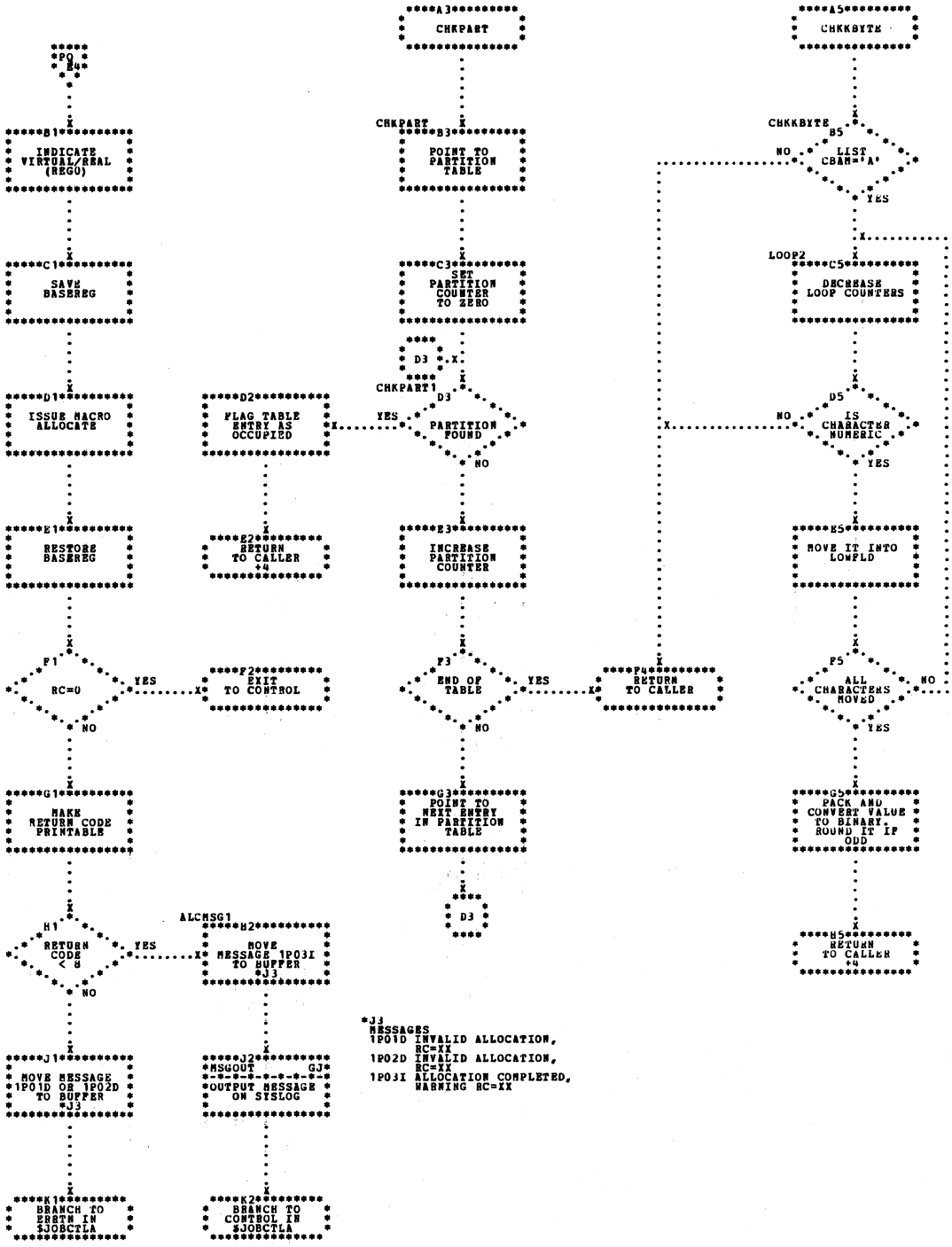


Chart PS. $JOBCTLJ - SIZE Command Processor
Refer to Charts 18 to 20.

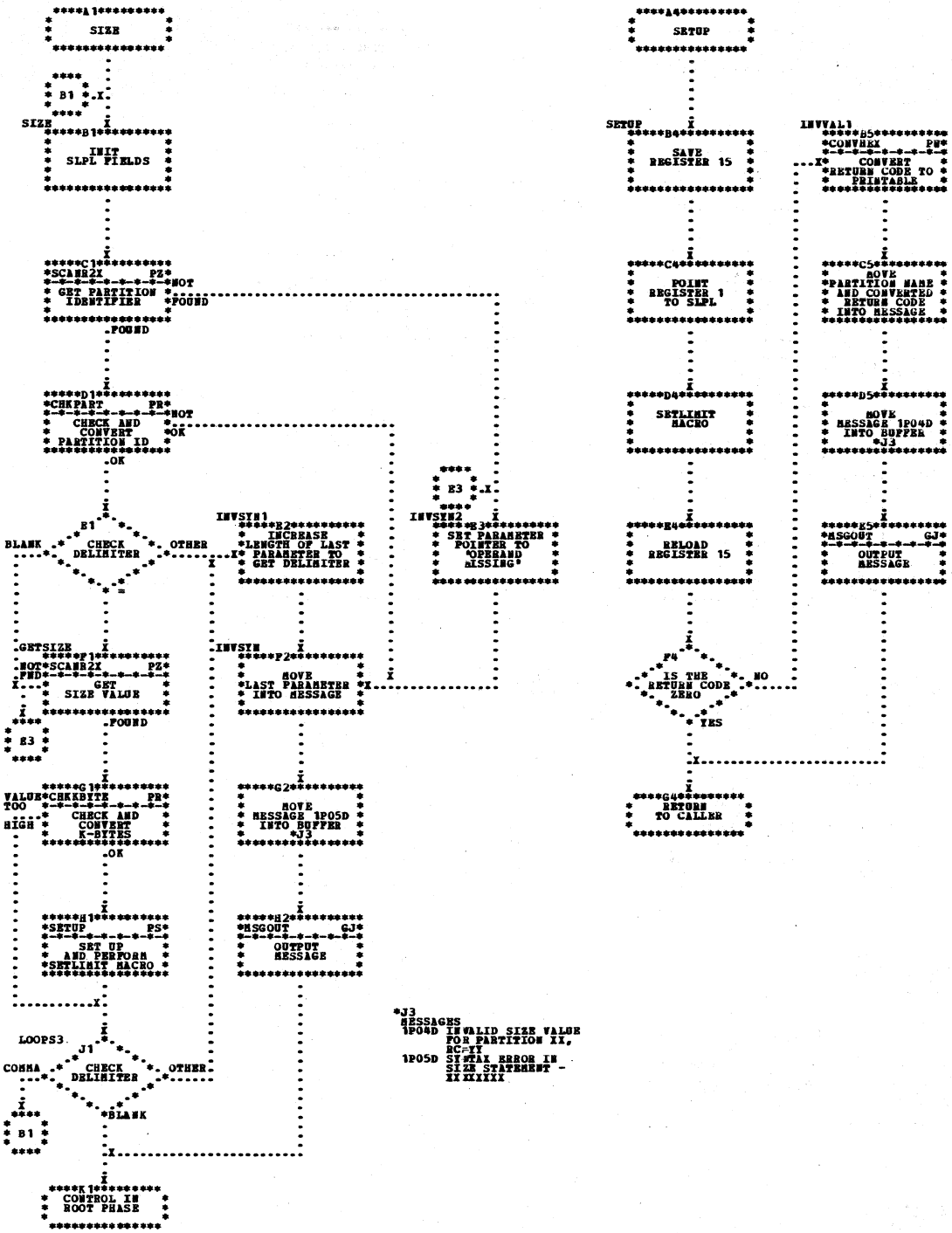
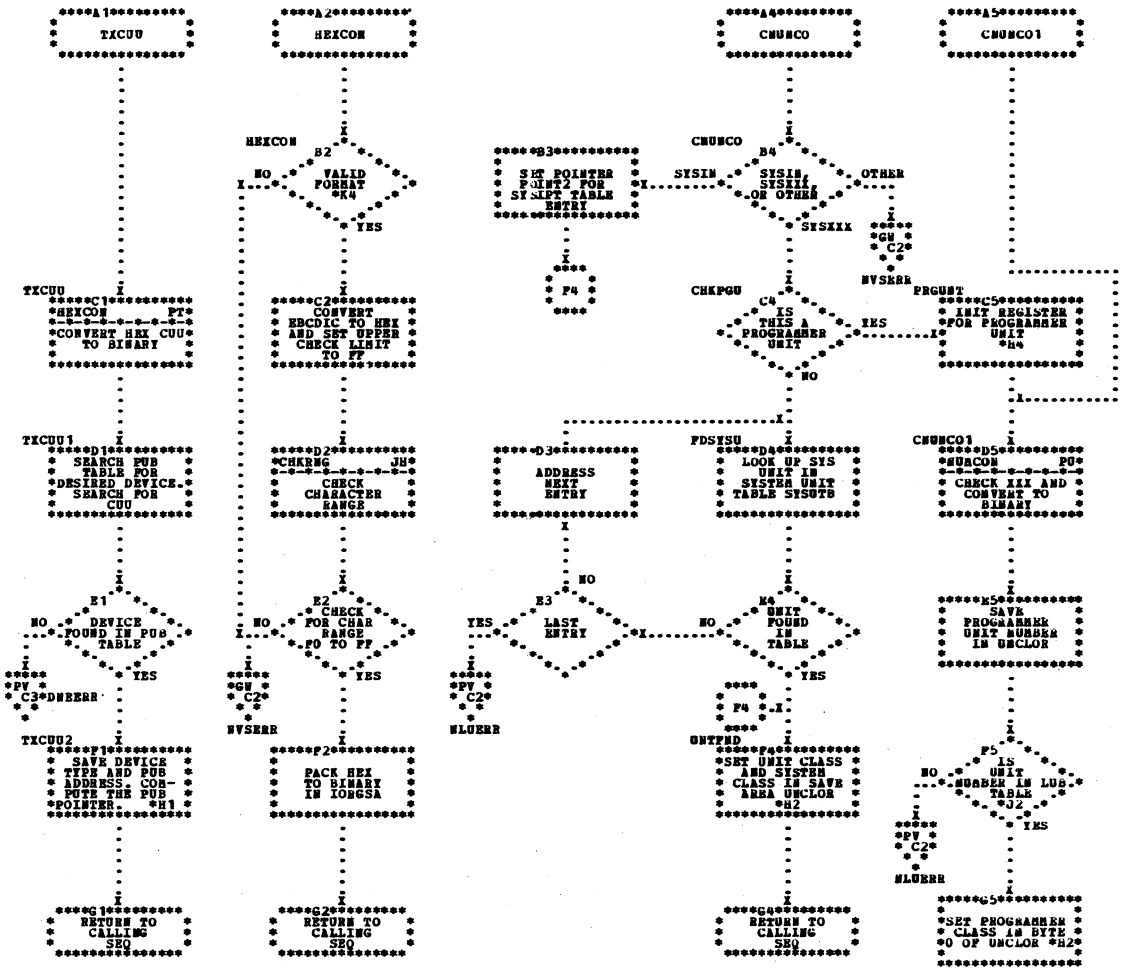


Chart PT. $JOBCTLJ - Subroutines
Refer to Charts 18 to 20.



*H1
1. THE DEVICE TYPE IS SAVED IN DVCTYP FROM THE PUB.
2. THE PUB ADDRESS IS SAVED IN REGISTER POINT4.
3. THE PUB POINTER IS COMPUTED IN WRKRG3.

*H2
THE SYSTEM CLASS, NICL AND FICL POINTER, IS SET TO A 1 IN UNCLOR BYTE 0.

*H4
1. POINT1 IS INCREMENTED BY 3 TO POINT TO THE FIRST CHARACTER OF III.
2. POINT3 IS INITIALIZED TO A 2 AS A CHARACTER COUNT REGISTER.

*J2
1. UNCLOR, BYTE 1, THE UNIT CLASS, LUB POINTER, IS THE FUNCTION OF THE SYS ARGUMENT IN THE TABLE SYSUTB.
2. UNCLOR, BYTE 0, THE SYSTEM CLASS, NICL AND FICL POINTER, IS SET TO ZERO BECAUSE THE ROUTINE IS WORKING WITH A SYSTEM UNIT.

*K2
CHECK ONLY LUBS FOR THIS PARTITION

*K4
VALID FORMATS ARE: X'HHH' OR HHH. IN THE SECOND FORMAT, LEADING ZEROS CAN BE OMITTED.

Chart PU. $JOBCTLJ - Subroutines
Refer to Charts 18 to 20.

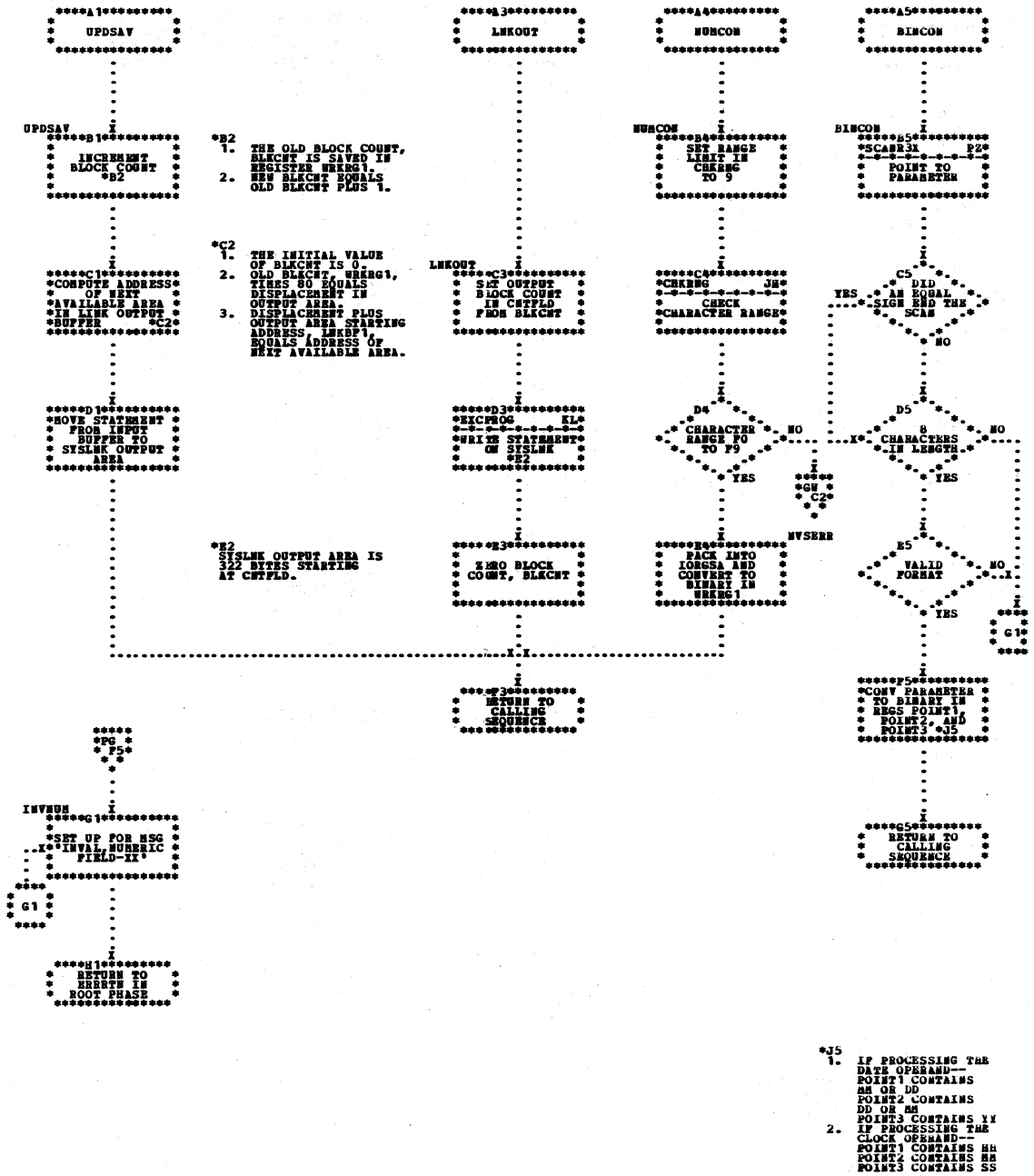
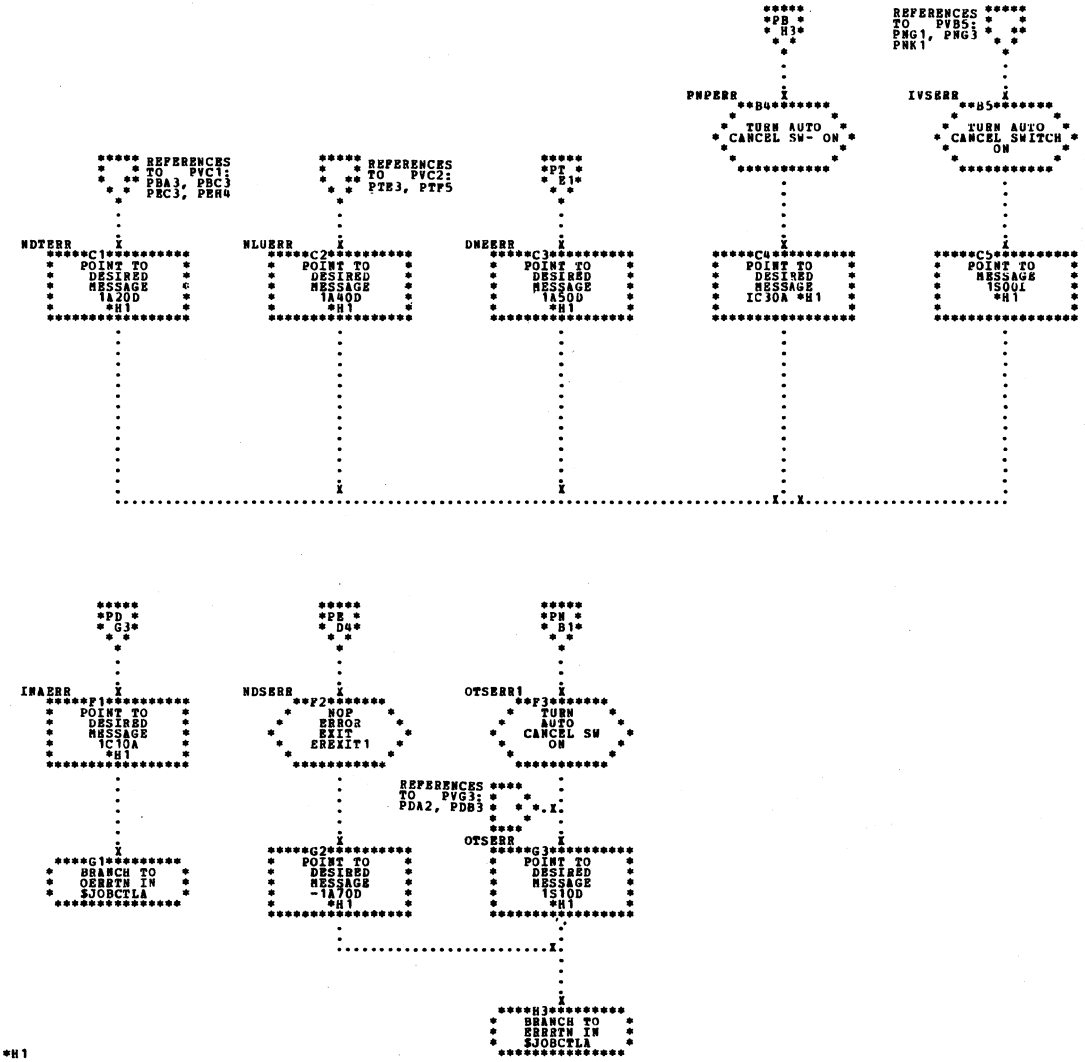


Chart PV. $JOBCTLJ - Error Subroutines
Refer to Charts 18 to 20.



*H1
MESSAGE. DESCRIPTION
1A20D INVALID DEVICE TYPE
1A40D INVALID LOGICAL UNIT SPECIFICATION
1A50D DEVICE NOT-DEFINED
1A70D INVALID DEVICE STATUS
1C10A PLEASE ASSIGN-SYSRDR,SYSIPT,SYSLNK-
1C30A PROGRAM NOT FOUND
1S00I INVALID STATEMENT
1S10D STATEMENT OUT OF SEQUENCE

*

Chart PW. $JOBCTLJ - Error Subroutines
Refer to Charts 18 to 20.

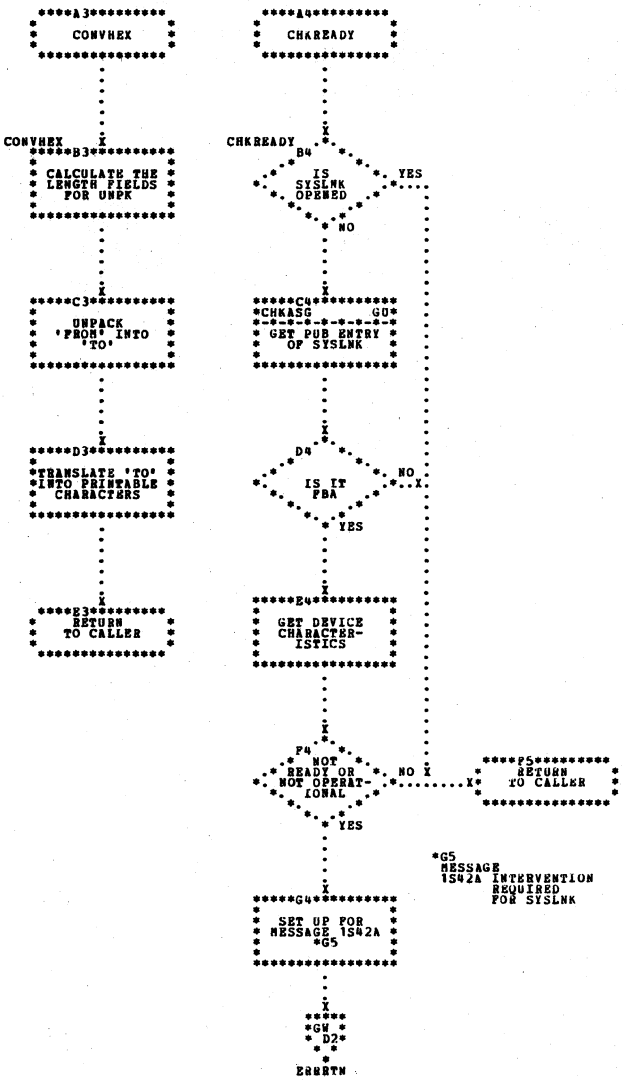


Chart PX. $JOBCTLJ - STDOPT Processor (Part 1 of 2)
Refer to Chart 19.

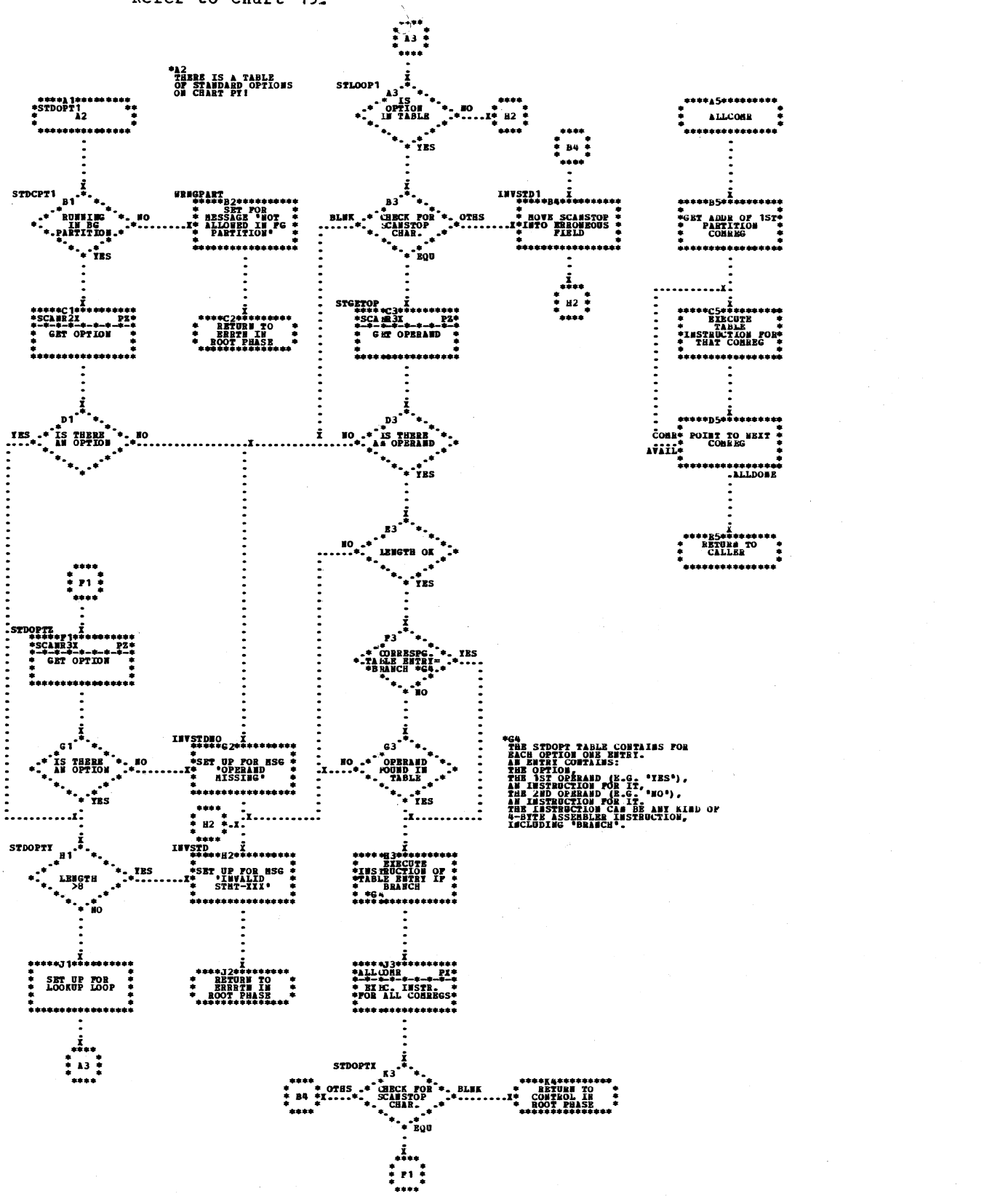


Chart PY. $JOBCTLJ - STDOPT Processor (Part 2 of 2)
Refer to Chart 19.

The standard options are set by the instructions in the STDOPT Table as follows:

DESCRIPTION OF JOB CONTROL OPTION BYTES IN PART.COMREGS

STANDARD OPTION FIELD	LTACT X'35'	SOB1 X'36'	SOB2 X'37'	SYSLINE X'4E'	STDOPT X'8C'	STDOPT2 X'AE'
TEMPORARY OPTION FIELD	---	JCSW3 X'3A'	JCSW4 X'3B'	RLNCNT+1 PARTIT.	TEMOPT X'8D'	TEMOPT2 X'AF'
PERMANENT OPTIONS	MASKS FOR BIT SETTING					
ALIGN=YES					X'40'	
ACANCEL=YES					X'01'	
CHARSET=48C		X'02'				
DATE=DMY	X'80'					
DECK=YES		X'80'				
DUMP=YES			X'40'			
DUMP=PART					X'20'	
EDECK=YES					X'80'	
ERRS=YES		X'04'				
LINES=YES				NUMBER		
LIST=YES		X'40'				
LISTX=YES		X'20'				
LOG=YES			X'10'			
RLD=YES					X'10'	
SXREF=YES					X'08'	
SYM=YES		X'10'				
XREF=YES		X'08'				
TERM=YES						

NOTE ADDITIONAL FUNCTIONS:
'DUMP=YES' AND 'NO' SETS OFF PARTDUMP INDICATOR
'DUMP=PART' SETS ON DUMP INDICATOR
'XREF=YES' AND 'NO' SETS OFF SXREF INDICATOR
'SXREF=YES' SETS OFF XREF INDICATOR
'LINES=NN' ADAPTS THE JCL-'RLNCNT' IN THE OWN PARTITION

The temporary options are not affected. At next // JOB or /& processing, the (new) standard options are moved into the temporary option fields in $JOBCTLG.

For the DATE operand, the format of the DATE-field will be reset with the next call of the GETTIME macro which is done by /& processing.

Chart PZ. $JOBCTLJ - Subroutines

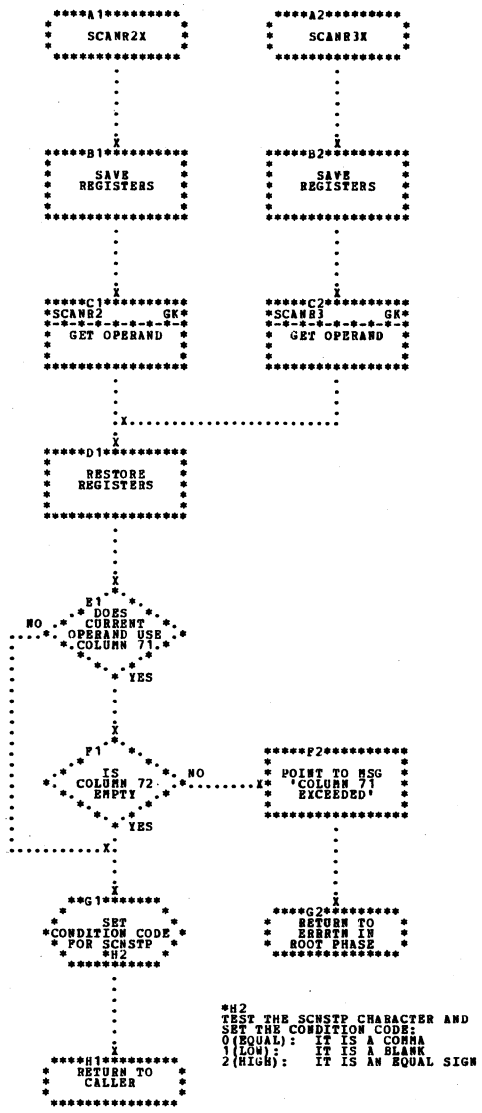


Chart QA. $JOBCTLK - LBLTYP, VOL, TPLAB Command Processors
Refer to Charts 21 to 22.

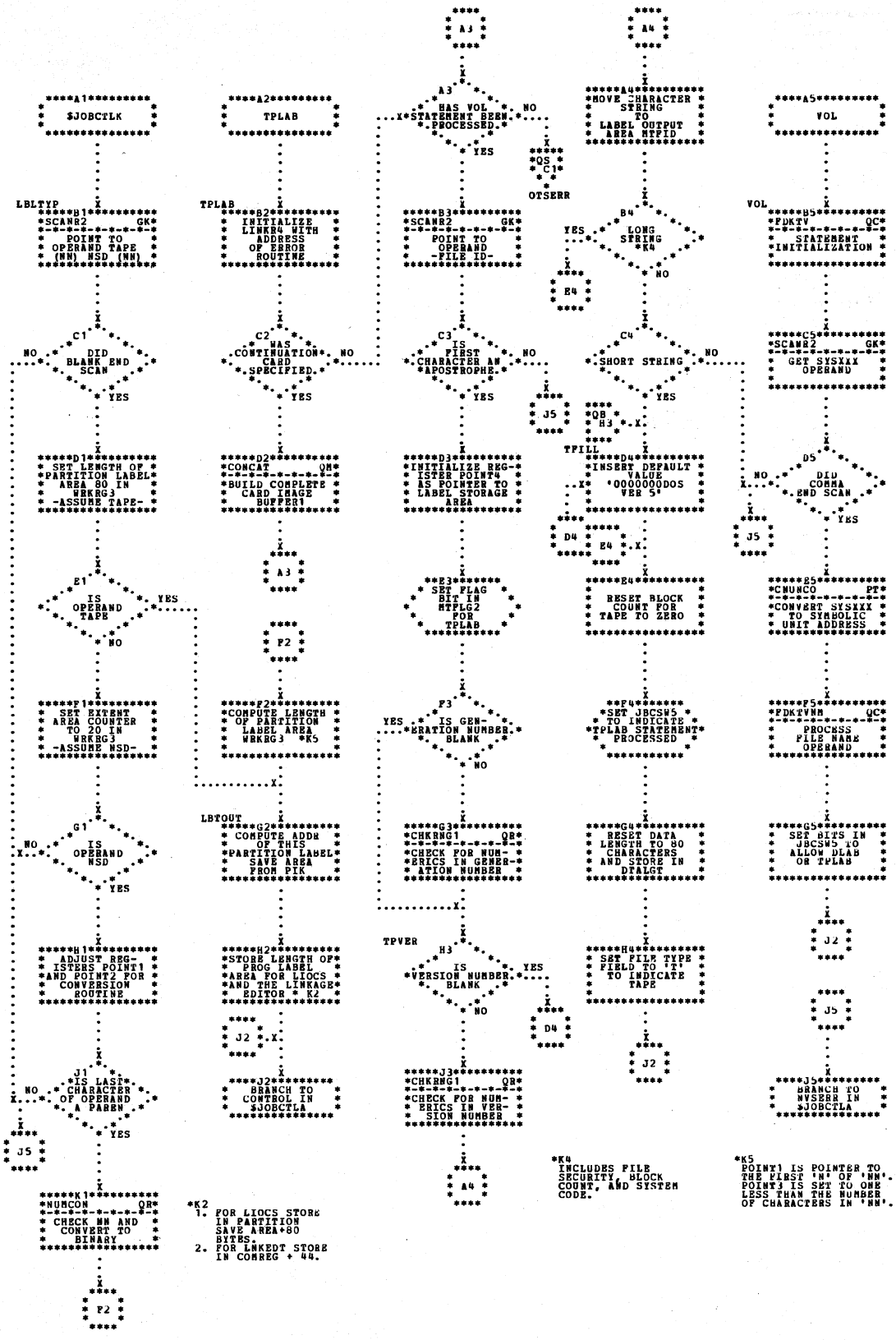
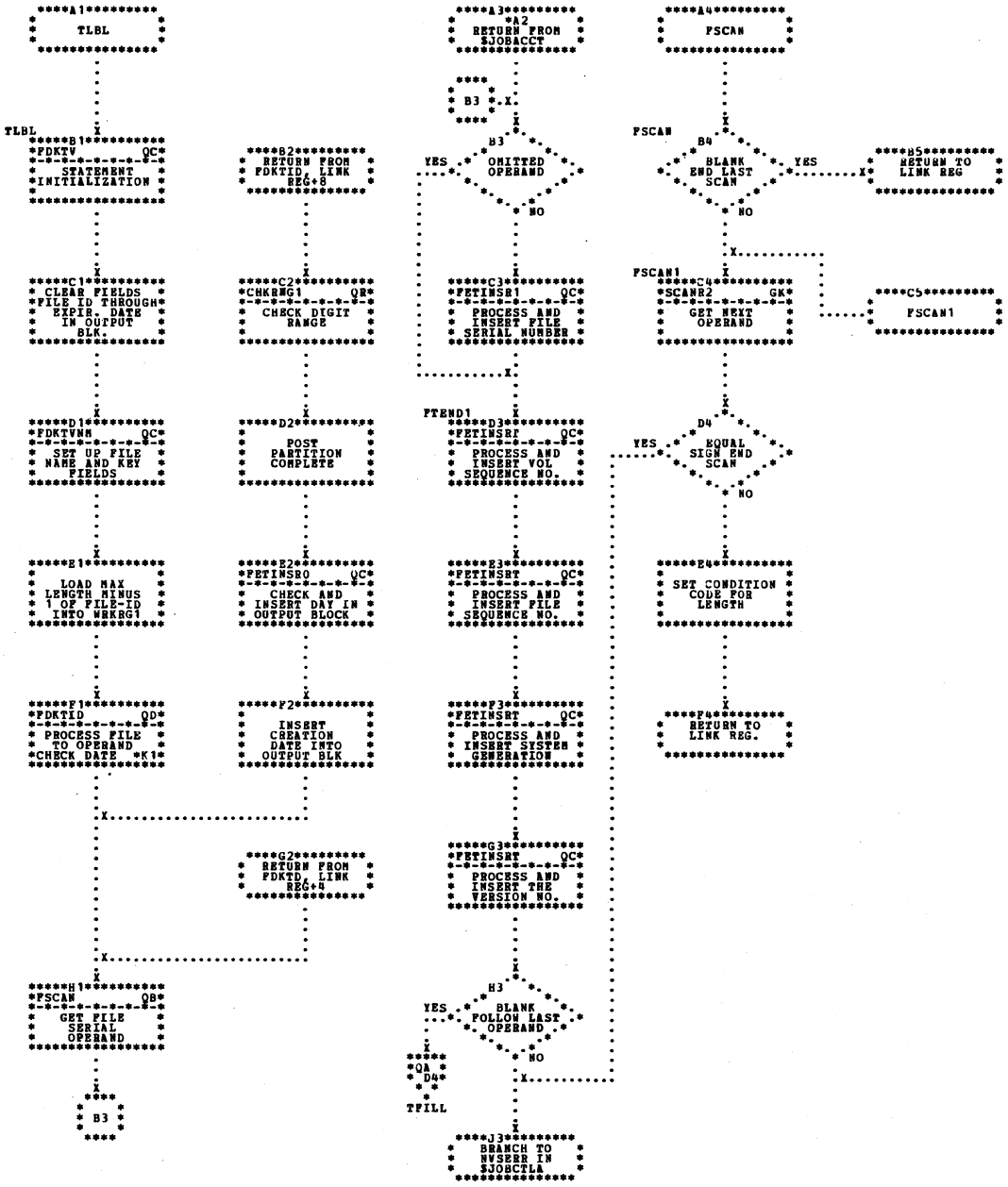


Chart QB. $JOBCTLK - TLBL Command Processor
Refer to Charts 21 to 22.



*K1
RETURN FROM FDKTID IS TO
1. B2 FOR ABSOLUTE DATE (LINK REG+8)
2. G2 FOR OMITTED DATE (LINK REG+4)
3. G1 FOR RETENTION (LINK REG)

Chart QC. $JOBCTLK - Label Processing Subroutines (Part 1 of 2)
Refer to Charts 21 to 22.

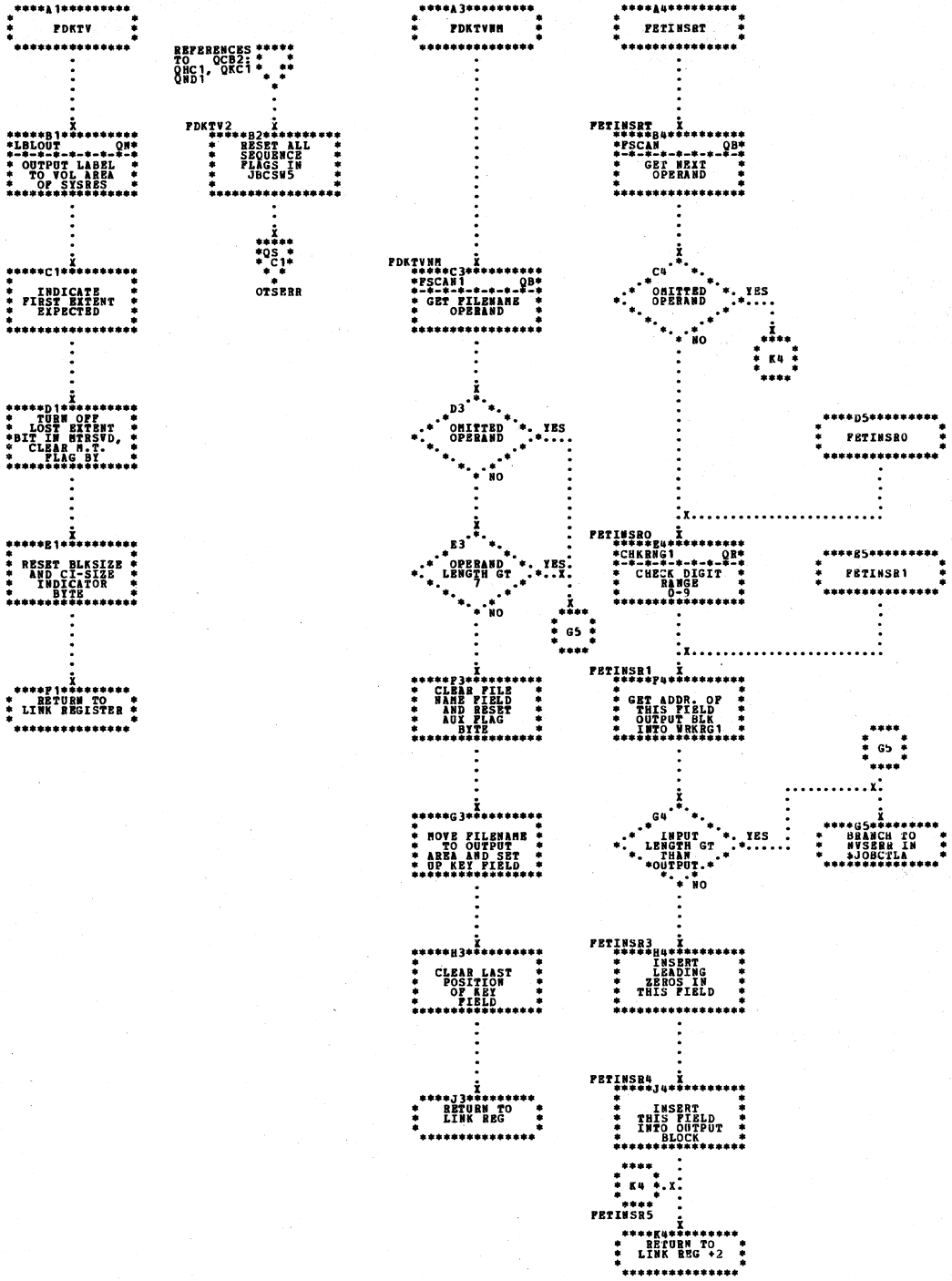


Chart QD. $JOBCTLK - Label Processing Subroutines (Part 2 of 2)
Refer to Charts 21 to 22.

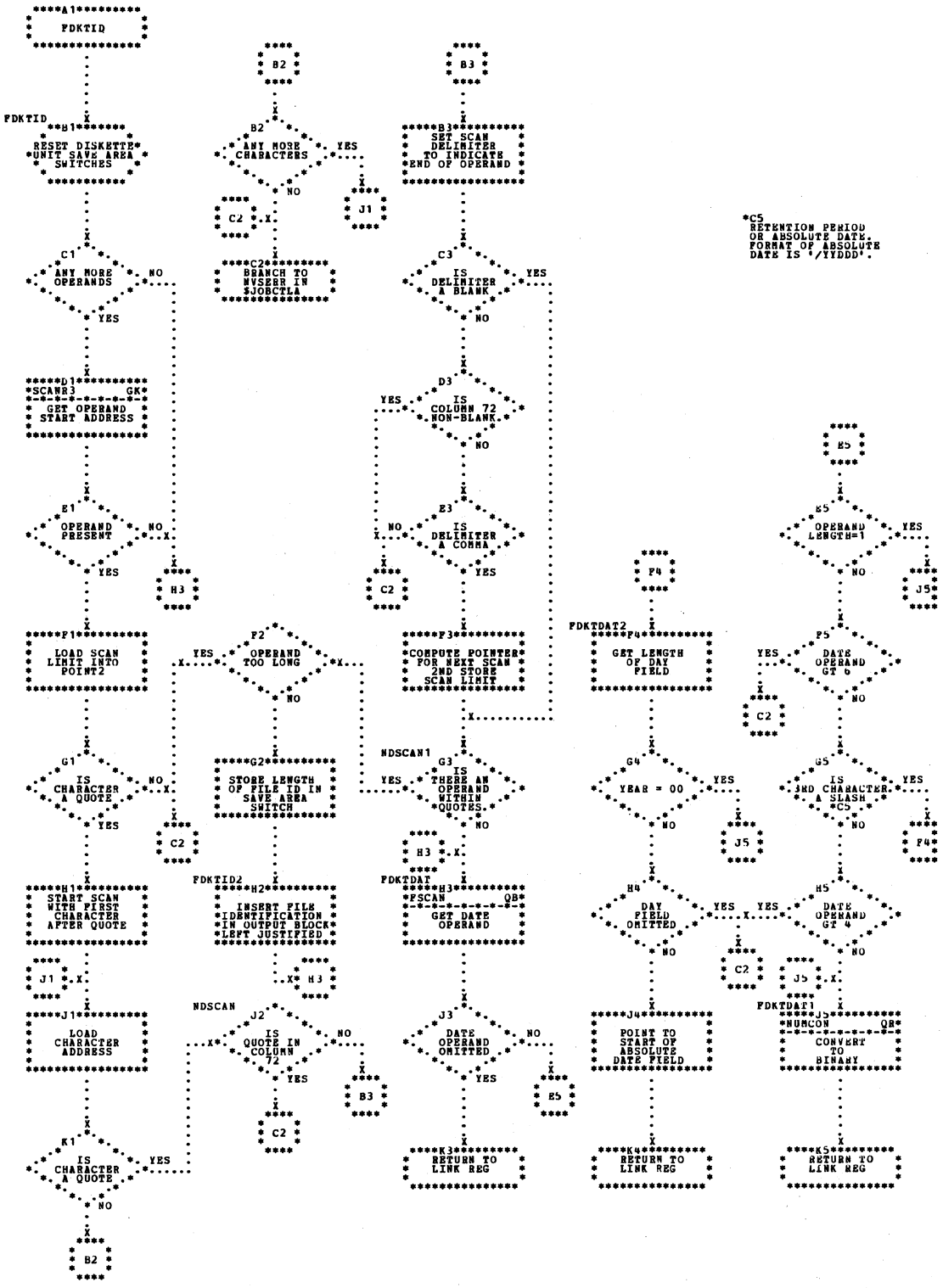


Chart QE. $JOBCTLK - DLBL Command Processor
Refer to Charts 21 to 22.

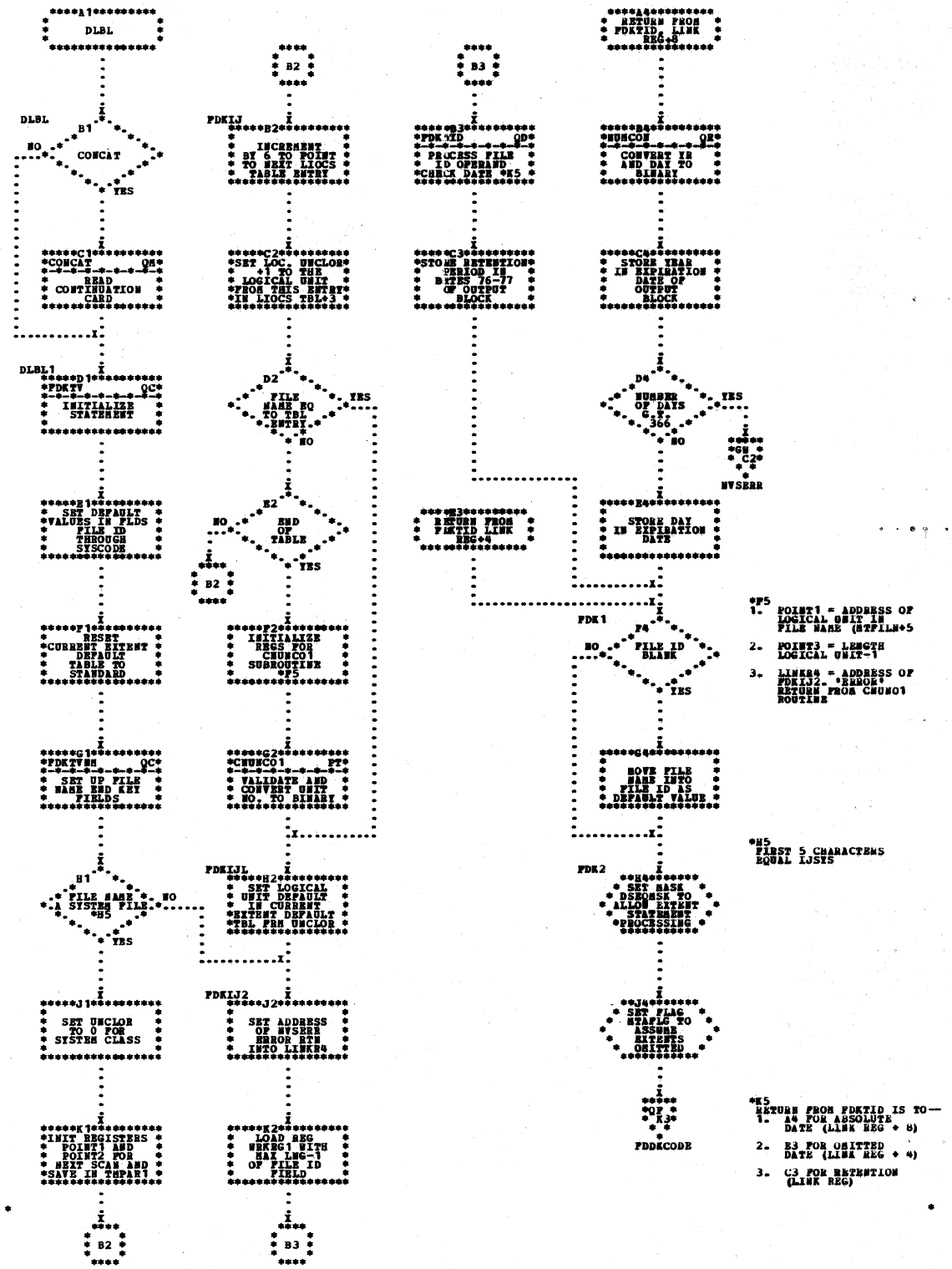


Chart QF. $JOBCTLK - DLAB Command Processor (Part 1 of 2)
Refer to Charts 21 to 22.

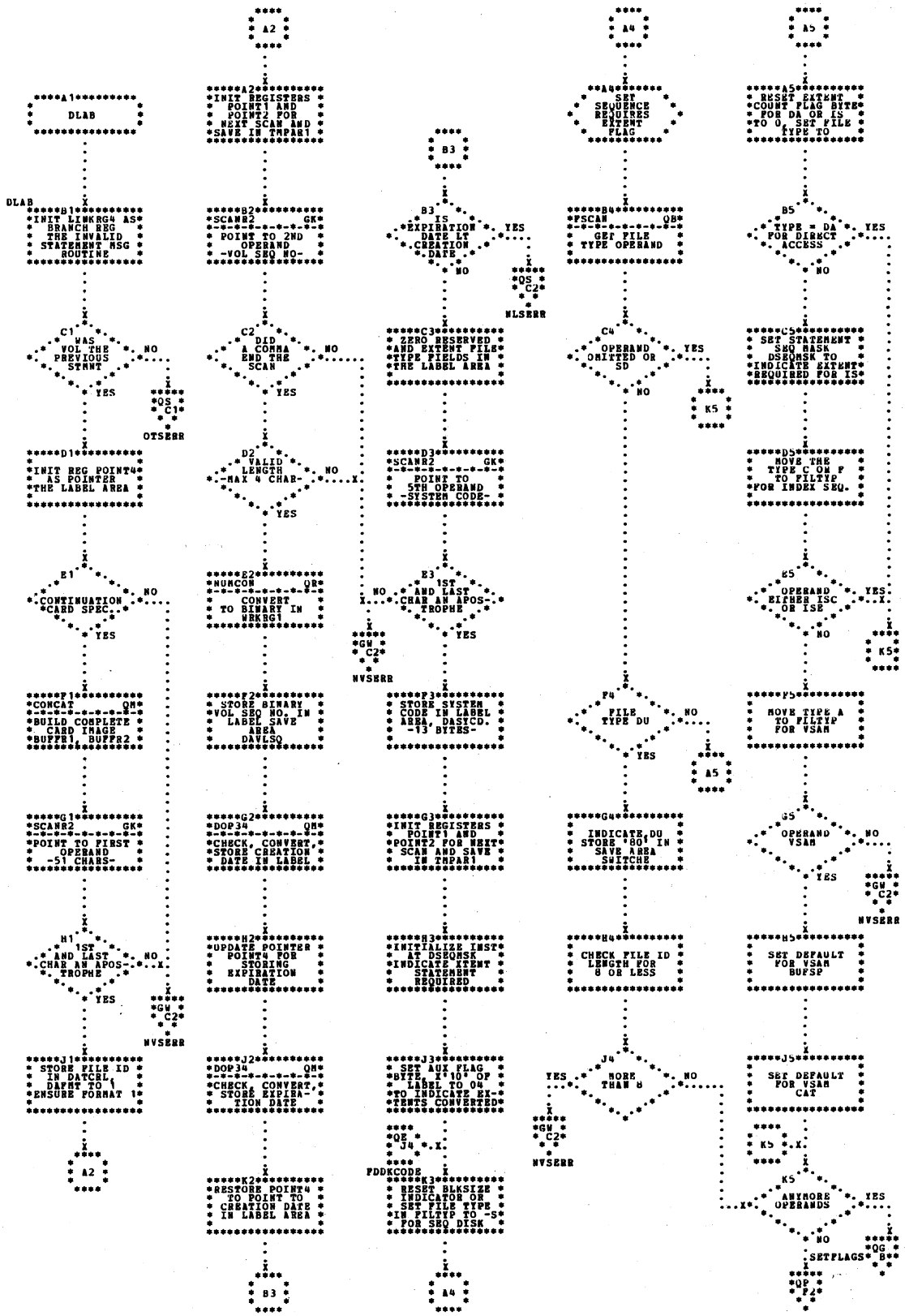


Chart QG. $JOBCTLK - DLAB Command Processor (Part 2 of 2)
Refer to Charts 21 to 22.

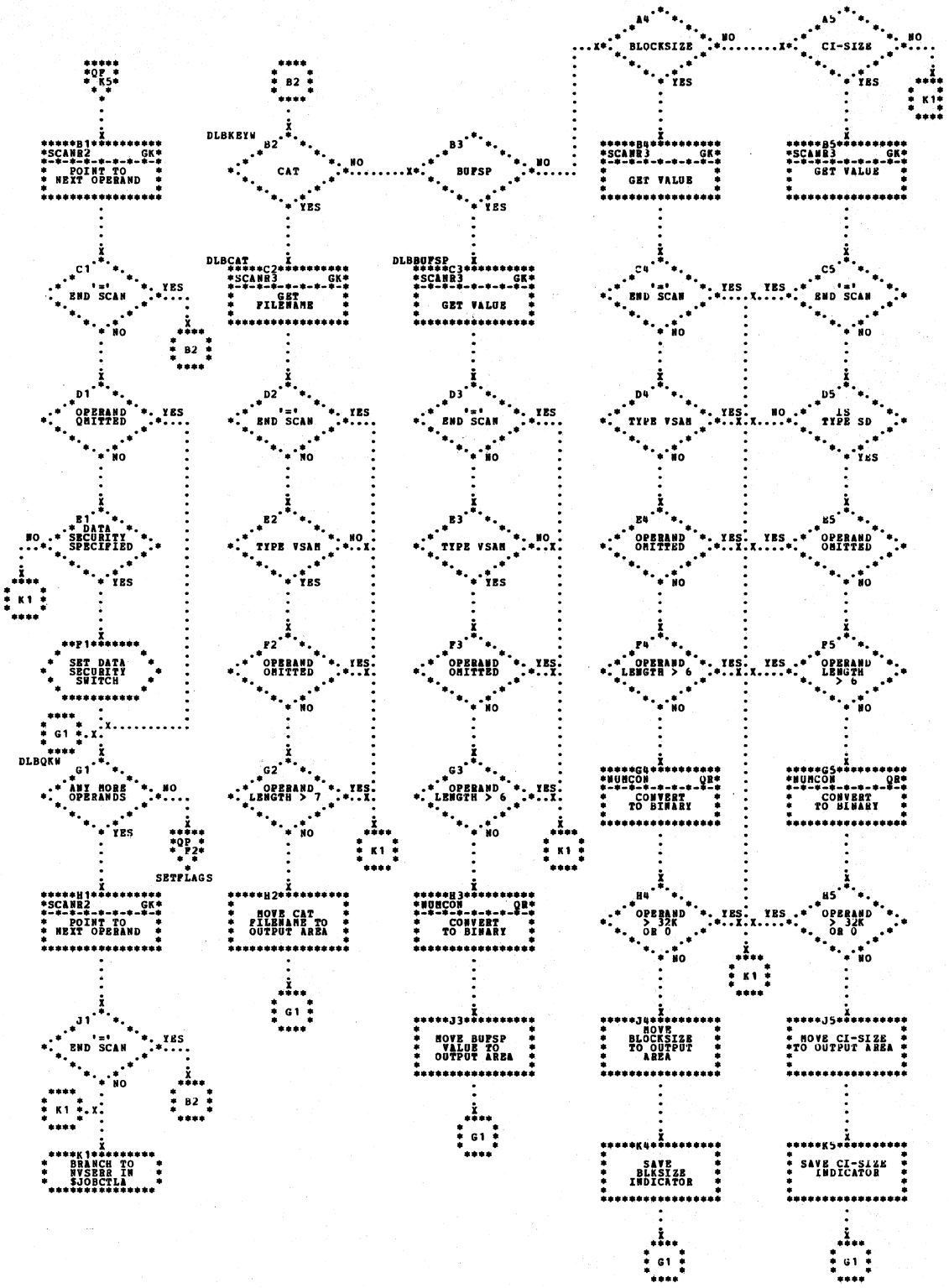


Chart QH. $JOBCTLK - XTENT Statement Processor (Part 1 of 2)
Refer to Charts 21 to 22.

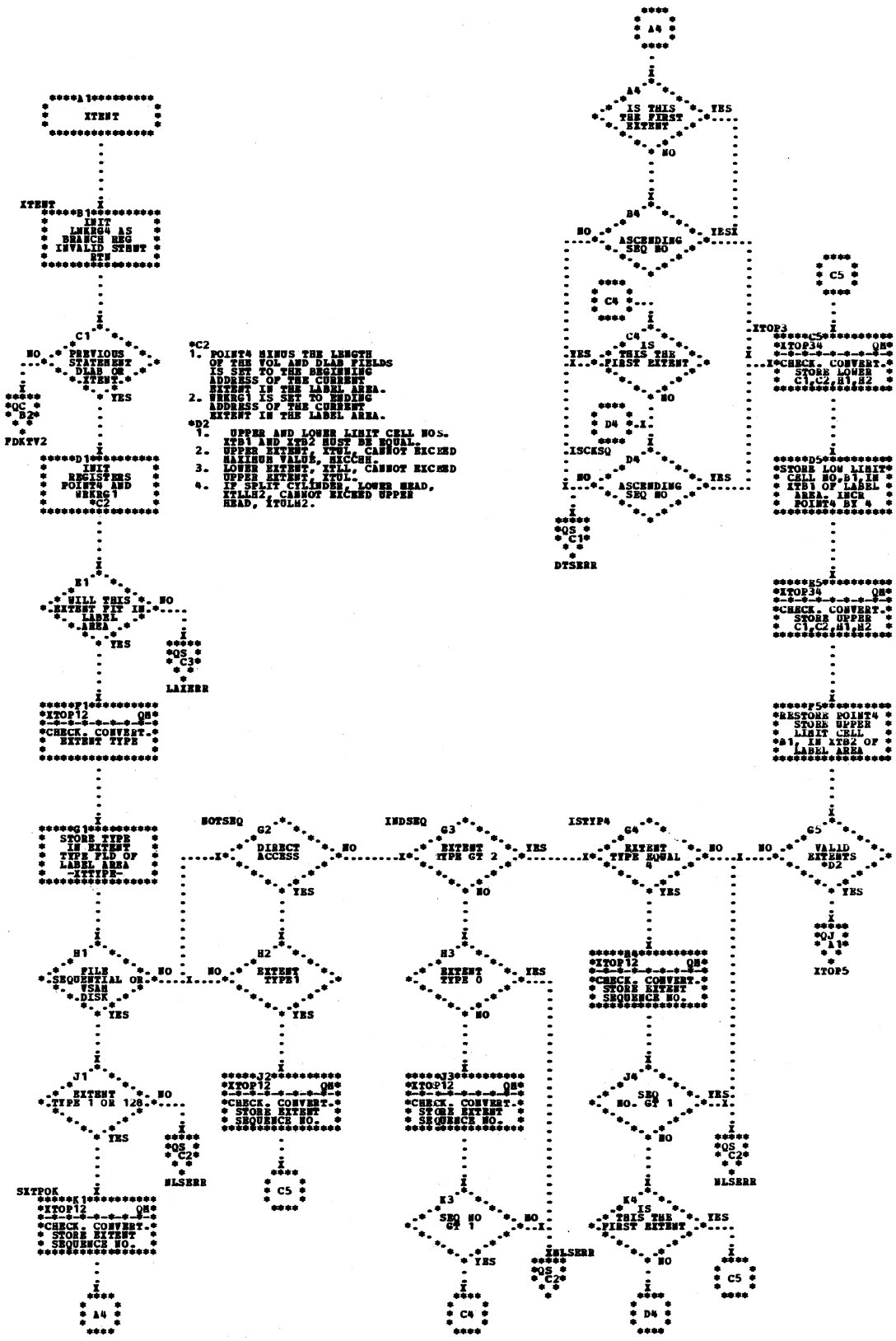


Chart QJ. $JOBCTLK - XTENT Statement Processor (Part 2 of 2)
Refer to Charts 21 to 22.

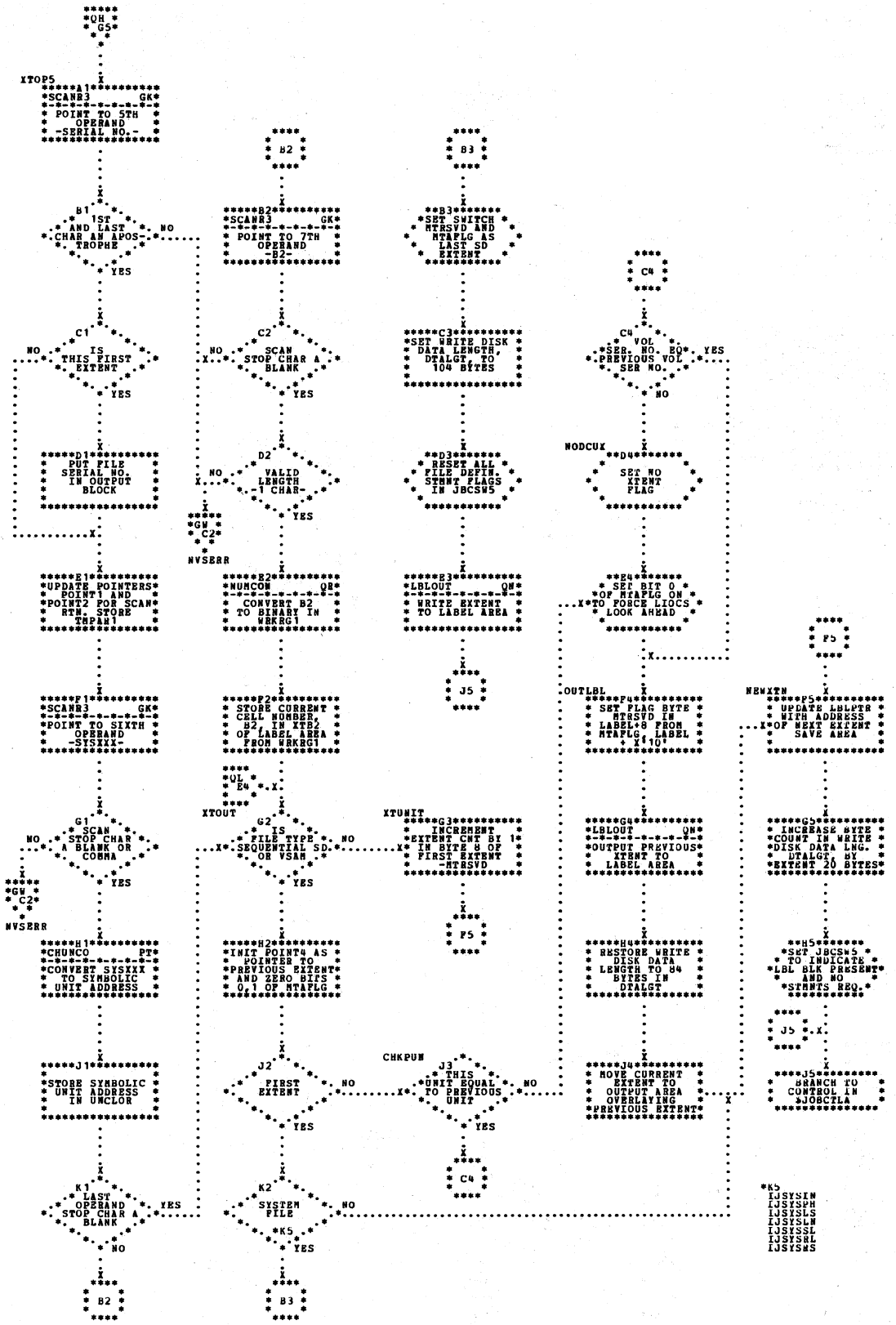


Chart QK. $JOBCTLK - EXTENT Statement Processor (Part 1 of 2)
Refer to Charts 21 to 22.

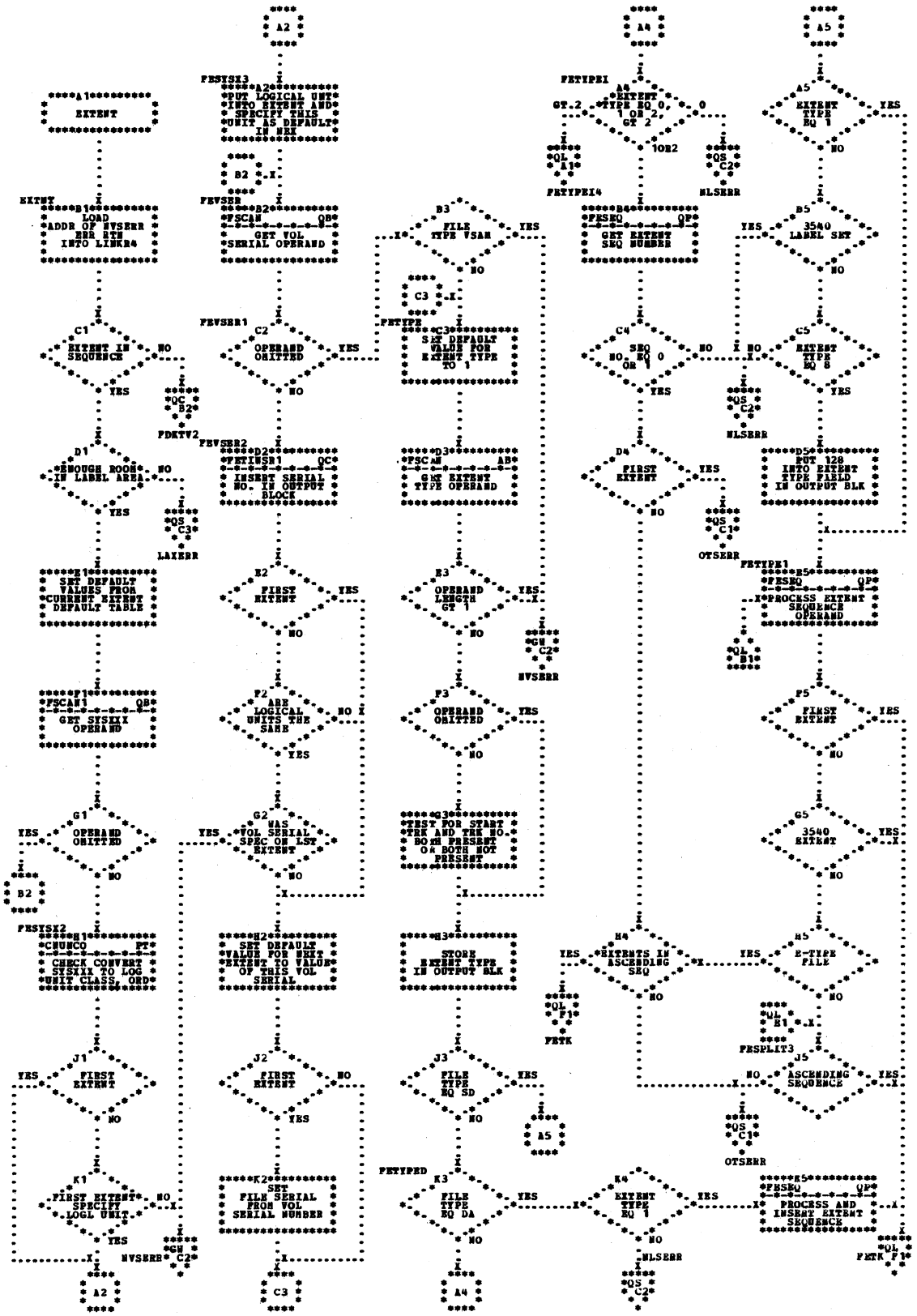


Chart QL. $JOBCTLK - EXTENT Statement Processor (Part 2 of 2)
Refer to Charts 21 to 22.

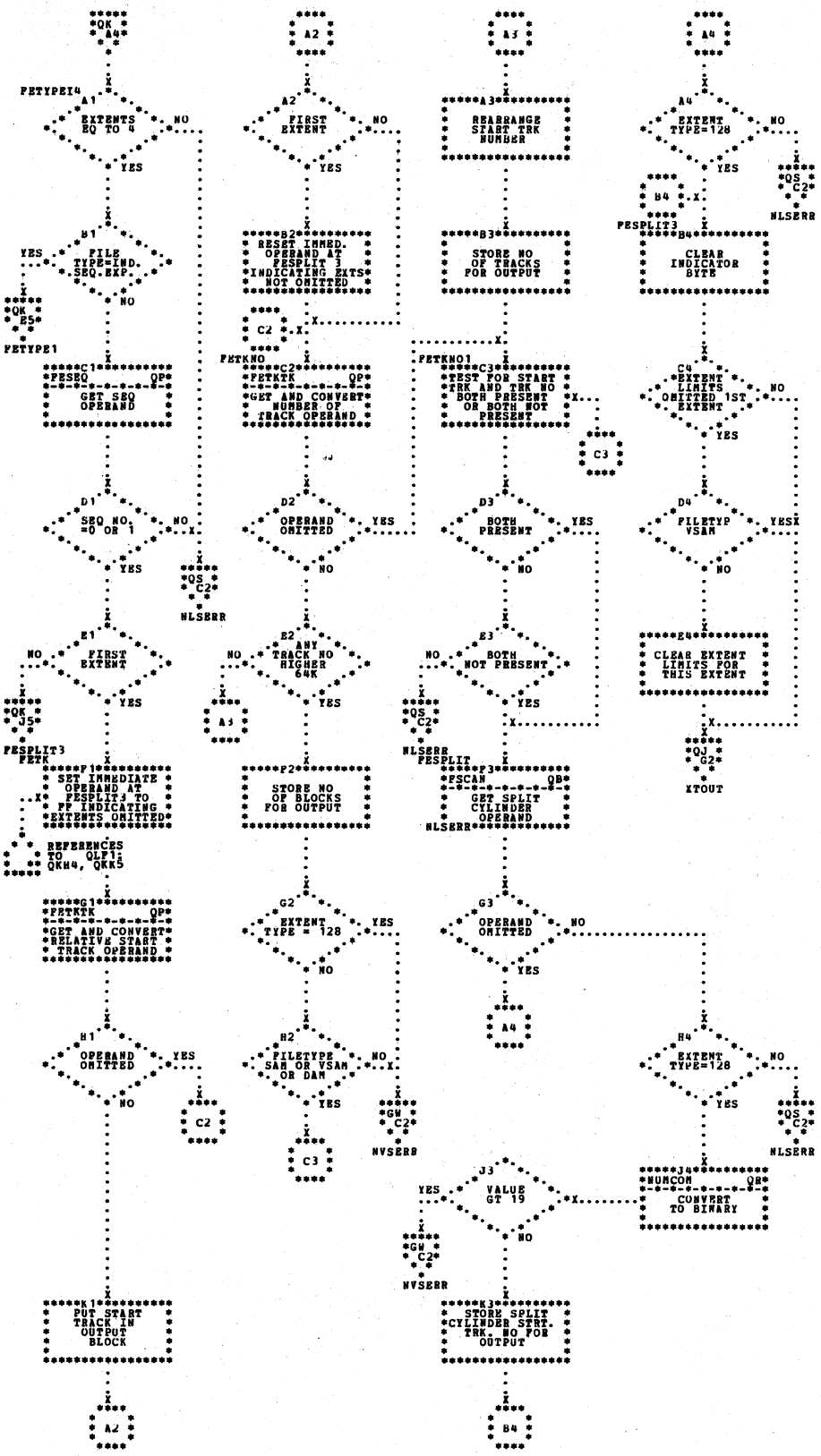


Chart QM. $JOBCTLK - Label Processing Subroutines (Part 1 of 3)
Refer to Charts 21 to 22.

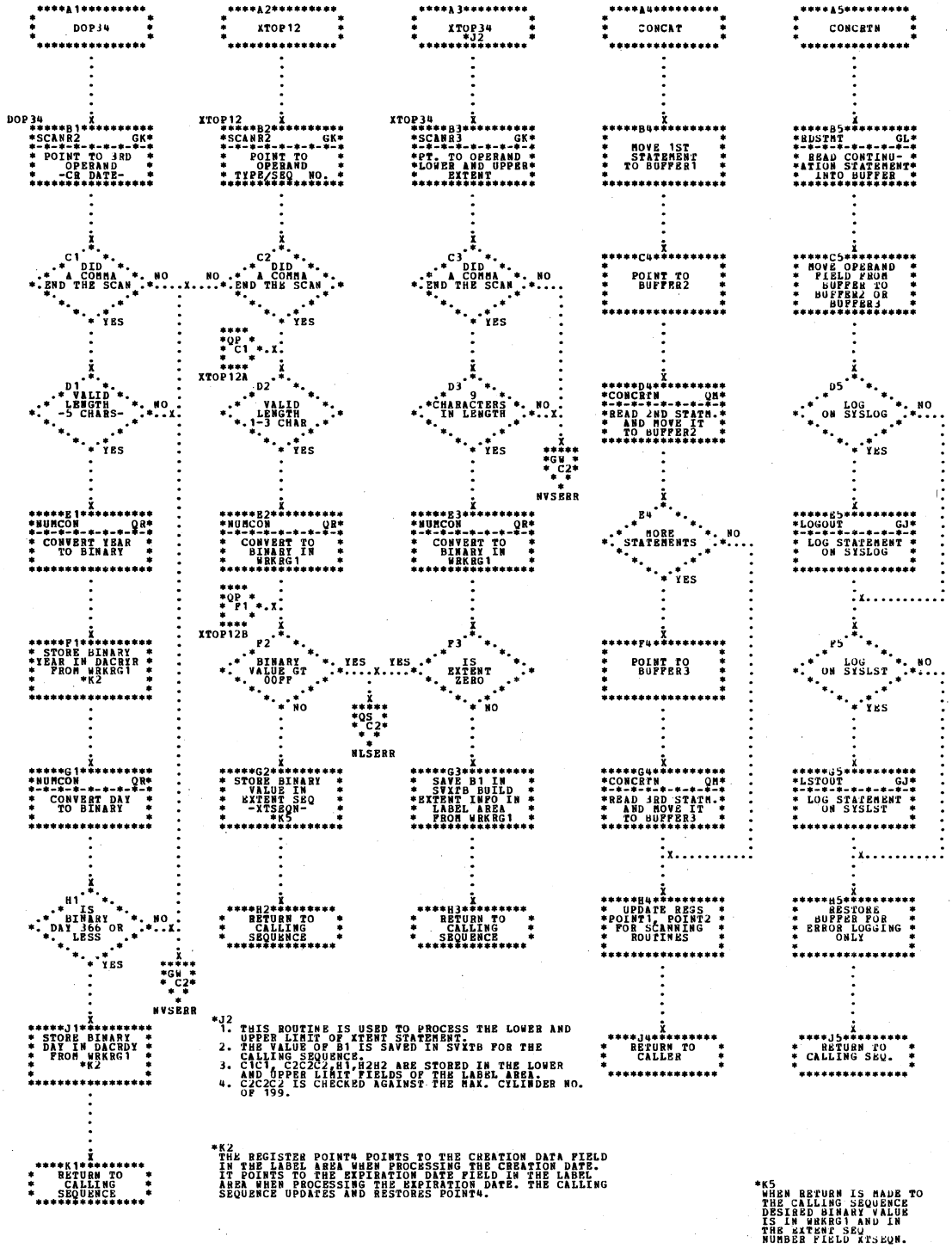


Chart QN. $JOBCTLK - Label Processing Subroutines (Part 2 of 3)
Refer to Charts 21 to 22.

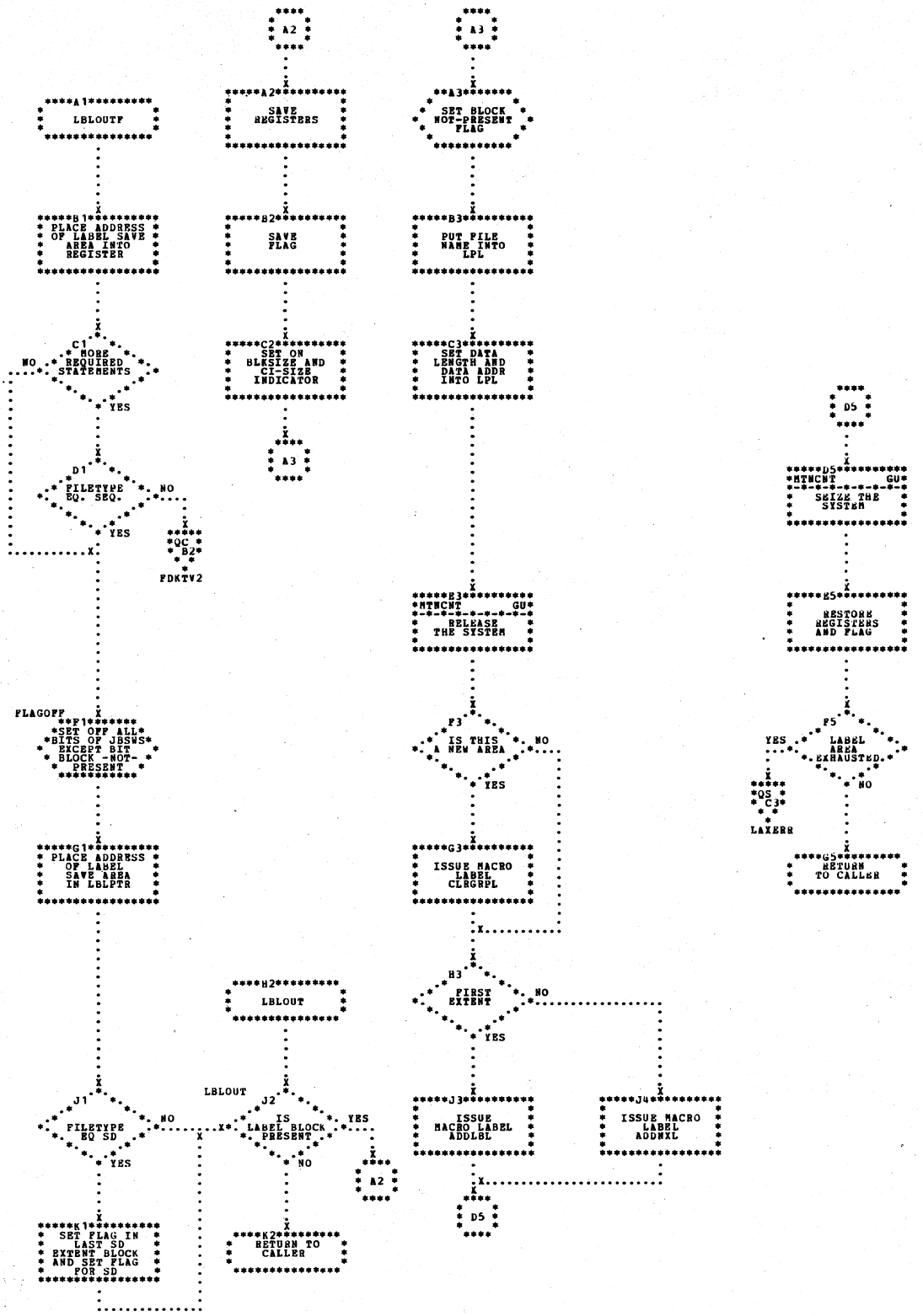


Chart QO. Reserved.

Chart QP. $JOBCTLK - Label Processing Subroutines (Part 3 of 3)
Refer to Charts 21 to 22.

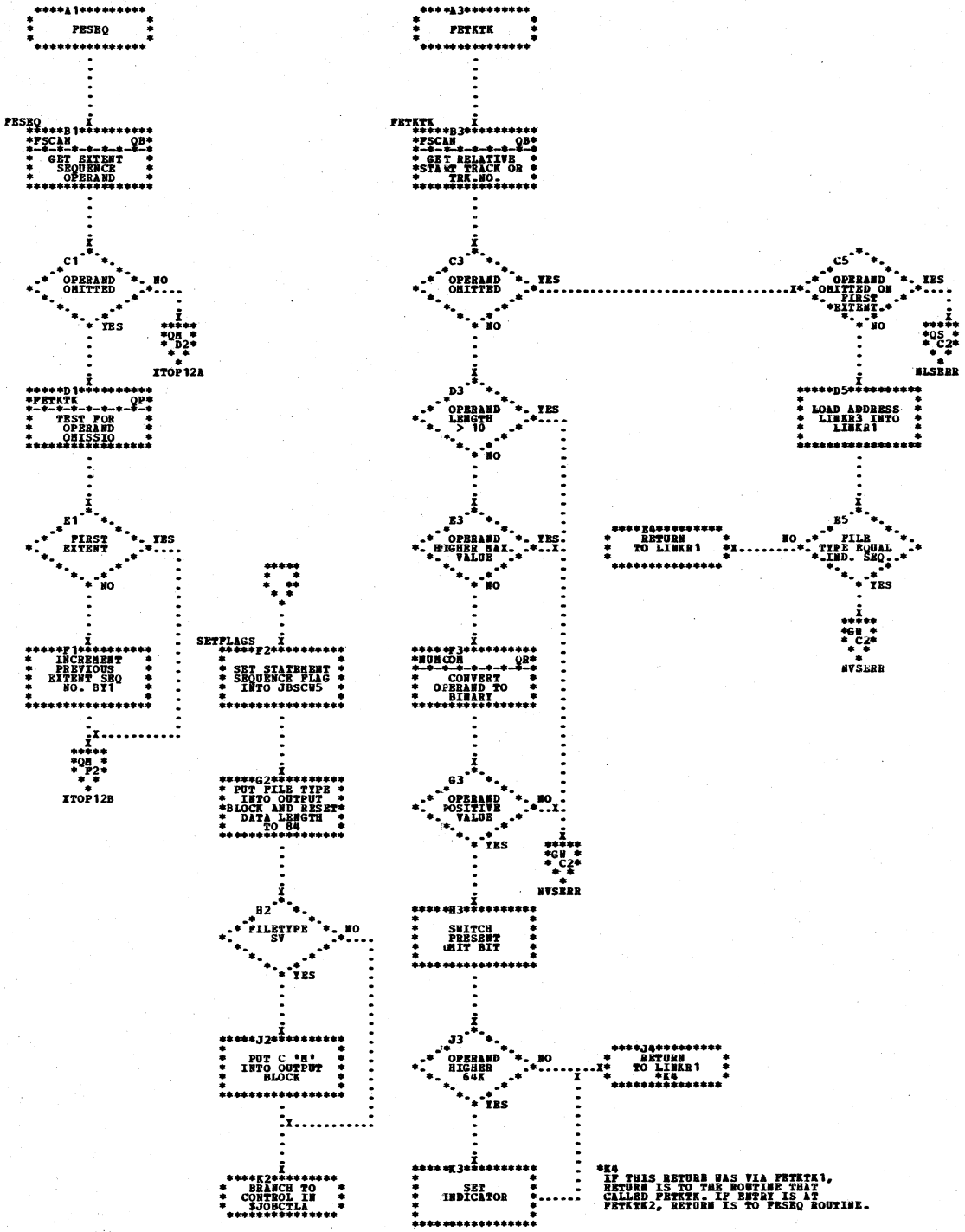


Chart QQ. $JOBCTLK - RSTRT Command Processor
Refer to Charts 21 to 22.

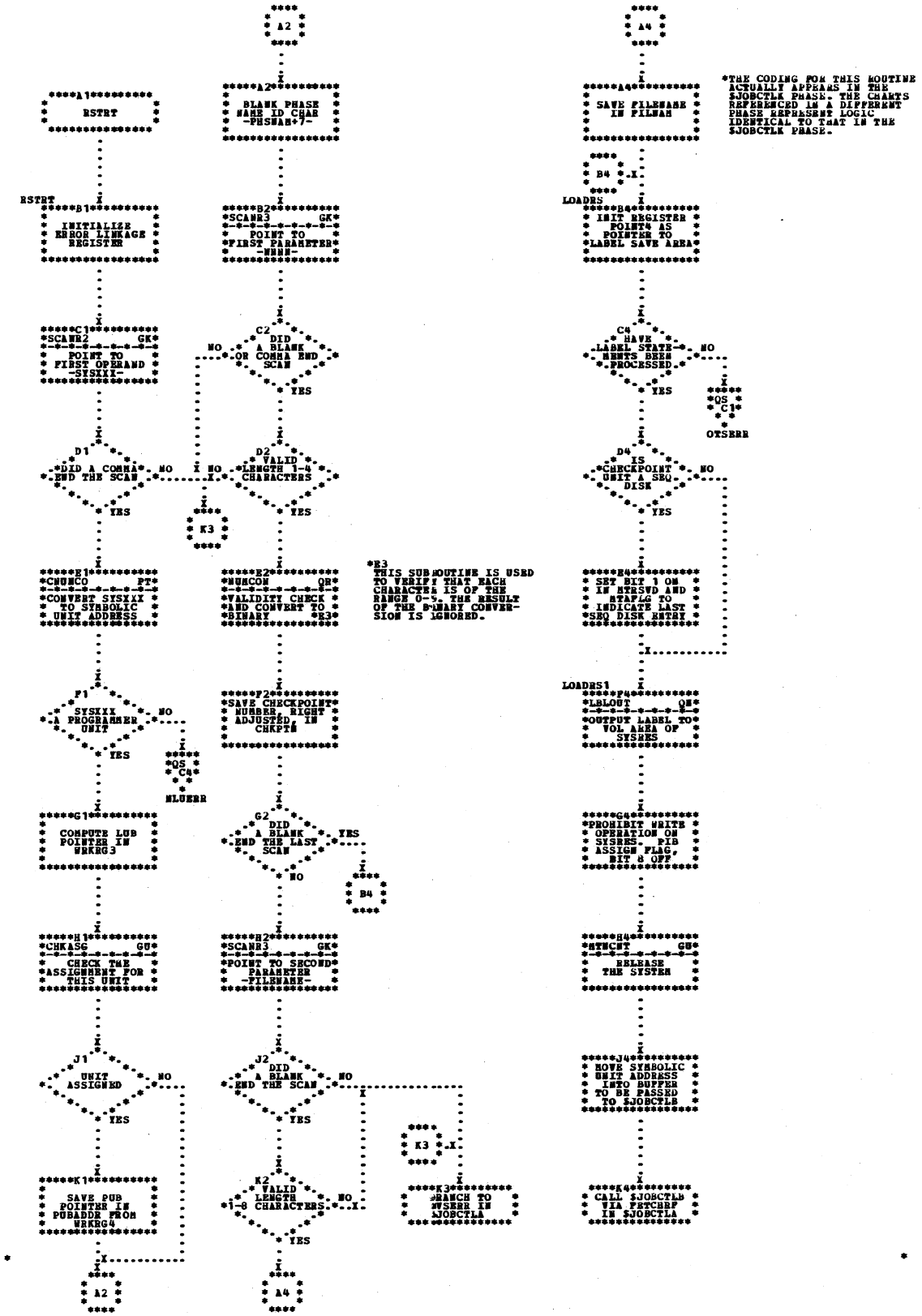


Chart QR. $JOBCTLK - Subroutines
Refer to Charts 21 to 22.

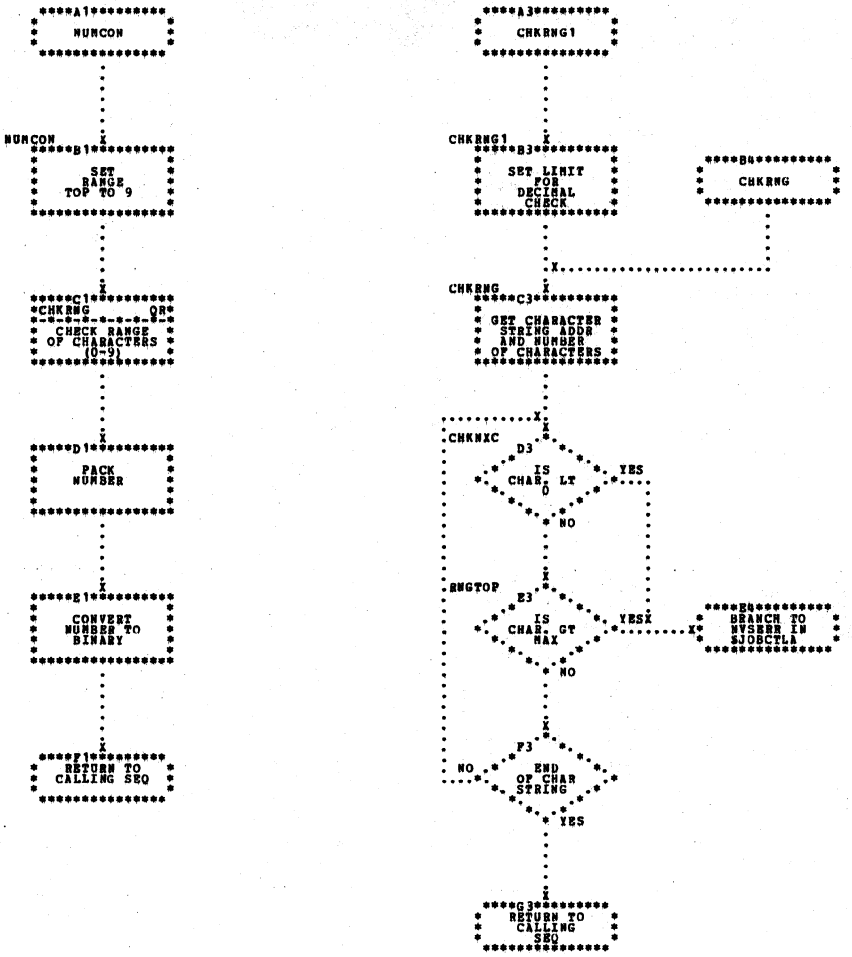
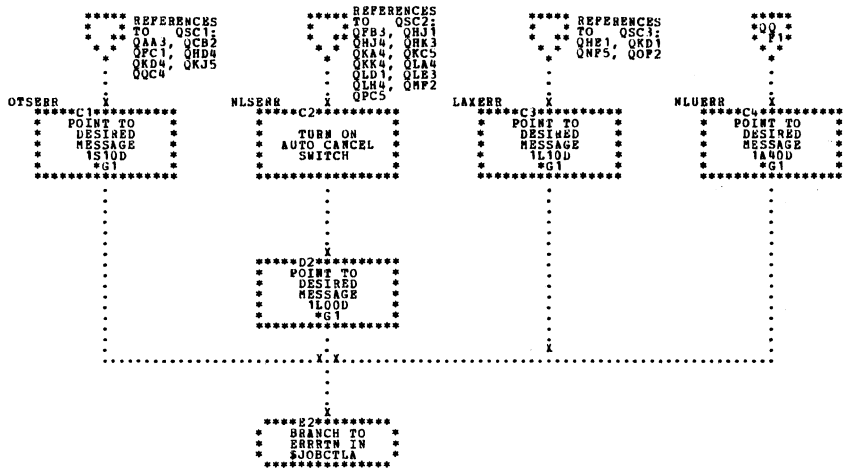


Chart QS. $JOBCTLK - Error Subroutines
Refer to Charts 21 to 22.



*G1

MESSAGE	DESCRIPTION
1A40D	INVALID LOGICAL UNIT SPECIFICATION
1L00D	INVALID LABEL SYNTAX
1L10D	LABEL AREA EXHAUSTED
1S10D	STATEMENT OUT OF SEQUENCE

Chart QT. $JOBCTLK - SETPRT Statement Processor (Part 1 of 3)
Refer to Charts 21 to 22.

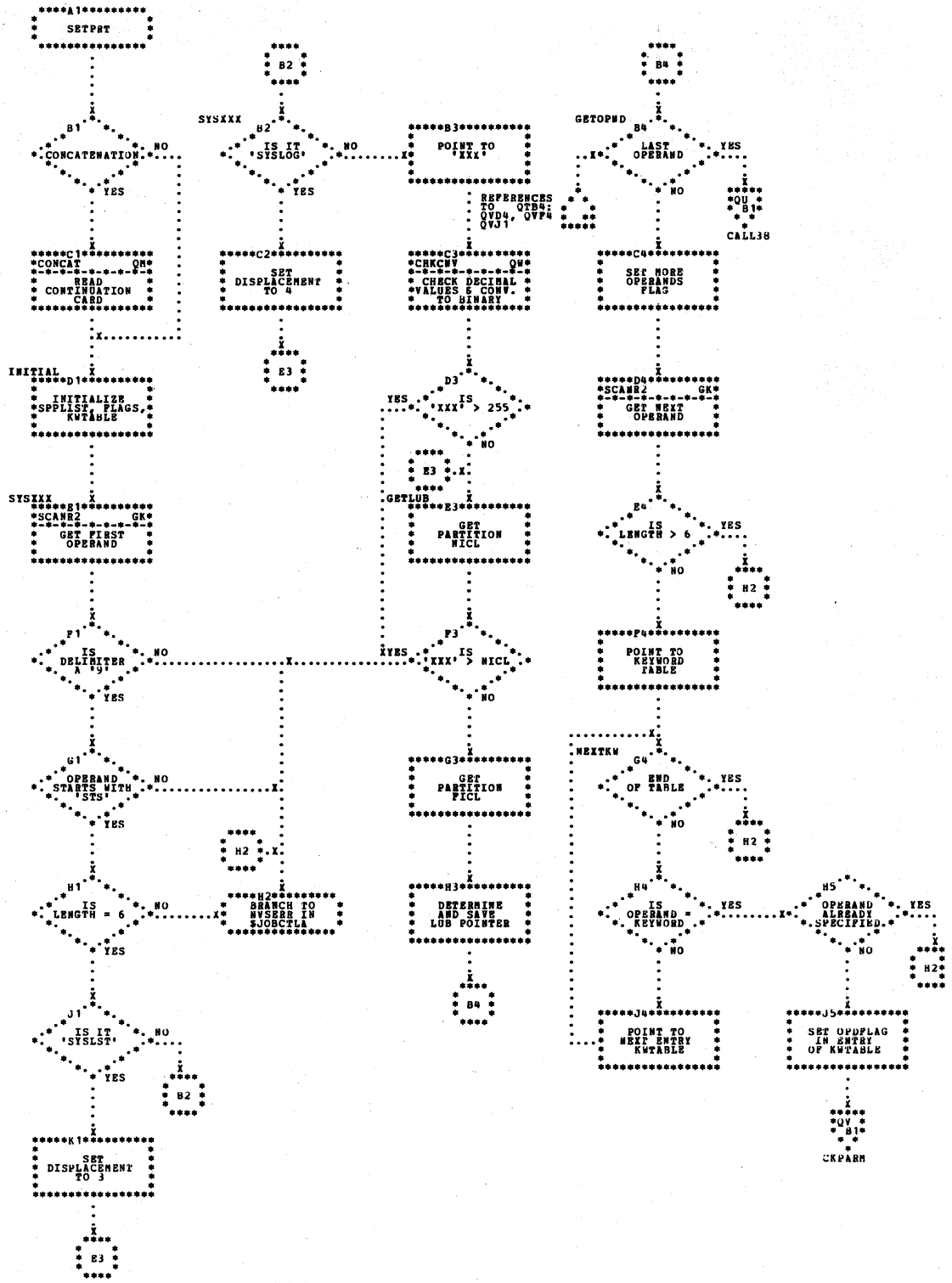


Chart QU. $JOBCTLK - SETPRT Statement Processor (Part 2 of 3)
Refer to Charts 21 to 22.

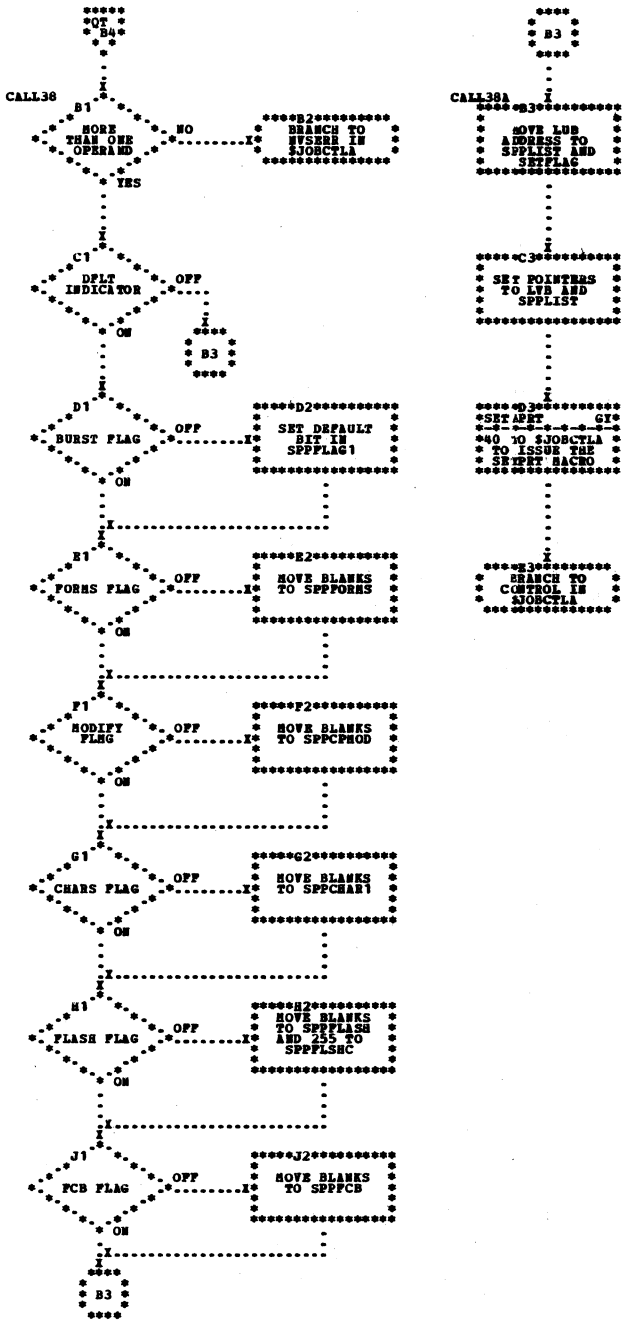


Chart QV. $JOBCTLK - SETPRT Statement Processor (Part 3 of 3)
Refer to Charts 21 to 22.

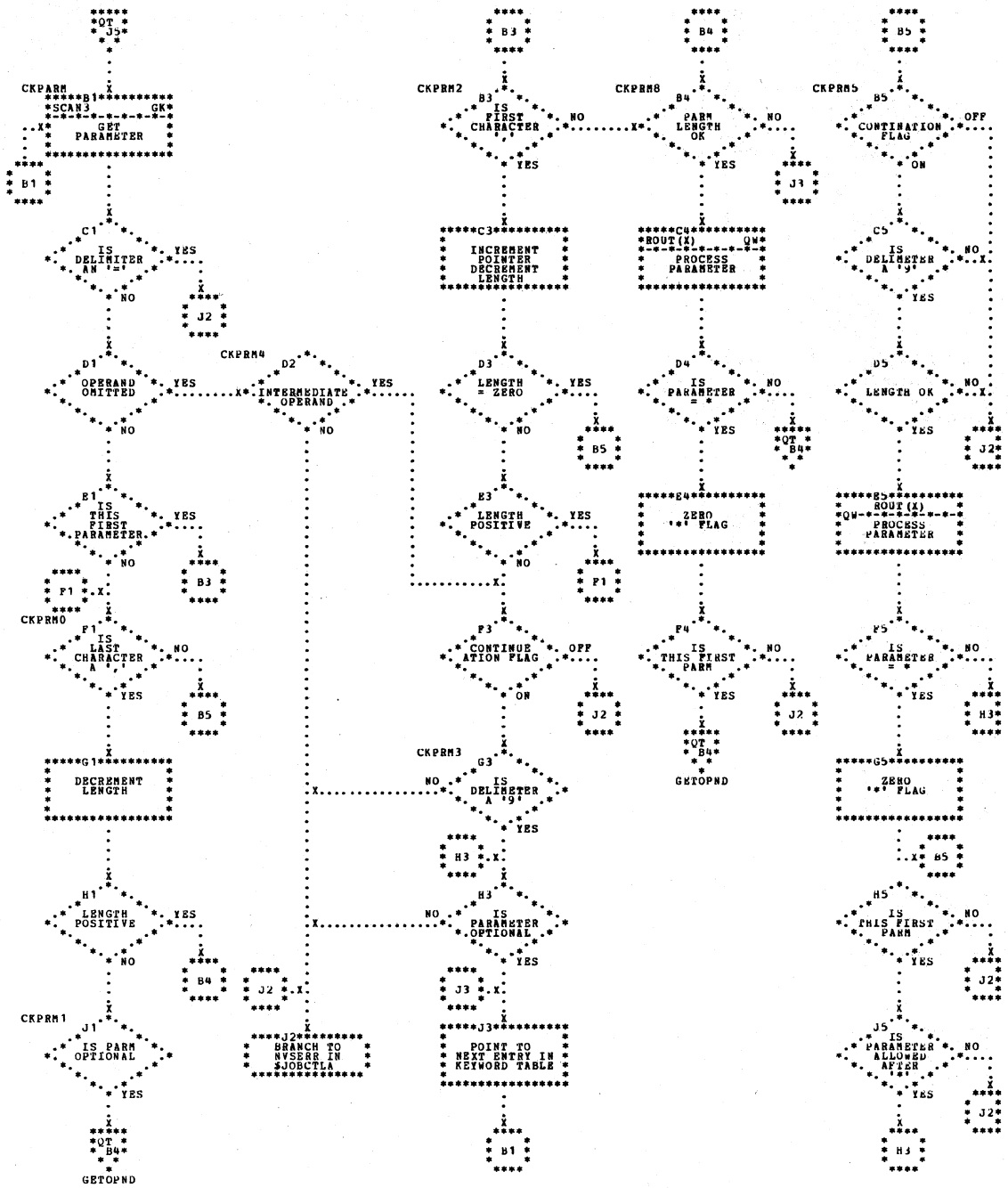


Chart QW. $JOBCTLK - SETPRT Processing Subroutines
Refer to Charts 21 to 22.

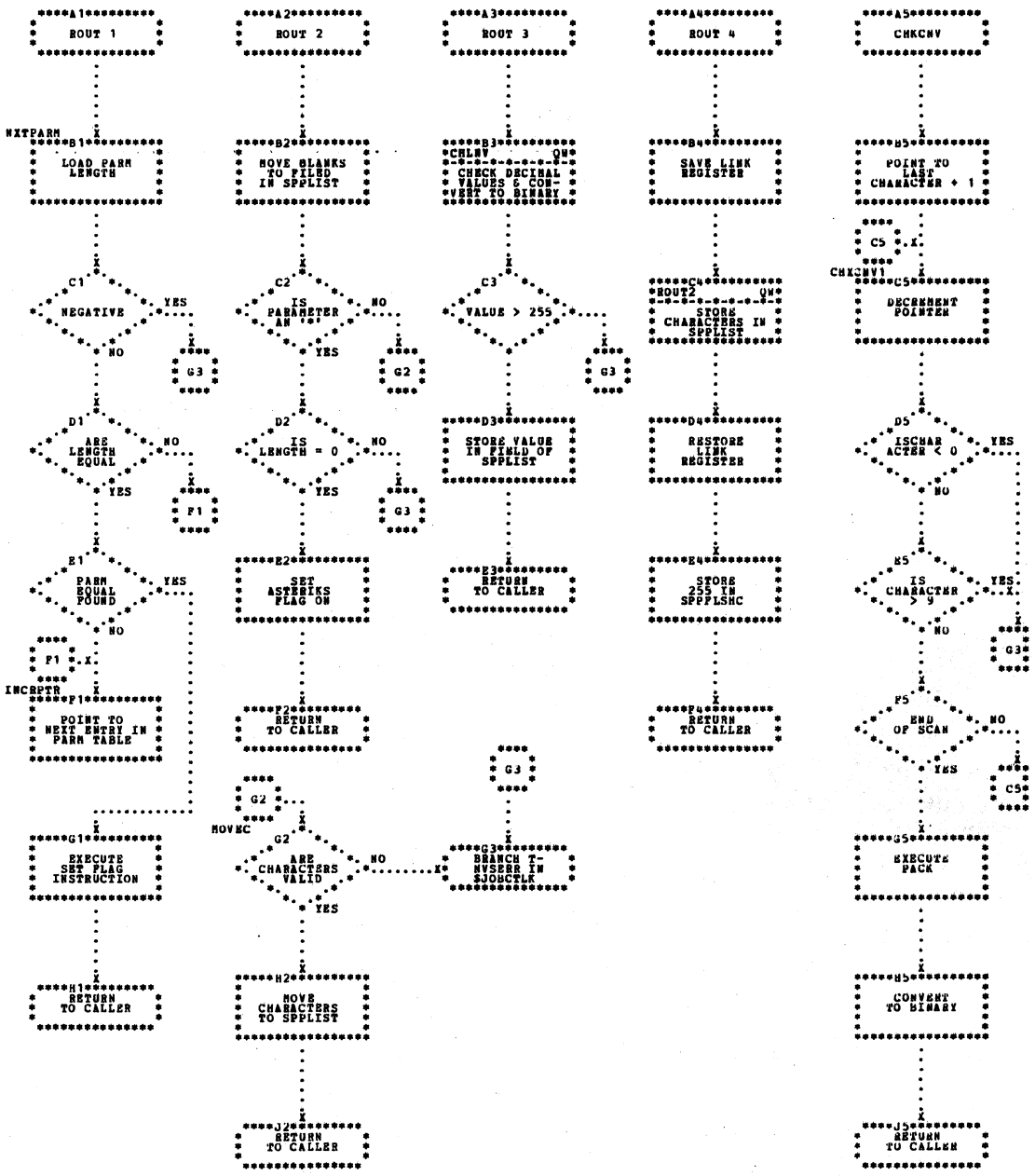


Chart RA. $JOBCTLM - Recorder File Initialization
Refer to Chart 23.

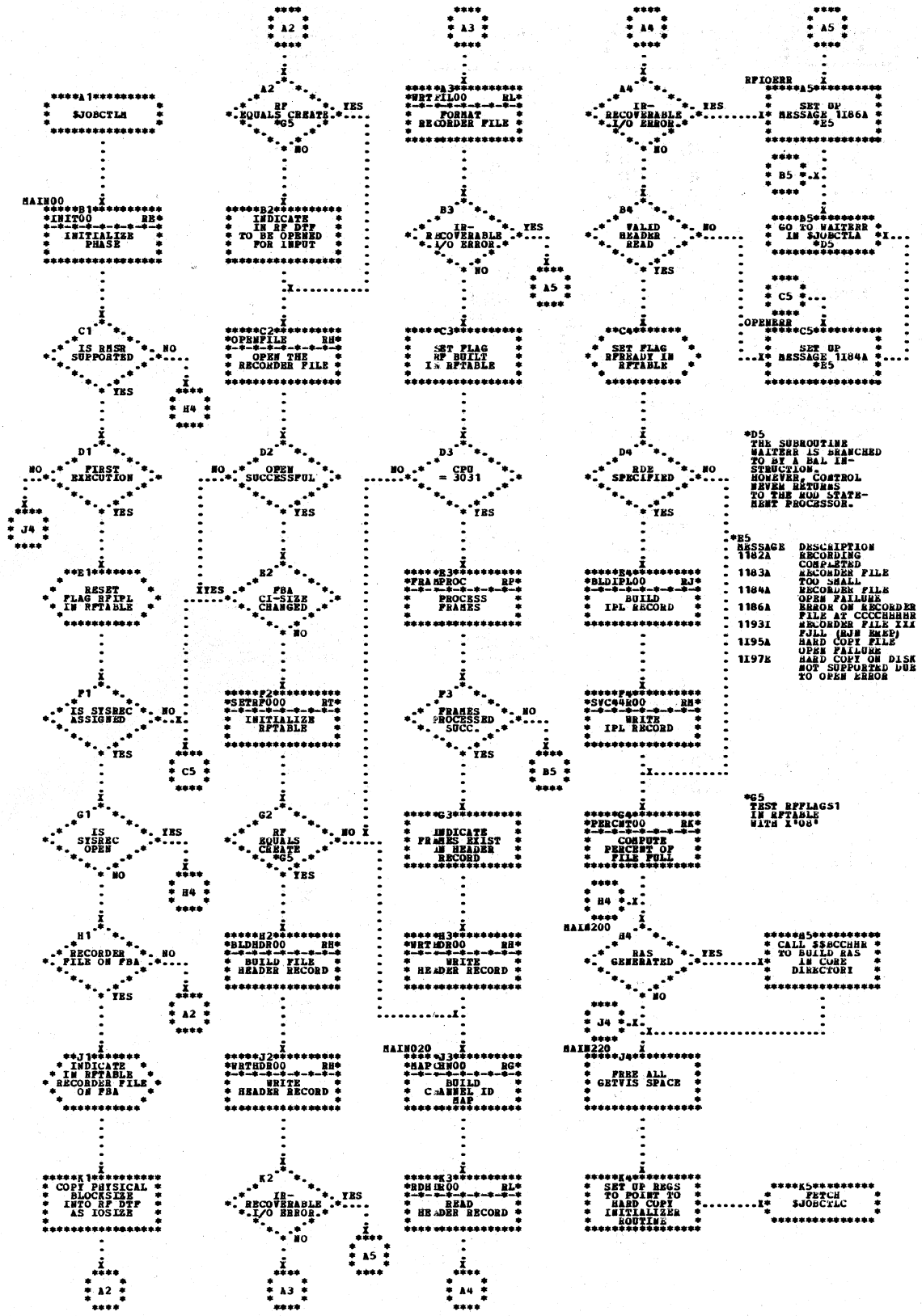


Chart RC. $JOBCTLM - ROD Command Processor
Refer to Chart 23.

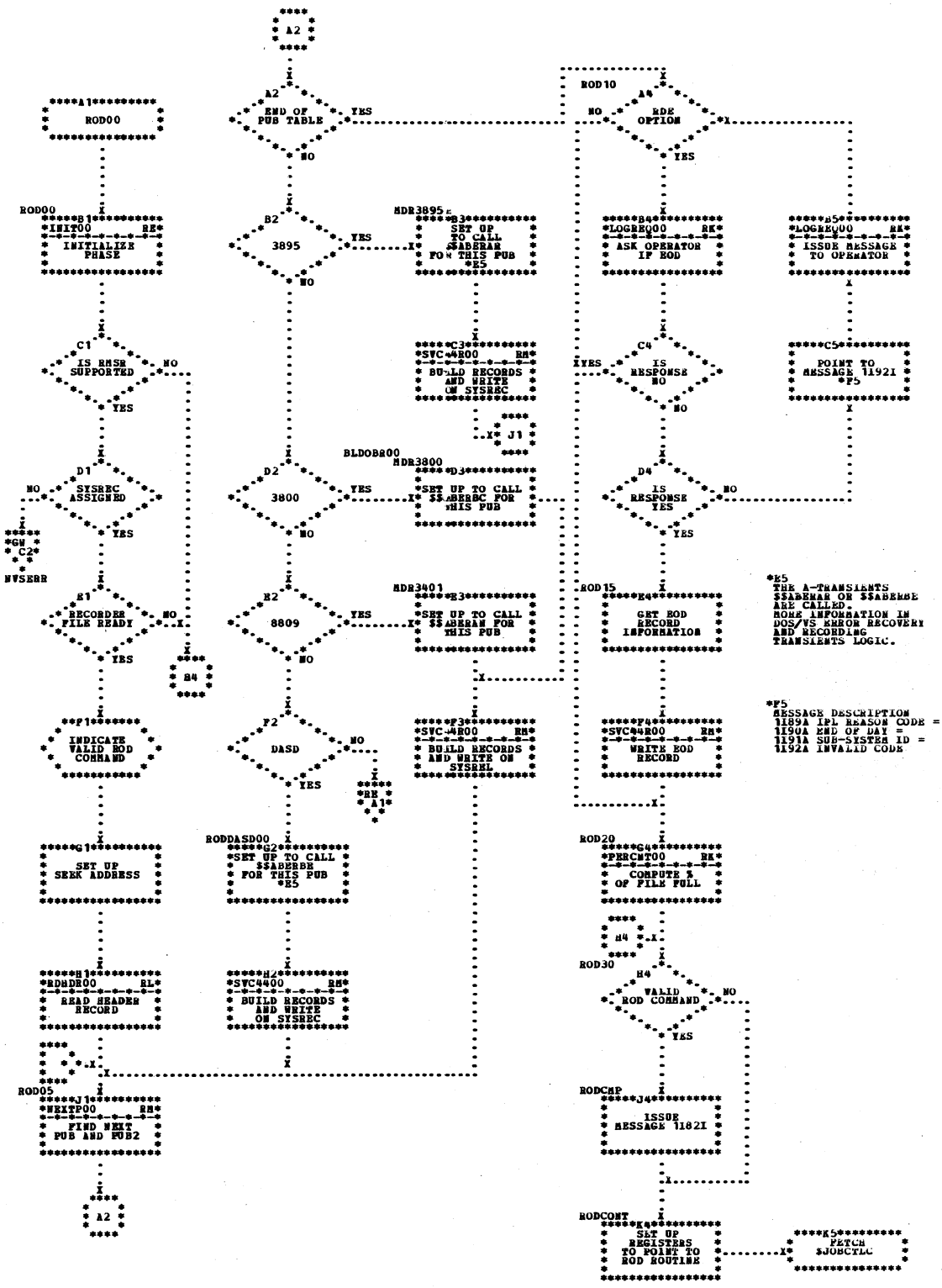


Chart RE. $JOBCTLM - Build EOD Outboard Record
Refer to Chart 23.

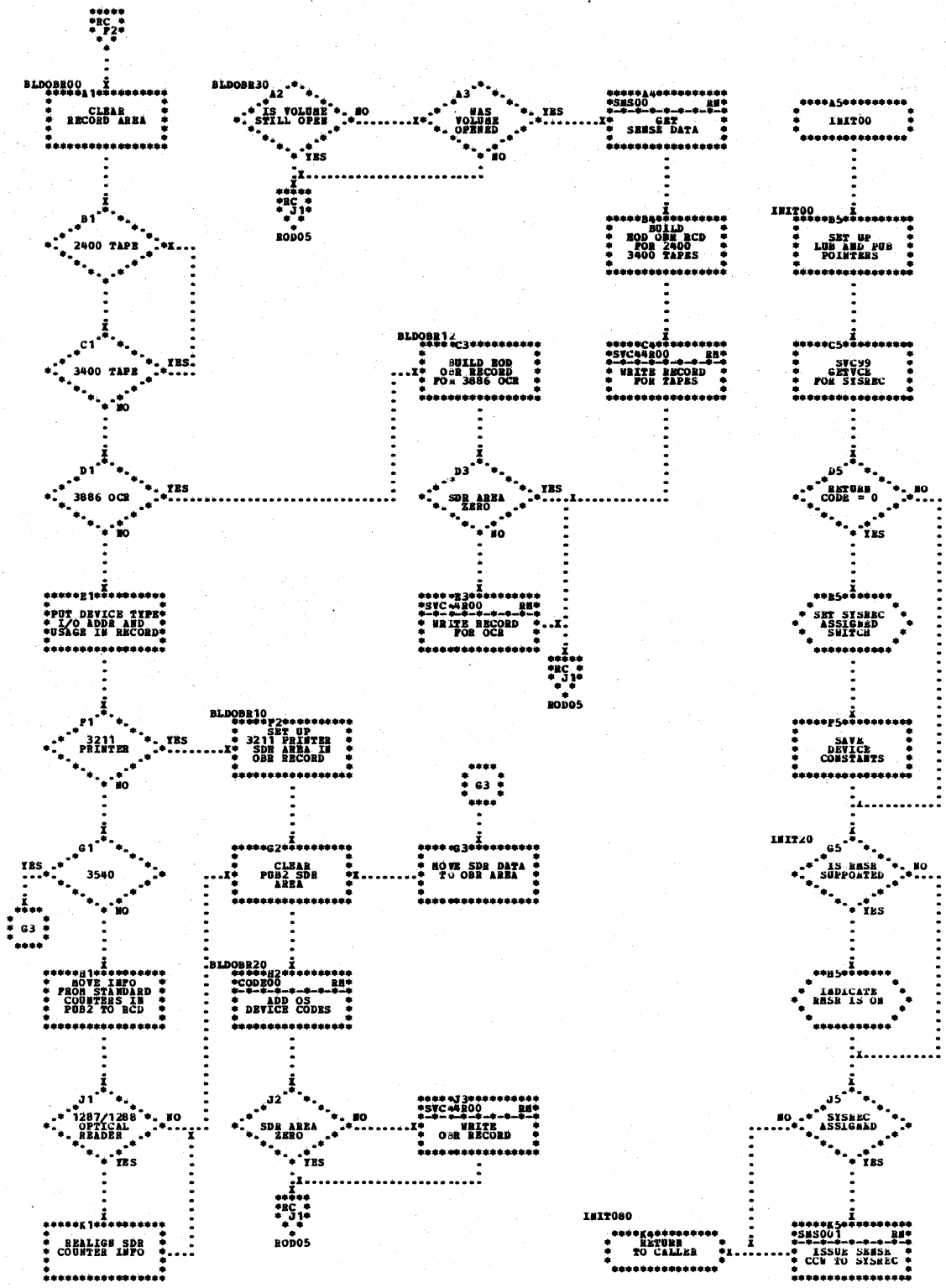


Chart RG. $JOBCTLM - Build Channel Map Routine
Refer to Chart 23.

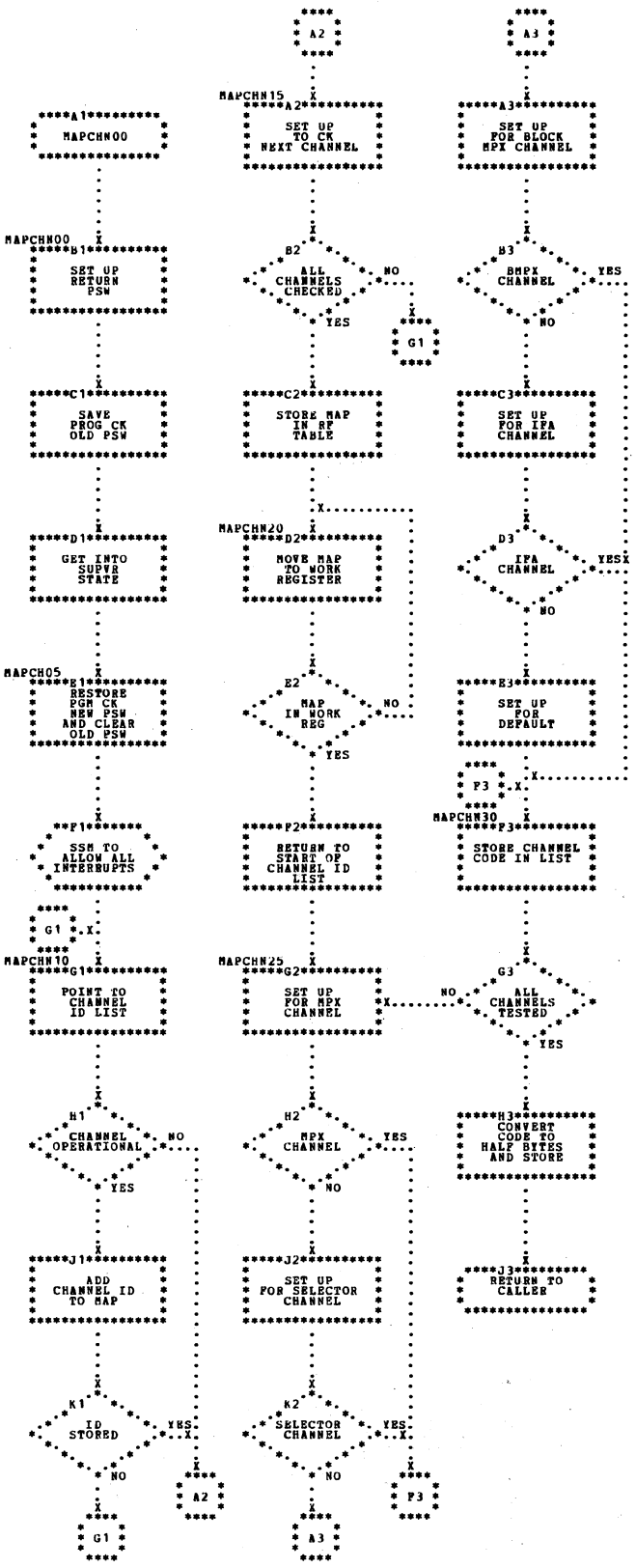


Chart RH. $JOBCTLM - Subroutines
Refer to Chart 23.

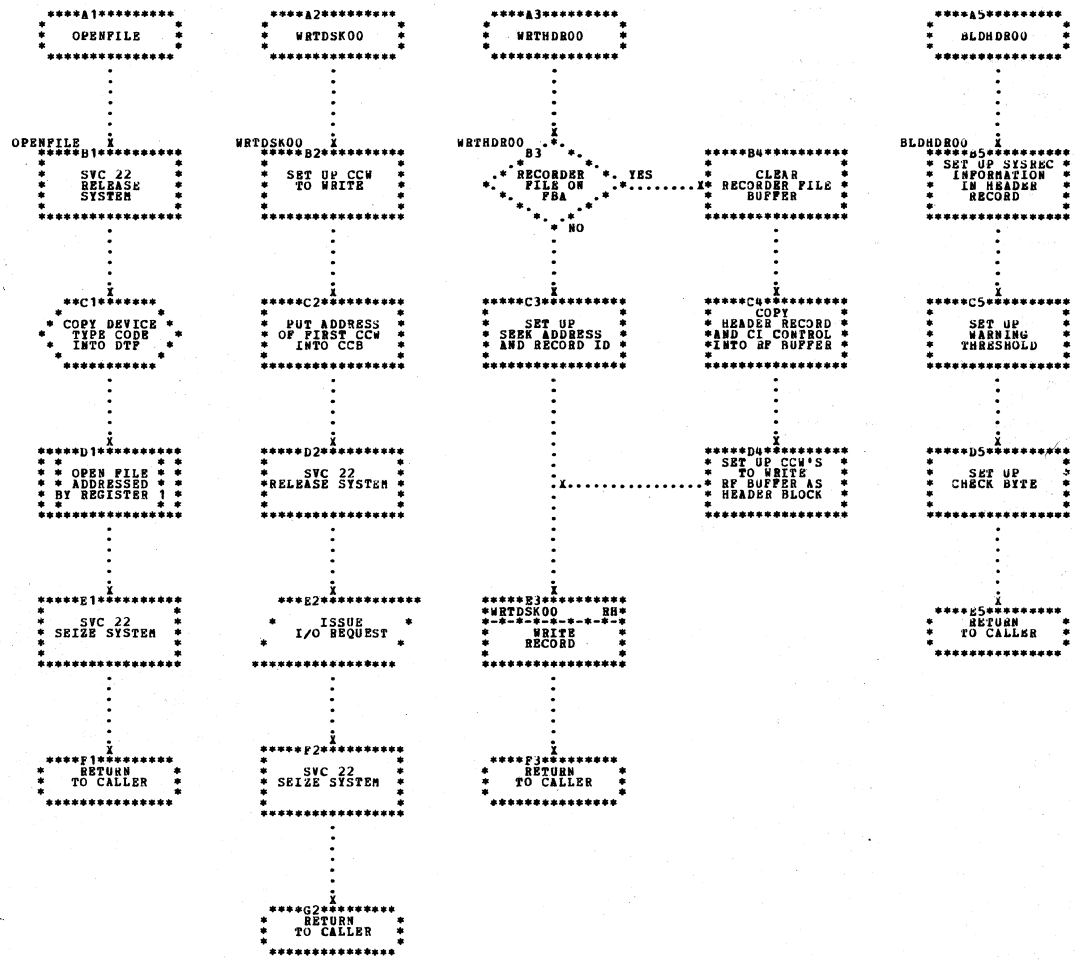


Chart RJ. $JOBCTLM - Subroutines
Refer to Chart 23.

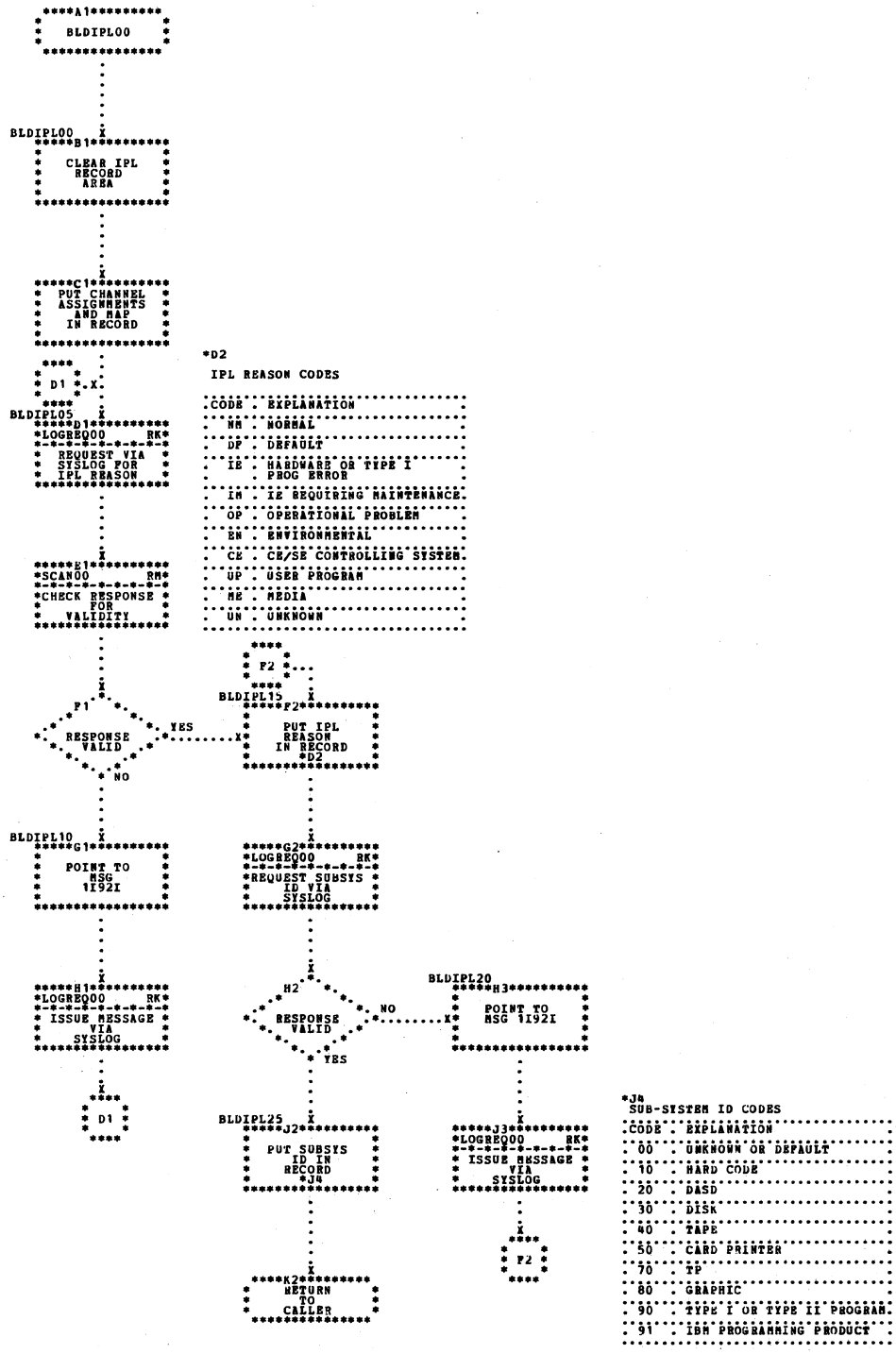


Chart RK. $JOBCTLM - Subroutines
Refer to Chart 23.

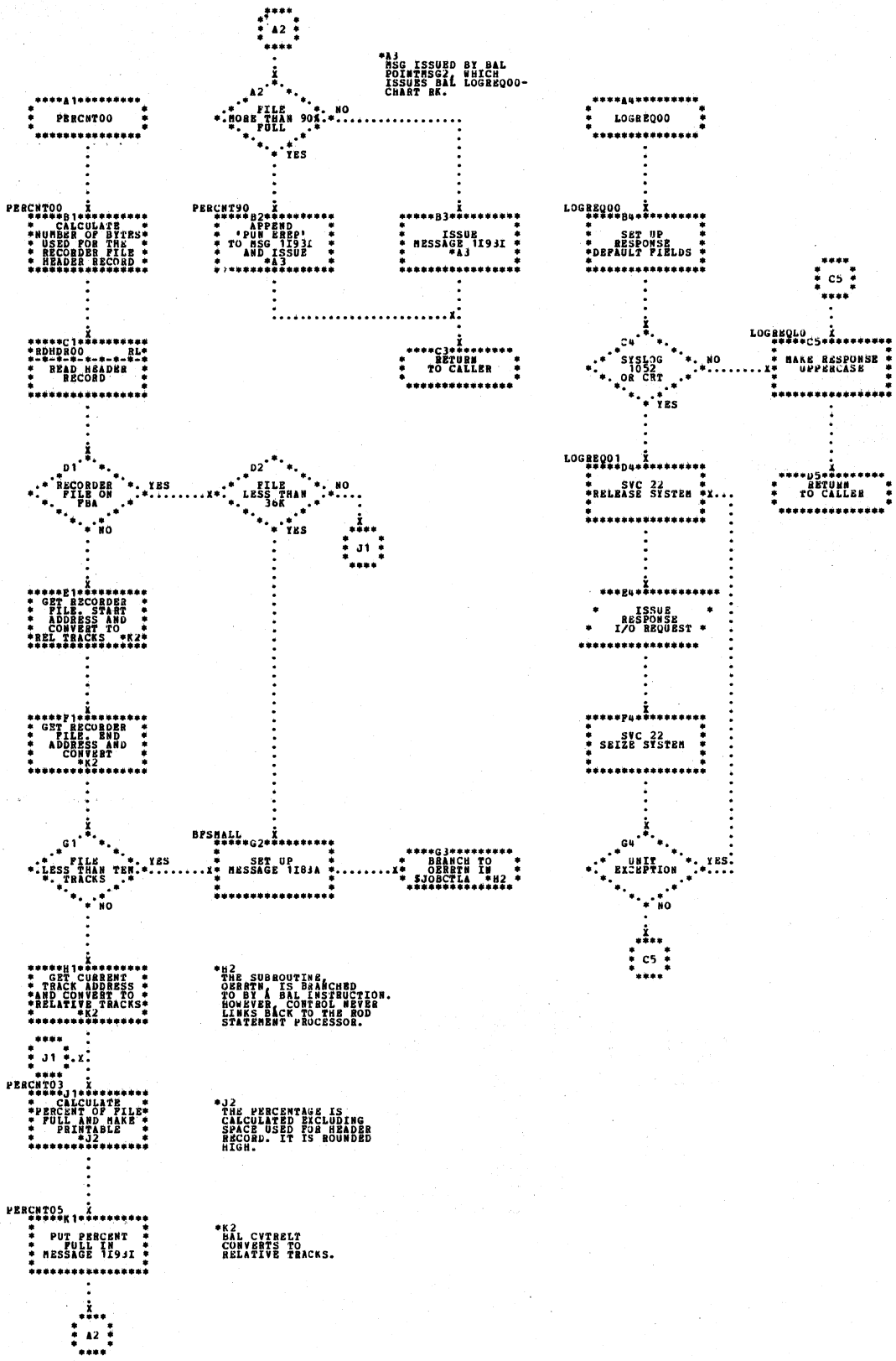


Chart RL. $JOBCTLM - Subroutines
Refer to Chart 23.

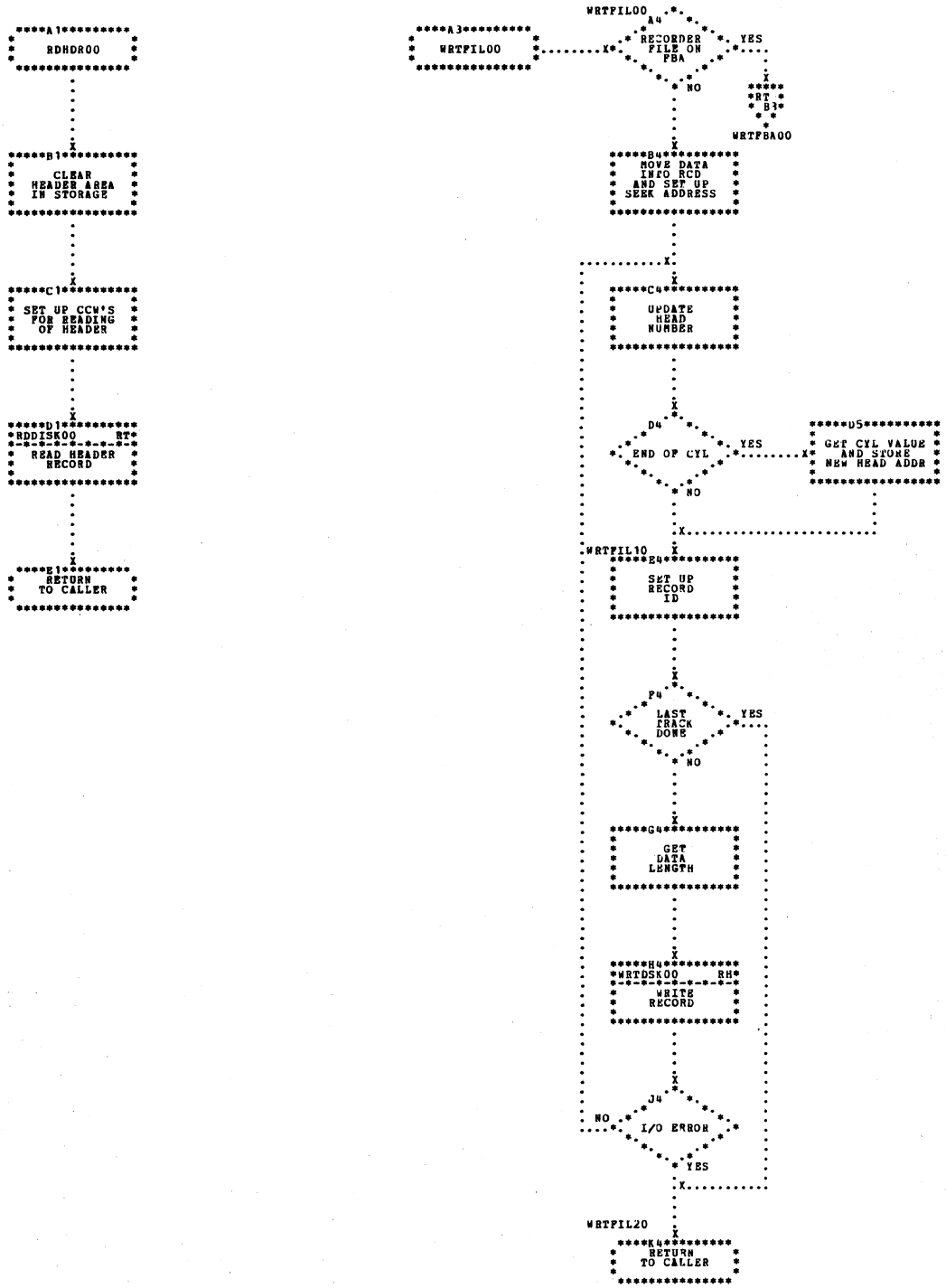


Chart RM. $JOBCTLM - Subroutines
Refer to Chart 23.

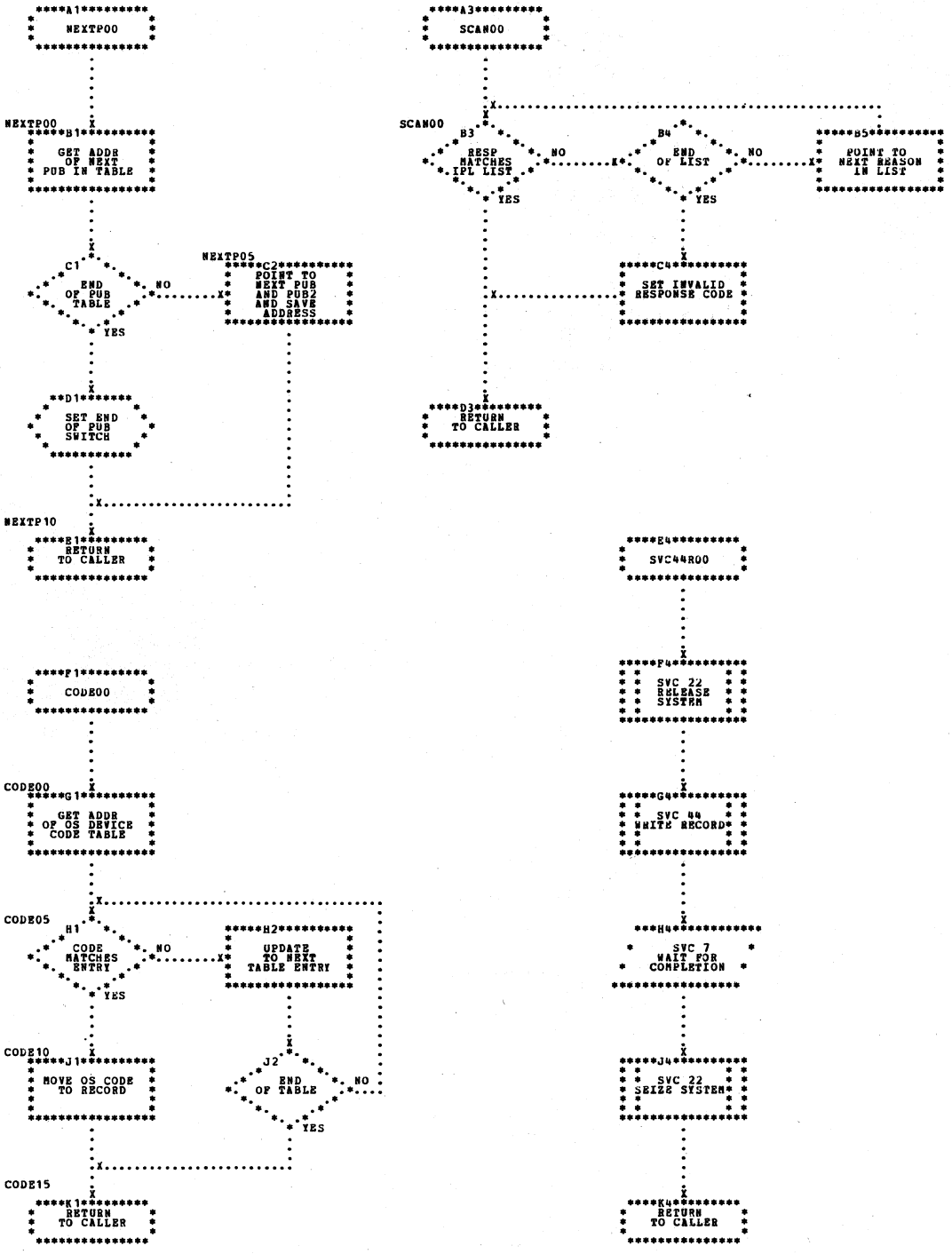


Chart RN. $JOBCTLM - Subroutines
Refer to Chart 23.

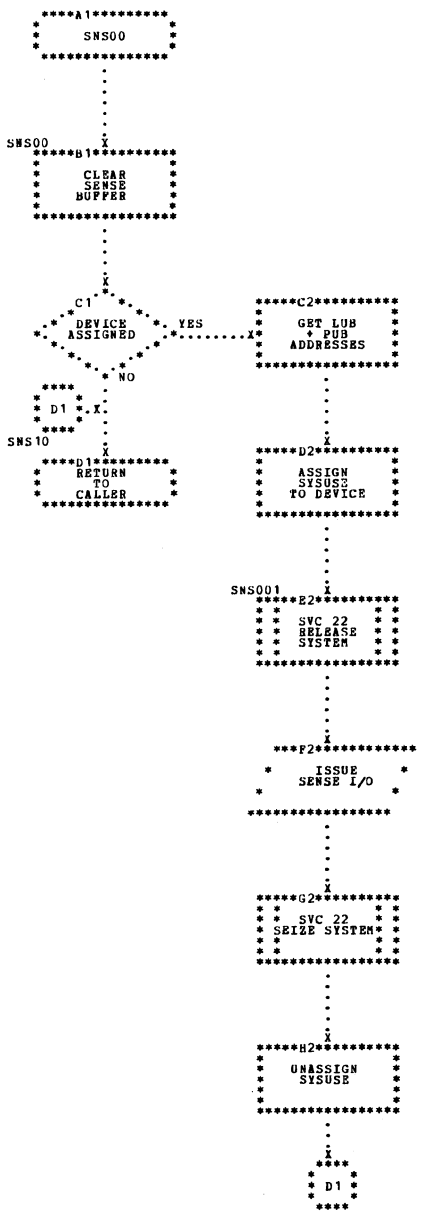
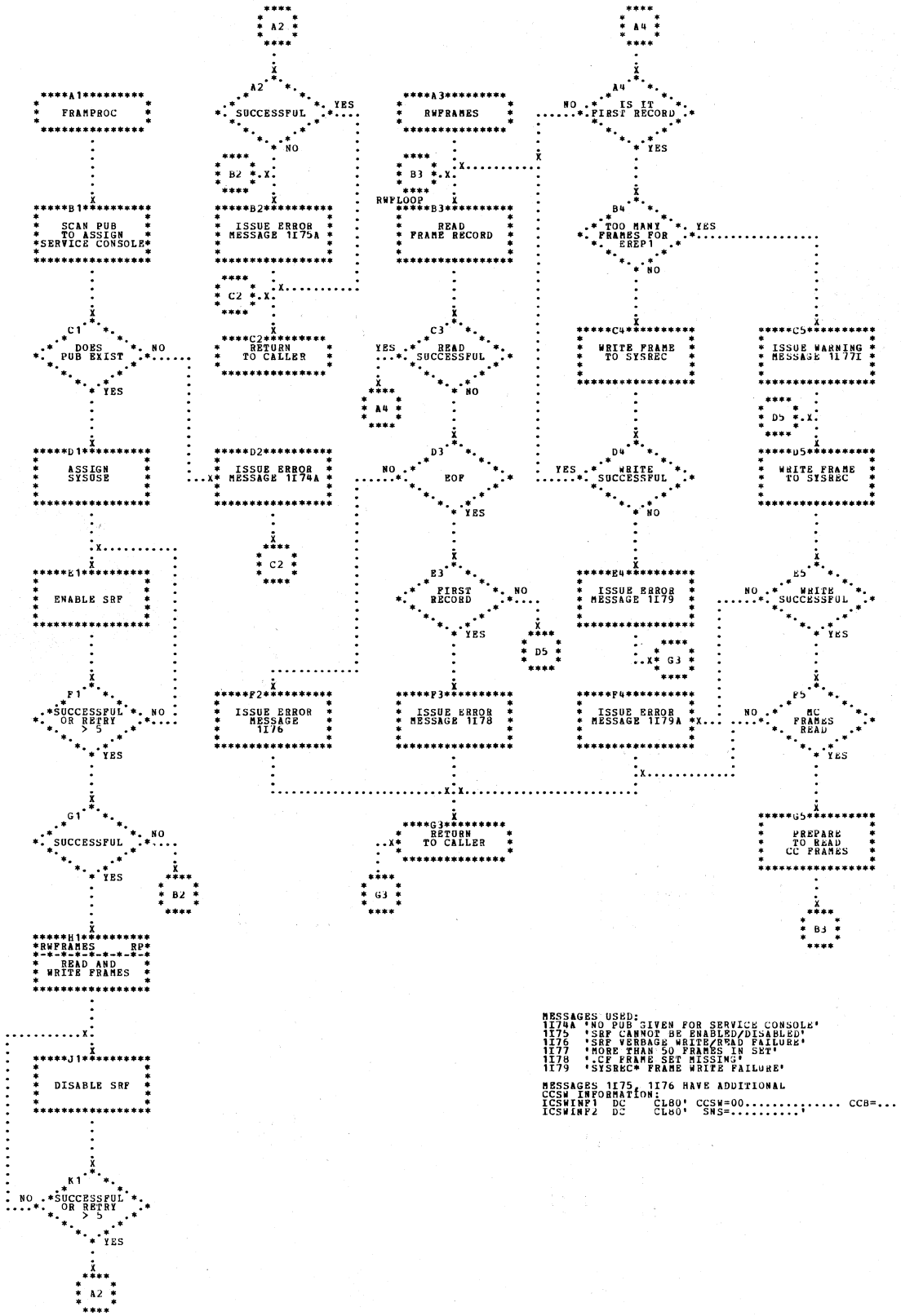


Chart RP. $JOBCTLM - Subroutines
Refer to Chart 23.




```
MESSAGES USED:
1I74A 'NO PUB GIVEN FOR SERVICE CONSOLE'
1I75  'SRF CANNOT BE ENABLED/DISABLED'
1I76  'SRF VERBAGE WRITE/READ FAILURE'
1I77  'MORE THAN 50 FRAMES IN SET'
1I78  'CF FRAME SET MISSING'
1I79  'SYSREC* FRAME WRITE FAILURE'

MESSAGES 1I75, 1I76 HAVE ADDITIONAL
CCSW INFORMATION:
ICSWINF1  DC    CL80' CCSW=00.............. CCB=...
ICSWINF2  DC    CL80'  SNS=..........'
```

Chart RT. $JOBCTLM - Subroutines
Refer to Chart 23.

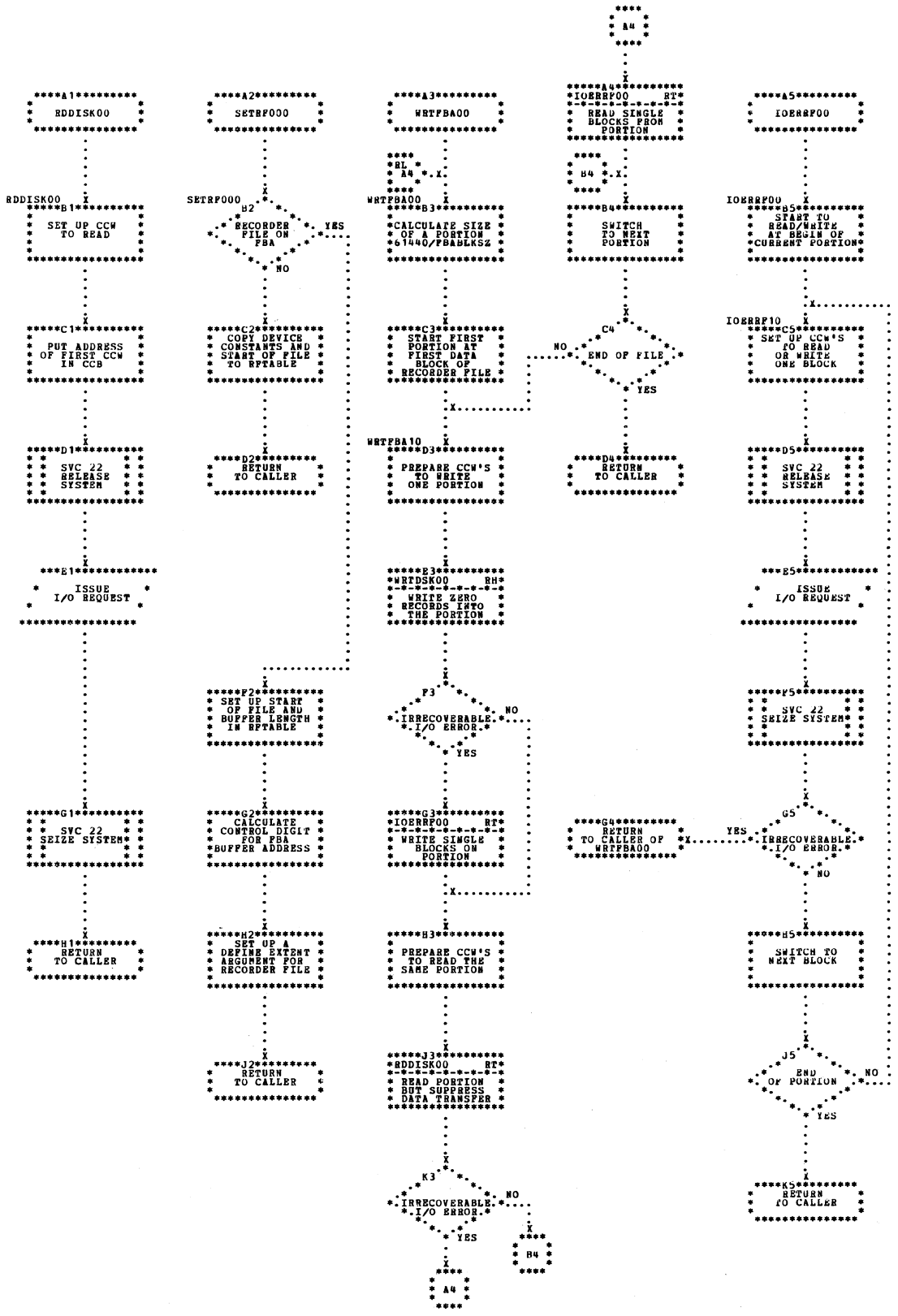


Chart SA. $JOBCTLN - Job Accounting Interface (Part 1 of 2)
Refer to Chart 24.

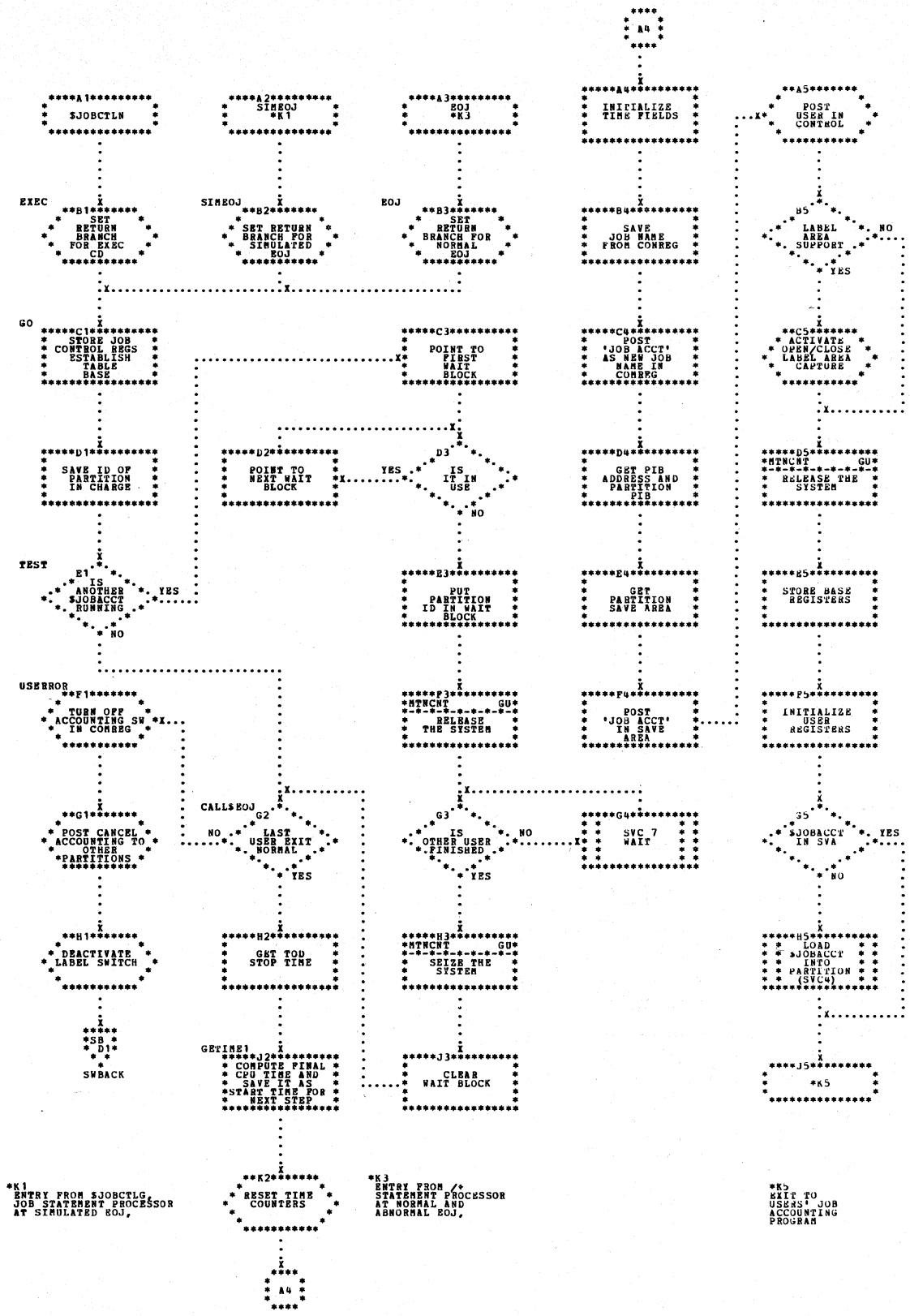


Chart SB. $JOBCTLN - Job Accounting Interface (Part 2 of 2)
Refer to Chart 24.

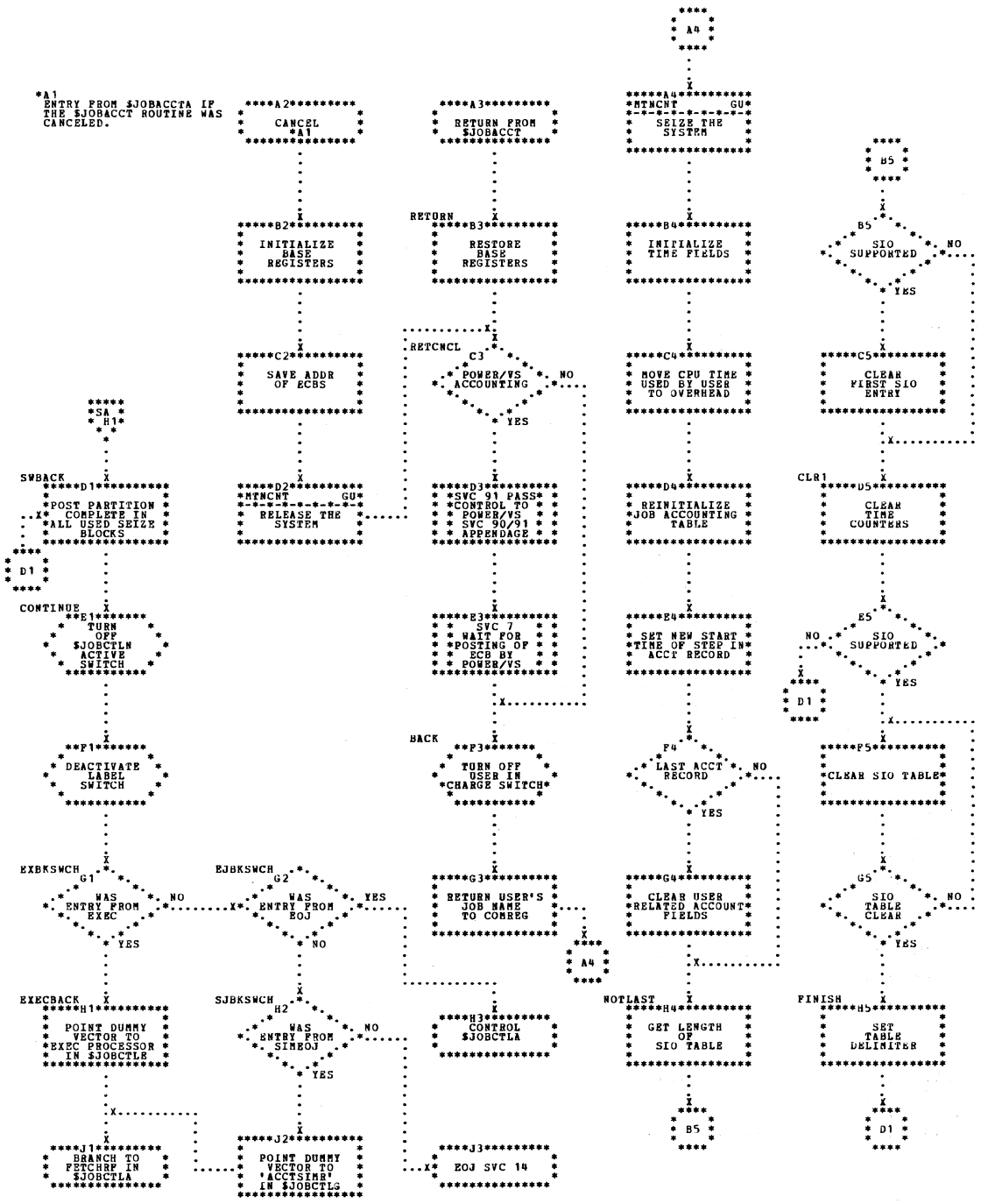


Chart TA. $$BLSTIO - Initialization
Refer to Chart 15.

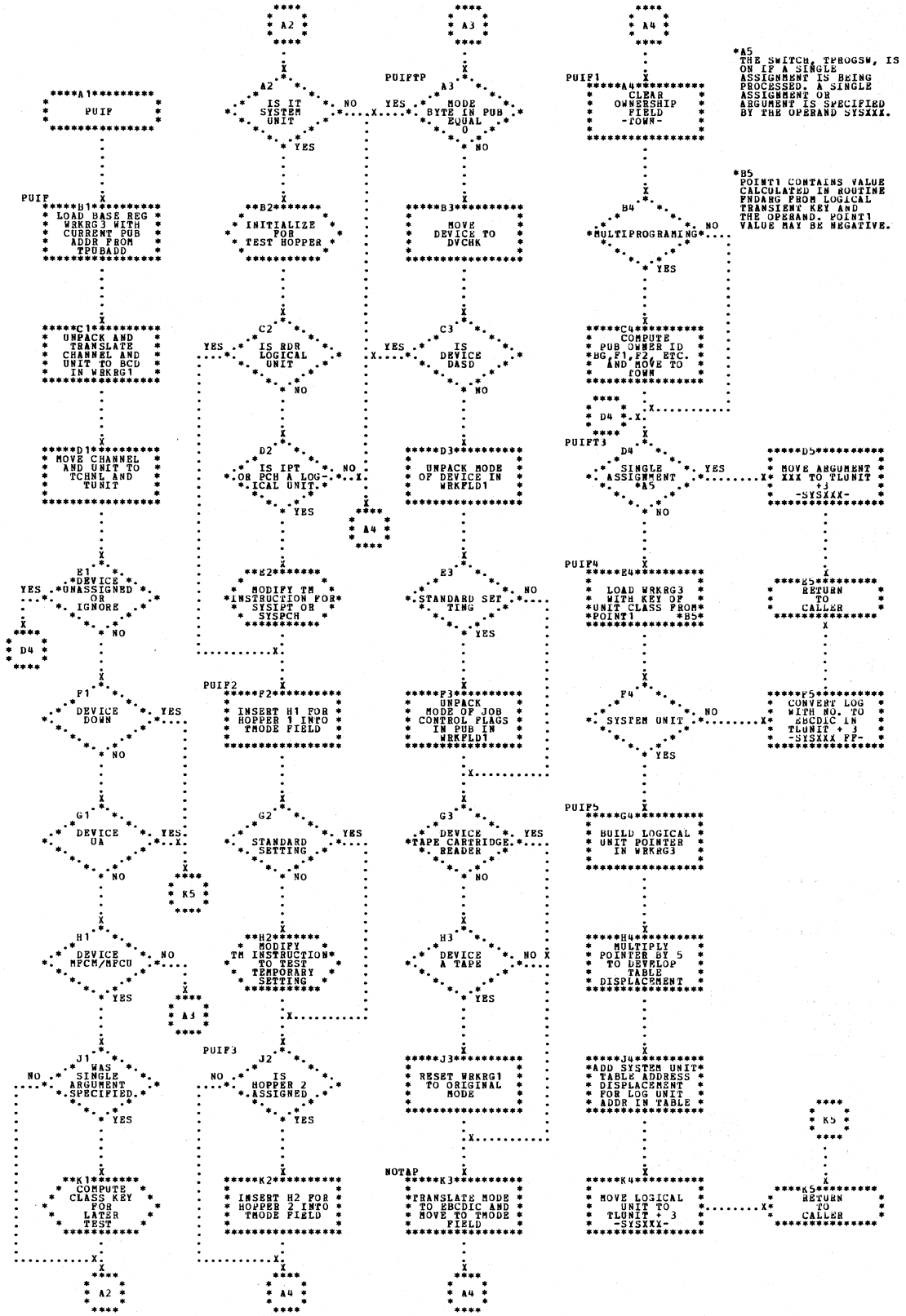


Chart TB. $$BLSTIO - Operand Identification
Refer to Chart 15.

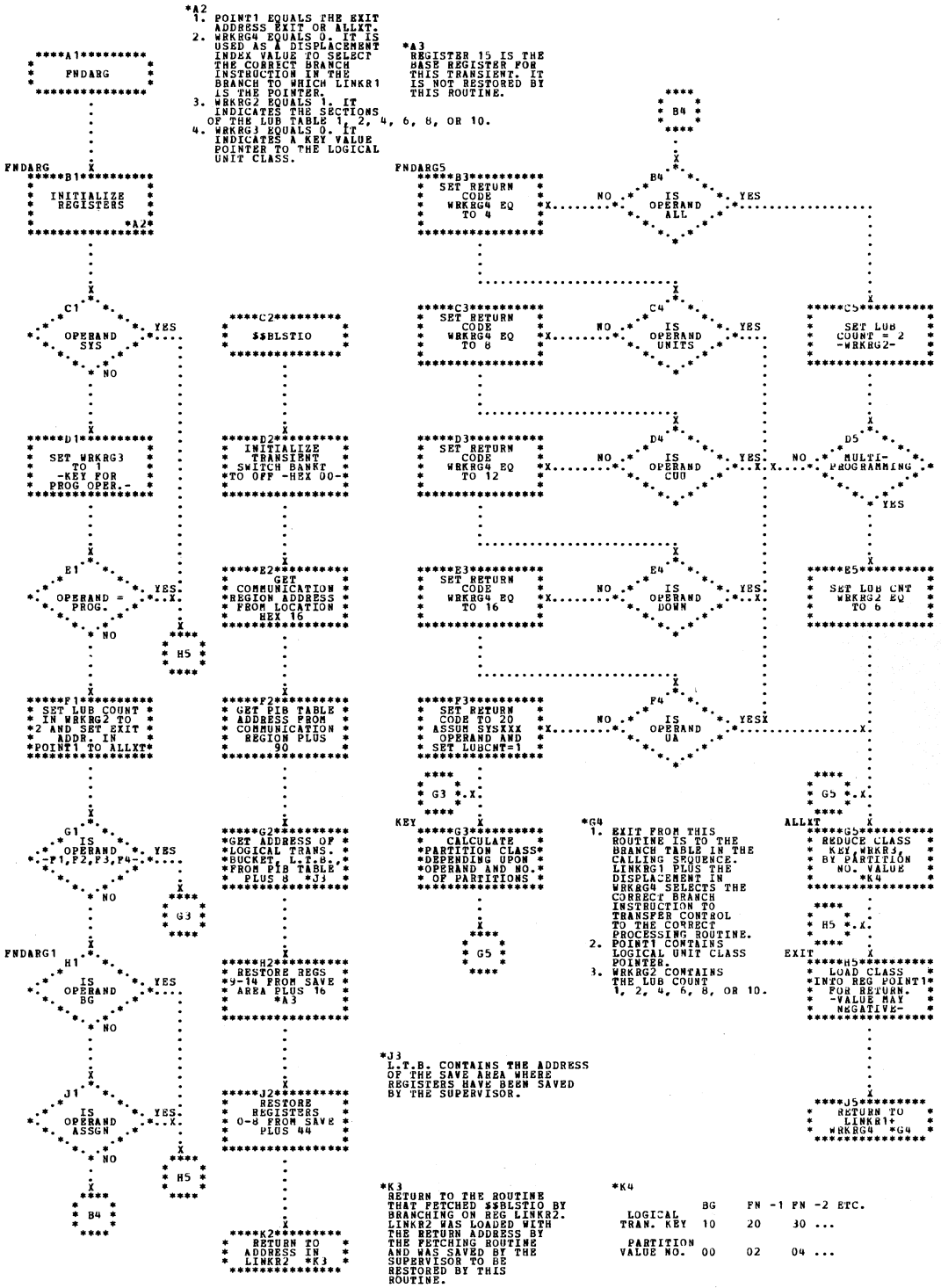
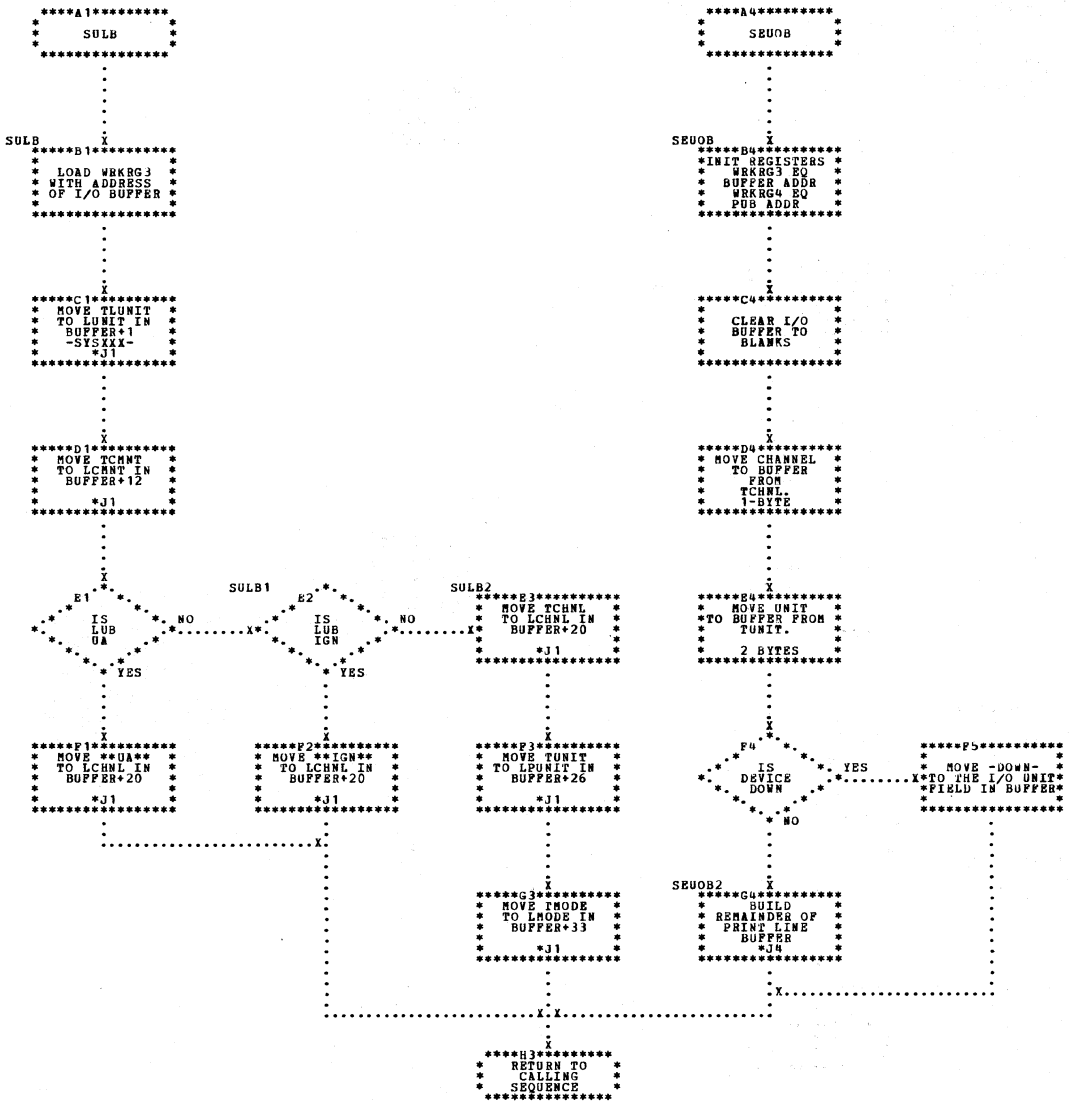


Chart TC. $$BLSTIO - Build Print Line
Refer to Chart 15.



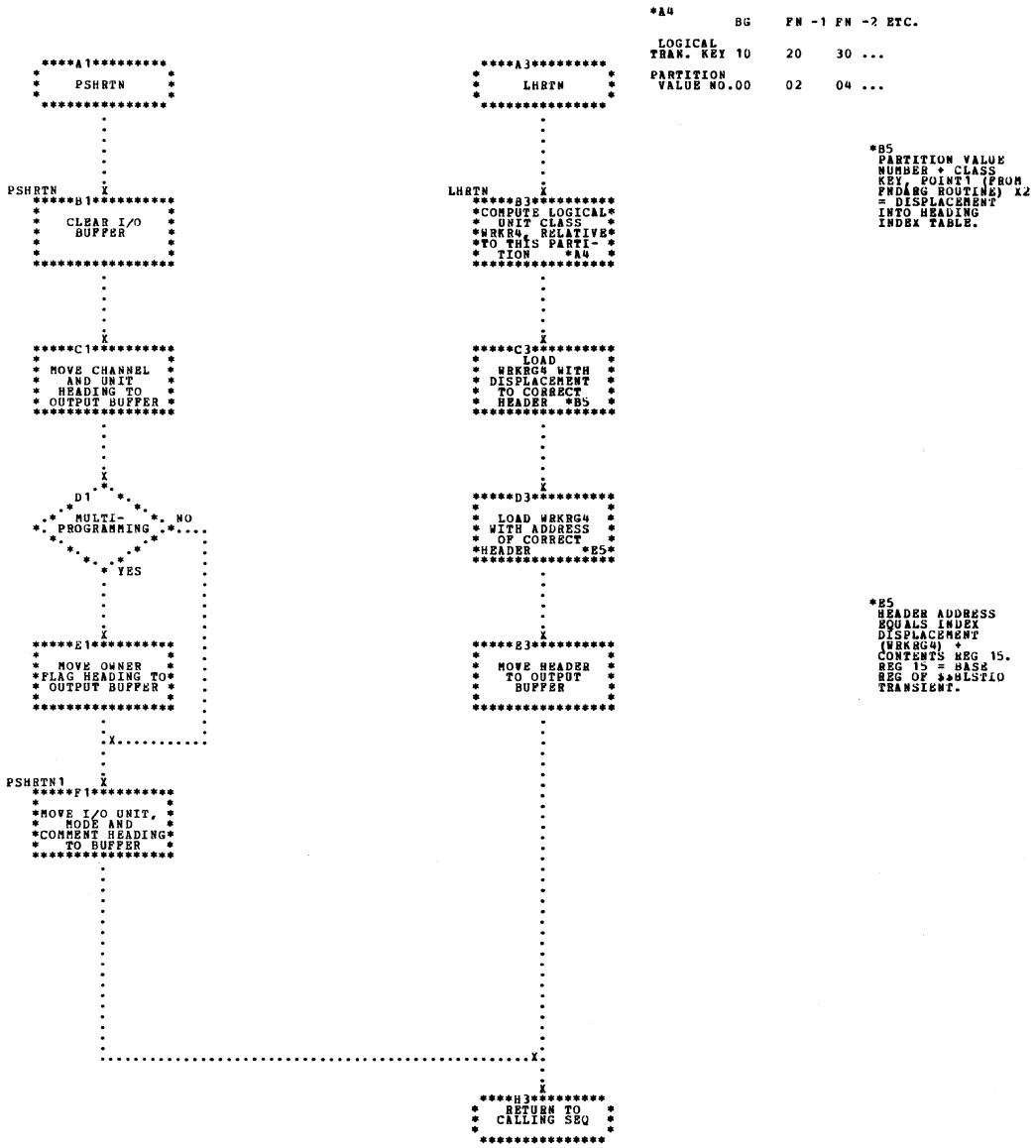
*J1

LUNIT	LCMNT	LCHNL	LPUNIT	LMODE
-SYSXXX-	XXX	**-UA-** **-IGN-**	XX	XX

*J4

1. MOVE OWNER TO OWNER FIELD
2. MOVE MODE FROM TMODE TO MODE FIELD
3. MOVE LOGICAL UNIT FROM TLUNIT TO I/O UNIT FIELD
4. MOVE THE COMMENT FROM TCMNT TO THE COMMENT FIELD-MODE LESS 6 BYTES

Chart TD. $$BLSTIO - Build Header
Refer to Chart 15.



APPENDIX A: LABEL LISTS

1. SLA LABEL LIST

Label		Chart
• ADDLBL	-	26
• ADDNXL	-	26
• CHAINREC	-	32
• CHECKLAS	-	35
• CLRGRPL	-	27
• DETENTRY	-	30
• DETLEN	-	34
• DOIO	-	30
• ENDLBL	-	26
• EXECADD	-	31
• FBAIN	-	35
• FBAOUT	-	32
• FBAREP	-	33
• FIRSTADD	-	31
• FIRSTREC	-	37
• GETLACB	-	29
• GETLBL	-	28
• GETNXGL	-	28
• GETNXL	-	28
• GGRPCKD	-	37
• GGRPFBA	-	37
• GNXTFBA	-	36
• GNXTHIS	-	36
• HISTORY	-	35
• IJBSLA	-	25
• INITALL	-	29
• INITLACB	-	29
• INITPART	-	29
• LOCGRPL	-	28
• RDCHREC	-	34
• RDIDREC	-	37
• RDLABREC	-	34
• READLAS	-	35
• REPLBL	-	27
• SRCHLAB	-	34
• SRCHLAS	-	31
• UPDHIS	-	35
• UPDLACB	-	30
• WRFBALAS	-	32
• WRLABREC	-	32
• WRLACB	-	29
• WRSYSRES	-	30

2. DETAILED FLOWCHART LABEL LIST

Label	Phase	Location
$MAINERR	$JOBCTLJ	PLF3
ACCNT1	$JOBCTLE	LAA3
ACCTBR1	$JOBCTLA	GBG2
ACCTBR2	$JOBCTLG	NBH1
ACCTBR3	$JOBCTLG	NBF4
ACCTBR4	$JOBCTLG	NBC4
ACCTBR5	$JOBCTLG	NCD1
ACCTBR6	$JOBCTLJ	PMB5
ACCTCL	$JOBCTLA	GBK2
ACCTCL	$JOBCTLJ	PMF5
ACCTIGN	$JOBCTLF	MMC3
ACCTIGN1	$JOBCTLE	LBE2
ACCTIGN2	$JOBCTLG	NED4
ACCTIGN4	$JOBCTLG	NAE5
ACCTIGN5	$JOBCTLG	NBK1
ACCTIGN6	$JOBCTLG	NBF5
ACCTIGN7	$JOBCTLJ	PMB3
ACCTIGN9	$JOBCTLA	GBA1
ACCTLV1	$JOBCTLA	GBH3
ACCTLV1B	$JOBCTLJ	PME5
ACCTSIMR	$JOBCTLG	NCF1
ACCTUPDT	$JOBCTLJ	PMK2
ACTION	$JOBCTLJ	PDB1
ACTRSP	$JOBCTLA	GCA1
ACTTEST	$JOBCTLF	MSH3
ADDRTN	$IPLRT3	DAB1
ADREST	$IPLRT4	EFB3
AGAIN	$JOBCTLD	JFE2
AGAIN	SYSBUFF1	FQC4
ALCMSG1	$JOBCTLJ	PRH2
ALIGN	$JOBCTLG	NHJ2
ALIGN	$JOBCTLJ	PPE5
ALLOC	$JOBCTLJ	PQB1
ALLOCR	$JOBCTLJ	PQB2
ALLXT	$$BLSTIO	TBG5
ALPHA	SYSBUFLD	FMB1
ANSVA	$IPLRT6	EUC1
ARND	$JOBCTLG	NNA2
ARND1	$JOBCTLG	NNJ4
ASCIILBL	$JOBCTLD	INC3
ASNRTN	$IPLRT7	EVC1
ASSGN	$JOBCTLD	IAB1
ASSGNALT	$JOBCTLD	IUA5
ASSGNB4	$JOBCTLD	INB1
ASSGNB6	$JOBCTLD	IHB1
ASSGNB6A	$JOBCTLD	IGF3
ASSGNB6D	$JOBCTLD	IGB4
ASSGNB7	$JOBCTLD	IUB1
ASSGNC	$JOBCTLD	IBH2
ASSGNDIB	$JOBCTLD	IGH5
ASSGNLST	$JOBCTLD	IGF5
ASSGNP	$JOBCTLD	IFJ1
ASSGNP1	$JOBCTLD	IFF2
ASSGNP2	$JOBCTLD	IFH5
ASSGNR	$JOBCTLD	IBE2
ASSGNV	$JOBCTLD	IBJ2
ASSGN0	$JOBCTLD	IBD3
ASSGN1	$JOBCTLD	IBE3
ASSGN10	$JOBCTLD	IPD2
ASSGN101	$JOBCTLD	IPJ1
ASSGN11	$JOBCTLD	IPD4
ASSGN12	$JOBCTLD	IPE2
ASSGN20	$JOBCTLD	ISC2
ASSGN20A	$JOBCTLD	ISF1
ASSGN21	$JOBCTLD	IGC3
ASSGN23	$JOBCTLD	ISB3
ASSGN23A	$JOBCTLD	ISG3
ASSGN23K	$JOBCTLD	ISB5
ASSGN24	$JOBCTLD	ISE2
ASSGN26	$JOBCTLD	IJE3
ASSGN27	$JOBCTLD	IJJ3
ASSGN28	$JOBCTLD	ITD1
ASSGN28A	$JOBCTLD	ITE4
ASSGN28B	$JOBCTLD	ITF3
ASSGN28C	$JOBCTLD	ITB3
ASSGN29	$JOBCTLD	IUC2
ASSGN3	$JOBCTLD	IBH3
ASSGN3A	$JOBCTLD	JDB1
ASSGN31	$JOBCTLD	IUF3
ASSGN34	$JOBCTLD	IHD3
ASSGN35	$JOBCTLD	IHC2
ASSGN36	$JOBCTLD	IJB1
ASSGN38	$JOBCTLD	JCJ3
ASSGN39	$JOBCTLD	IUF5
ASSGN40	$JOBCTLD	JCF4
ASSGN403	$JOBCTLD	IGA1
ASSGN404	$JOBCTLD	IGA3
ASSGN41	$JOBCTLD	JDE3
ASSGN41A	$JOBCTLD	JDH4
ASSGN41B	$JOBCTLD	JDC5
ASSGN42	$JOBCTLD	JDF5
ASSGN42A	$JOBCTLD	JEA1
ASSGN42B	$JOBCTLD	JED1
ASSGN42E	$JOBCTLD	JEA4
ASSGN43	$JOBCTLD	JFC4
ASSGN43A	$JOBCTLD	JFC5
ASSGN44	$JOBCTLD	JFB3
ASSGN6	$JOBCTLD	IPC1
ASSGN8	$JOBCTLD	IPF1
ASSGN9	$JOBCTLD	IJB5
ASSGN9A	$JOBCTLD	IJG5
ASSGN9B	$JOBCTLD	IJJ5
ASSGN9C	$JOBCTLD	IJG4
ASSGN901	$JOBCTLD	IHJ2
ASSIGN	$JOBCTLD	IQB1
ASSIGNEX	$JOBCTLD	IUK5
ASSIGN42D	$JOBCTLD	JEJ1
ASSPERM	$JOBCTLD	ISH1
ASSTEMP	$JOBCTLD	ITB1
ASS0	$JOBCTLD	IAD1
ASS1	$JOBCTLD	IAG1
ATNCUU	$JOBCTLA	GVB1
ATNCUU1	$JOBCTLA	GVF1
ATTNCHCK	$$A$IPLR	BFB1
ATTNCHCX	$$A$IPLR	BFK2
AUTONK	$JOBCTLE	LDF3
BACK	$JOBCTLN	SBF3
BACKSP	$JOBCTLD	JFG1
BAD	SYSBUFLD	FPE2

Label	Phase	Location
BAD2	SYSBUFLD	FPF2
BAL	SYSBUFF1	FSH2
BBGSSXXX	$JOBCTLD	IBE1
BCHEST	$IPLRT3	DED1
BEGIPLR	$$A$IPLR	BAB1
BEGXCUU	$JOBCTLD	IQC1
BFOCL	$$A$IPLR	BGC3
BFOCLC	$$A$IPLR	BGD3
BFOCLI	$$A$IPLR	BGJ3
BFSMALL	$JOBCTLM	RKG2
BINCON	$JOBCTLJ	PUB5
BLDAVRT	$IPLRT4	ENB4
BLDAVR10	$IPLRT4	END4
BLDAVR27	$IPLRT4	ENF4
BLDHDR00	$JOBCTLM	RHB5
BLDIPL00	$JOBCTLM	RJB1
BLDIPL05	$JOBCTLM	RJD1
BLDIPL10	$JOBCTLM	RJG1
BLDIPL15	$JOBCTLM	RJF2
BLDIPL20	$JOBCTLM	RJH3
BLDIPL25	$JOBCTLM	RJJ2
BLDLUB2	$IPLRT7	EVJ3
BLDOBR00	$JOBCTLM	REA1
BLDOBR10	$JOBCTLM	REF2
BLDOBR12	$JOBCTLM	REC3
BLDOBR20	$JOBCTLM	REH2
BLDOBR30	$JOBCTLM	REA2
BLDOBR50	$JOBCTLM	REB2
BLDPUB	$IPLRT3	DEB1
BLDP2T00	$IPLRT4	EKC1
BLDP2T05	$IPLRT4	EKH1
BLDP2T10	$IPLRT4	EKB3
BLDP2T12	$IPLRT4	EKC3
BLDP2T20	$IPLRT4	EKD3
BLDP2T25	$IPLRT4	EKE3
BLDP2T30	$IPLRT4	EKK3
BLDP2T40	$IPLRT4	EKF4
BLDSVA	$JOBCTLJ	PPA5
BLDTABL	$IPLRT4	EMB1
BLOCK	$JOBCTLC	HSB1
BMPXCHK	$IPLRT4	ESA2
BMPXINT	$IPLRT4	ERE1
BMPXLP	$IPLRT4	ERG3
BMPXMCR	$IPLRT4	ESE2
BMPXNBT	$IPLRT4	ERD3
BMPXNXT	$IPLRT4	ERG1
BMPXPT	$IPLRT4	ERB5
BMPXRTN	$IPLRT4	ERB1
BMPXSW1	$IPLRT4	ERJ3
BMPXSW2	$IPLRT4	ERE5
BMPXZER	$IPLRT4	ERF1
BMPX8FL	$IPLRT4	ESC4
BMSGI	$$A$IPLR	BAH1
BOTHFEAT	$JOBCTLD	KBK5
BPUBLUB	$$A$IPLR	BGF1
BPUBLUBC	$$A$IPLR	BGJ1
BPUBLUBL	$$A$IPLR	BGE2
BPUBLUBP	$$A$IPLR	BGH2
BPUBSC4	$$A$IPLR	BHE1
BPUBSC6	$$A$IPLR	BHH4
BPUBSC8	$$A$IPLR	BHG3
BPUPSCT	$$A$IPLR	BHB1
BSTOK	$IPLRT3	DFJ1
BTLOOP	$JOBCTLA	GCH5
BTLOOP	$JOBCTLG	NGB3
BTOFRT	$JOBCTLJ	PKG2
BTONRT	$JOBCTLJ	PKG1
BUILDCOM	$$A$IPLR	BDG5
BYPASS	$JOBCTLJ	PDH5
B3DSASLU	$$A$IPLR	BEB1
CALL	$JOBCTLE	LPA1
CALL$EOJ	$JOBCTLN	SAG2
CALLNORM	$JOBCTLE	LPA2
CALL38	$JOBCTLK	QUB1
CALL38A	$JOBCTLK	QUB3
CANCEL	$JOBCTLG	NFB1
CANCELIT	SYSBUFF1	FQE2
CATAL	$JOBCTLG	NJB1
CATAL	$JOBCTLJ	PNB1
CCW1	$$A$IPL1	AAC3
CCW2	$$A$IPL1	AAD3
CCW3	$$A$IPL1	AAE3
CCW4	$$A$IPL1	AAF3
CCW5	$$A$IPL1	AAG3
CDLMANU	$$A$IPLR	BMB5
CDLMANUX	$$A$IPLR	BMF5
CDLREAD	$$A$IPLR	BAG2
CDSYSRDR	$$A$IPLR	BDJ2
CHAIN	$JOBCTLA	GJE2
CHCKDOC	$IPLRT4	END1
CHCLOP	$IPLRT3	DEC1
CHCNT	$IPLRT3	DEE1
CHECKS1	$JOBCTLE	LEG1
CHECK1600	$JOBCTLD	IMG1
CHEK6250	$JOBCTLD	IMB4
CHEXT	$IPLRT3	DHD4
CHFIN	$IPLRT3	DEJ2
CHKASG	$JOBCTLA	GUB3
CHKASG3	$JOBCTLA	GUD3
CHKCNL	$JOBCTLA	GWB2
CHKCNT	$JOBCTLA	GJB4
CHKCSW	IPLBMAC	AHD2
CHKCSW1	IPLBMAC	AHC3
CHKCYL20	$IPLRT5	FGC5
CHKCYL30	$IPLRT5	FGD5
CHKDIB	$JOBCTLD	IGA5
CHKJIB	$JOBCTLA	GFD5
CHKKBYTE	$JOBCTLJ	PRB5
CHKLP	$IPLRT3	DHD1
CHKLST	$JOBCTLA	GFA3
CHKNXC	$JOBCTLD	JHC3
CHKNXC	$JOBCTLK	QRD3
CHKOPN	$JOBCTLD	JSB3
CHKOPN1	$JOBCTLD	JSC3
CHKOPN2	$JOBCTLD	JSB4
CHKOV	$JOBCTLE	LRB1
CHKPART	$JOBCTLJ	PRB3
CHKPART1	$JOBCTLJ	PRD3
CHKPGU	$JOBCTLJ	PTC4
CHKPUB	$JOBCTLF	MPB2
CHKPUN	$JOBCTLK	QJJ3
CHKREADY	$JOBCTLJ	PWB4
CHKRF	$JOBCTLJ	PHA1
CHKRNG	$JOBCTLD	JHB3
CHKRNG	$JOBCTLK	QRC3
CHKRNG1	$JOBCTLK	QRB3
CHKSDL	$JOBCTLJ	PGD1
CHKSTD	$JOBCTLG	NAJ3
CHKTYPE	SYSBUFLD	FNA1
CHKWAIT	$$A$IPLR	BEF4
CHSLOP	$IPLRT3	DHJ5

Label	Phase	Location
CHSWFCL	$IPLRT4	ETG3
CHSWIST	$IPLRT4	ETC3
CHSWLP	$IPLRT4	ETA3
CHSWNXT	$IPLRT4	ETD2
CHSWOFF	$IPLRT4	ETJ3
CHSWRTN	$IPLRT4	ETB1
CHULOP	$IPLRT3	DHE4
CHUPD	$IPLRT3	DHH4
CHURTN	$IPLRT3	DHB4
CHXCNV1	$JOBCTLK	QWC5
CINIT	$JOBCTLE	LPC3
CKCLAS	$JOBCTLF	MPJ1
CKDSCIL	$$A$IPLR	BEB2
CKMSG	SYSBUFLD	FOH1
CKPARM	$JOBCTLK	QVB1
CKPRM0	$JOBCTLK	QVF1
CKPRM1	$JOBCTLK	QVJ1
CKPRM2	$JOBCTLK	QVB3
CKPRM3	$JOBCTLK	QVG3
CKPRM4	$JOBCTLK	QVD2
CKPRM5	$JOBCTLK	QVB5
CKPRM8	$JOBCTLK	QVB4
CLBLUB	$JOBCTLF	MPJ3
CLCKCHK	$IPLRT4	EAH1
CLDLP1	$IPLRT4	EEE4
CLDLP2	$IPLRT4	EEF3
CLDLP2	$IPLRT4	EEJ4
CLDRTN	$IPLRT4	EEB4
CLEAR	$IPLRT2	CBH3
CLEAR	IPLBMAC	AEE3
CLEARFIN	$IPLRT2	CBJ3
CLEAR370	IPLBMAC	AED2
CLNOTOP	$IPLRT2	CCC2
CLOSALT	$JOBCTLD	JGH4
CLOSE	$JOBCTLD	IAB4
CLOSESYS	$JOBCTLD	JGB1
CLOSE1	$JOBCTLD	JJB2
CLOSE3	$JOBCTLD	JGB3
CLOSE4	$JOBCTLD	JGB4
CLOSE7	$JOBCTLD	JGC2
CLOSO	$JOBCTLD	IAF4
CLRDIB	$JOBCTLD	JKB1
CLRDIB	$JOBCTLE	LFJ1
CLRDIB1	$JOBCTLD	JLB2
CLRDIB2	$JOBCTLD	JLC2
CLRDIB3	$JOBCTLD	JKB5
CLRDIB4	$JOBCTLD	JKE5
CLRDIB5	$JOBCTLD	JKK5
CLRDIB6	$JOBCTLD	JLB1
CLRDIB7	$JOBCTLD	JLD1
CLRDN	$JOBCTLF	MPB3
CLRLBL	$JOBCTLG	NJD3
CLR1	$JOBCTLN	SBD5
CLSDU	$JOBCTLD	JKE4
CNCLMD	$JOBCTLA	GWJ5
CNCLMOD1	$JOBCTLA	GWF4
CNCL01	$JOBCTLG	NFE1
CNIOAG	$JOBCTLD	KAE2
CNKLNK	$JOBCTLG	NMB2
CNUNCO	$JOBCTLJ	PTB4
CNUNCO1	$JOBCTLJ	PTD5
CNVRT	$JOBCTLE	LTB2
CNVRTL	$JOBCTLE	LTD2
CNVRTS	$JOBCTLE	LTG2
CODE00	$JOBCTLM	RMG1
CODE05	$JOBCTLM	RMH1
CODE10	$JOBCTLM	RMJ1
CODE15	$JOBCTLM	RMK1
COMCHK	$IPLRT4	EEB2
COMLOG	$IPLRT4	EBF3
COMLOP	$IPLRT4	EEC2
COMPARE	SYSBUFLD	FOJ4
COMPDAY	$IPLRT4	EFE2
CONDSCK	$JOBCTLD	JCD2
CONTIN	IPLBMAC	AEH2
CONTINUE	$JOBCTLN	SBE1
CONTROL	$JOBCTLA	GCB2
CONT3	$JOBCTLA	GCC3
CONT4	$JOBCTLA	GCH3
CONT5	$JOBCTLA	GCJ3
CONT6	$JOBCTLA	GCC5
CONVHEX	$JOBCTLJ	PWB3
COPYLP	$JOBCTLJ	PDF3
CPBFEND	$IPLRT4	EEH3
CPYSLB	$IPLRT7	EVB1
CRBSQ1	$JOBCTLG	NAG3
CRJBSQ	$JOBCTLG	NAA3
CRTTST	$JOBCTLG	NBE3
CRTTST1	$JOBCTLG	NBF3
CSET	$JOBCTLE	LPJ3
CTRLSW	$JOBCTLA	GFJ3
CVHRC	$IPLRT5	FCC3
CVTOC	$IPLRT5	FBB5
CYLDP	$IPLRT4	EBD1
C48	$JOBCTLG	NHG3
C60	$JOBCTLG	NHG4
DASDOP0	$JOBCTLD	ILE1
DASDOP2	$JOBCTLD	ILJ1
DASDVOL	$JOBCTLD	INF1
DATE	$JOBCTLJ	PHB2
DATIMERT	$IPLRT4	EFB1
DBLSCN	$IPLRT3	DAA3
DCUXTN	$JOBCTLA	GFF5
DDLUB	$IPLRT3	DJF2
DEBLOCK	$JOBCTLC	HSB3
DECK	$JOBCTLG	NHB1
DECLP	$IPLRT2	CJC3
DECRL	$IPLRT2	CHD2
DECRR	$IPLRT2	CHE4
DECRTN	$IPLRT2	CJB3
DELLOP	$IPLRT3	DCA3
DELRTN	$IPLRT3	DCA1
DETACHR	$JOBCTLF	MME4
DETBG	$$A$IPLR	BJB2
DETBGLS	$$A$IPLR	BJC4
DETBGRS	$$A$IPLR	BJD2
DETBGX	$$A$IPLR	BJH4
DETCONT	$JOBCTLF	MMF3
DETHW	IPLBMAC	AEA2
DETHWX	IPLBMAC	AEE1
DETHWY	IPLBMAC	AEH1
DET370	IPLBMAC	AEC2
DEVFND	$$A$PLBK	ACB3
DEVTEST	$JOBCTLD	KDB1
DIBRC	$JOBCTLJ	PHH1
DISDEV2	$JOBCTLD	KFC2
DISKDEV	$JOBCTLD	KFB2
DISKET1	$IPLRT2	CDB4
DISKET2	$IPLRT2	CDC4
DISKET3	$IPLRT2	CDE4
DISKT	$IPLRT2	CAJ1

Label	Phase	Location
DISKTMSG	$IPLRT2	CDF5
DKETSCAN	$$A$IPLR	BGF4
DKETSCAR	$$A$IPLR	BGH5
DKSYSRDR	$$A$IPLR	BDC5
DLAB	$JOBCTLK	QFB1
DLBBUFSP	$JOBCTLK	QGC3
DLBCAT	$JOBCTLK	QGC2
DLBKEYW	$JOBCTLK	QGB2
DLBL	$JOBCTLK	QEB1
DLBL1	$JOBCTLK	QED1
DLBQKW	$JOBCTLK	QGG1
DNDERR	$JOBCTLD	KAB3
DNEERR	$JOBCTLJ	PVC3
DOCHCK	$$A$IPLR	BCF1
DOCRTN	$IPLRT2	CCF5
DOFFSET	$JOBCTLA	GRD1
DOIO	SYSBUFF1	FSC2
DOP34	$JOBCTLK	QMB1
DOWN	$JOBCTLF	MEB3
DPDOK	$IPLRT2	CFE4
DQNOT	$JOBCTLG	NNE2
DQXTNT	$JOBCTLG	NNF5
DSDP1	$IPLRT4	EDC4
DSDP2	$IPLRT4	EDE4
DSDP3	$IPLRT4	EDF4
DSDP4	$IPLRT4	EDG4
DSDRTN	$IPLRT4	EDB4
DSKDEV	$JOBCTLE	LTB5
DSKIND	$JOBCTLA	GRH2
DSKINIT	$JOBCTLE	LTB3
DUDFB	$JOBCTLA	GPB4
DUDFI	$JOBCTLA	GRB3
DUMP	$JOBCTLG	NKJ3
DUPKEYW	$IPLRT2	CLB3
DVCDN	$JOBCTLF	MHB1
DVCDNC	$JOBCTLF	MJF1
DVCDNS	$JOBCTLF	MJB1
DVCDNX	$JOBCTLF	MJF5
DVCDN10	$IPLRT4	EBC4
DVCDN10	$IPLRT4	ELC1
DVCDN11	$IPLRT4	EBJ4
DVCDN11	$IPLRT4	ELB4
DVCDN14	$JOBCTLF	MHF2
DVCDN14A	$JOBCTLF	MHK2
DVCDN14B	$JOBCTLF	MHA3
DVCDN2	$JOBCTLF	MHD3
DVCDN25	$IPLRT4	EBB5
DVCDN25	$IPLRT4	ELB2
DVCDN27	$IPLRT4	EBC5
DVCDN3	$JOBCTLF	MHB2
DVCDN30	$IPLRT4	ELF2
DVCDN5	$JOBCTLF	MHB4
DVCDN6	$JOBCTLF	MJJ1
DVCDN7	$JOBCTLF	MJF4
DVCDN8	$JOBCTLF	MJH2
DVCUP	$JOBCTLF	MLB2
EDECK	$JOBCTLG	NHJ4
EDITL	$IPLRT5	FDB3
EJBKSWCH	$JOBCTLN	SBG2
EMAP	$JOBCTLF	MKA3
END	SYSBUFLD	FME3
ENDDVTCK	$JOBCTLD	IJD4
ENDINC	$JOBCTLJ	PDH4
ENDLOOP	$IPLRT4	EAB4
ENDMONTH	$IPLRT4	EFG2
ENDMSG	$$BUFLDR	FHC5
ENDOPSCN	$JOBCTLD	IBA3
ENDROUT	$IPLRT4	EMB3
ENDSCAN	$JOBCTLD	INE5
ENDTIME	$JOBCTLG	NEF4
ENFND	$IPLRT3	DEK1
ENTWAIT	$$A$IPLR	BEH5
EODADDR	$JOBCTLD	JKD4
EOFLAB	$JOBCTLK	QOL2
EOFPRC	$JOBCTLA	GTJ3
EOJ	$JOBCTLG	NDE1
EOJ	$JOBCTLN	SAB3
EOJBY	$JOBCTLG	NBA4
EOJNRF	$JOBCTLG	NDB1
EOJPIB	$JOBCTLG	NDB2
EPART	$JOBCTLF	MKD3
EREXIT	$JOBCTLJ	PFE3
ERRAP	$JOBCTLA	GWG3
ERRAP1	$JOBCTLA	GWC4
ERRCHK	$JOBCTLD	IFB1
ERREND	$JOBCTLE	LQJ4
ERROFF	$JOBCTLA	GSC5
ERROI07A	$$A$PLBF	ADG3
ERROR	$$BUFLDR	FKD5
ERROR	$JOBCTLE	LQD4
ERROR0	IPLBMAC	AHB5
ERROR1	$JOBCTLE	LQC4
ERROR2	$JOBCTLE	LQE4
ERROR87	$IPLRT4	EMG4
ERRRTN	$JOBCTLA	GWD2
ERRRTN1	$JOBCTLF	MNK5
ERRRTN2	$JOBCTLA	GWF2
ERRRTN2	$JOBCTLF	MNJ5
ERRS	$JOBCTLG	NHG1
ERRTN	$JOBCTLF	MNC5
ERR0I00A	$$A$IPLR	BBF4
ERR0I00A	$$A$PLBF	ADH4
ERR0I00A	$$A$PLBK	ACC4
ERR0I02A	$$A$IPLR	BBD4
ERR0I05A	$$A$IPLR	BGE5
ERR0I06A	$$A$PLBK	ACA4
ERR0I07A	$$A$PLBK	ACH3
ERR0I08A	$$A$IPLR	BBE2
ERR0I09A	$$A$IPLR	BBG5
ERR0I09A	$$A$PLBF	ADJ3
ERR0I1A	$$A$IPLR	BDG3
ERR0I68A	$$A$IPLR	BCF2
ERR0I68B	$$A$IPLR	BCB2
ERR0I68C	$$A$IPLR	BCE2
ERR0I684	$$A$IPLR	BCA2
ERR1M21	$JOBCTLG	NAJ2
ERR1T10I	$JOBCTLJ	PLA4
ER1121	$IPLRT4	EDH4
ER3800	$$BUFLDR	FHJ4
ESETID	$JOBCTLF	MKC4
ESKIPRTY	$JOBCTLF	MKB4
ESTVOPEN	$JOBCTLJ	PFB5
ESUPV	$JOBCTLF	MKC3
EXBKSWCH	$JOBCTLN	SBG1
EXBLNK	$JOBCTLE	LBF1
EXCECP	$JOBCTLA	GRA4
EXCEDT	$JOBCTLE	LFB1
EXCP	$$BUFLDR	FKD1
EXCP	$JOBCTLD	JUB1
EXCPGO	$JOBCTLA	GTD3
EXCPIPT	SYSBUFLD	FPF3

Label	Phase	Location
EXCPRG	$JOBCTLA	GPB1
EXCPROG	$JOBCTLD	JUB4
EXCPROG1	$JOBCTLD	JUB5
EXCPROG2	$JOBCTLD	JUD4
EXCPROG3	$JOBCTLD	JUE5
EXCPROG4	$JOBCTLD	JUF4
EXEC	$JOBCTLE	LAB1
EXEC	$JOBCTLN	SAB1
EXECBACK	$JOBCTLN	SBH1
EXFBA	$JOBCTLE	LAE5
EXFBA10	$JOBCTLE	LAG5
EXIT	$$BLSTIO	TBH5
EXIT	$JOBCTLA	GDB3
EXIT	$JOBCTLD	IAC2
EXIT	SYSBUFF1	FRG5
EXPEXT	$JOBCTLA	GTH4
EXTNT	$JOBCTLK	QKB1
EXTR	$IPLRT5	FFB1
EX000	$JOBCTLE	LAJ3
EX005	$JOBCTLE	LAH1
EX010	$JOBCTLE	LAJ1
EX115	$JOBCTLE	LAC2
EX150	$JOBCTLE	LAE4
EX155	$JOBCTLE	LAH4
EX160	$JOBCTLE	LAK4
EX170	$JOBCTLE	LBD1
FBACDL0	$$A$IPLR	BAC2
FBADIB	$JOBCTLE	LTB4
FBAINT	$JOBCTLA	GPF4
FBAIN1	$$A$IPLR	BAG1
FBAPROC	$JOBCTLE	LRB5
FBAREAD	$JOBCTLE	LRE5
FBASCAN	$JOBCTLE	LRF5
FBASLD	$JOBCTLD	KGB1
FBASLD1	$JOBCTLD	KGE2
FBASLD4	$JOBCTLD	KGD4
FBASUPD	$$A$IPLR	BBC3
FBASUPL	$$A$IPLR	BBG4
FDDKCODE	$JOBCTLK	QFK3
FDKIJ	$JOBCTLK	QEB2
FDKIJL	$JOBCTLK	QEH2
FDKIJ2	$JOBCTLK	QEJ2
FDKTDAT	$JOBCTLK	QDH3
FDKTDAT1	$JOBCTLK	QDJ5
FDKTDAT2	$JOBCTLK	QDF4
FDKTID	$JOBCTLK	QDB1
FDKTID2	$JOBCTLK	QDH2
FDKTVNM	$JOBCTLK	QCC3
FDKTV2	$JOBCTLK	QCB2
FDK1	$JOBCTLK	QEF4
FDK2	$JOBCTLK	QEH4
FDSRTN	$IPLRT2	CJB2
FDSYSU	$JOBCTLJ	PTD4
FEED	$JOBCTLA	GSE2
FESEQ	$JOBCTLK	QPB1
FESPLIT	$JOBCTLK	QLF3
FESPLIT3	$JOBCTLK	QKJ5
FESPLIT3	$JOBCTLK	QLB4
FESYSX2	$JOBCTLK	QKH1
FESYSX3	$JOBCTLK	QKA2
FETCH	$$BUFLDR	FKH1
FETCHM	$JOBCTLG	NBJ3
FETCHN	$JOBCTLG	NBD4
FETCHRF	$JOBCTLA	GDB4
FETCHX	$IPLRT2	CFF3
FETCH1	$$BUFLDR	FHG4
FETCH2	SYSBUFF1	FQJ4
FETCH4	$IPLRT2	CFB3
FETCH51	$IPLRT2	CFG3
FETCH6	$IPLRT2	CFC4
FETCH7	$IPLRT2	CFC2
FETINSRO	$JOBCTLK	QCE4
FETINSRT	$JOBCTLK	QCB4
FETINSR1	$JOBCTLK	QCF4
FETINSR3	$JOBCTLK	QCH4
FETINSR4	$JOBCTLK	QCJ4
FETINSR5	$JOBCTLK	QCK4
FETK	$JOBCTLK	QLF1
FETKNO	$JOBCTLK	QLC2
FETKNO1	$JOBCTLK	QLC3
FETKTK	$JOBCTLK	QPB3
FETYPE	$JOBCTLK	QKC3
FETYPED	$JOBCTLK	QKK3
FETYPEI	$JOBCTLK	QKA4
FETYPEI4	$JOBCTLK	QLA1
FETYPE1	$JOBCTLK	QKE5
FEVSER	$JOBCTLK	QKB2
FEVSER1	$JOBCTLK	QKC2
FEVSER2	$JOBCTLK	QKD2
FICNIC	$JOBCTLA	GAH1
FIND	$JOBCTLE	LPJ2
FINDOPN	$JOBCTLA	GRK2
FINDPRC	$JOBCTLE	LRB2
FINDSIZE	$IPLRT2	CBA3
FINISH	$JOBCTLN	SBH5
FIRSTCYL	$JOBCTLK	QOD3
FJB3	$JOBCTLE	LSB2
FLAGOFF	$JOBCTLK	QNF1
FND	$JOBCTLE	LRG3
FNDARG	$$BLSTIO	TBB1
FNDARG1	$$BLSTIO	TBH1
FNDARG5	$$BLSTIO	TBB3
FNDOPER	$JOBCTLD	IBD4
FNDSTAT	$JOBCTLD	IKB5
FNDSTAT0	$JOBCTLD	ILE4
FNDSTAT1	$JOBCTLD	ILJ4
FNDTYP	$IPLRT3	DFB1
FNIOAG	$JOBCTLD	KAE3
FNXTDRBLK	IPLBMAC	AGE1
FORMAT	$IPLRT5	FDB1
FORMR	$IPLRT5	FEB1
FOUND	$$BUFLDR	FHA2
FOUND	$JOBCTLD	KBF5
FOUND	$JOBCTLE	LRH3
FPRTN	$IPLRT4	EJB1
FRDBCCW	IPLBMAC	AGF3
FRDNOIN	IPLBMAC	AGE3
FRDPHASE	IPLBMAC	AGC3
FSCAN	$JOBCTLK	QBB4
FSCAN1	$JOBCTLK	QBC4
FSCILDR	IPLBMAC	AGB1
FSCLDIRF	IPLBMAC	AGJ1
FSCLDIRN	IPLBMAC	AGJ2
FSCLDIRS	IPLBMAC	AGH1
FSCLDIRX	IPLBMAC	AGK2
FTCHJIB	$JOBCTLE	LSB1
FTEND1	$JOBCTLK	QBD3
GETADD	$IPLRT2	CFA2
GETASG	$JOBCTLJ	PEA3
GETCC	SYSBUFF1	FRD1

Label	Phase	Location
GETDAT	$JOBCTLG	NLB2
GETDEF	$IPLRT2	CFA5
GETDEL	$IPLRT2	CFA1
GETDIR	$JOBCTLE	LEA5
GETDPD	$IPLRT2	CFD3
GETIME1	$JOBCTLG	NLB1
GETIME1	$JOBCTLN	SAJ2
GETIME2	$JOBCTLG	NLC1
GETJIB	$JOBCTLD	JNB1
GETJIBSW	$JOBCTLD	JNK1
GETJIB1	$JOBCTLD	JNH1
GETJIB2	$JOBCTLD	JND2
GETK	$JOBCTLJ	PQB3
GETLAN	$JOBCTLD	JHB2
GETLUB	$JOBCTLF	MUB1
GETLUB	$JOBCTLG	NPB1
GETLUB	$JOBCTLK	QTE3
GETNAME	$JOBCTLJ	PPD2
GETNEXT	$JOBCTLD	IKD1
GETNICL	$JOBCTLF	MUF1
GETOPND	$JOBCTLK	QTB4
GETPBOWN	$JOBCTLD	JRB4
GETPID	$JOBCTLJ	PQD1
GETPUB	$JOBCTLG	NPB2
GETRESC	$$A$PLBF	ADD2
GETRESC	$$A$PLBK	ACD2
GETSET	$IPLRT2	CFA3
GETSIZE	$JOBCTLJ	PSF1
GETSVA	$IPLRT2	CFA4
GETSYS	$JOBCTLF	MUD2
GETVC	$IPLRT5	FGB1
GETVOL	$JOBCTLA	GSB2
GETVO1	$JOBCTLD	IKB2
GO	$JOBCTLN	SAC1
GPEXIT	$JOBCTLD	KBE2
GPEXIT1	$JOBCTLD	KBE3
GPEXIT2	$JOBCTLD	KBA5
GUNPA	$JOBCTLD	JFB4
HCCREATE	$JOBCTLC	HPB4
HCCREA10	$JOBCTLC	HPF4
HCCREA20	$JOBCTLC	HPC5
HCCREA30	$JOBCTLC	HPD5
HCEND	$JOBCTLJ	PJJ3
HCIOERR	$JOBCTLC	HQK4
HCMSG	$JOBCTLC	HRB1
HCNO	$JOBCTLJ	PJA4
HCOPNMSG	$JOBCTLC	HSB5
HCSET	$JOBCTLJ	PJK3
HCSMALL	$JOBCTLC	HPJ2
HCYES	$JOBCTLJ	PJF4
HCYES01	$JOBCTLC	HQE1
HCYES02	$JOBCTLC	HQG1
HCYES05	$JOBCTLC	HQK1
HCYES10	$JOBCTLC	HQB2
HCYES20	$JOBCTLC	HQE2
HCYES30	$JOBCTLC	HQB3
HCYES40	$JOBCTLC	HQE3
HCYES60	$JOBCTLC	HQH2
HEADNO	$JOBCTLA	GRC5
HEXCON	$JOBCTLD	JJB1
HEXCON	$JOBCTLJ	PTB2
HEXRTN	$IPLRT3	DHC1
HOLD	$JOBCTLJ	PAB3
HOLD1	$JOBCTLJ	PAF3
HOLD2	$JOBCTLJ	PAB4

Label	Phase	Location
IGNORE	$JOBCTLA	GCB4
IJBIPL6	$IPLRT6	EUB1
IJBJOBRS	$JOBCTLB	HAB2
ILLCARD	$IPLRT4	EAB5
ILLCD	$IPLRT2	CKB1
ILLCD1	$IPLRT2	CKC1
ILUS	$JOBCTLD	KAB1
ILUS	$JOBCTLF	MNE4
INAERR	$JOBCTLJ	PVF1
INCLUDE	$JOBCTLJ	PDB3
INCL02	$JOBCTLA	GLA2
INCL05	$JOBCTLA	GLE2
INCL15	$JOBCTLA	GLG3
INCL20	$JOBCTLA	GLH3
INCL25	$JOBCTLA	GLJ3
INCRPTR	$JOBCTLK	QWF1
INCRPUB	$JOBCTLD	KBK4
INCRTRK	$JOBCTLK	QOA1
INDIB	$JOBCTLA	GRD2
INDIGN	$JOBCTLD	ICD2
INDSEQ	$JOBCTLK	QHG3
INDUA	$JOBCTLD	ICD1
INDVTP	$JOBCTLD	KAB2
INIT	$JOBCTLE	LAA4
INIT	SYSBUFLD	FMD1
INITDIB	$JOBCTLA	GXB1
INITDIB1	$JOBCTLA	GXA3
INITDIB7	$JOBCTLA	GXA5
INITIAL	$IPLRT4	EMF1
INITIAL	$JOBCTLK	QTD1
INITL	$JOBCTLD	JHB1
INITSCAN	$JOBCTLJ	PPJ2
INITSEC	$JOBCTLA	GXD2
INIT00	$JOBCTLM	REB5
INIT020	$JOBCTLC	HNA4
INIT060	$JOBCTLC	HND4
INIT070	$JOBCTLC	HNE4
INIT080	$JOBCTLC	HNH4
INIT080	$JOBCTLM	REK4
INIT20	$JOBCTLM	REG5
INIT3800	$$BUFLDR	FHF5
INPCONS	$IPLRT2	CKB3
INPDKET	$IPLRT2	CKB4
INVASGN	$JOBCTLD	JCF3
INVCC	SYSBUFF1	FQE5
INVFOR	$JOBCTLD	JFH3
INVKEYW	$IPLRT2	CLB1
INVNUM	$JOBCTLJ	PUG1
INVSEQ	$IPLRT2	CLB5
INVSPEC	$IPLRT2	CLB2
INVSTD	$JOBCTLJ	PXH2
INVSTDNO	$JOBCTLJ	PXG2
INVSTD1	$JOBCTLJ	PXB4
INVSYN	$JOBCTLJ	PSF2
INVSYN1	$JOBCTLJ	PSE2
INVSYN2	$JOBCTLJ	PSE3
INVVAL1	$JOBCTLJ	PSB5
IOCOMPL	$JOBCTLA	GTA3
IOERR	IPLBMAC	AHA2
IOERRF00	$JOBCTLM	RTB5
IOERRF10	$JOBCTLM	RTC5
IOHALT	$IPLRT2	CKH1
IOHLD	$IPLRT2	CKB2
IORCKD	IPLBMAC	AHH3
IORFBA	IPLBMAC	AHJ4
IORTN	IPLBMAC	AHB1

Label	Phase	Location
IOSTOP	$IPLRT2	CKJ1
IPLCCW1	$$A$IPL0	ABC3
IPLCCW2	$$A$IPL0	ABD3
IPLCCW3	$$A$IPL0	ABE3
IPLCCW4	$$A$IPL0	ABF3
IPLPSW	$$A$IPL0	ABB3
IPLTERM	$IPLRT2	CFH2
IPTDATA	$JOBCTLE	LRH1
IS 81	$JOBCTLD	JEA5
ISCKSQ	$JOBCTLK	QHD4
ISPRNT	SYSBUFF1	FSG1
ISSMSG	$IPLRT2	CDD3
ISSMSG10	$IPLRT2	CDC3
ISSMSG50	$IPLRT2	CFD5
ISSMSG54	$IPLRT2	CFG2
ISSMSG94	$IPLRT2	CFF2
ISSUE66	$IPLRT2	CAH4
ISSUIO	$JOBCTLA	GTK1
ISSUMSG	SYSBUFF1	FQB1
ISTYP4	$JOBCTLK	QHG4
ISX	SYSBUFF1	FRA2
IVDS	$JOBCTLD	KAB4
IVDS	$JOBCTLF	MNA4
IVSERR	$JOBCTLJ	PVB5
JCBCNVBD	$JOBCTLB	HCC5
JCBERMSG	$JOBCTLB	HCB5
JCBMSGMV	$JOBCTLB	HCD5
JCBMSGM2	$JOBCTLB	HCE5
JCB0060	$JOBCTLB	HAC3
JCB0200	$JOBCTLB	HAE3
JCB0500	$JOBCTLB	HAC1
JCB0900	$JOBCTLB	HBB1
JCB1200	$JOBCTLB	HBJ1
JCB1220	$JOBCTLB	HBC2
JCB1800	$JOBCTLB	HBB3
JCB2000	$JOBCTLB	HBH4
JCB2010	$JOBCTLB	HCB1
JCB2160	$JOBCTLB	HCK1
JCB2350	$JOBCTLB	HCC2
JCB2600	$JOBCTLB	HCD3
JCB2800	$JOBCTLB	HCB4
JFB1	$JOBCTLE	LSE1
JIBPRT	$JOBCTLG	NNH5
JOB	$JOBCTLG	NAB1
JOBCTL	$JOBCTLA	GAB1
JOBCTL4	$JOBCTLA	GBG1
JOBCTL5	$JOBCTLA	GBF2
JOB05	$JOBCTLG	NAA2
KEY	$$BLSTIO	TBG3
KEYCHK	$IPLRT3	DBC2
KEYCHK1	$IPLRT3	DBA2
KEYCHK2	$IPLRT3	DAF4
KLEER	$$BUFLDR	FKF3
KNXTBLK	IPLBMAC	AFB3
KNXTBLK2	IPLBMAC	AFD3
KSCILDR	IPLBMAC	AFB1
LAXERR	$JOBCTLK	QSC3
LBLOUT	$JOBCTLK	QNJ2
LBLOUT1	$JOBCTLK	QOE1
LBLPED	$IPLRT3	DJK1
LBLTYP	$JOBCTLK	QAB1
LBTOUT	$JOBCTLK	QAG2
LDMNT	$JOBCTLA	GAB3
LDSVA3	$IPLRT7	EVG3
LDSVA5	$IPLRT7	EVH3
LHRTN	$$BLSTIO	TDB3
LINK	$JOBCTLG	NJB2
LINKINT	$JOBCTLA	GQB2
LINKINTF	$JOBCTLA	GQB4
LIOCUU	$JOBCTLF	MDB2
LIOEOJ	$JOBCTLF	MAA4
LIOEOJ1	$JOBCTLF	MAB4
LIOL	$JOBCTLF	MBA2
LIOLL201	$JOBCTLF	MBD1
LIOLX	$JOBCTLF	MBB1
LIOL1	$JOBCTLF	MBA3
LIOL2	$JOBCTLF	MBA4
LIOL202	$JOBCTLF	MBE1
LIOL3	$JOBCTLF	MBE3
LIOL4	$JOBCTLF	MBF3
LIOSYX	$JOBCTLF	MDB3
LIST	$JOBCTLG	NHB3
LISTIO	$JOBCTLF	MAB1
LISTX	$JOBCTLG	NHB5
LLUPD	$IPLRT5	FGB3
LMOD0G	$IPLRT4	EQD1
LMOD02	$IPLRT4	EQE2
LMOD03	$IPLRT4	EQK2
LMOD04	$IPLRT4	EQA1
LMOD07	$IPLRT4	EQE3
LNAERR	$JOBCTLG	NME3
LNKNOP	$JOBCTLE	LHA3
LNKOUT	$JOBCTLJ	PUC3
LOADA	$IPLRT2	CAH1
LOADFCB	SYSBUFF1	FRH2
LOADPH1	$IPLRT2	CFE1
LOADPH11	$IPLRT2	CFG1
LOADRS	$JOBCTLK	QQB4
LOADRS1	$JOBCTLK	QQF4
LOAD1	$$BUFLDR	FHB4
LOAD2	$$BUFLDR	FHE4
LOG	$JOBCTLJ	PKB5
LOGCHK	$JOBCTLA	GLB4
LOGIN	$JOBCTLA	GMC2
LOGIN0	$JOBCTLA	GME2
LOGIN1	$JOBCTLA	GMF2
LOGIPL	$IPLRT2	CCF2
LOGOUT	$JOBCTLA	GJC1
LOGPRT	$JOBCTLA	GLH4
LOGRED	$IPLRT2	CDE3
LOGREQ	$$A$IPLR	BBA1
LOGREQL0	$JOBCTLM	RKC5
LOGREQ00	$JOBCTLM	RKB4
LOGREQ01	$JOBCTLC	HSB4
LOGREQ01	$JOBCTLM	RKD4
LOGREQ5	$$A$IPLR	BBB1
LOGRITE	$$BUFLDR	FKB1
LOGSTR	$IPLRT2	CDB3
LOOKDOC	$IPLRT4	ENE1
LOOP	$JOBCTLG	NKA2
LOOPS3	$JOBCTLJ	PSJ1
LOOP2	$JOBCTLJ	PRC5
LOOP3	$JOBCTLJ	PPD3
LPDOPT	$JOBCTLJ	PDE5
LSTOUT	$JOBCTLA	GJB3
LSTTRACK	$JOBCTLK	QOA2
LUBUPL	$IPLRT3	DJE1
LUB2INI1	$IPLRT7	EVK3
LUDRTN	$IPLRT3	DJB2

Label	Phase	Location
LUURTN	$IPLRT3	DJB1
MAIN00	$JOBCTLC	HMB1
MAIN00	$JOBCTLM	RAB1
MAIN020	$JOBCTLM	RAJ3
MAIN100	$JOBCTLC	HMF1
MAIN200	$JOBCTLC	HMG2
MAIN200	$JOBCTLM	RAH4
MAIN220	$JOBCTLM	RAJ4
MAPCHN00	$JOBCTLM	RGB1
MAPCHN10	$JOBCTLM	RGG1
MAPCHN15	$JOBCTLM	RGA2
MAPCHN20	$JOBCTLM	RGD2
MAPCHN25	$JOBCTLM	RGG2
MAPCHN30	$JOBCTLM	RGF3
MAPCH05	$JOBCTLM	RGE1
MAPDMC	IPLBMAC	AEH4
MAP370	$JOBCTLF	MKC1
MCRETURN	$IPLRT2	CBA4
MDR3401	$JOBCTLM	RCE3
MDR3800	$JOBCTLM	RCD3
MDR3895R	$JOBCTLM	RCB3
MDVTPLST	$JOBCTLD	KBA3
MICRTST	$IPLRT4	ESB5
MIC30A	$JOBCTLE	LUD2
MIP20D	$JOBCTLE	LUB4
MISKEYW	$IPLRT2	CLB4
MMDD1	$JOBCTLJ	PGJ3
MODEOK	$IPLRT2	CBE3
MODERTN	$IPLRT4	EQB2
MODLOP	$IPLRT3	DDD1
MODMOVE	$IPLRT3	DDE2
MODSCAN	$JOBCTLD	KCD3
MONITOR	$IPLRT2	CDA1
MONITOR2	$IPLRT2	CDC1
MOVEC	$JOBCTLK	QWG2
MOVEMSG	SYSBUFF1	FSA1
MOVINFO	$IPLRT7	EVE1
MSGMICR	$IPLRT4	ESH2
MSGOR19I	$JOBCTLB	HBB2
MSGOUT	$JOBCTLA	GJB2
MSGOR11I	$JOBCTLB	HCG2
MTC	$JOBCTLJ	PEB1
MTCMTCS	$JOBCTLJ	PEC3
MTCTEST	$JOBCTLJ	PED4
MTCTEST1	$JOBCTLJ	PFB1
MTC2	$JOBCTLJ	PEG2
MTC3	$JOBCTLJ	PEE3
MTC4	$JOBCTLJ	PEG4
MTNCNT	$JOBCTLA	GUB1
MVDU	$JOBCTLA	GRA1
MVN	$JOBCTLD	JAD1
MVN1	$JOBCTLD	JAB2
MVN2	$JOBCTLD	JAC3
MVN3	$JOBCTLD	JAB4
MVQUO	$JOBCTLG	NKC2
MYJOBDTE	$IPLRT4	EAH4
M1S10D	$JOBCTLE	LUC5
M1S10D1	$JOBCTLE	LUB5
NDSCAN	$JOBCTLA	GKG1
NDSCAN	$JOBCTLG	NKE4
NDSCAN	$JOBCTLK	QDJ2
NDSCAN1	$JOBCTLK	QDG3
NDSERR	$JOBCTLJ	PVF2
NDSERR1	$JOBCTLJ	PFC4
NDTERR	$JOBCTLJ	PVC1
NEWXTN	$JOBCTLK	QJF5
NEXT	$$BUFLDR	FHC2
NEXT	$JOBCTLJ	PNC5
NEXTFCB	$$BUFLDR	FKE3
NEXTFCB	SYSBUFF1	FRK4
NEXTKW	$JOBCTLK	QTG4
NEXTP00	$JOBCTLM	RMB1
NEXTP05	$JOBCTLM	RMC2
NEXTP10	$JOBCTLM	RME1
NEXTRK	$JOBCTLK	QOB1
NEXTSYS	$IPLRT4	EDD1
NEXTUP	SYSBUFLD	FPA1
NK	$JOBCTLE	LDB5
NLISTX	$JOBCTLG	NHD1
NLSERR	$JOBCTLK	QSC2
NLUERR	$JOBCTLJ	PVC2
NLUERR	$JOBCTLK	QSC4
NOALIGN	$JOBCTLG	NHJ3
NOASINO	$$A$IPLR	BCB1
NOCAT	$JOBCTLE	LBB2
NOCATL	$JOBCTLG	NEA1
NOCLEAR	$JOBCTLG	NDF1
NODASD	$IPLRT2	CBE2
NODASD	$JOBCTLD	INH1
NODASDIP	SYSBUFLD	FMH1
NODATE	$JOBCTLE	LAG3
NODCUX	$JOBCTLK	QJD4
NODECK	$JOBCTLG	NHB2
NODSYS	$JOBCTLA	GFE3
NODUMP	$JOBCTLG	NKJ2
NODUR1	$JOBCTLG	NEH3
NOEDECK	$JOBCTLG	NHJ5
NOEERT	$JOBCTLA	GVD3
NOEOC	$JOBCTLD	JBF5
NOERRS	$JOBCTLG	NHG2
NOESTV	$JOBCTLJ	PFD2
NOFEAT	$JOBCTLD	KBD5
NOFP	$JOBCTLE	LEK2
NOIDAT	$$A$IPLR	BCE4
NOINCL	$JOBCTLA	GCE2
NOINTENT	$JOBCTLD	KBJ1
NOIPT	$JOBCTLG	NEC2
NOIPTA	$JOBCTLG	NEB2
NOIPT3	$JOBCTLG	NEA2
NOIPT7	$JOBCTLG	NEE2
NOLEAP	$JOBCTLJ	PGG3
NOLINK	$JOBCTLG	NKF2
NOLIST	$JOBCTLG	NHB4
NOLIST	SYSBUFLD	FMH3
NOLNKDIR	$JOBCTLE	LFD1
NOLOG	$JOBCTLJ	PKH4
NOLOG	SYSBUFF1	FQD1
NOMFCXT	$JOBCTLD	KDF4
NOMFCXT	$JOBCTLD	KED5
NOMODE	$JOBCTLD	IQE1
NOMOVE	$JOBCTLG	NEA4
NOMRJB	$JOBCTLD	KAE5
NONFPR	$JOBCTLD	IAC3
NOPAUSE	$JOBCTLA	GBE1
NOPHSE	SYSBUFF1	FQC2
NOPHYS	SYSBUFLD	FNE3
NOPL	$$A$PLBF	ADF5
NOPL	$$A$PLBK	ACF5
NOPRT	$JOBCTLJ	PJB4
NOPSE	$JOBCTLG	NED5

Label	Phase	Location
NOPUB	$JOBCTLG	NNG1
NOPUB1	$JOBCTLG	NND4
NOPUB2	$JOBCTLG	NNG4
NORESET	$JOBCTLJ	PFG3
NOREW	$JOBCTLD	INJ2
NORLD	$JOBCTLG	NHK2
NOSET	$JOBCTLE	LAK2
NOSLD	$JOBCTLD	JCH1
NOSOAD	$IPLRT3	DEJ5
NOSODL1	$IPLRT3	DDF4
NOSODL2	$IPLRT3	DDA1
NOSVAIPL	$JOBCTLA	GAA4
NOSYM	$JOBCTLG	NHD3
NOSYS	SYSBUFF1	FQG3
NOTAP	$$BLSTIO	TAK3
NOTAPE	$JOBCTLD	IMH5
NOTASS	$JOBCTLA	GNF2
NOTDEFIN	$JOBCTLD	IMA3
NOTDISKT	$IPLRT2	CAK1
NOTDSK	$JOBCTLA	GTA1
NOTDU	$JOBCTLD	JEB1
NOTERR	$JOBCTLG	NMG3
NOTFBA	$JOBCTLD	JBC2
NOTICA1	$IPLRT2	CBC1
NOTLAST	$JOBCTLN	SBH4
NOTLIST	$JOBCTLG	NFE2
NOTNOS	$JOBCTLG	NGG1
NOTPRNT	SYSBUFLD	FMA2
NOTPTST	$JOBCTLD	KDB3
NOTRDY	$JOBCTLD	INC4
NOTRECPT	$JOBCTLD	KED3
NOTRPS	$JOBCTLD	JBF2
NOTSEQ	$JOBCTLK	QHG2
NOTSTRES	$JOBCTLD	KEA2
NOTTCR	$JOBCTLD	IKE3
NOTUNIQ	$IPLRT4	EMD2
NOTVIRT	$JOBCTLE	LEK1
NOTYPE	SYSBUFF1	FQG2
NOTYP2	SYSBUFLD	FNE1
NOT370B	$IPLRT4	EBH2
NOUAIGN	$JOBCTLD	IFD1
NOWVIRT1	$IPLRT7	EVE3
NOXREF	$JOBCTLG	NHD5
NO1052T	$JOBCTLD	KEA5
NO3330	$JOBCTLD	KEF1
NO3340	$JOBCTLD	KED1
NO81STD	$JOBCTLD	JME4
NPT01	$JOBCTLA	GRF3
NPWR	$JOBCTLA	GSD1
NSTADSK	$JOBCTLD	KEA1
NTINJB	$JOBCTLA	GVD2
NULCHK	$JOBCTLJ	PBJ3
NUMCON	$JOBCTLD	JHB4
NUMCON	$JOBCTLJ	PUB4
NUMCON	$JOBCTLK	QRB1
NUMCVT	$IPLRT3	DBG2
NVOERR	$JOBCTLG	NQG4
NVSERR	$JOBCTLA	GWC2
NVSERR1	$JOBCTLJ	PLB2
NVSERR2	$JOBCTLJ	PLA2
NVSLST	$JOBCTLJ	PLC3
NVS1	$JOBCTLJ	PLE2
NXTBIT	$JOBCTLJ	PKH1
NXTDPD	$JOBCTLG	NQC5
NXTDRCYL	IPLBMAC	AFC1
NXTENTRX	$JOBCTLJ	PPB2
NXTENTRY	$JOBCTLJ	PPA2
NXTJIB	$JOBCTLA	GFG3
NXTLUB	$JOBCTLA	GFF3
NXTNUNIQ	$IPLRT4	EMG2
NXTOPRD	$JOBCTLJ	PNE1
NXTPARM	$JOBCTLK	QWB1
NXTPARTN	$IPLRT3	DJJ3
NXTPUB	$JOBCTLD	KCG5
NXTTYPE	$JOBCTLD	KDC1
OERRTN	$JOBCTLA	GVF3
OKSUP	$$A$IPLR	BCG1
OPENERR	$JOBCTLM	RAC5
OPENEX	$JOBCTLD	JFH1
OPENFILE	$JOBCTLC	HSB2
OPENFILE	$JOBCTLM	RHB1
OPERSCN	$JOBCTLD	IBB1
OPH1	$JOBCTLD	ICH4
OPH2	$JOBCTLD	IDB1
OPIGN	$JOBCTLD	ICB2
OPLOG	$JOBCTLJ	PKC4
OPMODE	$JOBCTLD	IDB3
OPMODE1	$JOBCTLD	IDB5
OPMODE2	$JOBCTLD	IDC3
OPNEND	$IPLRT4	EDH1
OPNHCERR	$JOBCTLC	HME4
OPNLNK	$JOBCTLG	NLB5
OPNLOP	$IPLRT4	EDC2
OPNRTN	$IPLRT4	EDC1
OPNUSN	$IPLRT4	EDG1
OPPERM	$JOBCTLD	IEA4
OPRSP	$IPLRT5	FCE1
OPRTN	$IPLRT2	CEB1
OPRTN30	$IPLRT2	CED1
OPSHR	$JOBCTLD	IEA1
OPSYS	$JOBCTLD	IEB5
OPTEMP	$JOBCTLD	IEA3
OPTION	$JOBCTLG	NGB1
OPTLOG	$JOBCTLG	NKH4
OPTNLG	$JOBCTLG	NHG5
OPTNOONL	$JOBCTLG	NIB4
OPTONL	$JOBCTLG	NIB3
OPUA	$JOBCTLD	ICB1
OPVOL	$JOBCTLD	IEB4
OPXLIST	$JOBCTLD	IDB2
OPX2	$JOBCTLD	IBB5
OPX3	$JOBCTLD	IBJ5
OTSERR	$JOBCTLG	NMK4
OTSERR	$JOBCTLJ	PVG3
OTSERR	$JOBCTLK	QSC1
OTSERR1	$JOBCTLJ	PVF3
OUTLBL	$JOBCTLK	QJF4
OUTMSG	$JOBCTLD	IQA3
OUTP	$IPLRT2	CKB5
OUTPUTS	$JOBCTLF	MQC1
OUTPUT1	$JOBCTLF	MQC2
OUTPUT2	$JOBCTLF	MQE1
OVDERR	$JOBCTLA	GNH4
OVEND	$JOBCTLA	GNH5
OVRLPI	$JOBCTLA	GTA2
OVTOC	$IPLRT5	FBB1
OWNED	$JOBCTLD	IMB3
PAGEKN	$JOBCTLD	KBA4
PARCHK	$JOBCTLE	LDB1
PARSTD	$JOBCTLG	NJB4

Label	Phase	Location
PART	$JOBCTLF	MKH1
PARTDUMP	$JOBCTLG	NKJ1
PASSGN	$JOBCTLE	LSB3
PASSMC	$IPLRT2	CBA5
PASS1	$IPLRT4	EBB1
PASS2	$IPLRT4	EBE2
PASS2EX	$IPLRT4	EBH3
PAUSE	$JOBCTLJ	PKB3
PBFEND	$IPLRT4	EEH2
PBFLOP	$IPLRT4	EEF1
PBFRTN	$IPLRT4	EEE1
PCHOUT	$JOBCTLJ	PNB2
PCILSUP	$JOBCTLG	NJD1
PCILSUPR	$JOBCTLG	NJD2
PCLOSE	$JOBCTLG	NTB4
PEND	$JOBCTLG	NSB1
PENDR1	$JOBCTLG	NDA2
PENDR1	$JOBCTLG	NDJ1
PEND1	$JOBCTLG	NSE1
PERCNT00	$JOBCTLM	RKB1
PERCNT03	$JOBCTLM	RKJ1
PERCNT05	$JOBCTLM	RKK1
PERCNT90	$JOBCTLM	RKB2
PEREXIT	$JOBCTLD	KAK4
PEREXIT	$JOBCTLF	MFH2
PFORMAT	$IPLRT5	FDB4
PFREE	$JOBCTLG	NTC2
PFT10	$JOBCTLA	GFJ1
PJB	$JOBCTLE	LSE3
PLOG	$JOBCTLG	NUB1
PLOG05	$JOBCTLG	NUC1
PLOG07	$JOBCTLG	NUD1
PLOG08	$JOBCTLG	NUH2
PND02	$JOBCTLG	NSF1
PND03	$JOBCTLG	NSH1
PND05	$JOBCTLG	NSG3
PND10	$JOBCTLG	NSB5
PNPERR	$JOBCTLJ	PVB4
POSTAN	$IPLRT6	EUD1
POUT	$JOBCTLG	NUE3
PRCDIB	$JOBCTLA	GRC1
PRCINIT	$JOBCTLA	GGC2
PREP000	$JOBCTLC	HPB1
PREP020	$JOBCTLC	HPD2
PREP030	$JOBCTLC	HPH1
PRGUNT	$JOBCTLJ	PTC5
PRINTBR	SYSBUFF1	FQD2
PRINTMSG	SYSBUFF1	FQC5
PROCDIB	$JOBCTLE	LTB1
PROCOP	$$A$IPLR	BFB4
PROMPT	$JOBCTLJ	PLG5
PROMPTTM	$IPLRT2	CFF5
PROTP	$JOBCTLD	IFF1
PRSYSLOG	$$A$IPLR	BDE5
PRSYSRDR	$$A$IPLR	BDG2
PRT1F	SYSBUFLD	FOF1
PSHRTN	$$BLSTIO	TDB1
PSHRTN1	$$BLSTIO	TDF1
PSUP	$$A$IPLR	BAB4
PSUPERR	$$A$IPLR	BAJ5
PSUPX	$$A$IPLR	BAK5
PTEST	$JOBCTLA	GEB1
PUBDTS	$JOBCTLD	KBB3
PUBEXD	$IPLRT3	DAC2
PUBLOOP	$IPLRT4	EMC1
PUBMKE	$IPLRT3	DBJ2
PUBMK1	$IPLRT3	DBB5
PUBMSK	$JOBCTLF	MRE4
PUBOK	$$A$IPLR	BEE1
PUBSCAN	$IPLRT7	EVA1
PUIF	$$BLSTIO	TAB1
PUIFTP	$$BLSTIO	TAA3
PUIFT3	$$BLSTIO	TAD4
PUIF1	$$BLSTIO	TAA4
PUIF2	$$BLSTIO	TAF2
PUIF3	$$BLSTIO	TAJ2
PUIF4	$$BLSTIO	TAE4
PUIF5	$$BLSTIO	TAG4
QEND	$JOBCTLG	NKE3
RANXJB	$JOBCTLG	NPB5
RASCAN	$JOBCTLG	NPB4
RASSGN	$JOBCTLG	NPE5
RASWAIT	$IPLRT2	CBG4
RCMOVE	$JOBCTLD	IRF5
RCMOVE	$JOBCTLD	KFC4
RDDISK00	$JOBCTLM	RTB1
RDERR	$JOBCTLA	GSH4
RDOVER	$JOBCTLA	GNB4
RDOVERI	$JOBCTLA	GND4
RDOV06	$JOBCTLA	GNG1
RDOV02	$JOBCTLA	GND1
RDOV04	$JOBCTLA	GNE1
RDOV08	$JOBCTLA	GNA3
RDOV10	$JOBCTLA	GNE2
RDRESP	SYSBUFF1	FQC1
RDRIN	$JOBCTLA	GMA3
RDRIN3	$JOBCTLA	GMD3
RDRIN5	$JOBCTLA	GME4
RDSTMT	$JOBCTLA	GMB1
RDVOL2	$JOBCTLD	IQC4
RDVOL2A	$JOBCTLD	IRG3
READ	$IPLRT5	FBB3
READGO	$IPLRT2	CBE1
READIN	SYSBUFLD	FMD2
READOV	$JOBCTLA	GNB1
READRT	$IPLRT2	CDB2
READRT3	$IPLRT2	CDD2
REALALOC	$IPLRT7	EVB5
RELO	$$BUFLDR	FHB1
REQASI	$$A$IPLR	BLB4
REQLATIN	$$A$IPLR	BKC2
REQLERR1	$$A$IPLR	BKF4
REQLERR2	$$A$IPLR	BKH5
REQLINIT	$$A$IPLR	BKA3
REQLOG	$$A$IPLR	BKB1
REQLOGL	$$A$IPLR	BKH2
REQLOGX	$$A$IPLR	BKK1
REQLREST	$$A$IPLR	BKA2
REQLWAIT	$$A$IPLR	BKB2
REQL00	$$A$IPLR	BKD4
REQL04	$$A$IPLR	BKH4
REQSPOP	$$A$IPLR	BLB3
REQSTERM	$$A$IPLR	BLJ3
REQSUP	$$A$IPLR	BLB1
REQSUPX	$$A$IPLR	BLK4
REQSWAIT	$$A$IPLR	BLB2
REQS00	$$A$IPLR	BLF1
REQS08	$$A$IPLR	BLE5
RESET	$JOBCTLF	MFB1
RESET	$JOBCTLJ	PFF3

Label	Phase	Location
RESET	SYSBUFF1	FRA4
RESETBUF	$JOBCTLG	NQB2
RESETCL	$JOBCTLF	MPJ2
RESETOWN	$JOBCTLD	IMC3
RESET01	$JOBCTLF	MFD2
RESET015	$JOBCTLF	MFF2
RESET1	$JOBCTLF	MFC2
RESET101	$JOBCTLF	MGC1
RESET2	$JOBCTLF	MGE1
RESET3	$JOBCTLF	MGE5
RESET5	$JOBCTLF	MGA2
RESET6	$JOBCTLF	MGF4
RESET7	$JOBCTLF	MGB4
RESET8	$JOBCTLF	MGJ3
RESFCH	$JOBCTLE	LBJ2
RESIDL	$IPLRT2	CHF2
RESIDR	$IPLRT2	CHG4
RESTART	$$A$IPLR	BGB4
RETADD	$JOBCTLF	MAJ1
RETCNCL	$JOBCTLN	SBC3
RETSVA	$JOBCTLJ	PPH5
RETURN	$JOBCTLN	SBB3
REVERSE	SYSBUFF1	FQG5
RFEND	$JOBCTLJ	PJK1
RFIOERR	$JOBCTLM	RAA5
RFYES	$JOBCTLJ	PJG2
RIGHT	$IPLRT2	CHC4
RLD	$JOBCTLG	NHJ1
RLINDT	$JOBCTLA	GMC1
RLSENT	$JOBCTLJ	PAC3
RMSBCRET	IPLBMAC	AEG4
RNGTOP	$JOBCTLD	JHD3
RNGTOP	$JOBCTLK	QRE3
RNVWB	$JOBCTLG	NEH5
RODCMP	$JOBCTLM	RCJ4
RODCONT	$JOBCTLM	RCK4
RODDASD00	$JOBCTLM	RCG2
ROD00	$JOBCTLM	RCB1
ROD05	$JOBCTLM	RCJ1
ROD10	$JOBCTLM	RCA4
ROD15	$JOBCTLM	RCE4
ROD20	$JOBCTLM	RCG4
ROD30	$JOBCTLC	HNB2
ROD30	$JOBCTLM	RCH4
ROD40	$JOBCTLC	HNH2
RSPASG	$JOBCTLG	NME4
RSSASG	$JOBCTLG	NAA4
RSTASG	$JOBCTLG	NMB4
RSTCOM	$JOBCTLG	NBA1
RSTDPD	$JOBCTLG	NQB5
RSTLUB	$IPLRT3	DJJ1
RSTRET	$JOBCTLG	NQD2
RSTRT	$JOBCTLK	QQB1
RSTSTD	$JOBCTLD	JMB1
RSTSTD1	$JOBCTLD	JMF1
RSTSTD4	$JOBCTLD	JMD2
RSTSW4	$JOBCTLG	NCC1
RWFLOOP	$JOBCTLM	RPB3
SAVEBOTH	$JOBCTLD	KBH4
SAVECUU	$JOBCTLD	ISJ2
SAVEONE	$JOBCTLD	KBH5
SCANCDL	$$A$IPLR	BGB1
SCANRES	$$A$PLBK	ACJ2
SCANRL3	$JOBCTLA	GKE3
SCANR1	$JOBCTLA	GKC1
SCANR2	$JOBCTLA	GKB1
SCANR3	$JOBCTLA	GKB3
SCAN00	$JOBCTLM	RMB3
SCHLOP	$IPLRT3	DFC1
SCHSCH	$IPLRT3	DEJ1
SCHSTA	$IPLRT3	DEF1
SCHTST	$IPLRT3	DEH1
SCILDIRF	IPLBMAC	AFH1
SCILDIRN	IPLBMAC	AFJ2
SCILDIRS	IPLBMAC	AFF1
SCILDIRX	IPLBMAC	AFK1
SCNEND	$IPLRT3	DCB4
SCNINT	$JOBCTLG	NMB1
SCNJIB	$JOBCTLD	JSB2
SCNLOP	$IPLRT3	DCD3
SCNLUB	$JOBCTLD	JSB1
SCNRL1	$JOBCTLA	GKD1
SCNRL2	$JOBCTLA	GKD2
SCNRT	$JOBCTLE	LPD2
SCRTCH	$IPLRT5	FBH3
SDLFULL	$JOBCTLJ	PLA5
SDLNEXT	$JOBCTLJ	PPB4
SDLOVRFL	$JOBCTLJ	PLC4
SDL2	$JOBCTLJ	PPF1
SEND	$JOBCTLE	LQG3
SENSE	IPLBMAC	AHC2
SET	$JOBCTLJ	PGB2
SETAPRT	$JOBCTLA	GYB2
SETAPRTA	$JOBCTLA	GYB4
SETAPRTB	$JOBCTLA	GYE4
SETAPRTE	$JOBCTLA	GYH4
SETAPRTF	$JOBCTLA	GYF2
SETAREA	$JOBCTLG	NKH5
SETBTN	$IPLRT4	EAC1
SETCLOCK	$IPLRT4	EAG4
SETDIB1	$JOBCTLD	JFC2
SETEXT	$JOBCTLJ	PGA1
SETFCL	$IPLRT3	DHG4
SETFLAGS	$JOBCTLK	QPF2
SETID	$JOBCTLF	MKD2
SETKEY1	$$A$IPLR	BCB5
SETKEY2	$$A$IPLR	BCC5
SETLAST	$JOBCTLG	NME1
SETLUB	$$BUFLDR	FHC4
SETOK3	$IPLRT2	CFD4
SETOUT	$JOBCTLA	GSD2
SETOUTPT	$JOBCTLJ	PDC2
SETRF000	$JOBCTLM	RTB2
SETSDL	$JOBCTLJ	PPB1
SETSIZE	$JOBCTLD	JEC5
SETSKIP	$JOBCTLA	GTH2
SETSVA	$JOBCTLJ	PLB1
SETUP	$JOBCTLG	NND2
SETUP	$JOBCTLJ	PSB4
SETWRT	$JOBCTLA	GTJ1
SEUOB	$$BLSTIO	TCB4
SEUOB2	$$BLSTIO	TCG4
SFNC	$JOBCTLF	MNB5
SFPPE	$JOBCTLD	JPB1
SFPPE	$JOBCTLF	MRB1
SFPPEDQ2	$JOBCTLF	MJC4
SFPPE01	$JOBCTLF	MRE2
SFPPE02	$JOBCTLD	JPE1
SFPPE02	$JOBCTLF	MRE1
SFPPE03	$JOBCTLD	JRE1
SFPPE03	$JOBCTLF	MRF5

Label	Phase	Location
SFPPE04	$JOBCTLD	JPH3
SFPPE1	$JOBCTLF	MRC1
SFPPE10	$JOBCTLD	JQB1
SFPPE10	$JOBCTLF	MSA1
SFPPE11	$JOBCTLF	MSD3
SFPPE12	$JOBCTLD	JQD2
SFPPE12	$JOBCTLF	MSB2
SFPPE13	$JOBCTLD	JQC4
SFPPE13	$JOBCTLF	MSC4
SFPPE14	$JOBCTLD	JQG2
SFPPE14	$JOBCTLF	MSF2
SFPPE15	$JOBCTLD	JQD4
SFPPE15	$JOBCTLF	MSD4
SFPPE16	$JOBCTLF	MSG4
SFPPE17	$JOBCTLD	JQJ5
SFPPE17	$JOBCTLF	MSJ5
SFPPE18	$JOBCTLF	MSK4
SFPPE19	$JOBCTLF	MSA4
SFPPE2	$JOBCTLD	JPD1
SFPPE2	$JOBCTLF	MRD1
SFPPE201	$JOBCTLF	MSE3
SFPPE3	$JOBCTLD	JRJ3
SFPPE3	$JOBCTLF	MRJ5
SFPPE4	$JOBCTLD	JPG1
SFPPE4	$JOBCTLF	MRH1
SFPPE5	$JOBCTLD	JPB2
SFPPE5	$JOBCTLF	MRC3
SFPPE5A	$JOBCTLF	MRB3
SFPPE6	$JOBCTLF	MRC4
SFPPE7	$JOBCTLF	MRD5
SFPPE8	$JOBCTLD	JQE1
SFPPE8	$JOBCTLF	MSC1
SFPPE8A	$JOBCTLF	MSG1
SFPPE8B	$JOBCTLF	MSB4
SFPPE8C	$JOBCTLF	MSA5
SFPPE9	$JOBCTLD	JQF2
SFPPE9	$JOBCTLF	MSE2
SHARE	$JOBCTLD	IKF5
SHORTFCB	$$BUFLDR	FKG3
SHORTFCB	SYSBUFF1	FRC5
SIMEND	$JOBCTLG	NCA1
SIMEOF	SYSBUFLD	FPD5
SIMEOJ	$JOBCTLN	SAB2
SIMRET	$JOBCTLG	NBB1
SIO	$JOBCTLC	HRB5
SIO	IPLBMAC	AHC1
SIZE	$JOBCTLJ	PSB1
SJBKSWCH	$JOBCTLN	SBH2
SKIP	$JOBCTLJ	PHF5
SKIPDC	$$A$PLBF	ADB1
SKIPDC	$$A$PLBK	ACB1
SKIPLOAD	$$BUFLDR	FKJ3
SKIPMSG	$JOBCTLD	IAE2
SKIPRTY	$JOBCTLF	MKC2
SKPINC	$IPLRT4	EFD3
SKPKEY	$IPLRT3	DAG1
SLDLOOP	$JOBCTLD	JBD4
SLIBAE	$JOBCTLG	NKD5
SLIBDF	$JOBCTLG	NKE5
SNS00	$JOBCTLM	RNB1
SNS001	$JOBCTLM	RNE2
SNS10	$JOBCTLM	RND1
SOLVE	SYSBUFF1	FRK3
SPRSW	$IPLRT3	DEC3
STARTIO	$$BUFLDR	FKE1
STDASSGN	$JOBCTLG	NNG3
STDCPT1	$JOBCTLJ	PXB1
STDLBL	$JOBCTLG	NJB3
STDOPTX	$JOBCTLJ	PXK3
STDOPTY	$JOBCTLJ	PXH1
STDOPTZ	$JOBCTLJ	PXF1
STDUCHN	$JOBCTLA	GPC5
STGETOP	$JOBCTLJ	PXC3
STIDC	$$A$IPLR	BCH3
STIDEL	$$A$IPLR	BCJ3
STLOOP1	$JOBCTLJ	PXA3
STOP	$JOBCTLJ	PMB1
STORERJ	SYSBUFF1	FQH5
STORNAM	$$BUFLDR	FHG3
STORUNIT	$JOBCTLG	NQA2
STSCLB	$JOBCTLG	NPG1
ST370	$$A$IPLR	BCH2
SUBLIB	$JOBCTLG	NKB5
SULB	$$BLSTIO	TCB1
SULB1	$$BLSTIO	TCE2
SULB2	$$BLSTIO	TCE3
SUPCHCK	$$A$IPLR	BBD3
SUPREQ	$$A$IPLR	BBF1
SUPV	$JOBCTLF	MKG1
SVCBTRNS	$JOBCTLF	MAD5
SWBACK	$JOBCTLN	SBD1
SWDEV1	$IPLRT4	EMC3
SWDEV3	$IPLRT4	EMF3
SWITCHING	$IPLRT7	EVG1
SWOFF1	$JOBCTLD	INC5
SWOFF2	$JOBCTLD	IND5
SWVIRT1	$IPLRT7	EVB3
SWVIRT2	$IPLRT7	EVC3
SXTPOK	$JOBCTLK	QHK1
SYM	$JOBCTLG	NHD2
SYNTAX	$JOBCTLE	LQB2
SYN02	$JOBCTLE	LQE2
SYSDATE	$JOBCTLJ	PGH2
SYSERR	$JOBCTLA	GWB5
SYSERR1	$JOBCTLG	NCA2
SYSHC	$JOBCTLJ	PJA3
SYSMVC	$IPLRT2	CHB2
SYSPRM	$JOBCTLG	NKB1
SYSR	$IPLRT5	FFB3
SYSRF	$JOBCTLJ	PJB1
SYSRTURN	SYSBUFLD	FME4
SYSTERR	$JOBCTLD	KFB4
SYSUPI	$JOBCTLJ	PGG1
SYSUPT1	$JOBCTLJ	PKC1
SYSXXX	$JOBCTLK	QTB2
SYSXXX	$JOBCTLK	QTE1
SYSYYY	$JOBCTLD	KFB1
SYSZZZ	$JOBCTLD	JTB1
SYSZZZ01	$JOBCTLD	JTG3
SYSZZZ1	$JOBCTLD	JTD1
SYSZZZ2	$JOBCTLD	JTE1
SYSZZZ3	$JOBCTLD	JTE3
SYSZZZ4	$JOBCTLD	JTA4
SYSZZZ5	$JOBCTLD	JTA5
SYSZZZ6	$JOBCTLD	JTH2
TAPINT	$JOBCTLA	GQB1
TEMPASS1	$JOBCTLD	JEG5
TEMPEND	$JOBCTLE	LHE2
TEST	$JOBCTLN	SAE1
TESTACCT	$JOBCTLA	GAF4
TESTAUTO	$JOBCTLE	LHB1

Label	Phase	Location
TESTCIOA	$JOBCTLD	IPB1
TESTCSW	IPLBMAC	AHH1
TESTDEV	SYSBUFF1	FRJ1
TESTEND	$JOBCTLA	GNE5
TESTIPT	$JOBCTLD	ITF2
TESTNK	$JOBCTLE	LDH2
TESTOV	$JOBCTLA	GNB5
TESTREAL	$JOBCTLE	LUE5
TESTSIZE	$JOBCTLE	LDG1
TEST1	$JOBCTLF	MEG2
TEST2	$JOBCTLF	MEG3
TEST3420	$JOBCTLD	IMD4
TFILL	$JOBCTLK	QAD4
TIAERR	$JOBCTLD	KAE1
TIAERR	$JOBCTLF	MNG4
TIMERT	$IPLRT4	EFB4
TIMOUT	$JOBCTLG	NLB3
TIMOUT1	$JOBCTLG	NLK3
TIMOUT2	$JOBCTLG	NLE4
TIO	IPLBMAC	AHG1
TLBL	$JOBCTLK	QBB1
TLOOP	$JOBCTLE	LDC5
TM1	$IPLRT2	CFD1
TOOMANY	$JOBCTLJ	PNJ1
TPLAB	$JOBCTLK	QAB2
TPVER	$JOBCTLK	QAH3
TRLBL	$JOBCTLD	IQE4
TRTDO	SYSBUFF1	FQE4
TSLNKSW	$JOBCTLJ	PDA2
TSTCLASS	$JOBCTLF	MUC2
TSTDF	$JOBCTLA	GSE4
TSTEXT	$JOBCTLD	JKJ3
TSTLOG	$JOBCTLA	GDF1
TSTMNT	$JOBCTLD	IQJ1
TSTPAUSE	$JOBCTLA	GBD1
TSTPROC	$JOBCTLD	KAG4
TSTPRT1	SYSBUFLD	FNA2
TSTTYPE	$JOBCTLA	GCK4
TTRCONV	IPLBMAC	AFB5
TXCUU	$JOBCTLF	MQB4
TXCUU	$JOBCTLJ	PTC1
TXCUU1	$JOBCTLD	KCC1
TXCUU1	$JOBCTLF	MQE4
TXCUU1	$JOBCTLJ	PTD1
TXCUU1&8	$JOBCTLF	MNC4
TXCUU2	$JOBCTLF	MQH4
TXCUU2	$JOBCTLJ	PTF1
TXCUU3	$JOBCTLF	MQJ3
TXCUU4	$JOBCTLF	MQF4
TYPEXCP	SYSBUFF1	FQB5
TYPRTURN	SYSBUFLD	FMK3
TYP2RTRN	SYSBUFLD	FNC1
T125BG	$IPLRT4	EPC3
T125CH	$IPLRT4	EPD1
T125EN	$IPLRT4	EPH3
T125HW	$IPLRT4	EPJ1
T125NOC	$IPLRT4	EPG3
T125NOR	$IPLRT4	EPB4
T125RT	$IPLRT4	EPA4
T125SCA	$IPLRT4	EPA3
T1442	$JOBCTLJ	PNE2
T9TRK	$JOBCTLD	KCB4
T9T3401	$JOBCTLD	KCC5
UA	$JOECTLF	MEB1
UADN1	$JOBCTLF	MEE1
UADN2	$JOBCTLF	MEJ1
UADN3	$JOBCTLF	MEE5
UADN4	$JOBCTLF	MEF3
UADN5	$JOBCTLF	MEA4
UADN6	$JOBCTLF	MEH3
UALUB	$JOBCTLF	MJC3
UALUB1	$JOBCTLF	MJD3
UANXJB	$JOBCTLG	NNC5
UASCAN	$JOBCTLG	NNB1
UASTD	$JOBCTLF	MJB5
UCBLOAD	SYSBUFLD	FOB1
UCS	$JOBCTLJ	PBB1
UCSSCN	$JOBCTLJ	PCB2
UCSVN	$JOBCTLJ	PCC1
UCS1	$JOBCTLJ	PBH1
UCS3	$JOBCTLJ	PCE3
UCS4	$JOBCTLJ	PCE4
UNA	$JOBCTLF	MNB1
UNAE	$JOBCTLF	MNF1
UNANENT	$JOBCTLF	MRG5
UNASGN	$JOBCTLG	NNA4
UNASSGN	$JOBCTLD	JFJ3
UNA1	$JOBCTLF	MNB3
UNA1A	$JOBCTLF	MNE3
UNA2	$JOBCTLF	MPA1
UNA2A	$JOBCTLF	MPE1
UNA2AA	$JOBCTLF	MPF1
UNA2B	$JOBCTLF	MPG1
UNBATCH	$JOBCTLF	MMB1
UNBLKD1	$JOBCTLJ	PDC5
UNCLOG	$JOBCTLA	GDF2
UNCU	$JOBCTLF	MCA3
UNITCHCK	$$A$IPLR	BFB3
UNITPUB	SYSBUFLD	FMJ5
UNITR	$IPLRT5	FEB3
UNITS	$JOBCTLF	MCB2
UNITS1	$JOBCTLF	MCA1
UNITS2	$JOBCTLF	MCH1
UNITS3	$JOBCTLF	MCH2
UNITS401	$JOBCTLF	MCE1
UNITS402	$JOBCTLF	MCB3
UNITS5	$JOBCTLF	MCC3
UNITS501	$JOBCTLF	MCC4
UNITS6	$JOBCTLF	MCA5
UNITS8	$JOBCTLF	MCH4
UNITS9	$JOBCTLF	MCD3
UNPA	$JOBCTLD	JRB1
UNPA3	$JOBCTLD	JRB2
UNTFND	$JOBCTLJ	PTF4
UPDATE	$JOBCTLC	HRB3
UPDBLK	$JOBCTLC	HRD3
UPDCHBKT	$$A$IPLR	BEH1
UPDHDN	$JOBCTLA	GRD5
UPDSAV	$JOBCTLJ	PUB1
UPLUB	$IPLRT3	DJH1
UPSICH	$JOBCTLJ	PKE1
UPSIPROC	$JOBCTLJ	PKB1
USERENTZ	$JOBCTLA	GEF3
USERETRN	$JOBCTLA	GEJ3
USEREXIT	$JOBCTLA	GEB3
USERNONE	$JOBCTLA	GEJ5
USERNOXT	$JOBCTLA	GEE5
USEROUT	$JOBCTLA	GEH5
USERPTST	$JOBCTLA	GED4
USERROR	$JOBCTLN	SAF1
USERSEIZ	$JOBCTLA	GEG5

Label	Phase	Location
USERSKIP	$JOBCTLA	GEF5
USNRES	$IPLRT4	EAJ4
USRLBL	$JOBCTLG	NJB5
VMBUCK	$JOBCTLJ	PNB5
VMEND	$JOBCTLJ	PNF5
VMRTN	$JOBCTLJ	PNB3
VNFERR	$JOBCTLD	KAE4
VOL	$JOBCTLK	QAB5
VOLIDR	$IPLRT5	FEB5
VOLOK	$JOBCTLD	IQJ4
WAIT	$$A$IPLR	BDD3
WAITERR	$JOBCTLA	GVH5
WAITHARD	$$A$IPLR	BED5
WAITLOG	$$A$IPLR	BMB4
WAITLOGW	$$A$IPLR	BMD4
WAITLOGX	$$A$IPLR	BMJ5
WAITR	$JOBCTLJ	PMA5
WRITE	$IPLRT5	FCA1
WRITLOG	$JOBCTLE	LED4
WRNGPART	$JOBCTLJ	PXB2
WRONG	$JOBCTLD	IQH2
WRONGVOL	$JOBCTLD	IQJ2
WRTDSK00	$JOBCTLM	RHB2
WRTERR	$JOBCTLA	GSG5
WRTFBA00	$JOBCTLM	RTB3
WRTFBA10	$JOBCTLM	RTD3
WRTFIL00	$JOBCTLM	RLA4
WRTFIL10	$JOBCTLM	RLE4
WRTFIL20	$JOBCTLM	RLK4
WRTHDR00	$JOBCTLM	RHB3
WTORLOG	$$A$IPLR	BMA1
WTORLOGC	$$A$IPLR	BMC1
WTORLOGU	$$A$IPLR	BMC2
WTORLOGX	$$A$IPLR	BMK1
WTORLOG3	$$A$IPLR	BMD2
XCTL	$$A$IPLR	BED2
XCTL2	$JOBCTLE	LEG5
XFERINS	$$A$IPLR	BEE2
XREF	$JOBCTLG	NHD4
XTENT	$JOBCTLK	QHB1
XTOP12	$JOBCTLK	QMB2
XTOP12A	$JOBCTLK	QMD2
XTOP12B	$JOBCTLK	QMF2
XTOP3	$JOBCTLK	QHC5
XTOP34	$JOBCTLK	QMB3
XTOP5	$JOBCTLK	QJA1
XTOUT	$JOBCTLK	QJG2
XTUNIT	$JOBCTLK	QJG3
X203	SYSBUFLD	FOD2
YEOLST	$JOBCTLD	KBK3
ZEROSLD	$JOBCTLD	JBJ5
ZONE	$JOBCTLJ	PHB4
ZONECHK	$IPLRT4	EAJ1
ZRMVDN	$JOBCTLA	GJE4
ZRMVLP	$JOBCTLA	GJD4

APPENDIX B: PHASE TO MODULE CROSS REFERENCE

IPL:

Phase:	Module:
$$A$IPLR	$$A$IPLR
0	0
1	1
$$A$PLBF	$$A$PLBF
K	K
$$BATTF0	$$BATTF0
2	2
3	3
$$BFCB	$$BFCB
B00	B00
10	10
3	3
5	5
$$BUCB	$$BUCB
$$BUCB00	$$BUCB00
$$BUCB3	$$BUCB3
$$BUCB4	$$BUCB4
$$BUCB5	$$BUCB5
$$BUFLDR	$$BUFLDR
LD1	$$BUFLDR
2	$$BUFLDR
$IPLRT2	IJBIPL
3	IJBIPL
4	IJBIPL
5	IJBIPL
6	IJBIPL
7	IJBIPL
$SVABAM	$SVABAM
$SVACSC	$SVACSC
$$A$SVA	$$A$SVA
$SVADLI	
$SVASORT	
$SVAVSAM	
$SVA0000	
$SVA3800	
$SVACICS	
$SVASEC	

These phases are dummy phases to be replaced later by real phases of the same name, which contain their own modules. These do not belong to the IPL component.

JCL:

Phase	Module
SYSBUFF1	IJBSBUF1
SYSBUFLD	IJBSBUFF
$$BJCOPT	$$BJCOPT
JCOP1	JCOP1
LSTIO	LSTIO
$IJBSLA	IJBSLA
$JOBACCT	$JOBACCT
$JOBCTLA	IJBJC1
$JOBCTLB	IJBJCB
$JOBCTLC	IJBJCC
$JOBCTLD	IJBJC2
$JOBCTLE	IJBJC9
$JOBCTLF	IJBJC5
$JOBCTLG	IJBJC3
$JOBCTLJ	IJBJC4
$JOBCTLK	IJBJC6
$JOBCTLM	IJBJC7
$JOBCTLN	IJBJC8
$JOBEXIT	$JOBEXIT

APPENDIX C: ERROR MESSAGES CROSS REFERENCE

Message	Phase
0I00	$$A$IPLR
01	$$A$IPLR
02	$$A$IPLR
03	$$A$IPLR
04	$$A$IPLR
05	$$A$IPLR
06	$$A$PLBK
	$$A$PLBF
07	$$A$IPLR
08	$$A$IPLR
09	$$A$IPLR
10	$IPLRT2
11	$IPLRT2
12	$IPLRT3
13	$IPLRT3
15	$IPLRT3
16	$IPLRT4
17	$IPLRT4
18	$IPLRT2
19	$IPLRT4
20	$IPLRT7
21	$IPLRT2
22	$IPLRT3
23	$IPLRT4
24	$$A$IPLR
25	$IPLRT4
26	$$BUFLDR
	$$BUFLD2
27	$$BUFLD2
28	$$BUFLDR
29	$IPLRT4
30	$IPLRT2
31	$IPLRT2
32	$IPLRT2
34	$IPLRT4
35	$IPLRT4
36	$IPLRT2
37	$IPLRT5
38	$IPLRT5
39	$IPLRT5
40	$IPLRT4
41	$IPLRT5
42	$IPLRT5
43	$IPLRT5
44	$IPLRT5
45	$IPLRT5
46	$IPLRT5
50	$IPLRT2
51	$IPLRT5
52	$IPLRT5
54	$IPLRT2
58	$IPLRT4
59	$IPLRT7
60	$IPLRT3
61	$IPLRT4
63	$IPLRT4
64	$IPLRT2
65	$IPLRT2
66	$IPLRT2
67	$IPLRT4
68	$$A$IPLR
69	$$BUFLDR

Message	Phase
0I70	$$BUFLDR
71	$$BUFLDR
72	$IPLRT3
73	$IPLRT5
74	$IPLRT7
75	$IPLRT5
76	$IPLRT5
77	$IPLRT5
78	$IPLRT5
79	$IPLRT5
81	$IPLRT7
82	$IPLRT7
83	$IPLRT6
84	$IPLRT6
85	$IPLRT6
86	$IPLRT2
	$IPLRT4
	$IPLRT5
	$IPLRT6
87	$IPLRT2
	$IPLRT4
	$IPLRT5
	$IPLRT6
88	$IPLRT2
	$IPLRT5
	$IPLRT6
89	$IPLRT2
	$IPLRT4
	$IPLRT5
90	$IPLRT6
91	$IPLRT6
92	$IPLRT7
94	$IPLRT2
95	$IPLRT6
96	$$A$IPLR
99	$IPLRT5
0J18	$IPLRT7
0R00I	$JOBCTLB
01I	$JOBCTLB
02I	$JOBCTLB
03I	$JOBCTLB
04I	$JOBCTLB
05I	$JOBCTLB
06I	$JOBCTLB
07I	$JOBCTLB
08I	$JOBCTLB
09I	$JOBCTLB
10I	$JOBCTLB
11I	$JOBCTLB
12I	$JOBCTLB
13I	$JOBCTLB
14I	$JOBCTLB
15I	$JOBCTLB
19I	$JOBCTLB
1A0n	$JOBCTLD
1n	$JOBCTLD
2n	$JOBCTLD
	$JOBCTLG
	$JOBCTLJ

Message	Phase
1A3n	$JOBCTLD
4n	$JOBCTLD
	$JOBCTLF
	$JOBCTLJ
	$JOBCTLK
5n	$JOBCTLD
	$JOBCTLF
	$JOBCTLJ
6n	$JOBCTLD
	$JOBCTLF
7n	$JOBCTLD
	$JOBCTLF
	$JOBCTLJ
80	$JOBCTLD
81	$JOBCTLA
82	$JOBCTLD
83	$JOBCTLD
9n	$JOBCTLD
	$JOBCTLF
1B01	SYSBUFLD
08	$JOBCTLJ
10	SYSBUFLD
11	SYSBUFLD
1C00	$JOBCTLA
10	$JOBCTLJ
	$JOBCTLA
	$JOBCTLG
3n	$JOBCTLE
	$JOBCTLJ
70	$JOBCTLA
8n	$JOBCTLA
	$JOBCTLG
1I00	$JOBCTLA
	$JOBCTLG
10	$JOBCTLF
20	$JOBCTLG
50	$JOBCTLA
70	$JOBCTLA
74	$JOBCTLM
75	$JOBCTLM
76	$JOBCTLM
77	$JOBCTLM
78	$JOBCTLM
79	$JOBCTLM
80	$JOBCTLG
82	$JOBCTLM
83	$JOBCTLM
84	$JOBCTLM
85	$JOBCTLM
86	$JOBCTLM
89	$JOBCTLM
90	$JOBCTLM
91	$JOBCTLM
92	$JOBCTLM
93	$JOBCTLM
95	$JOBCTLM
	$JOBCTLC
96	$JOBCTLM
	$JOBCTLC
97	$JOBCTLM
	$JOBCTLC

Message	Phase
1I98	$JOBCTLJ
99	$JOBCTLM
	$JOBCTLC
1L0n	$JOBCTLK
1n	$JOBCTLK
2n	$JOBCTLK
40	$IJBSLA
1M10	$JOBCTLA
	$JOBCTLG
20	$JOBCTLA
21	$JOBCTLG
3n	$JOBCTLA
4n	$JOBCTLG
5n	$JOBCTLD
	$JOBCTLF
6n	$JOBCTLE
7n	$JOBCTLE
	$JOBCTLG
8n	$JOBCTLE
	$JOBCTLG
9n	$JOBCTLE
	$JOBCTLG
1N00	$JOBCTLA
10	$JOBCTLE
2n	$JOBCTLE
30	$JOBCTLE
40	$JOBCTLE
50	$JOBCTLE
60	$JOBCTLE
7n	$JOBCTLG
80	$JOBCTLG
90	$JOBCTLG
91	$JOBCTLA
92	$JOBCTLE
1P01	$JOBCTLJ
02	$JOBCTLJ
	$JOBCTLE
03	$JOBCTLJ
04	$JOBCTLJ
05	$JOBCTLJ
2n	$JOBCTLE
1S0n	$JOBCTLA
	$JOBCTLG
	$JOBCTLJ
1n	$JOBCTLG
	$JOBCTLE
	$JOBCTLJ
	$JOBCTLK
2n	$JOBCTLJ
3n	$JOBCTLE
40	$JOBCTLA
	$JOBCTLD
	$JOBCTLE
41	$JOBCTLJ
42	$JOBCTLJ
50	$JOBCTLJ
51	$JOBCTLJ
52	$JOBCTLJ

Message	Phase
1S6n	$JOBCTLE
1T10	$JOBCTLA
	$JOBCTLJ
20	$JOBCTLD
40	$JOBCTLD
1T50	$JOBCTLD
60	$JOBCTLD
70	$JOBCTLD
80	$JOBCTLA
3M91	$JOBCTLD

APPENDIX D: SAMPLE LISTIO PRINTOUTS

1

```
// LISTIO SYS

              *** BACKGROUND ***

I/O UNIT   CMNT    CHNL    UNIT    MODE

 SYSRDR              0       0C
 SYSIPT              0       0C
 SYSPCH              0       0D
 SYSLST              1       0A
 SYSLOG              0       1F
 SYSLNK              1       91
 SYSRES              1       92
 SYSSLB             ** UA **
 SYSRLB             ** UA **
 SYSREC              1       91
```

2

```
// LISTIO PROG

              *** BACKGROUND ***

I/O UNIT   CMNT    CHNL    UNIT    MODE

 SYS000              0       91
 SYS001              0       91
 SYS002              0       91
 SYS003              0       91
 SYS004             ** UA **
 SYS005             ** UA **
```

3

```
// LISTIO F2

              *** FOREGROUND 2 ***

I/O UNIT   CMNT    CHNL    UNIT    MODE

 SYSRDR             ** UA **
 SYSIPT             ** UA **
 SYSPCH             ** UA **
 SYSLST             ** UA **
 SYSLOG             ** UA **
 SYSLNK             ** UA **
 SYSRES              1       92
 SYSSLB             ** UA **
 SYSRLB             ** UA **
 SYSREC              1       91

              *** FOREGROUND 2 ***

I/O UNIT   CMNT    CHNL    UNIT    MODE

 SYS000             ** UA **
 SYS001             ** UA **
 SYS002             ** UA **
 SYS003             ** UA **
 SYS004             ** UA **
 SYS005             ** UA **
 SYS006             ** UA **
 SYS007             ** UA **
 SYS008             ** UA **
 SYS009             ** UA **
 SYS010             ** UA **
 SYS011             ** UA **
 SYS012             ** UA **
 SYS013             ** UA **
 SYS014             ** UA **
 SYS015             ** UA **
```

Note: The 1st line of each sample shows the control statement as it was logged by job control.

1. List all background system units.
2. List all background programmer units.
3. List all foreground 2 units.
4. List all units.
5. List a specific unit (SYSXXX).
6. List the logical units assigned to all physical devices.
7. List all unassigned units.
8. List all down units.
9. List all logical units assigned to a specified physical unit.

4

```
// LISTIO ALL

              *** BACKGROUND ***

I/O UNIT   CMNT    CHNL    UNIT    MODE

 SYSRDR              0       0C
 SYSIPT              0       0C
 SYSPCH              0       0D
 SYSLST              1       0A
 SYSLOG              0       1F
 SYSLNK              1       91
 SYSRES              1       92
 SYSSLB             ** UA **
 SYSRLB             ** UA **
 SYSREC              1       91

              *** BACKGROUND ***

I/O UNIT   CMNT    CHNL    UNIT    MODE

 SYS000             ** UA **
 SYS001              1       91
 SYS002              1       91
 SYS003              1       91
 SYS004             ** UA **
 SYS005             ** UA **
 SYS006             ** UA **
 SYS007             ** UA **
 SYS008             ** UA **
 SYS009             ** UA **
 SYS010             ** UA **
 SYS011             ** UA **
 SYS012             ** UA **
 SYS013             ** UA **
 SYS014             ** UA **
 SYS015             ** UA **

              *** FOREGROUND 2 ***

I/O UNIT   CMNT    CHNL    UNIT    MODE

 SYSRDR             ** UA **
 SYSIPT             ** UA **
 SYSPCH             ** UA **
 SYSLST             ** UA **
 SYSLOG             ** UA **
 SYSLNK             ** UA **
 SYSRES              1       92
 SYSSLB             ** UA **
 SYSRLB             ** UA **
 SYSREC              1       91

              *** FOREGROUND 2 ***

I/O UNIT   CMNT    CHNL    UNIT    MODE

 SYS000             ** UA **
 SYS001             ** UA **
 SYS002             ** UA **
 SYS003             ** UA **
 SYS004             ** UA **
 SYS005             ** UA **
 SYS006             ** UA **
 SYS007             ** UA **
 SYS008             ** UA **
 SYS009             ** UA **
 SYS010             ** UA **
 SYS011             ** UA **
 SYS012             ** UA **
 SYS013             ** UA **
 SYS014             ** UA **
 SYS015             ** UA **

              *** FOREGROUND 1 ***

I/O UNIT   CMNT    CHNL    UNIT    MODE

 SYSRDR             ** UA **
 SYSIPT             ** UA **
 SYSPCH             ** UA **
 SYSLST             ** UA **
 SYSLOG             ** UA **
 SYSLNK             ** UA **
 SYSRES              1       92
 SYSSLB             ** UA **
 SYSRLB             ** UA **
 SYSREC              1       91
```

```
              *** FOREGROUND 1 ***

I/O UNIT   CMNT    CHNL    UNIT    MODE

 SYS000             ** UA **
 SYS001             ** UA **
 SYS002             ** UA **
 SYS003             ** UA **
 SYS004             ** UA **
 SYS005             ** UA **
 SYS006             ** UA **
 SYS007             ** UA **
 SYS008             ** UA **
 SYS009             ** UA **
 SYS010             ** UA **
 SYS011             ** UA **
 SYS012             ** UA **
 SYS013             ** UA **
 SYS014             ** UA **
 SYS015             ** UA **
```

5

```
// LISTIO SYSRDR

              *** BACKGROUND ***

I/O UNIT   CMNT    CHNL    UNIT    MODE

 SYSRDR              0       0C
```

6

```
// LISTIO UNITS

CHNL    UNIT     OWNER    I/O UNIT    CMNT    MOD

  0      0C       BG       SYSRDR
  0      0C       BG       SYSIPT
  0      0D       BG       SYSPCH
  0      0E                * UA *
  0      1F       BG       SYSLOG
  0      1F       BG       SYSIN
  1      0A       BG       SYSLST
  1      90                * UA *
  1      91       BG       SYSLNK
  1      91       BG       SYSREC
  1      91       BG       SYS001
  1      91       BG       SYS002
  1      91       BG       SYS003
  1      91       F2       SYSREC
  1      91       F1       SYSREC
  1      92       BG       SYSRES
  1      92       F2       SYSRES
  1      92       F1       SYSRES
  1      80                * UA *
  1      81                * UA *
  1      82                * UA *
  1      83                * UA *
  1      84                * UA *
```

7

```
// LISTIO UA

        *** UNASSIGNED ***

          CHNL    UNIT

            1      80
            1      81
            1      82
            1      83
```

8

```
// LISTIO DOWN

              *** DOWN ***

              CHNL    UNIT

               ** NONE **
```

9

```
// LISTIO X'01F'

CHNL    UNIT     OWNER    I/O UNIT    CMNT    MODE

  0      1F       BG       SYSLOG
```

Figure 17. Sample LISTIO Printouts

APPENDIX E: VARIOUS CONTROL BLOCKS AND TABLES

1. THE PUB TABLE

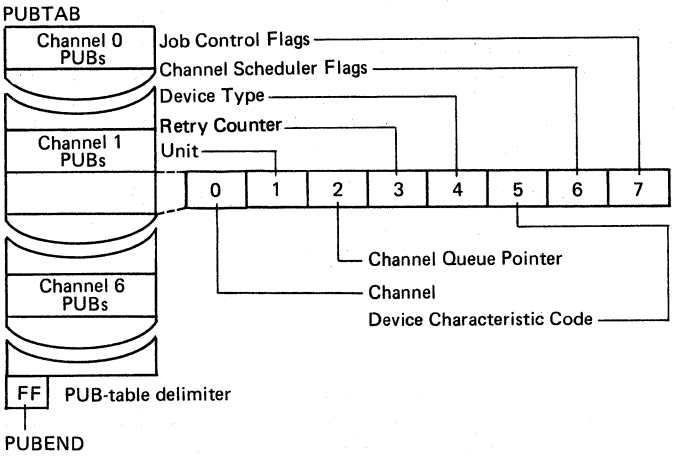


Figure 18. Physical Unit Block (PUB) Table

Bytes:

0 Channel number. (Hex 0-9, FF=NULL)

1 I/O device unit number.

2 Hex 0, 1, 2, ... first channel queue entry number for this device.

3 ERP retry counter.

4 Device type code.

5 For ICA line (Model 115, or 125), this byte contains the displacement index of the entry in the Line Mode Table (LMT). The address of the LMT is in SYSCOM.

For DASD with Track Hold this byte contains a printer to the Track Hold table entry or X'FF'.

For MICR type devices, this byte indicates which external interrupt line is in use.

For a 3704/3705 Communications Controller, this byte contains the type number of the Channel Adapter.

For MFCM or MFCU:

Bit
0: 1 = Repositioning required

1: 0 = SYSPCH temporarily assigned to hopper 1
1 = SYSPCH temporarily assigned to hopper 2

2: 0 = SYSIPT temporarily assigned to hopper 1
1 = SYSIPT temporarily assigned to hopper 2

3: 0 = SYSRDR temporarily assigned to hopper 1
1 = SYSRDR temporarily assigned to hopper 2

5: 0 = SYSPCH permanently assigned to hopper 1
1 = SYSPCH permanently assigned to hopper 2

6: 0 = SYSIPT permanently assigned to hopper 1
1 = SYSIPT permanently assigned to hopper 2

7: 0 = SYSRDR permanently assigned to hopper 1
1 = SYSRDR permanently assigned to hopper 2

6 Channel Scheduler Flags

Bit 0: 1 = Device busy
1: 1 = Switchable device
2: 1 = EOJ for SYSRDR or SYSIPT
3: 1 = I/O error queued for recovery
4: 1 = Operator intervention required
5: 1 = I/O interrupt outstanding
6: 1 = Burst or overrunable device
7: 1 = 7-track tape unit

7 Job Control Flags

Bit 0-4: Standard MODE assignment for 7- and 9-track tape (all ones if not tape, all zeros if device is down)
5: 1 = Device supports RPS
6: Reserved
7: Unit check pending on alternate channel

Notes:

- A null entry is generated at supervisor generation time for each device to be supported by the supervisor. Then standard physical unit assignments are made to the PUB table. Physical unit assignments can also be made during IPL. PUBs are ordered by channel and priority within a channel. A PUB entry must be generated or added during IPL for any device of the installation.
- An entry in the PUB Ownership Table is associated with each entry in the PUB table, if the supervisor has been generated to support multiprogramming. For the PUBOWNER table see DOS/VS Supervisor Logic.
- Bytes 64-65 (X'40' - X'41') of the partition communication region contain the address of the PUB table entry. Label PUBTAB identifies the first byte of the table.
- Bytes 76-77 (X'4C' - X'4D') of the partition communication region contain the address of the LUB table. Label LUBTAB identifies the first byte of the table.
- SYSUSE may be called SYSCTL in error recovery messages.

2. THE NICL, FICL, AND LUB TABLES

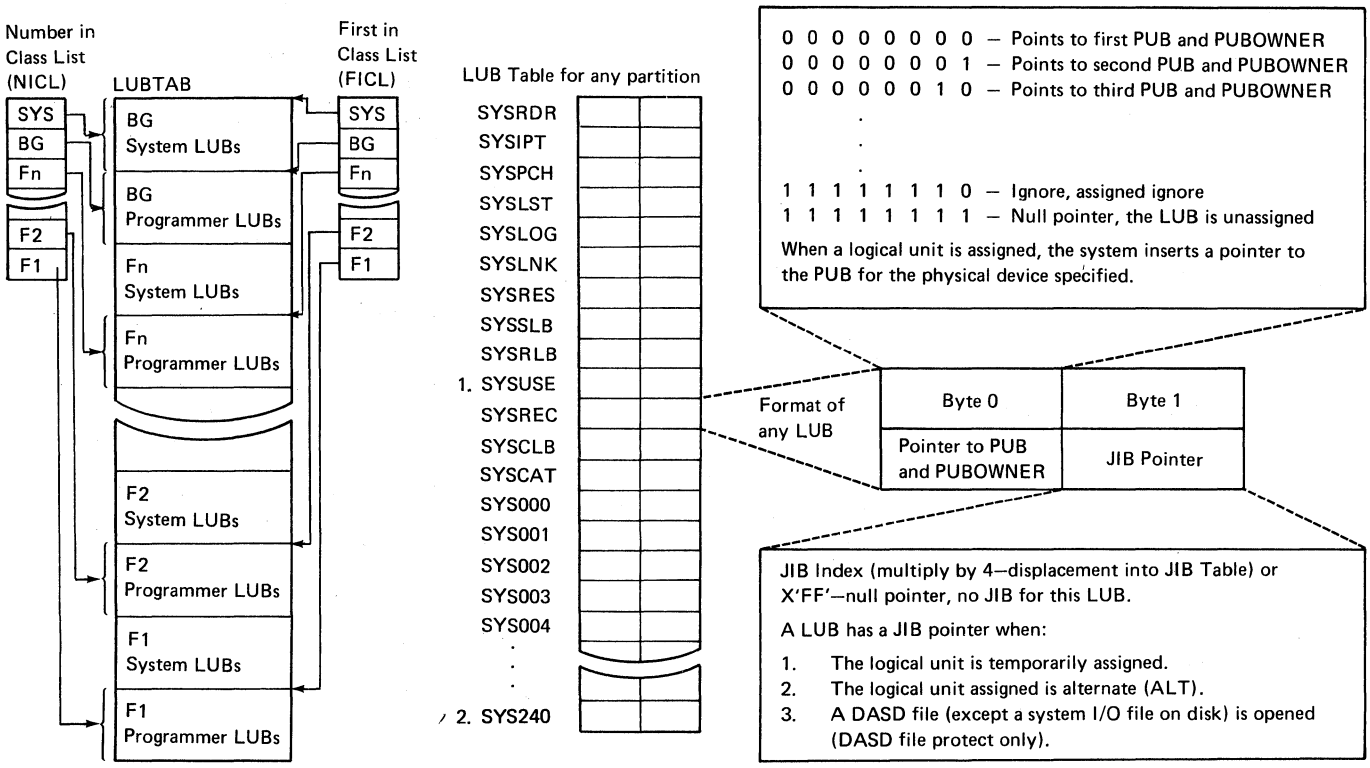
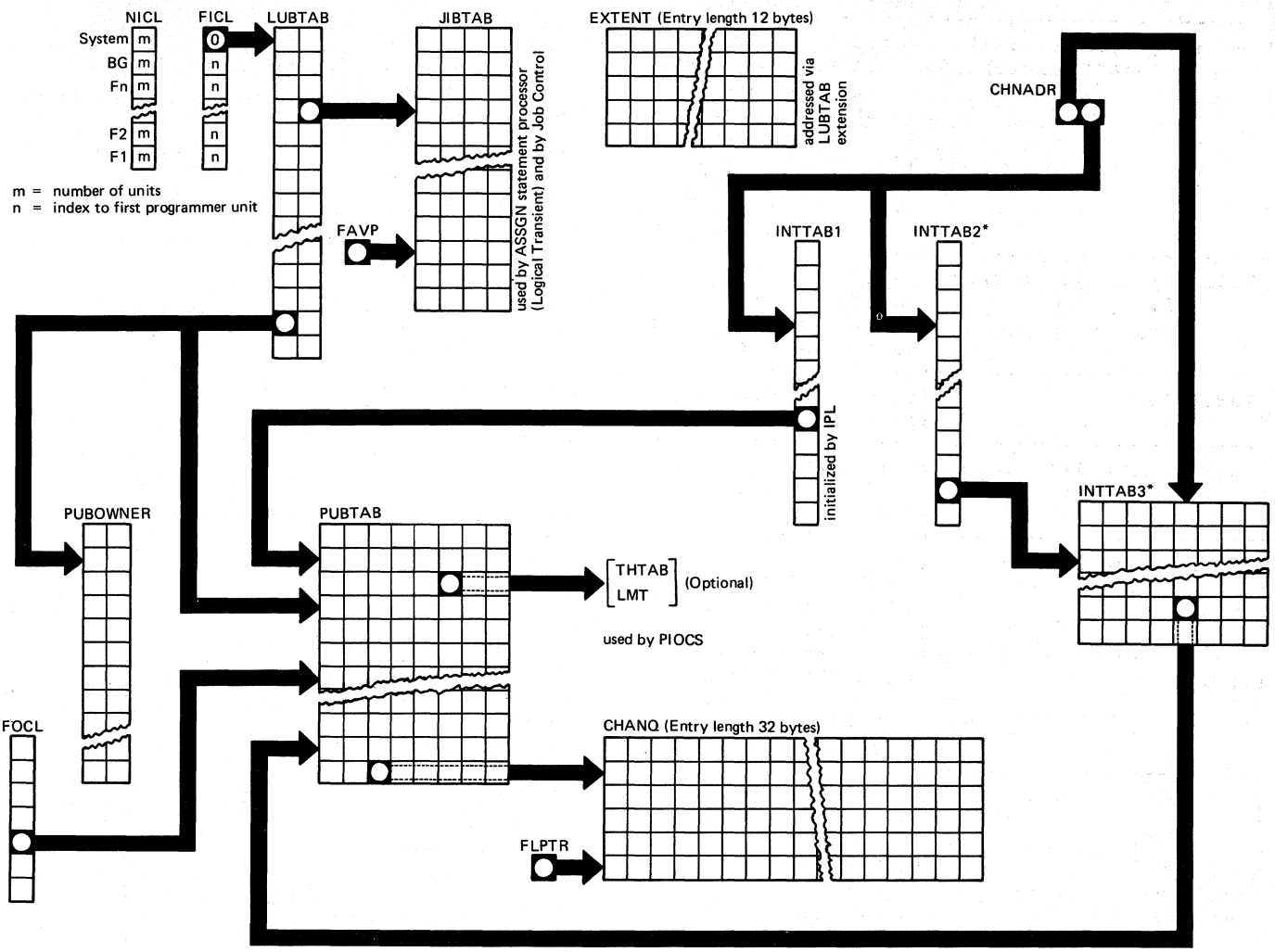


Figure 19. Logical Unit Block (LUB) Table



* Optionally allocated and initialized by IPL

Key	Explanation
NICL (Number in Class)	Byte 0 contains the number of system class LUBs. The remaining bytes contain the number of programmer class LUBs for each partition. The total number of bytes is one more than the number of partitions supported.
FICL (First in Class)	Byte 0 points to the first system class LUB in the LUB table (LUBTAB). This is always the first entry in the LUB table. The remaining bytes point to the first programmer class LUBs in the LUB table partition areas. The total number of bytes is one more than the number of partitions supported.
LUBTAB (Logical Unit Block Table)	Byte 0 of each entry is an index pointer to an entry in the PUB Table (PUBTAB) and to an entry in the PUB Ownership Table (PUBOWNER), or contains X'FF' if no logical unit is assigned. Byte 1 points to an entry in the JIB Table (JIBTAB) or contains X'FF'.

Figure 20. I/O Table Interrelationships (Part 1 of 3)

Key	Explanation
PUBTAB (Physical Unit Block Table)	Bytes 0 and 1 of each entry contain the channel and unit address of the physical device. Byte 2 points to the entry in the channel queue (CHANQ) table or contains X'FF' if no request is queued to the PUB. Byte 4 contains the device type code. Byte 5 is an index pointer to the entry in the Track-Hold Table or the line mode table (Models 115/125 only) when one of these options is active for the device. Otherwise this byte contains X'FF'. Byte 6 contains the channel scheduler flags and byte 7 the job control flags.
FOCL (First on Channel List)	Byte 0 points to the first PUBTAB entry for a device on channel 0. Byte 1 points to the first PUBTAB entry for a device on channel 1, and so on. X'FF' indicates that the associated channel is not supported.
PUBOWNER	Byte 0 of each entry is reserved. Byte 1 identifies the partition that owns the corresponding PUB.
FAVP (First Available Pointer)	This one-byte pointer to the next available entry in the JIB Table (JIBTAB) is used by the ASSGN statement processor and by job control.
JIBTAB (Job Information Block Table)	Bytes 0 and 1 of each entry contain extent or LUB information used by job control and the ASSGN statement processor. Byte 2 is a flag byte. Byte 3 is a chain byte to the next JIB for this LUB.
INTTAB1	Consists of 255 one byte entries that contain an index to a related PUB entry, or zero if an entry is to be obtained from the INTTAB2 table. The one-byte device address, stored in low core at interrupt time, is used as an index.
INTTAB2	Consists of 255 one-byte entries that contain an index to an entry in the INTTAB3 table, or zero. The one-byte device address, stored in low core of interrupt time, is used as an index.
INTTAB3	Contains one 16-byte entry for each PUB entry that has a device address (PUB byte 1), which exists at least two times on different channels. The channel-ID, stored in low core at interrupt time, is used to index a byte within the entry. This byte contains an index to the related PUB, or zero if the PUB is not defined.
CHANQ (Channel Queue	Byte 0 in each entry points to the next entry in sequence, or it contains X'FF' if the entry is the last in a chain. There are two types of chains, one is based on a PUBTAB entry (byte 2), the other on the FLPTR entry. Bytes 17 through 19 contain the address of the PUB2 entry.
FLPTR (Free List Pointer)	This one-byte pointer contains the entry index of the next free entry in the channel queue table (CHANQ).

Figure 20. I/O Table Interrelationships (Part 2 of 3)

Key	Explanation
THTAB (Track-Hold Table)	This table is built at supervisor generation time if the TRKHLD=n parameter is included in the FOPT macro. Byte 0 in each entry points to the next entry in the chain of requests for a track to be held on a specific DASD (or the next free entry if in the free list), or it contains X'FF' if the entry is the last in a chain. Bytes 1, 2, and 3 contain the CCB address. Bytes 4-11 describe the resource to be held. For CKD devices they contain the track address (BBCCHH) padded with X'0000'. For FBA devices they contain the range of FBA blocks, where bytes 4-7 present the physical block number of the first block of the range and bytes 8-11 of the last block of the range. Byte 12 contains a backward pointer. The backward pointer of the first Track Hold table entry contains the PUB index. Byte 13 is a flag and counter byte: bit 0 is turned on when a task requests a track already held by another task. Bit 1 is turned on if this entry is the first entry of a chain. The value in the low-order half-byte is incremented by one each time a task requests a hold on a track that it already holds itself. When multiple holds by one task are effective, the value in the lower-order half-byte is one less than the actual number of holds. Bytes 14 and 15 contain the key of the owning task, or zeros if the entry is free.
LMT (Line Mode Table)	This table is built at supervisor generation time if the processor is model 115 or 125. An entry is built for each device in the address range X'20' to X'3F', specified as 2703.

Figure 20. I/O Table Interrelationships (Part 3 of 3)

Job Accounting Interface Common Table (ACCTCOMN)

This table is used by the supervisor and job control to save and to communicate general job accounting information.

Displacement	Label	Description
	(ACCTCOMN)	
0-15	ACCTSVRG	Temporary register save area
16-17	ACCCTSVRX	Reserved
18-19	ACCTSVRE	Reserved
20-23	ACCTPCNT	Count of partitions using the Job Accounting interface
24	ACCTSAID	Reserved
25	ACCTFAID	Reserved
26	ACCTRAID	Accounting ID: X'00' = overhead time X'04' = allbound time byte 1 of PIK for CPU time
27	ACCTSWCH	Accounting switches: if bit=1, true; if bit=0, not true bit 0: cancel accounting bit 4: IPL indicator bit 1: no active partitions bit 5: not used bit 2: catalog in process bit 6: not used bit 3: alternate label area bit 7: not used
28-31	ACCTIME	Reserved
32-33	ACCTRESC	Reserved
34-35	ACCTUSEP	Address of user save area (ACCTUSER)
36-37	ACCTUSEL	Length of user save area (set with 1st operand of FOPT macro parameter JALIOCS)
38-39	ACCT$JOB	Job accounting partition indication
40-43	ACCTBLES	Address of BG Job Accounting Table

This table is extended with the following fields depending on the number of supported partitions.

Figure 21. Job Accounting Interface Common Table (ACCTCOMN) (Part 1 of 2)

44-47		Address of Fn Job Accounting Table
.		.
.		.
.		.
.		.
+4		Address of F2 Job Accounting Table
+4		Address of F1 Job Accounting Table
+(n*2)-1	ACCTSEAS	Control Field: prevents the accounting routine being loaded twice (depending on the number of supported partitions), otherwise the table ends here.

Bytes 124-127 (X'7C' - X'7F') of the System Communication Region (SYSCOM) contain the address of the Job Accounting Interface Common Table. Label ACCTCOMN identifies the first byte of the table.

Figure 21. Job Accounting Interface Common Table (ACCTCOMN) (Part 2 of 2)

Job Accounting Interface Partition Tables (ACCTxx)

These tables are maintained by the supervisor and job control. They contain job accounting information for single partitions.

Displacement	Label	Description
	(ACCTABLE)	
0-3	ACCTWK1	Work area used by job control in SIO update
4-7	ACCTWK2	Work area used by job control with ACCTWK1 in start/stop time routine
8-11	ACCTSVPT	Job card pointer; address of job card field following jobname
12-13	ACCTPART	ID of partition in charge (partition switch name)
14-15	ACCTLEN	Length of SIO area=6n+1, where n=number of devices for this partition in SYSGEN option JA=(n,n,...)
16-21	ACCTLOAD	Label area instruction; moves JAI label area address to OPEN/CLOSE transients
22-23	ACCTRES3	Reserved
24-27	ACCTLADD	Address of alternate label area
28-31	ACCTCPUT	Counter for program execution time ('CPU-time') in the current job step, counted in 300th of a second
32-35	ACCTOVHT	Counter for system execution time ('overhead time') in the current job step distributed in proportion to the CPU time.
36-39	ACCTBNDT	Counter for all-bound time; system wait state time divided between running partitions

Figure 22. Job Accounting Interface Partition Table (ACCTxx) (Part 1 of 2)

Displacement	Label	Description
40-47	ACCTSVJN	Save area for job name during simulated EOJ
48-55	ACCTJBNM	Job name; taken from job statement
56-71	ACCTUSRS	User information; 16 bytes from job statement
72-73	ACCTPTID	Partition ID: 'BG', 'F1', 'F2', etc. in EBCDIC format
74	ACCTCNCL	Cancel code
75	ACCTYPER	Type of record: 'S' = job step, 'L' = last step of job
76-83	ACCTDATE	Date in format specified at SYSGEN (MM/DD/YY or DD/MM/YY)
84-87	ACCTSTRT	Start time of job step in packed decimal (DHHMMSSF; F = sign)
88-91	ACCTSTOP	Stop time of job step. This value is used as the start time for the next step (same format as ACCTSTRT)
92-95	ACCTRES	Reserved
96-103	ACCTEXEC	Phase name; taken from EXEC statement
104-107	ACCTHICR	2K number of partition pages referenced (or PFIX-ed for real execution) by the problem program in the current job step.
108-111	ACCTIMES	Same as ACCTCPUT at the end of the job step
112-115		Same as ACCTOVHT at the end of the job step
116-119		Same as ACCTBNDT at the end of the job step
120	ACCTSIOS	SIO tables: 6 bytes for each device specified by SYSGEN options, as follows: 2 bytes for device address (0cuu), 4 bytes for count of SIOs in current jobstep Overflow byte: normally X'20', but is X'30' if more devices are used within a partition than specified by SYSGEN options

Notes:

1. DSECT ACCTABLE symbolically addresses the JAI Partition Tables with labels as shown. Each partition in which JAI is supported has its own JAI Partition Table. This table is labelled ACCTBG for the active partition BG; ACCTF1 for the active partition F1; ACCTF2 for F2 etc.

2. The address of this table is in the partition Communication Region at displacement 116 (X'74').

Figure 22. Job Accounting Interface Partition Table (ACCTxx) (Part 2 of 2)

Decimal Displacement	Byte Length	Description
0	3	Usage Count (number of non-ERP SIOs)
3	1	Flag Byte Bit 0: 1 = Device in intensive mode 1: 1 = Device in diagnostic mode 2: 1 = No recording mode 3: 1 = Call statistics transient 2 4: 1 = Use PUB2 name completion field 5: 1 = Volume opened on this device 6,7: Reserved
4	1	CE mode limit byte
5	1	CE mode byte/bit mask
6	6	Statistical data counters
Total length 12		

* This bit is also set (during IPL) to indicate hardware recording for the device on the Model 115/125.

Figure 23. PUB2 Table Entry Format for Unit Record and Unsupported Devices

Decimal Displacement	Byte Length	Description
3	3	Usage Count (number of non-ERP SIOs)
3	1	Flag Byte Bit 0: 1 = Device in intensive mode 1: 1 = Device in diagnostic mode 2: 1 = No recording mode 3: 1 = Call statistics transient 2 4: 1 = Use PUB2 name completion field 5: 1 = Volume opened on this device 6,7: Reserved
4	1	CE mode limit byte
5	1	CE mode byte/bit mask
6	20	Statistical data counters
Total length 26		

Figure 24. PUB2 Table Entry Format for 3886 Optical Character Reader

Decimal Displacement	Byte Length	Description
0	3	Usage Count (number of non-ERP SIOs)
3	1	Flag Byte Bit 0: 1 = Device in intensive mode 1: 1 = Device in diagnostic mode 2: 1 = No recording mode 3: 1 = Call statistics transient 2 4: 1 = Use PUB2 name completion field 5: 1 = Volume openend on this device 6,7: Reserved
4	1	CE mode limit byte
5	1	CE mode byte/bit mask
6	1	Flag byte Bit 0: 1 = Soft DASD error is queued 1: 1 = ERP requests logging of error 2-7: Reserved
7	2	Reserved
9	1	Physical module identifier
10	6	Volume serial number
End 3340. Total length 16		
16	8	Statistical data counters
End DASD except 3330/3333/3350/FBA. Total length 24		
24	8	Additional statistical data counters (3330/3333/3350/FBA only)
End 3330/3333/3350/FBA. Total length 24		

Figure 25. PUB2 Table Entry Format for DASD

Decimal Displacement	Byte Length	Description
0	3	Usage Count (number of non-ERP SIOs)
3	1	Flag Byte Bit 0: 1 = Device in intensive mode 1: 1 = Device in diagnostic mode 2: 1 = No recording mode 3: 1 = Call statistics transient 2 4: 1 = Use PUB2 name completion field 5: 1 = Volume opened on this device 6,7: Reserved
4	1	CE mode limit byte
5	1	CE mode byte/bit mask
6	2	Name of ERP that wants control
8	1	Flag byte 1 Bit 0: Reserved 1: 1 = Unsolicited interrupts 2: 1 = ERP is in control 3: 1 = ERP requests repositioning 4: 1 = Use original TIE byte 0 = Use opposite TIE byte 5: 1 = Intercept next SIO request 6: 1 = ERP read opposite request 7: 1 = Restart user's CCW chain
9	1	Flag byte 2 Bit 0: 1 = Last ERP operation was ERG 1: 1 = Last ERP operation was reposition 2: 1 = Cleaner action in progress 3: 1 = Read Opposite Recovery in progress 4: 1 = Message stored in P20RGTIE 5: 1 = Error on attempt to recover by respositioning 6: 1 = Data check after ERP in control 7: Reserved
10	1	Flag byte 3 Bit 0: 1 = Failing CCW is Writer or Control command 1: 1 = User reading backwards 2: 1 = Read Opposite Recovery (ROR) 3: 1 = Maximum ROR retries 4: 1 = Command chaining ROR 5: 1 = ROR suppressed incorrect length 6: 1 = User used SILI 7: Reserved

Figure 26. PUB2 Table Entry Format for Tapes (Part 1 of 2)

Decimal Displacement	Byte Length	Description
11	1	Temporary read count
12	1	Temporary write count
13	1	Noise record count
14	2	Erase gap count
16	2	Cleaner action count
18	1	Permanent read errors count
19	1	Permanent write errors count
20	1	TIE original direction
21	1	TIE opposite direction
22	1	ERP counter 0
23	1	ERP counter 1
24	8	ERP work area
32	6	Tape serial number
38	2	Block length
40	4	User ROR command address from CSW
44	2	User ROR residual count from CSW
46	2	Reserved
48	16	Work area for rewind-unload error action
64	10	2400-series statistical data counter area
74	2	Reserved
End 2400-series: Total length 76		
64	20	3410/3420 statistical data counter area
End 3410/3420/8809: Total entry length 84		

Note: PUB2 Table is not generated for Model 115 or 125 without RMSR support.

Figure 26. PUB2 Table Entry Format for Tapes (Part 2 of 2)

Decimal Displacement	Byte Length	Description
0	6	Standard PUB2 area prefix
6	1	SDR counter 1 (channel data checks)
7	1	2 (channel control checks)
8	1	3 (interface control checks)
9	1	4 (stacker misfolds)
10	1	5 (burster-trimmer jams)
11	1	6 (no-burst checks)
12	1	7 (burster-stacker jams)
13	1	8 Reserved
14	1	9 Reserved
15	1	Default flags Bit 0: 1 = BURST defaults to Y 1-7: Reserved
16	4	Default FCB identifier
20	4	Default character arrangement table identifier
24	4	Default copy modification identifier
28	4	Default forms overlay frame identifier
32	4	Default paper forms identifier
36	4	IDs of character sets loaded in printer
40	1	WCGMs with modified character sets Bit 0: contains a modified character set 1: " " " " " 2: " " " " " 3: " " " " " 4-7: Reserved
41	1	Flag byte Bit 0: Reserved 1: Reserved 2: 0 = Threaded in continuous forms stacker 2: 1 = Threaded in burster-trimmer stacker 3: Reserved 4: 0 = DCHK=B was coded or DCHK was omitted 4: 1 = DCHK=U was specified 5-7: Reserved

Figure 27. PUB2 Table Entry Format for 3800 (Part 1 of 2)

Decimal Displacement	Byte Length	Description
42	1	Flag byte Bit 0-1: Reserved 2: 0 = Records do not contain TRCs 2: 1 = Records contain TRCs 3-7: Reserved
43	1	Flag byte Reserved
44	4	ID for currently loaded FCB
48	4	ID for first character arrangement table currently loaded
52	4	ID for second character arrangement table currently loaded
56	4	ID for third character arrangement table currently loaded
60	4	ID for fourth character arrangement table currently loaded
64	4	ID of the character arrangement table used for the current copy modifications
68	4	ID of the copy modification module currently loaded
72	4	ID of forms currently loaded
76	4	ID of current forms overlay frame
80	8	The eight copy group count received by SETPRT
88	1	Copy group numbers if grouping, transmission number if not grouping
89	1	Flash count presently set
90	2	Reserved

Figure 27. PUB2 Table Entry Format for 3800 (Part 2 of 2)

This glossary includes definitions developed by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). This material is reproduced from the American National Dictionary for Information Processing, copyright 1977 by the Computer and Business Equipment Manufacturers Association, copies of which may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018.

For a more complete list of data processing terms, refer to IBM Data Processing Techniques, A Data Processing Glossary, GC20-1699.

address translation: The process of changing the address of an item of data or an instruction from its virtual address to its real storage address. See also dynamic address translation.

American national standard label format: The tape file format used when the label is written in the ASCII mode.

ASCII (American national standard code for information interchange): A 128-character, 7-bit code. The high-order bit in the System/370 8-bit environment is zero.

ASI (automatic system initialization)

block prefix: An optional, 0-99 byte field preceding an ASCII record. It contains user specified data or, for variable-length (format D) records, the physical record length.

CCH (channel check handler): A feature that assesses channel errors to determine if the system can continue operations.

change bit: A bit associated with a page in real storage: the change bit is turned "ON" by hardware whenever the associated page in real storage is modified. In DOS/VS, there is a change bit in the storage key associated with each 2K storage block.

channel inboard error: An error that occurs between one I/O device and the central processing unit.

channel program translation: In a channel program, replacement, by software, of virtual addresses with real addresses.

CKD (count key data): The DASD architecture with variable block size and addresses of the format BBCCHHR (bin, cylinder, head, record); as opposed to Fixed Block Architecture.

data conversion: The process of changing data from one form of representation to another.

diskette: A flexible magnetic oxide coated disk, permanently enclosed in a semi-rigid protective plastic jacket approximately 8 inches square. During data processing operations, the disk turns freely within the jacket. It is capable of storing 1898 128-character data records.

dynamic address translation (DAT): (1) The change of a virtual storage address to a real storage address during execution of an instruction. See also address translation. (2) A hardware feature that performs the translation.

ECPS:DOS/VS (extended control program support for DOS/VS: An implementation of the virtual storage concept which does not require software participation in the translation of virtual addresses into real addresses.

EREP (environmental recording, editing, and printing): A program that processes the data contained on the system recorder file.

EVA (error volume analysis): An option of TES that causes the system to issue a message to the operator when a number of temporary read or write errors (specified by the user at system generation time) has been exceeded on a currently accessed tape volume.

extended control (EC) mode: A mode in which all the features of a System/370 computing system, including dynamic address translation, are operational.

FBA (fixed block architecture): The DASD architecture where data addresses are by block numbers only, since blocks have a fixed size, as opposed to CKD.

FCB (forms control buffer): A buffer that is used to store vertical formatting information for printing, each position corresponding to a line on the form.

file section number: For ASCII files, the number indicates the order of a volume in a given file or multifile set. The first file must be numbered 0001. The ASCII file section number is equivalent to the EBCDIC

volume sequence number.

fixed: In DOS/VS, not capable of being paged out.

fixed page: A page in real storage that is not to be paged out.

frame: Same as page frame

frame number: In DOS/VS, the part of a real storage address needed to refer to a frame. See also page number.

frame table: Same as page frame table.

frame table entry (FTE): In DOS/VS, an entry in the page frame table that describes how a frame is being used.

FTE: Frame table entry.

hold page queue: A queue to which pages in real storage are initially assigned through a page-in.

invalid page: A page that cannot be directly addressed by the dynamic address translation feature of the central processing unit.

I/O (Input/Output) error logging: The process of recording OBR and SDE records on the system recorder file.

job accounting interface: A function that accumulates accounting information for each job step to charge usage of the system, help plan new applications, and help supervise system operation more efficiently.

load point: The beginning of the recording area on a reel of magnetic tape.

main page pool: The set of all page frames in real storage not assigned to the supervisor or one of the real partitions.

MCAR (machine check analysis and recording): A feature that records System/370 machine check interrupt error information on the system recorder file and then attempts to recover from the interrupt.

MCI (machine check interrupt): The interrupt that occurs if the central processing unit fails to operate.

nonstandard labels: Labels that do not conform to the System/370 standard label specifications. They can be any length, need not have a specified identification, and do not have a fixed format.

page: (1) A fixed-length block of instructions, data, or both, that can be transferred between real storage and page data set. (2) To transfer instructions, data, or both between real storage and external page storage.

page data set: A file in external page storage, in which pages are stored.

page fault: A program interruption that occurs when a page that is marked "not in real storage" is referred to by an active page. Synonymous with page translation exception.

page fixing: Marking a page as nonpageable so that it remains in real storage.

page frame: A block of real storage that can contain a page. Synonymous with frame.

page frame table: In DOS/VS, a table that contains an entry for each frame. Each frame table entry describes how the frame is being used.

page-in: The process of transferring a page from the external page storage to real storage.

page number: The part of a virtual storage address needed to refer to a page. See also frame number:

page-out: The process of transferring a page from real storage to external page storage.

page pool: The set of page frames available for paging virtual mode programs.

page table (PT): A table that indicates whether a page is in real storage and correlates virtual addresses with real storage addresses.

paging: The process of transferring pages between real storage and external page storage.

PCIL (private core image library): A file referenced in the same manner and for the same purposes as the system core image library, but distinct from the system core image library. PCIL increases available core image library space to enable compiling, linkage editing, and executing in the foreground partition, when a private core image library is assigned to that foreground partition.

real address: The address of a location in real storage.

real address area: In DOS/VS, the area of virtual storage where virtual addresses are equal to real addresses.

real mode: In DOS/VS, the mode of a program

that may not be paged.

real partition: (1) In DOS/VS, a division of the real address area of virtual storage that may be allocated for programs that are not to be paged. (2) An area of real storage that can contain pages that are to be PFIXed.

real storage: The storage of a System/370 computing system from which the central processing unit can directly obtain instructions and data, and to which it can directly return results.

reference bit: A bit associated with a page in real storage; the reference bit is turned "ON" by hardware whenever the associated page in real storage is referred to (read or stored into). In DOS/VS, there is a reference bit in the storage key associated with each 2K storage block.

reliability data extractor (RDE): A function that provides hardware reliability data that is analyzed by IBM.

RMSR (Recovery Management Support Recorder): A feature that consists of the MCAR (Machine Check Handler) functions. RMSR gathers information about System/370 hardware reliability and attempts certain error recovery operations. RMSR is a part of the entire reliability, availability, and serviceability support for System/370.

rotational position sensing: A standard feature of IBM 3330/3333 and an optional feature of IBM 3340 disk storage devices. It permits a device to disconnect from a block multiplexer channel during rotational positioning operations, thereby allowing the channel to service other devices on the channel during the positioning delay.

second level directory: A table located in the supervisor, and containing the first phasenames found on the corresponding directory tracks of the system core image library.

segment: A continuous 64K area of virtual storage starting and ending on a 64K boundary.

segment table (SGT): A table used in dynamic address translation to control user access to virtual storage segments. Each entry indicates the length, location, and availability of a corresponding page table.

shared virtual area: An area located in the highest addresses of virtual storage. It can contain a system directory list (SDL) of frequently used phases, resident programs that can be shared between partitions, and an area for system GETVIS support.

SLA (symbolic label access): A reentrant program, resident in the SVA to read, write, and modify labels from the label area.

system directory list: A list containing directory entries of frequently used phases and of all phases resident in the shared virtual area. This list is placed in the shared virtual area.

system recorder file: The file that is used to record hardware reliability data.

TES (tape error statistics): A facility provided by RMSR that monitors and records read and write errors per volume for IBM 2400 and 3400 series tapes.

thrashing: A condition in which the system can do little useful work because of excessive paging.

TOD-clock: A 64-bit binary counter. It provides a consistent measure of elapsed time for time-of-day indication.

translation tables: Page tables and segment tables.

UCB (universal character set buffer): A buffer that is used to store print control information for output with a UCS print train.

UCS (universal character set): A printer feature that permits the use of a variety of character arrays.

virtual address: An address that refers to virtual storage and must, therefore, be translated into a real storage address when it is used.

virtual address area: In DOS/VS, the area of virtual storage whose addresses are greater than the highest address of the real address area.

virtual mode: In DOS/VS, the mode of a program which may be paged.

virtual partition: In DOS/VS, a division of the virtual address area of virtual storage that is allocated for programs that are to be paged.

virtual storage: Addressable space that appears to the user as real storage, from which instructions and data are mapped into real storage locations. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, rather than by the actual number of real storage locations.

working set: The set of a user's pages that must be in real storage in order to avoid excessive paging.

zone: Difference (hours and minutes) between Greenwich Mean Time and local time.

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

X

Z

Initial Program Load and Job Control Logic Printed in U.S.A. SY33-8555-4

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

READER'S COMMENT FORM

SY33-8555-4

This sheet is for comments and suggestions about this manual. We would appreciate *your* views, favorable or unfavorable, in order to aid us in improving *this* publication. This form will be sent directly to the author's department. Please include your name and address if you wish a reply. Contact your IBM branch office for answers to technical questions about the system or when requesting additional publications. Thank you.

Name

Address

What is your occupation?

How did you use this manual?

As a reference source

As a classroom text

As a self-study text

Your comments* and suggestions:

*** We would especially appreciate your comments on any of the following topics:**

Clarity of the text	Accuracy	Index	Illustrations	Appearance	Paper
Organization of the text	Cross-references	Tables	Examples	Printing	Binding

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for system analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
1133 Westchester Avenue
White Plains, N.Y. 10604

Attention: Department 813 BP

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

CUT ALONG THIS LINE

Initial Program Load and Job Control Logic Printed in U.S.A. SY33-8555-4